Programmer's Guide

Agilent Technologies E4406A VSA Series Transmitter Tester

Agilent Technologies

Manufacturing Part Number: E4406-90303
Supersedes E4406-90176

Printed in USA

May 2007

Notice

The information contained in this document is subject to change without notice.

Agilent Technologies makes no warranty of any kind with regard to this material, including but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Agilent Technologies shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Technology Licenses

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

Restricted Rights Legend

If software is for use in the performance of a U.S. Government prime contract or subcontract, Software is delivered and licensed as "Commercial computer software" as defined in DFAR 252.227-7014 (June 1995), or as a "commercial item" as defined in FAR 2.101(a) or as "Restricted computer software" as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Agilent Technologies' standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

Safety Information

The following safety symbols are used throughout this manual. Familiarize yourself with the symbols and their meaning before operating this instrument.

WARNING ***Warning*** **denotes a hazard. It calls attention to a procedure which, if not correctly performed or adhered to, could result in injury or loss of life. Do not proceed beyond a warning note until the indicated conditions are fully understood and met.**

CAUTION *Caution* denotes a hazard. It calls attention to a procedure that, if not correctly performed or adhered to, could result in damage to or destruction of the instrument. Do not proceed beyond a caution sign until the indicated conditions are fully understood and met.

NOTE *Note* calls out special information for the user's attention. It provides operational information or additional instructions of which the user should be aware.

WARNING **This is a Safety Class 1 Product (provided with a protective earth ground incorporated in the power cord). The mains plug shall be inserted only in a socket outlet provided with a protected earth contact. Any interruption of the protective conductor inside or outside of the product is likely to make the product dangerous. Intentional interruption is prohibited.**

WARNING **No operator serviceable parts inside. Refer servicing to qualified personnel. To prevent electrical shock do not remove covers.**

CAUTION Always use the three-prong AC power cord supplied with this product. Failure to ensure adequate grounding may cause product damage.

Where to Find the Latest Information

Documentation is updated periodically. For the latest information about Agilent Technologies E4406 VSA Series Transmitter Tester, including firmware upgrades and application information, please visit the following Internet URL:

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

Contents

Contents

Contents

Contents

Contents

Contents

Contents

Contents

List of Commands

List of Commands

List of Commands

List of Commands

List of Commands

1 Preparing for Use

This instrument uses the Standard Commands for Programmable Instruments (SCPI) programming language. For information on writing SCPI commands see "SCPI Language Basics" on page 79.

What's in This Chapter?

www.agilent.com/find/vsa

Get the latest listing of SCPI commands for this instrument at the above web location. Look under technical support information.

Digital Communications Measurements Information

Additional measurement application information is available through your local Agilent Technologies sales and service office. Some application notes are listed below:

Description	Agilent Part Number
Digital Modulation in Communications Systems - An Introduction Application Note 1298	5965-7160E
Understanding CDMA Measurements for Base Stations and Their Components Application Note 1311	5968-0953E
Designing and Testing 3GPP W-CDMA User Equipment (UE) Application Note 1356	5980-1238E
Designing and Testing 3GPP W-CDMA Base Stations (BTS) Application Note 1355	5980-1239E
Designing and Testing IS-2000 Mobile Stations Application Note 1358	5980-1237E
Designing and Testing IS-2000 Base Stations Application Note 1357	5980-1303E

Description	Agilent Part Number
Understanding GSM Transmitter Measurements for Base Transceiver Stations and Mobile Stations Application Note 1312	5966-2833E
Understanding PDC and NADC Transmitter Measurements for Base Transceiver Stations and Mobile Stations Application Note 1324	5968-5537E

Programming the Transmitter Tester

The E4406A VSA Series Transmitter Tester has several different measurement modes. The measurement commands that are available to you change, depending on which mode is selected. Use INSTrument:SELect to select the desired mode.

Most modes are optional and must be installed into instrument memory before they can be used. See "Installing Optional Measurement Personalities" on page 44, if your measurement mode is not installed.

Table 1-1 Available Modes and Measurements

Modes	Measurement Keywords
Basic - standard `INST:SELECT BASIC`	• ACP - adjacent channel power measurement • CHPower - channel power measurement • PSTatistic - power statistics (CCDF) measurement • SPECtrum - spectrum (frequency domain) measurement • WAVeform - waveform (time domain) measurement
cdmaOne - Option BAC `INST:SELECT CDMA`	• ACP - adjacent channel power ratio measurement • CDPower - code domain power measurement • CHPower - channel power measurement • CSPur - close spurs measurement • RHO - rho (waveform quality) measurement • SPECtrum - spectrum (frequency domain) measurement • TSpur - transmit band spurs measurement • WAVeform - waveform (time domain) measurement

Table 1-1 Available Modes and Measurements

Modes	Measurement Keywords
cdma2000 - Option B78 `INST:SELECT CDMA2K`	• ACP - adjacent channel power ratio measurement • CHPower - channel power measurement • PSTatistic - power statistics (CCDF) measurement • CDPower - code domain power measurement • EVMQpsk - QPSK error vector magnitude measurement • RHO - modulation accuracy (composite rho) measurement • OBW - occupied bandwidth measurement • SEMask - spectrum emission mask measurement • IM - intermodulation measurement • SPECtrum - spectrum (frequency domain) measurement • WAVeform - waveform (time domain) measurement
W-CDMA (3GPP) - Option BAF `INST:SELECT WCDMA`	• ACP - adjacent channel power ratio measurement • CDPower - code domain power measurement • CHPower - channel power measurement • PSTatistic - power statistics (CCDF) measurement • EVMQpsk - QPSK error vector magnitude measurement • RHO - modulation accuracy (composite EVM) measurement • OBW - occupied bandwidth measurement • SEMask - spectrum emission mask measurement • IM - intermodulation measurement • MCPower - multi carrier power measurement • SPECtrum - spectrum (frequency domain) measurement • WAVeform - waveform (time domain) measurement

Table 1-1 Available Modes and Measurements

Modes	Measurement Keywords
EDGE w/GSM - Option 202 or EDGE w/GSM - Option 252 `INST:SELECT EDGEGSM`	• ORFSpectrum - GMSK output RF spectrum measurement • PFERror - GMSK phase and frequency error measurement • PVTime - GMSK power versus time measurement • TXSPurs - GMSK transmit band spurs measurement • EEVM - EDGE error vector magnitude measurement • EPVTime - EDGE power versus time measurement • EORFspectr - EDGE output RF spectrum measurement • ETXSpurs - EDGE transmit band spurs measurement • SPECtrum - spectrum (frequency domain) measurement • TXPower - transmit power measurement • WAVeform - waveform (time domain) measurement
GSM - Option BAH `INST:SELECT GSM`	• ORFSpectrum - output RF spectrum measurement • PFERror - phase and frequency error measurement • PVTime - power versus time measurement • TXSPurs - transmit band spurs measurement • SPECtrum - spectrum (frequency domain) measurement • TXPower - transmit power measurement • WAVeform - waveform (time domain) measurement

Table 1-1 Available Modes and Measurements

Modes	Measurement Keywords

NADC - Option BAE
`INST:SELECT NADC`
or
PDC - Option BAE
`INST:SELECT PDC`
	• ACP - adjacent channel power measurement
	• EVM - error vector magnitude measurement
	• OBWidth - occupied bandwidth measurement
	• SPECtrum - spectrum (frequency domain) Measurement
	• WAVeform - waveform (time domain) measurement

iDEN - Option HN1 `INST:SELECT IDEN`	• ACP - adjacent channel power measurement • BER - bit error rate measurement • OBWidth - occupied bandwidth measurement • SPECtrum - spectrum (frequency domain) Measurement • WAVeform - waveform (time domain) measurement
Service - standard `INST:SELECT SERVICE`	• AREFerence - (internal) 50 MHz amplitude reference measurement • PVTime - power versus time measurement • SENSors - (internal) temperature sensors measurement • SPECtrum - spectrum (frequency domain) measurement • TBFRequency - (internal) timebase frequency measurement • WAVeform - waveform (time domain) measurement

Installing Optional Measurement Personalities

When you install a measurement personality, you need to follow a three step process:

1. Determine whether your memory capacity is sufficient to contain all the options you want to load. If not, decide which options you want to install now, and consider upgrading your memory. Details follow in "Do You Have Enough Memory to Load All Your Personality Options?" on page 44.
2. Install the measurement personality firmware into the instrument memory. Details follow in "Loading an Optional Measurement Personality" on page 47.
3. Enter a license key that activates the measurement personality. Details follow in "Obtaining and Installing a License Key" on page 48.

Adding measurement personalities requires the purchase of an upgrade kit for the desired option. The upgrade kit contains the measurement personality firmware and an entitlement certificate that is used to generate a license key from the internet website. A separate license key is required for each option on a specific instrument serial number and host ID.

For the latest information on Agilent Spectrum Analyzer options and upgrade kits, visit the following web location:

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

Do You Have Enough Memory to Load All Your Personality Options?

If you do not have memory limitations then you can skip ahead to the next section "Loading an Optional Measurement Personality" on page 47. If after installing your options you get error messages relating to memory issues, you can return to this section to learn more about how to optimize your configuration.

If you have 64 MBytes of memory installed in your instrument, you should have enough memory to install at least four optional personalities, with plenty of memory for data and states.

The optional measurement personalities require different amounts of memory. So the number of personalities that you can load varies. This is also impacted by how much data you need to save. If you are having memory errors you must swap the applications in or out of memory as needed. If you only have 48 MBytes of memory, you can upgrade your

hardware to 64 MBytes.

To see the size of your installed memory for E4406A Transmitter Testers:

1. Press the **System** key, **MORE (1 of 3)**, and **MORE (2 of 3)** keys.
2. Read the **File System** Key - The total of the entries for `Used` and `Free` memory will total the installed flash memory, either 48 or 64 MBytes.

If you have 48 MBytes of memory, and you want to install more than 3 optional personalities, you may need to manage your memory resources. The following section, "How to Predict Your Memory Requirements" on page 46, will help you decide how to configure your installed options to provide optimal operation.

How to Predict Your Memory Requirements

If you plan to install many optional personalities, you should review your memory requirements, so you can determine whether you have enough memory. There is an Agilent "Memory Calculator" available online that can help you do this, or you can make a calculated approximation using the information that follows. You will need to know your instrument's installed memory size as determined in the previous section and then select your desired applications.

To calculate the available memory on your E4406, see:
http://www.agilent.com/find/e4406a_firmware

Select the "Memory Calculator" link. You can try any combination of available personalities to see if your desired configuration is compatible with your installed memory.

You can manually estimate your total memory requirements by adding up the memory allocations described in the following steps. Compare the desired total with the available memory that you identified in the previous section.

1. Program memory - Select option requirements from the table "Measurement Personality Options and Memory Required" on page 46.
2. Screens - .gif files need 20-25 kB each
3. State memory - State file sizes range from 21 kB for SA mode to 40 kB for W-CDMA. The state of every mode accessed since power-on will be saved in the state file. File sizes can exceed 150 kB each when several modes are accessed, for each state file saved.

TIP

State memory retains settings for all states accessed before the **Save State** command. To reduce this usage to a minimum, reduce the modes accessed before the **Save State** is executed. You can set the PSA to boot into a selected mode by accessing the desired mode, then pressing the **System**, **Power On/Preset**, **Power On** keys and toggle the setting to **Last**.

Measurement Personality Options and Memory Required

Personality Options for E4406A Transmitter Tester [a]	**Option**	**File Size (E4406A Rev: A.10)**
cdmaOne measurement personality	**BAC**	1.82 Mbytes
NADC measurement personality	**BAE**	1.10 Mbytes
PDC measurement personality	**BAE**	1.23 Mbytes
W-CDMA or W-CDMA, HSDPA, HSUPA measurement personality	**BAF**, **210**	5.00 Mbytes

Personality Options for E4406A Transmitter Tester [a]	Option	File Size (E4406A Rev: A.10)
cdma2000 or cdma2000 w/ 1xEV-DV measurement personality	**B78, 214**	3.88 Mbytes
1xEV-DO measurement personality	**204**	4.84 Mbytes
GSM (with EDGE) measurement personality	**202**	3.56 Mbytes
GSM measurement personality	**BAH**	2.51 Mbytes
EDGE upgrade from BAH measurement personality	**252 (202)**	3.56 Mbytes
iDEN measurement personality	**HN1**	2.10 Mbytes
WiDEN measurement personality	**HN1**	1.58 Mbytes
Baseband I/Q Inputs	**B7C**	n/a (hardware only)

a. Available as of the print date of this guide.

Memory Upgrade Kits

The VSA 64 MByte Memory Upgrade kit part number is E4406AU-ANE.

For more information about memory upgrade kits contact your local sales office, service office, or see:
http://www.agilent.com/find/sa_upgrades

Loading an Optional Measurement Personality

You must use a PC to load the desired personality option into the instrument memory. Loading can be done from a firmware CD-ROM or by downloading the update program from the internet. An automatic loading program comes with the files and runs from your PC.

You can check the Agilent internet website for the latest E4406 firmware versions available for downloading:

http://www.agilent.com/find/e4406a_firmware

NOTE When you add a new option, or update an existing option, you will get the updated versions of all your current options as they are all reloaded simultaneously. This process may also require you to update the instrument core firmware so that it is compatible with the new option.

Depending on your installed hardware memory, you may not be able to fit all of the available measurement personalities in instrument memory at the same time. You may need to delete an existing option file from memory and load the one you want. Use the automatic update

program that is provided with the files. Refer to the table showing "Measurement Personality Options and Memory Required" on page 46. The approximate memory requirements for the options are listed in this table. These numbers are worst case examples. Some options share components and libraries, therefore the total memory usage of multiple options may not be exactly equal to the combined total.

Obtaining and Installing a License Key

If you purchase an optional personality that requires installation, you will receive an "Entitlement Certificate" which may be redeemed for a license key specific to one instrument. Follow the instructions that accompany the certificate to obtain your license key.

To install a license key for the selected personality option, use the following procedure:

NOTE You can also use this procedure to reinstall a license key that has been deleted during an uninstall process, or lost due to a memory failure.

For E4406:

1. Press **System**, **More**, **More**, **Install**, **Choose Option** to accesses the alpha editor. Use this alpha editor to enter letters (upper-case), and the front-panel numeric keys to enter numbers for the option designation. You will validate your option entry in the active function area of the display. Then, press the **Done** key.

NOTE Before you enter the license key for the EDGE Retrofit Option 252, you must already have entered the license key for the GSM Option BAH.

2. Press **License Key** to enter the letters and digits of your license key. You will validate your license key entry in the active function area of the display. Then, press the **Done** key.
3. Press the **Install Now** key. The message "`New option keys become active after reboot.`" will appear, along with the **Yes**/**No** menu: press the **Yes** key and cycle the instrument power off and then on to complete your installation process, or press the **No** key to cancel the installation process.

Viewing a License Key

Measurement personalities purchased with your instrument have been installed and activated at the factory before shipment. The instrument requires a **License Key** unique to every measurement personality purchased. The license key is a hexadecimal number specific to your measurement personality, instrument serial number and host ID. It enables you to install, or reactivate that particular personality.

Use the following procedure to display the license key unique to your personality option that is already installed in your E4406:

Press **System**, **More**, **More**, **Install**, **Choose Option** to enter the letters and numbers for the option you want. You can see the key on the **License Key** menu key. Press the **Done** key.

NOTE *You will want to keep a copy of your license key in a secure location. Press **System**, **More**, then **Show System**, and print out a copy of the display that shows the license numbers. If you should lose your license key, call your nearest Agilent Technologies service or sales office for assistance.*

Using the Uninstall Key on E4406A

This key will make the option unavailable for use, but will not delete it from memory. The message "`Application Not Licensed`" will appear in the Status/Info bar at the bottom of the display. Record the 12-digit license key for the option before you delete it. If you want to use that measurement personality later, you will need the license key to reactivate the personality firmware.

NOTE Using the **Uninstall** key does not remove the personality firmware from the instrument memory, and does not free memory to be available to install another option. If you need to free memory to install another option, refer to the instructions for loading firmware updates available at the URL: **http://www.agilent.com/find/vsa/**

1. Press **System**, **More (1 of 3)**, **More (2 of 3)**, **Uninstall**, **Choose Option** to access the alpha editor. Use this alpha editor to enter the letters (upper-case), and the front-panel numeric keys to enter the numbers (if required) for the installed option. You will validate your option entry in the active function area of the display. Then, press the **Done** key.
2. Pressing the **Uninstall Now** key will activate the **Yes**/**No** menu: press the **Yes** key to continue your uninstall process, or press the **No** key to cancel the uninstall process.
3. Cycle the instrument power off and then on to complete the uninstall process.

Ordering Optional Measurement Personalities

When you order a personality option, you will receive an entitlement certificate. Then you will need to go to the Web site to redeem your entitlement certificate for a license key. You will need to provide your instrument serial number and host ID, and the entitlement certificate number.

Required Information:	**Front Panel Key Path:**
Model #: (Ex. E4440A)	
Host ID: _____	**System**, **Show System**
Instrument Serial Number: _____	**System**, **Show System**

Writing Your First Program

When the instrument has been connected to a computer, the computer can be used to send instrument instructions to make fast, repeatable measurements. A variety of different programming languages, computer types, and interface buses can be used for this process. The following section describes some basic steps for making a measurement program.

NOTE

Remember that in any type programming using LAN you should avoid constantly opening and closing connections. This uses up processing resources, adds to your system overhead, and can cause problems with asynchronous implementation of successive commands. When you are sending the instrument multiple commands: open the connection, send all the commands, and close the connection.

Three Basic Steps in a Measurement

Step	Tasks (SCPI Command Subsystem)
1. Set system parameters	• Printer setup (HCOPy) • I/O & addressing (SYSTem) • Display configuration (DISPlay) • Data formatting (FORMat) • Status and errors (*IEEE and STATus)
2. Select mode & setup mode	• Mode selection (INSTrument:SELect) • Standard selection (SENSe:RADio) • RF channel (SENSe:CHANnel) • Frequency (SENSe:FREQuency) • Triggering (TRIGger) • Input (INPut)
3. Select measurement & setup measurement	• Measurement selection (MEASure) • Meas control/restart (INITiate) • Markers (CALCulate:<meas>:MARKer) • Averaging (SENSe:<meas>:AVERage) • Bandwidth (SENSe:<meas>:BWIDth) • FFT & meas window (SENSe:<meas>:FFT)

Programming a Measurement

General recommendations for writing a measurement program:

- Include comment lines in your program to describe what is happening at each point. The way you include comment lines is dependent on the controller and the programming language that you are using.

- Use variables for function values. List the variables at the beginning of the program.
- Perform the measurement manually, keeping track of the key functions used. Identify the programming commands equivalent to these front-panel keys.
- Select the instrument mode with INST:SELect. Set the mode setup for things like your desired communications standard, channel frequency and triggering.
- In the program, execute an instrument preset (*RST) and select single-sweep mode (INITiate:CONTinuous OFF) before setting other instrument functions.
- Use the MEASure group of commands, described in Chapter 5 , "Language Reference" on page 229. MEASure commands make the measurement using the standard procedure and limits. You can alter some of the measurement defaults by using commands in the SENSe:<meas> subsystem. Once altered, use the CONFigure, FETCh, READ, and INITiate commands to perform the measurements.
- The instrument can return different types of results for a particular measurement. These results are described in the language reference section on the MEASure group of commands.
- Execute the desired commands in logical order. Multiple SCPI commands can be included on one line. See "SCPI Language Basics" on page 79.

File Naming Rules

File names for storing instrument states or other data files in the analyzer should follow DOS conventions.

- They can be up to eight characters long. In addition, they can have a file extension up to three characters long. The analyzer can assign the extension.
- They are not case sensitive. It does not matter whether you use upper case or lower case letters when you enter them.
- They can only contain the letters A through Z and the numbers 0 through 9.
- They cannot contain any special characters (except the period that separates the name from the extension).
- They cannot be identical to the name of another file in the same directory.

Cables for Connecting to RS-232

There are a variety of cables and adapters available for connecting to PCs, and printers. Several of these are documented in the following wiring diagrams. You need to find out what connections your equipment uses to identify the cables and/or adapters that you will need.

HP/Agilent 34398A
RS-232
Cable Kit — This kit comes with an RS-232, 9-pin female to 9-pin female null modem/printer cable and one adapter 9-pin male to 25-pin female (part number 5181-6641). The adapter is also included in the 34399A RS-232 Adapter Kit.

HP/Agilent 34399A
RS-232
Adapter Kit — This kit includes four adapters to go from an DB9 female cable (34398A) to a PC/printer DB25 male or female, or to a modem DB9 female or DB25 female.

Figure 1-1 HP/Agilent 24542U Cable

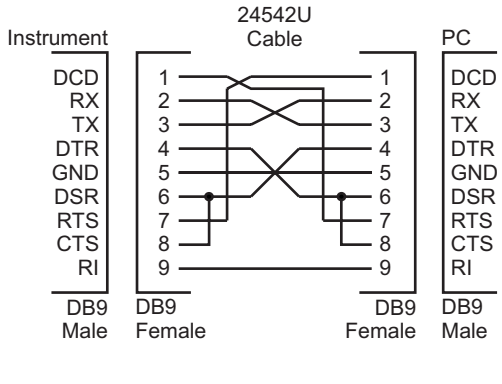
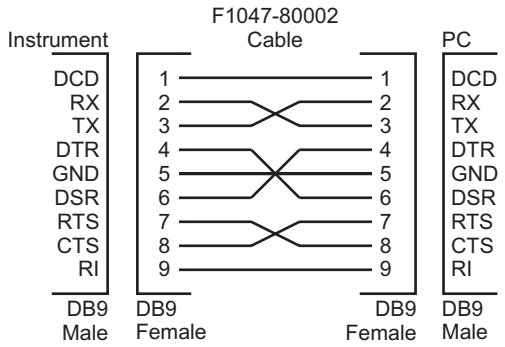
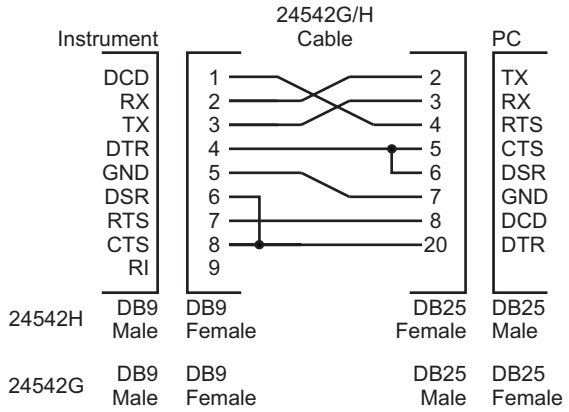


Figure 1-2 HP/Agilent F1047-80002 Cable



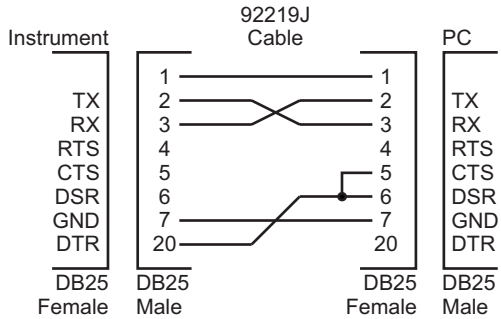
ca86a

Figure 1-3 HP/Agilent 24542G/H Cable



ca87a

Figure 1-4 HP/Agilent 92219J Cable



ca83a

Figure 1-5 HP/Agilent 13242G Cable

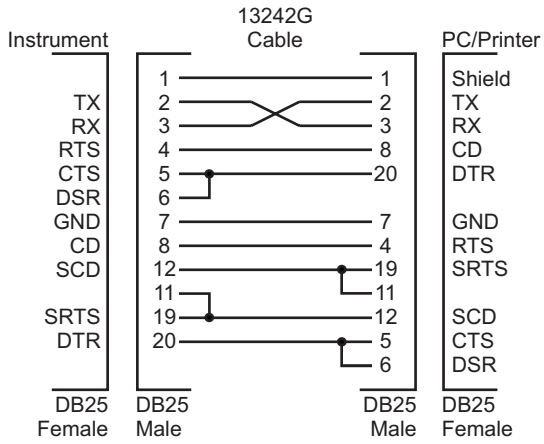


Figure 1-6 HP/Agilent 24542M Modem Cable

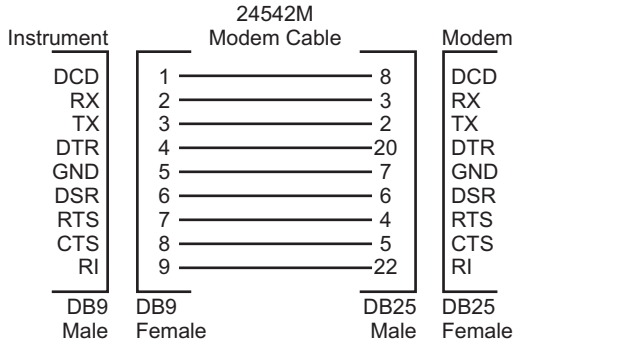


Figure 1-7 HP/Agilent C2913A/C2914A Cable

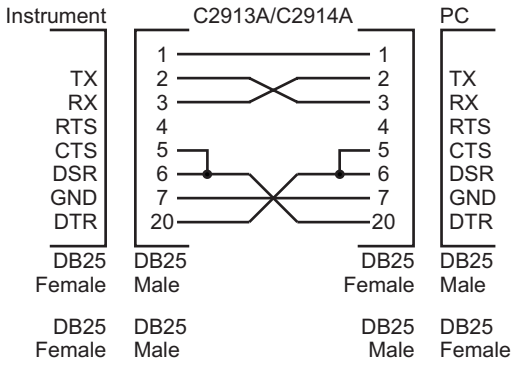


Figure 1-8 Mouse Adapter (typical)

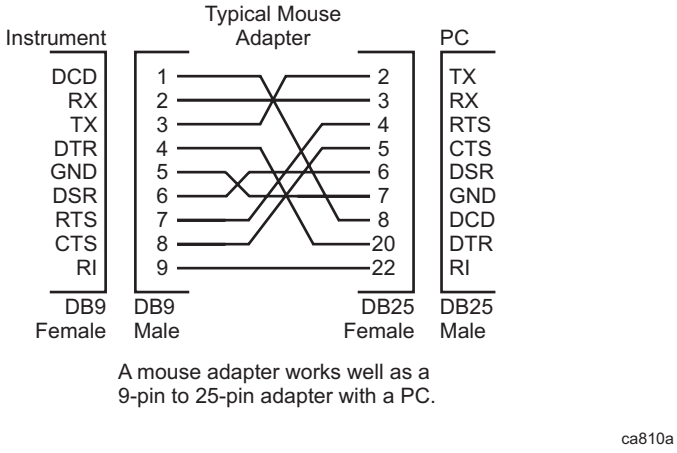


Figure 1-9 HP/Agilent 24542U Cable with 5181-6641 Adapter

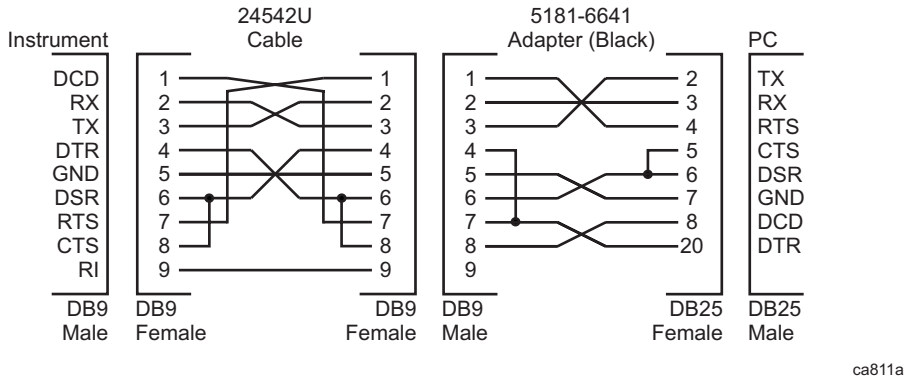


Figure 1-10 HP/Agilent 24542U Cable with 5181-6640 Adapter

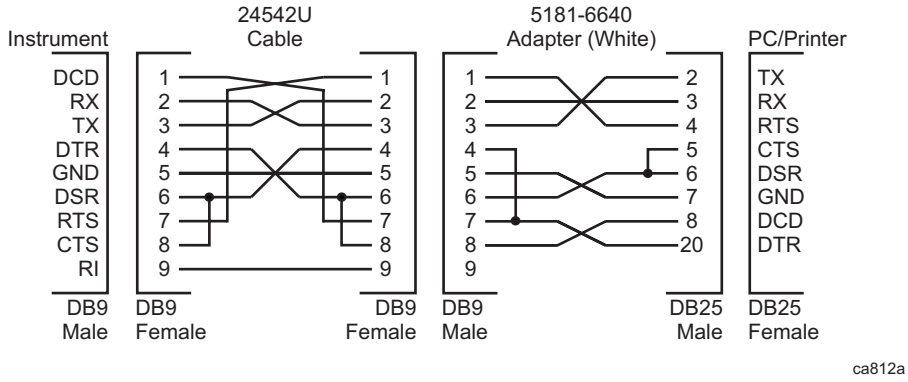


Figure 1-11 HP/Agilent 24542U Cable with 5181-6642 Adapter

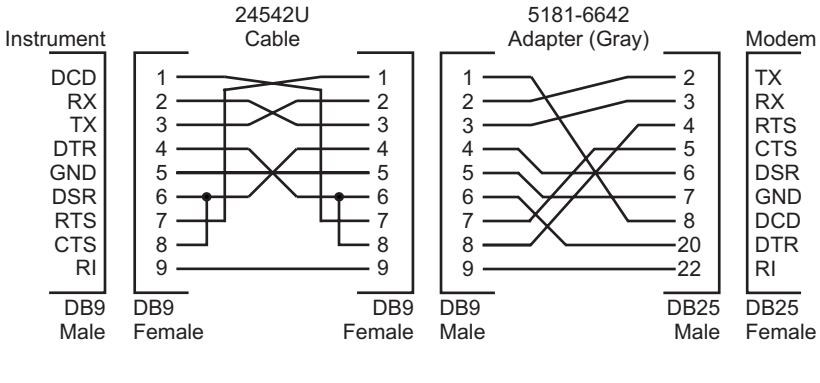


Figure 1-12 HP/Agilent 24542U Cable with 5181-6639 Adapter

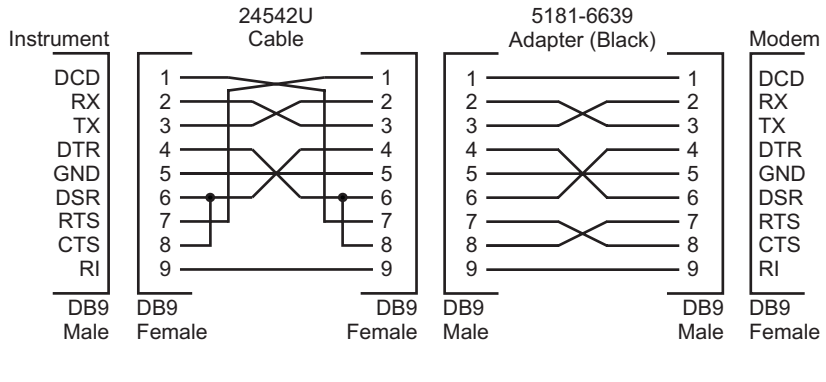


Figure 1-13 HP/Agilent F1047-80002 Cable with 5181-6641 Adapter

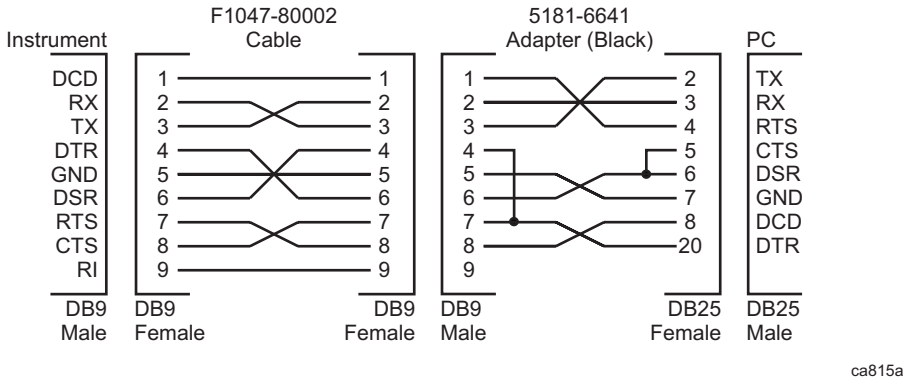


Figure 1-14 HP/Agilent F1047-80002 Cable with 5181-6640 Adapter

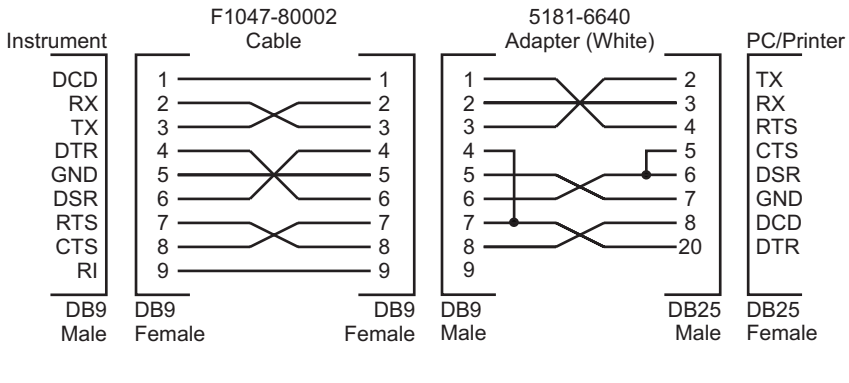


Figure 1-15 HP/Agilent F1047-80002 Cable with 5181-6642 Adapter

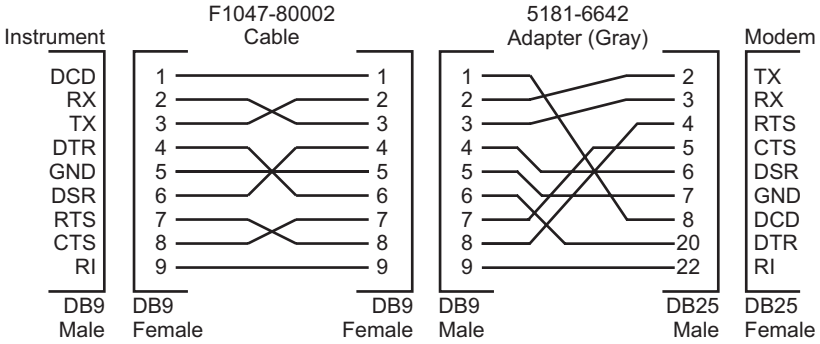
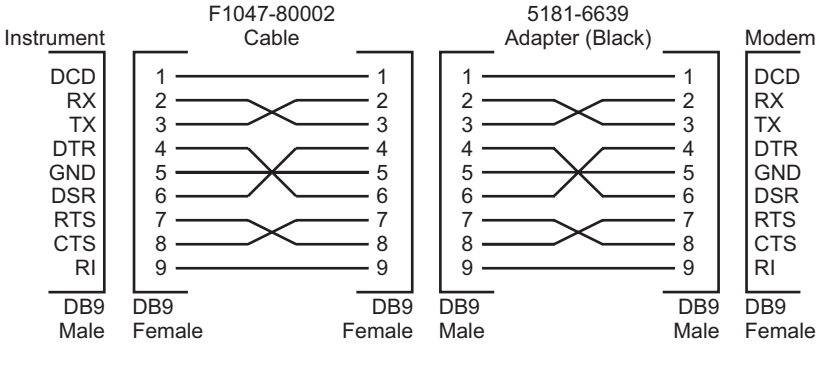


Figure 1-16 HP/Agilent F1047-80002 Cable with 5181-6639 Adapter



Connecting to a LAN Server

Connect a cable to the standard LAN connector on the rear panel of the instrument. The LAN can then be used several different ways:

- To ftp files from the instrument
- To use telnet to send SCPI commands
- To use sockets to send SCPI commands
- To use as a SICL server emulating IEEE 488.2 GPIB

Several LAN parameters can be queried from the front-panel key menus by pressing **System, Config I/O** and then pressing the appropriate keys. Configuration of some LAN parameters can only be done from the front panel. The IP address can be set or queried remotely using SYST:COMM:LAN:IP. The LAN default configuration settings do not usually have to be changed for you to use the functionality. More detailed LAN use and troubleshooting information can be found in Chapter 2 , “Programming Fundamentals” on page 63.

The different types of LAN functionality can be turned on and off from the front-panel keys under **System, Config I/O**. If you are running programs on the analyzer, you might want to turn off the other types of LAN access to make sure other users do not accidentally send commands to your analyzer in the middle of the program execution.

Pressing **Preset** will not change the LAN configuration settings. Since the settings are persistent they will stay at the last user-defined setting. However, you can return the instrument to its original factory defaults by pressing **System, Restore Sys Defaults**. If you want to use the LAN after restoring defaults, you will have to re-set the instrument IP address (and any other appropriate configuration settings) found in **System, Config I/O**.

Connecting to a GPIB Server

Connect a cable to the standard GPIB connector on the rear panel of the instrument. The GPIB can then be used to send SCPI commands to control the instrument and to return measurement data to the computer.

The GPIB address can be queried and set from the front-panel key menus by pressing **System, Config I/O, GPIB Address**. This can also be done remotely using SYST:COMM:GPIB:ADDR.

Pressing **Preset** will not change the GPIB address. It is persistent and will stay at the last user-defined setting. However, you can return the instrument to its original factory defaults by pressing **System, Restore Sys Defaults**. If you want to use a GPIB address other than 18 after restoring defaults, you will have to re-set the address.

2 Programming Fundamentals

Measure

For key and remote command information on each measurement, refer to the section which describes the measurement of interest.

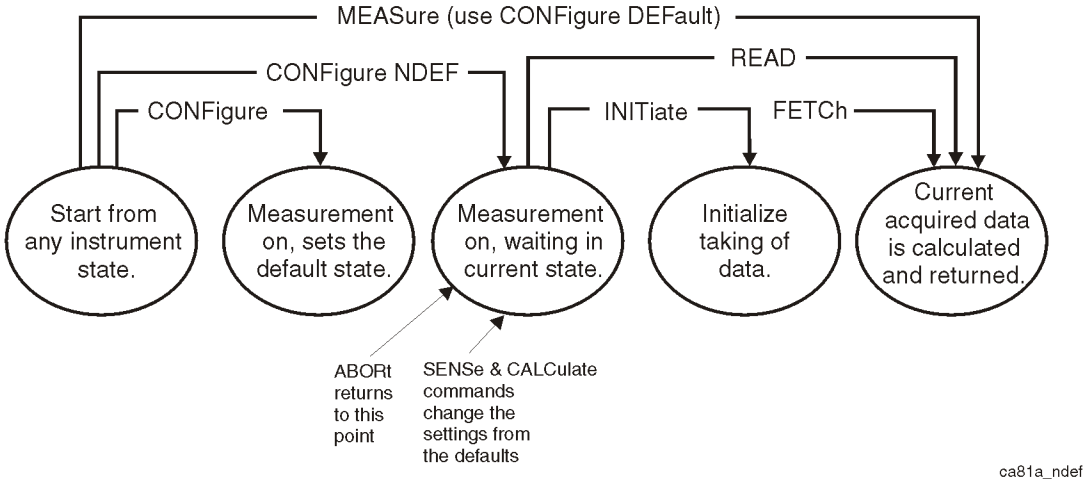
Measurements available under the **Measure** key are specific to the current Mode.

Key Path	Front-panel key
Help Map ID	4008

Command Interactions: MEASure, CONFigure, FETCh, INITiate and READ

Each one-button measurement has a group of commands that work together to make the measurement fast, but flexible.

Figure 2-1 Measurement Group of Commands



Measure Commands:

`:MEASure:<measurement>[n]?`

This is a fast single-command way to make a measurement using the factory default instrument settings. These are the settings and units that conform to the Mode Setup settings (e.g. radio standard) that you have currently selected.

- Stops the current measurement (if any) and sets up the instrument for the specified measurement using the factory defaults
- Initiates the data acquisition for the measurement
- Blocks other SCPI communication, waiting until the measurement is complete before returning results.
- After the data is valid it returns the scalar results, or the trace data, for the specified measurement. The type of data returned may be defined by an [n] value that is sent with the command.

  The scalar measurement results will be returned if the optional [n] value is not included, or is set to 1. If the [n] value is set to a value other than 1, the selected trace data results will be returned. See each command for details of what types of scalar results or trace data results are available.

  ASCII is the default format for the data output. The binary data formats should be used for handling large blocks of data since they are smaller and faster than the ASCII format. Refer to the FORMat:DATA command for more information.

If you need to change some of the measurement parameters from the factory default settings you can set up the measurement with the CONFigure command. Use the commands in the SENSe:<measurement> and CALCulate:<measurement> subsystems to change the settings. Then you can use the READ? command to initiate the measurement and query the results.

If you need to repeatedly make a given measurement with settings other than the factory defaults, you can use the commands in the SENSe:<measurement> and CALCulate:<measurement> subsystems to set up the measurement. Then use the READ? command to initiate the measurement and query results.

Measurement settings persist if you initiate a different measurement and then return to a previous one. Use READ:<measurement>? if you want to use those persistent settings. If you want to go back to the default settings, use MEASure:<measurement>?.

Configure Commands:

`:CONFigure:<measurement>`

This command stops the current measurement (if any) and sets up the instrument for the specified measurement using the factory default instrument settings. It sets the instrument to single measurement mode but should not initiate the taking of measurement data unless INIT:CONTinuous is ON. If you change any measurement settings after using the CONFigure command, the READ command can be used to initiate a measurement without changing the settings back to their defaults.

The CONFigure? query returns the current measurement name.

Fetch Commands:

`:FETCh:<measurement>[n]?`

This command puts selected data from the most recent measurement into the output buffer. Use FETCh if you have already made a good measurement and you want to return several types of data (different [n] values, e.g. both scalars and trace data) from a single measurement. FETCh saves you the time of re-making the measurement. You can only FETCh results from the measurement that is currently active, it will not change to a different measurement. An error is reported if a measurement other than the current one, is specified.

If you need to get new measurement data, use the READ command, which is equivalent to an INITiate followed by a FETCh.

The scalar measurement results will be returned if the optional [n] value is not included, or is set to 1. If the [n] value is set to a value other than 1, the selected trace data results will be returned. See each command for details of what types of scalar results or trace data results are available. The binary data formats should be used for handling large blocks of data since they are smaller and transfer faster then the ASCII format. (FORMat:DATA)

FETCh may be used to return results other than those specified with the original READ or MEASure command that you sent.

INITiate Commands:

`:INITiate:<measurement>`

This command is not available for measurements in all the instrument modes:

- Initiates a trigger cycle for the specified measurement, but does not output any data. You must then use the FETCh<meas> command to return data. If a measurement other than the current one is specified, the instrument will switch to that measurement and then initiate it.

  For example, suppose you have previously initiated the ACP measurement, but now you are running the channel power measurement. If you send INIT:ACP? it will change from channel power to ACP and will initiate an ACP measurement.

- Does not change any of the measurement settings. For example, if you have previously started the ACP measurement and you send INIT:ACP? it will initiate a new ACP measurement using the same instrument settings as the last time ACP was run.

- If your selected measurement is currently active (in the idle state) it triggers the measurement, assuming the trigger conditions are met. Then it completes one trigger cycle. Depending upon the measurement and the number of averages, there may be multiple data acquisitions, with multiple trigger events, for one full trigger cycle. It also holds off additional commands on GPIB until the acquisition is complete.

READ Commands:

`:READ:<measurement>[n]?`

- Does not preset the measurement to the factory default settings. For example, if you have previously initiated the ACP measurement and you send READ:ACP? it will initiate a new measurement using the same instrument settings.
- Initiates the measurement and puts valid data into the output buffer. If a measurement other than the current one is specified, the instrument will switch to that measurement before it initiates the measurement and returns results.

  For example, suppose you have previously initiated the ACP measurement, but now you are running the channel power measurement. Then you send READ:ACP? It will change from channel power back to ACP and, using the previous ACP settings, will initiate the measurement and return results.
- Blocks other SCPI communication, waiting until the measurement is complete before returning the results

  If the optional [n] value is not included, or is set to 1, the scalar measurement results will be returned. If the [n] value is set to a value other than 1, the selected trace data results will be returned. See each command for details of what types of scalar results or trace data results are available. The binary data formats should be used when handling large blocks of data since they are smaller and faster then the ASCII format. (FORMat:DATA)

Current Measurement Query (Remote Command Only)

This command returns the name of the measurement that is currently running.

Mode	All
Remote Command	`:CONFigure?`
Example	`CONF`

Test current results against all limits (Remote Command Only)

Queries the status of the current measurement limit testing. It returns a 0 if the measured results pass when compared with the current limits. It returns a 1 if the measured results fail any limit tests.

Mode	All
Remote Command	`:CALCulate:CLIMits:FAIL?`
Range	0 | 1
Help Map ID	0

Data Query (Remote Command Only)

Returns the designated measurement data for the currently selected measurement and subopcode.

n = any valid subopcode for the current measurement. See the measurement command results table in each measurement section for information about what data is returned for the subopcodes.

Mode	All
Remote Command	`:CALCulate:DATA[n]? <real>,…`
Notes	The return trace depends on the measurement. In CALCulate:DATA[n], n is any valid subopcode for the current measurement.
Help Map ID	0

Calculate/Compress Trace Data Query (Remote Command Only)

`:CALCulate:DATA<n>:COMPress? BLOCk|CFIT|MAXimum|MINimum|MEAN|DMEan|RMS|SAMPle|SDEViation |PPHase [,<soffset>[,<length>[,<roffset>[,<rlimit>]]]]`

Returns compressed data for the specified trace data. The data is returned in the same units as the original trace and only works with the currently selected measurement. The command is used with a subopcode <*n*> since measurements usually return several types of trace data. See the following table for the subopcodes for the trace data names that are available in each measurement. For subopcodes that return scalar data use the `:CALCulate:DATA[n]?` command above.

NOTE This description of CALC:DATA:COMP? operation applies to all measurements except Swept SA measurement. See the description in the **Trace/Detector** section for use in Swept SA.

This command is used to compress or decimate a long trace to extract and return only the desired data. A typical example would be to acquire N frames of GSM data and return the mean power of the first burst in each frame. The command can also be used to identify the best curve fit for the data.

- BLOCk or block data - returns all the data points from the region of the trace data that you specify. For example, it could be used to return the data points of an input signal over several timeslots, excluding the portions of the trace data that you do not want.
- CFIT or curve fit - applies curve fitting routines to the data. <soffset> and <length> are required to define the data that you want. <roffset> is an optional parameter for the desired order of the

curve equation. The query will return the following values: the x-offset (in seconds) and the curve coefficients ((order + 1) values).

MAX, MEAN, MIN, RMS, SAMP, SDEV and PPH return one data value for each specified region (or <length>) of trace data, for as many regions as possible until you run out of trace data (using <roffset> to specify regions). Or they return the number regions you specify (using <rlimit>) ignoring any data beyond that.

- MAXimum - returns the maximum data point for the specified region(s) of trace data. For I/Q trace data, the maximum magnitude of the I/Q pairs is returned.
- MEAN - returns the arithmetic mean of the data point values for the specified region(s) of trace data. See "Mean Value of I/Q Data Points for Specified Region(s)" on page 70. For I/Q trace data, the mean of the magnitudes of the I/Q pairs is returned. See "Mean Value of I/Q Data Pairs for Specified Region(s)" on page 70.

  Note: If the original trace data is in dB, this function returns the arithmetic mean of those log values, not log of the mean power, which is a more useful value.

Equation 2-1 Mean Value of I/Q Data Points for Specified Region(s)

$$\text{MEAN} = \frac{1}{n} \sum_{Xi \in \text{region(s)}} Xi$$

where Xi is a data point value, and n is the number of data points in the specified region(s).

Equation 2-2 Mean Value of I/Q Data Pairs for Specified Region(s)

$$\text{MEAN} = \frac{1}{n} \sum_{Xi \in \text{region(s)}} |Xi|$$

where |Xi| is the magnitude of an I/Q pair, and n is the number of I/Q pairs in the specified region(s).

- MINimum - returns the minimum data point for the specified region(s) of trace data For I/Q trace data, the minimum magnitude of the I/Q pairs is returned.
- RMS - returns the arithmetic rms of the data point values for the specified region(s) of trace data. See "RMS Value of Data Points for

Specified Region(s)" on page 71.

For I/Q trace data, the rms of the magnitudes of the I/Q pairs is returned. See "RMS Value of I/Q Data Pairs for Specified Region(s)" on page 71.

Note: This function is very useful for I/Q trace data. However, if the original trace data is in dB, this function returns the rms of the log values which is not usually needed.

Equation 2-3 RMS Value of Data Points for Specified Region(s)

$$RMS = \sqrt{\frac{1}{n} \sum_{Xi \in \text{region(s)}} Xi^2}$$

where Xi is a data point value, and n is the number of data points in the specified region(s).

Equation 2-4 RMS Value of I/Q Data Pairs for Specified Region(s)

$$RMS = \sqrt{\frac{1}{n} \sum_{Xi \in \text{region(s)}} Xi \, Xi^*}$$

where Xi is the complex value representation of an I/Q pair, Xi its conjugate complex number, and n is the number of I/Q pairs in the specified region(s).*

Once you have the rms value for a region of I/Q trace data, you may want to calculate the mean power. You must convert this rms I/Q value (peak volts) to power in dB.

$$10 \times \log[10 \times (\text{rms value})^2]$$

- SAMPle - returns the first data value for the specified region(s) of trace data. For I/Q trace data, the first I/Q pair is returned.
- SDEViation - returns the arithmetic standard deviation for the data point values for the specified region(s) of trace data. See "Standard Deviation of Data Point Values for Specified Region(s)" on page 71.

  For I/Q trace data, the standard deviation of the magnitudes of the I/Q pairs is returned. See "Standard Deviation of I/Q Data Pair Values for Specified Region(s)" on page 72.

Equation 2-5 Standard Deviation of Data Point Values for Specified Region(s)

$$SDEV = \sqrt{\frac{1}{n} \sum_{Xi \in region(s)} (Xi - \overline{X})^2}$$

where Xi is a data point value, X is the arithmetic mean of the data point values for the specified region(s), and n is the number of data points in the specified region(s).

Equation 2-6 Standard Deviation of I/Q Data Pair Values for Specified Region(s)

$$SDEV = \sqrt{\frac{1}{n} \sum_{Xi \in region(s)} (|Xi| - \overline{X})^2}$$

where |Xi| is the magnitude of an I/Q pair, X is the mean of the magnitudes for the specified region(s), and n is the number of data points in the specified region(s).

- PPH - returns the pairs of rms power (dBm) and arithmetic mean phase (radian) for every specified region and frequency offset (Hz). The number of pairs is defined by the specified number of regions. Assuming this command can be used for I/Q vector (n=0) in Waveform (time domain) measurement and all parameters are specified by data point in PPH.

  The rms power of the specified region may be expressed as:

  Power = 10 x log [10 x (RMS I/Q value)] + 10

  The RMS I/Q value (peak volts) =

  $$\sqrt{\frac{1}{n} \sum_{Xi \in region} XiXi^*}$$

  where Xi is the complex value representation of an I/Q pair, Xi* its conjugate complex number, and n is the number of I/Q pairs in the specified region.

  The arithmetic mean phase of the specified region may be expressed as:

  Phase = $\frac{1}{n} \sum_{Yi \in region} Yi$

  Where Yi is the unwrapped phase of I/Q pair with applying frequency correction and n is the number of I/Q pairs in the specified region.
  The frequency correction is made by the frequency offset calculated

by the arithmetic mean of every specified region's frequency offset. Each frequency offset is calculated by the least square method against the unwrapped phase of I/Q pair.

Figure 2-2 Sample Trace Data - Constant Envelope

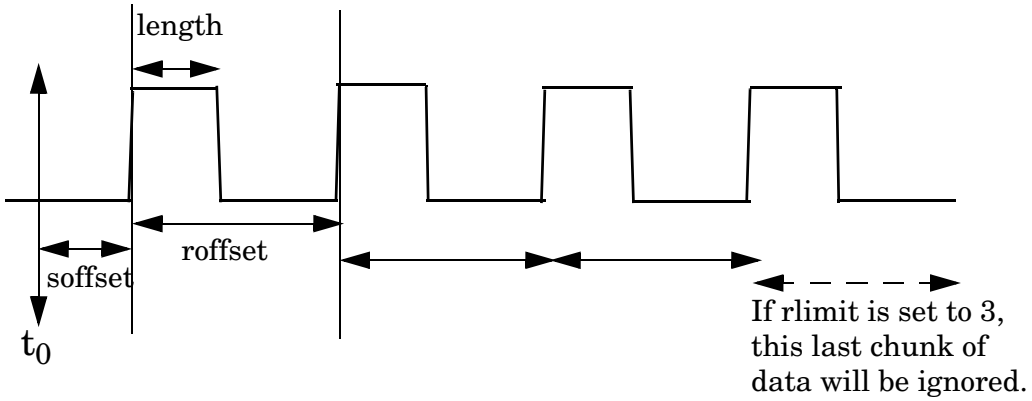
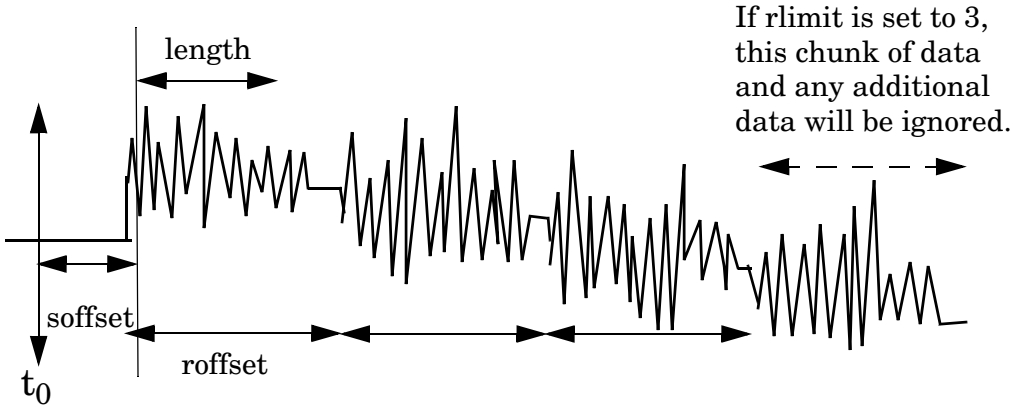


Figure 2-3 Sample Trace Data - Not Constant Envelope



<soffset> - start offset is an optional real number (in seconds). It specifies the amount of data at the beginning of the trace that will be ignored before the decimation process starts. It is the time from the start of the trace to the point where you want to start using the data. The default value is zero.

<length> - is an optional real number (in seconds). It defines how much data will be compressed into one value. This parameter has a default value equal to the current trace length.

<roffset> - repeat offset is an optional real number (in seconds). It defines the beginning of the next field of trace elements to be compressed. This is relative to the beginning of the previous field. This parameter has a default value equal to the <length> variable.

<rlimit> - repeat limit is an optional integer. It specifies the number of data items that you want returned. It will ignore any additional

items beyond that number. You can use the Start offset and the Repeat limit to pick out exactly what part of the data you want to use. The default value is all the data.

Example: To query the mean power of a set of GSM bursts:

1. Set the waveform measurement sweep time to acquire at least one burst.
2. Set the triggers such that acquisition happens at a known position relative to a burst.
3. Then query the mean burst levels using, `CALC:DATA2:COMP? MEAN,24e-6,526e-6` (These parameter values correspond to GSM signals, where 526e-6 is the length of the burst in the slot and you just want 1 burst.)

Remarks: The optional parameters must be entered in the specified order. For example, if you want to specify <length>, you must also specify <soffset>.

This command uses the data in the format specified by FORMat:DATA, returning either binary or ASCII data.

Measurement	**Available Traces**	**Markers Available?**
ACP - adjacent channel power (Basic, cdmaOne, cdma2000, W-CDMA, NADC, PDC modes)	no traces (n=0)[a] for I/Q points	no markers
CDPower - code domain power (cdmaOne mode)	POWer (n=2)[a] TIMing (n=3)[a] PHASe (n=4)[a] (n=0)[a] for I/Q points	yes
CDPower - code domain power (cdma2000, W-CDMA modes)	CDPower (n=2)[a] EVM (n=5)[a] MERRor (n=6)[a] PERRor (n=7)[a] SPOWer (n=9)[a] CPOWer (n=10)[a] (n=0)[a] for I/Q points	yes
CHPower - channel power (Basic, cdmaOne, cdma2000, W-CDMA modes)	SPECtrum (n=2)[a] (n=0)[a] for I/Q points	no markers

Measurement	Available Traces	Markers Available?
CSPur - spurs close (cdmaOne mode)	SPECtrum (n=2)[a] ULIMit (n=3)[a] (n=0)[a] for I/Q points	yes
EEVM - EDGE error vector magnitude (EDGE mode)	EVMerror (n=2)[a] MERRor (n=3)[a] PERRor (n=4)[a] (n=0)[a] for I/Q points	yes
EORFspectr - EDGE output RF spectrum (EDGE mode)	RFEMod (n=2)[a] RFESwitching (n=3)[a] SPEMod (n=4)[a] LIMMod (n=5)[a] (n=0)[a] for I/Q points	yes, only for a single offset yes, only for multiple offsets
EPVTime - EDGE power versus time (EDGE mode)	RFENvelope (n=2)[a] UMASk (n=3)[a] LMASk (n=4)[a] (n=0)[a] for I/Q points	yes
ETSPur - EDGE transmit band spurs (EDGE mode)	SPECtrum (n=2)[a] ULIMit (n=3)[a] (n=0)[a] for I/Q points	yes
EVM - error vector magnitude (NADC, PDC modes)	EVM (n=2)[a] MERRor (n=3)[a] PERRor (n=4)[a] (n=0)[a] for I/Q points	yes
EVMQpsk - QPSK error vector magnitude (cdma2000, W-CDMA modes)	EVM (n=2)[a] MERRor (n=3)[a] PERRor (n=4)[a] (n=0)[a] for I/Q points	yes

Measurement	Available Traces	Markers Available?
IM - intermodulation (cdma2000, W-CDMA modes)	SPECtrum (n=2)[a] (n=0)[a] for I/Q points	yes
MCPower - multi-carrier power (W-CDMA mode)	no traces (n=0)[a] for I/Q points	no markers
OBW - occupied bandwidth (cdmaOne, cdma2000, PDC, W-CDMA modes)	no traces (n=0)[a] for I/Q points	no markers
ORFSpectrum - output RF spectrum (GSM, EDGE mode)	RFEMod (n=2)[a] RFESwitching (n=3)[a] SPEMod (n=4)[a] LIMMod (n=5)[a] (n=0)[a] for I/Q points	yes, only for a single offset yes, only for multiple offsets
PFERror - phase and frequency error (GSM, EDGE mode)	PERRor (n=2)[a] PFERror (n=3)[a] RFENvelope (n=4)[a] (n=0)[a] for I/Q points	yes
PSTatistic - power statistics CCDF (Basic, cdma2000, W-CDMA modes)	MEASured (n=2)[a] GAUSian (n=3)[a] REFerence (n=4)[a] (n=0)[a] for I/Q points	yes
PVTime - power versus time (GSM, EDGE modes)	RFENvelope (n=2)[a] UMASk (n=3)[a] LMASk (n=4)[a] (n=0)[a] for I/Q points	yes

Measurement	Available Traces	Markers Available?
RHO - modulation quality (cdmaOne, cdma2000, W-CDMA mode)	(n=0)[a] for I/Q points EVM (n=2)[a] MERRor (n=3)[a] PERRor (n=4)[a] (n=0)[a] for I/Q points	yes
SEMask - spectrum emissions mask (cdma2000, W-CDMA mode)	SPECtrum (n=2)[a] (n=0)[a] for I/Q points	yes
TSPur - transmit band spurs (GSM, EDGE mode)	SPECtrum (n=2)[a] ULIMit (n=3)[a] (n=0)[a] for I/Q points	yes
TXPower - transmit power (GSM, EDGE mode)	RFENvelope (n=2)[a] IQ (n=8)[a] (n=0)[a] for I/Q points	yes
SPECtrum - (frequency domain) (all modes)	IQ (n=3)[a] SPECtrum (n=4)[a] ASPectrum (n=7)[a] (n=0)[a] for I/Q points	yes
WAVEform - (time domain) (all modes)	RFENvelope (n=2)[a] (also for Signal Envelope trace) IQ (n=5)[a] (n=0)[a] for I/Q points	yes

a. The *n* number indicates the subopcode that corresponds to this trace. Detailed descriptions of the trace data can be found in the MEASure subsystem documentation by looking up the subopcode for the appropriate measurement.

Calculate peaks of trace data (Remote Command Only)

Returns a list of peaks for the designated trace data n for the currently selected measurement. The peaks must meet the requirements of the peak threshold and excursion values. The command can only be used

with specific [n] (subopcode) values, for measurement results that are trace, or scalar, data. See the remote command section of each measurement for the appropriate subopcodes. Both real and complex traces can be searched, but complex traces are converted to magnitude in dBm. subopcode n=0, is the raw trace data which cannot be searched for peaks. subopcode n=1, is the scaler data which also cannot be searched for peaks.

Mode	All
Remote Command	`:CALCulate:DATA[n]:PEAKs?`
	`<threshold>,<excursion>[,AMPLitude|FREQuency|TIME]`
Notes	The return trace depends on the measurement.
Help Map ID	0

SCPI Language Basics

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

Topics covered in this chapter include:

For more information refer to:

IEEE Standard 488.1-1987, *IEEE Standard Digital Interface for Programmable Instrumentation*. New York, NY, 1998.

IEEE Standard 488.2-1987, *IEEE Standard Codes, Formats, Protocols and Comment Commands for Use with ANSI/IEEE Std488.1-1987*. New York, NY, 1998.

Command Keywords and Syntax

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

Example: `SENSe:FREQuency:STARt 1.5 MHZ`

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

Example: Sens:Freq:Star 1.5 mhz

is the same as `SENSE:FREQ`:start 1.5 MHz

NOTE The command `SENS:FREQU:STAR` is not valid because `FREQU` is neither the short, nor the long form of the command. Only the short and long forms of the keywords are allowed in valid commands.

Creating Valid Commands

Commands are not case sensitive and there are often many different

ways of writing a particular command. These are examples of valid commands for a given command syntax:

Command Syntax	Sample Valid Commands
`[SENSe:]BANDwidth[:RESolution] <freq>`	The following sample commands are all identical. They will all cause the same result. • `Sense:Band:Res 1700` • `BANDWIDTH:RESOLUTION 1.7e3` • `sens:band 1.7KHZ` • `SENS:band 1.7E3Hz` • `band 1.7kHz` • `bandwidth:RES 1.7e3Hz`
`MEASure:SPECtrum[n]?`	• `MEAS:SPEC?` • `Meas:spec?` • `meas:spec3?` The number 3 in the last meas example causes it to return different results then the commands above it. See the command description for more information.
`[:SENSe]:DETector[:FUNCtion] NEGative|POSitive|SAMPle`	• `DET:FUNC neg` • `Detector:Func Pos`
`INITiate:CONTinuous ON|OFF|1|0`	The sample commands below are identical. • `INIT:CONT ON` • `init:continuous 1`

Special Characters in Commands

Special Character	Meaning	Example
|	A vertical stroke between **parameters** indicates alternative choices. The effect of the command is different depending on which parameter is selected.	Command: `TRIGger:SOURce EXTernal|INTernal|LINE` The choices are external, internal, and line. Ex: `TRIG:SOURCE INT` is one possible command choice.

Special Character	Meaning	Example
	A vertical stroke between **keywords** indicates identical effects exist for both keywords. The command functions the same for either keyword. Only one of these keywords is used at a time.	Command: `SENSe:BANDwidth|BWIDth:OFFSet` Two identical commands are: Ex1: `SENSE:BWIDTH:OFFSET` Ex2: `SENSE:BAND:OFFSET`
[ ]	keywords in square brackets are optional when composing the command. These implied keywords will be executed even if they are omitted.	Command: `[SENSe:]BANDwidth[:RESolution]:AUTO` The following commands are all valid and have identical effects: Ex1: `bandwidth:auto` Ex2: `band:resolution:auto` Ex3: `sense:bandwidth:auto`
< >	Angle brackets around a word, or words, indicates they are not to be used literally in the command. They represent the needed item.	Command: `SENS:FREQ <freq>` In this command example the word <freq> should be replaced by an actual frequency. Ex: `SENS:FREQ 9.7MHz.`

{ }	Parameters in braces can optionally be used in the command either not at all, once, or several times.	Command:
		`MEASure:BW <freq>{,level}`
		A valid command is:
		`meas:BW 6 MHz, 3dB, 60dB`

Parameters in Commands

There are four basic types of parameters: booleans, keywords, variables and arbitrary block program data.

OFF | ON | 0 | 1
(Boolean) This is a two state boolean-type parameter. The numeric value 0 is equivalent to OFF. Any numeric value other than 0 is equivalent to ON. The numeric values of 0 or 1 are commonly used in the command instead of OFF or ON. Queries of the parameter always return a numeric value of 0 or 1.

keyword The keywords that are allowed for a particular command are defined in the command syntax description.

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

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

The following keywords may also be used in commands, but not all commands allow keyword variables.

- DEFault - resets the parameter to its default value.
- UP - increments the parameter.
- DOWN - decrements the parameter.
- MINimum - sets the parameter to the smallest possible value.
- MAXimum - sets the parameter to the largest possible value.

The numeric value for the function's MINimum, MAXimum, or DEFault can be queried by adding the keyword to the command in its query form. The keyword must be entered following the question mark.

Example query: `SENSE:FREQ:CENTER? MAX`

Variable Parameters

<freq>
<bandwidth> Is a positive rational number followed by optional units. The default unit is Hz. Acceptable units include: HZ,

KHZ, MHZ, GHZ.

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

<voltage> Is a rational number followed by optional units. The default units are V. Acceptable units include: Volts, V, MV, UV.

<power>
<ampl> Is a rational number followed by optional units. The default units are dBm. Acceptable units include: DBM, DBMV, W.

<rel_power>
<rel_ampl> Is a positive rational number followed by optional units. The default units are dB. Acceptable units include: DB.

<angle>
<degrees> Is a rational number followed by optional units. The default units are degrees. Acceptable units include: DEG, RAD.

<integer> An integer value has no units.

<real> Is a floating point number, with no units.

<percent> Is a rational number between 0 and 100, with no units.

<string> Is a series of alpha numeric characters.

<bit_pattern> Specifies a series of bits rather than a numeric value. The bit series is the binary representation of a numeric value. There are no units.

Bit patterns are most often specified as hexadecimal numbers, though octal, binary or decimal numbers may also be used. In the SCPI language these numbers are specified as:

- Hexadecimal, #Hdddd or #hdddd where ‘d’ represents a hexadecimal digit 0 to 9 and ‘a’ to ‘f’. So #h14 can be used instead of the decimal number 20.
- Octal, #Oddddddd or #oddddddd where ‘d’ represents an octal digit 0 to 7. So #o24 can be used instead of the decimal number 20.
- Binary, #Bddddddddddddddddd or #bddddddddddddddddd where ‘d’ represents a 1 or 0. So #b10100 can be used instead of the decimal number 20.

Block Program Data

Some parameters consist of a block of data. There are a few standard types of block data. Arbitrary blocks of program data can also be used.

<trace> Is an array of rational numbers corresponding to displayed trace data. The default uses “display units” with 600 trace points with amplitudes from 0 to 1024. See FORMat:DATA for information about available data formats.

A SCPI command often refers to a block of current trace data with a variable name such as: Trace1, TRACE2, or trace3, depending on which trace is being accessed.

<arbitrary block data> Consists of a block of data bytes. The first information sent in the block is an ASCII header beginning with #. The block is terminated with a semi-colon. The header can be used to determine how many bytes are in the data block. There are no units. (You will not get block data if your data type is ASCII, using `FORMat:DATA ASCII` command. Your data will be comma separated ASCII values.

Block data example: suppose the header is #512320.

- The first digit in the header (5) tells you how many additional digits/bytes there are in the header.
- The 12320 means 12 thousand, 3 hundred, 20 data bytes follow the header.
- Divide this number of bytes by your current data format (bytes/data point), either 8 (for real,64), or 4 (for real,32). For this example, if you’re using real64 then there are 1540 points in the block.

Putting Multiple Commands on the Same Line

Multiple commands can be written on the same line, reducing your code space requirement. To do this:

- Commands must be separated with a semicolon (;).
- If the commands are in different subsystems, the key word for the new subsystem must be preceded by a colon (:).
- If the commands are in the same subsystem, the full hierarchy of the command key words need not be included. The second command can start at the same key word level as the command that was just executed.

SCPI Termination and Separator Syntax

A terminator must be provided when an instrument is controlled using RS-232. There are several issues to be understood about choosing the proper SCPI terminator and separator when this is the case. There is no current SCPI standard for RS-232. Although one intent of SCPI is to be interface independent, <END> is only defined for IEEE 488 operation. At the time of this writing, the RS-232 terminator issue was in the process of being addressed in IEEE standard 1174.

A semicolon (;) is not a SCPI terminator, it is a separator. The purpose of the separator is to queue multiple commands or queries in order to obtain multiple actions and/or responses. Make sure that you do not attempt to use the semicolon as a terminator when using RS-232 control.

All binary trace and response data is terminated with <NL><END>, as defined in Section 8.5 of IEEE Standard 488.2-1992, *IEEE Standard Codes, Formats, Protocols and Common Commands for Use with ANSI/IEEE Std 488.1-1987*. New York, NY, 1992.

The following are some examples of good and bad commands. The examples are created from a theoretical instrument with the simple set of commands indicated below:

```
[:SENSe]
     :POWer
          [:RF]
          :ATTenuation 40dB

:TRIGger
     [:SEQuence]
     :EXTernal [1]
          :SLOPe
               POSitive

[:SENSe]
     :FREQuency
          :STARt
     :POWer
     [:RF]
          :MIXer
               :RANGe
              [:UPPer]
```

Bad Command	Good Command
`PWR:ATT 40dB`	`POW:ATT 40dB`
The short form of **POWER** is **POW**, not **PWR**.	

Bad Command	Good Command
`FREQ:STAR 30MHz;MIX:RANG -20dBm`	`FREQ:STAR 30MHz;POW:MIX:RANG -20dBm`
The **MIX:RANG** command is in the same **:SENSE** subsystem as **FREQ**, but executing the **FREQ** command puts you back at the **SENSE** level. You must specify **POW** to get to the **MIX:RANG** command.	
`FREQ:STAR 30MHz;POW:MIX RANG -20dBm`	`FREQ:STAR 30MHz;POW:MIX:RANG -20dBm`
MIX and **RANG** require a colon to separate them.	
`:POW:ATT 40dB;TRIG:FREQ:STAR 2.3GHz`	`:POW:ATT 40dB;:FREQ:STAR 2.3GHz`
:FREQ:STAR is in the **:SENSE** subsystem, not the **:TRIGGER** subsystem.	
`:POW:ATT?:FREQ:STAR?`	`:POW:ATT?;:FREQ:STAR?`
:POW and **FREQ** are within the same **:SENSE** subsystem, but they are two separate commands, so they should be separated with a semicolon, not a colon.	
`:POW:ATT -5dB;:FREQ:STAR 10MHz`	`:POW:ATT 5dB;:FREQ:STAR 10MHz`
Attenuation cannot be a negative value.	

Improving Measurement Speed

There are a number of things you can do in your programs to make them run faster:

Turn off the display updates

:DISPlay:ENABle OFF turns off the display. That is, the data may still be visible, but it will no longer be updated. Updating the display slows down the measurement. For remote testing, since the computer is processing the data rather than a person, there is no need to display the data on the analyzer screen.

Use binary data format instead of ASCII

The ASCII data format is the instrument default since it is easier for people to understand and is required by SCPI for *RST. However, data input/output is faster using the binary formats.

:FORMat:DATA REAL,64 selects the 64-bit binary data format for all your numerical data queries. You may need to swap the byte order if you are using a PC rather than UNIX. **NORMal** is the default byte order. Use **:FORMat:BORDer SWAP** to change the byte order so that the least significant byte is sent first. (Real,32 which is smaller and somewhat faster, should only be used if you don't need full resolution for your data. Some frequency data may require full 64 bit resolution.)

When using the binary format, data is sent in a block of bytes with an ASCII header. A data query would return the block of data in the following format: `#DNNN<nnn binary data bytes>`

To parse the data:

- Read two characters (#D), where D tells you how many N characters follow the D character.
- Read D characters, the resulting integer specifies the number of data bytes sent.
- Read the bytes into a real array.

For example, suppose the header is #512320.

- The first character/digit in the header (5) tells you how many additional digits there are in the header.
- The 12320 means 12 thousand, 3 hundred, 20 data bytes follow the header.
- Divide this number of bytes by your current data format (bytes/data point), 8 for real,64. For this example, there are 1540 data points in the block of data.

Avoid unnecessary use of *RST

Remember that while *RST does not change the current Mode, it presets all the measurements and settings to their factory defaults. This forces you to reset your analyzer's measurement settings even if they use similar mode setup or measurement settings. See Minimize DUT/instrument setup changes below. (Also note that *RST may put the instrument in single measurement/sweep for some modes.)

Minimize DUT/instrument setup changes

- Some instrument setup parameters are common to multiple measurements. You should look at your measurement process with an eye toward minimizing setup changes. If your test process involves nested loops, make sure that the inner-most loop is the fastest. Also, check if the loops could be nested in a different order to reduce the number of parameter changes as you step through the test.
- Remember that if you have already set your Meas Setup parameters for a measurement, and you want to make another one of these measurements later, use READ:<meas>?. The MEASure:<meas>?. command resets all the settings to the defaults, while READ changes back to that measurement without changing the setup parameters from the previous use.
- Are you are using the Measurements under the **MEASURE** key? Remember that *Mode Setup* parameters remain constant across all the measurements in that mode (e.g. center/channel frequency, amplitude, radio standard, input selection, trigger setup). You don't have to re-initialize them each time you change to a different measurement.

Consider using USB (Option 111) or LAN instead of GPIB

USB and LAN allow faster I/O of data, especially if you are moving large blocks of data. You will not get this improved throughput using LAN if there is excessive LAN traffic (i.e. your test instrument is connected to enterprise LAN). You may want to use a private LAN that is only for your test system.

Minimize the number of GPIB transactions

When you are using the GPIB for control of your instrument, each transaction requires driver overhead and bus handshaking, so minimizing these transactions reduces the time used.

- You can reduce bus transactions by sending multiple commands per transaction. See the information on “Putting Multiple Commands on the Same Line” in the SCPI Language Basics section.
- If you are making the same measurement multiple times with small changes in the measurement setup, use the READ command. It is faster then using INITiate and FETCh.
- If you are changing the frequency and making a measurement repeatedly, you can reduce transactions by sending the optional frequency parameter with your READ command.
  (for example, READ:<meas>? {<freq>})

  The CONFigure/MEASure/READ commands for measurements in the option Modes allow you to send center frequency setup information along with the command. (for example, **`MEAS:PVT? 935.2MHz`**) This sets the power vs. time measurement to it’s defaults, then changes the center frequency to 935.2 MHz, initiates a measurement, waits until it is complete and returns the measurement data.
- If you are doing bottom/middle/top measurements on base stations, you can reduce transactions by making a time slot active at each of the B,M,T frequencies. Then issue three measurements at once in the programming code and retrieve three data sets with just one GPIB transaction pair (write, read).

  For example, send READ:PFER? <Freq_bottom>;PFER? <Freq_middle>;PFER? <Freq_top> This single transaction initiates three different phase and frequency error measurements at each of the three different frequencies provided and returns the data. Then you read the three sets of data.

Avoid automatic attenuator setting

The internal process for automatically setting the value of the attenuator requires measuring an initial burst to identify the proper

attenuator setting before the next burst can be measured properly. If you know the amount of attenuation or the signal level needed for your measurement, just set it.

Note that spurious types of measurements must be done with the attenuator set to automatic (for measurements like: output RF spectrum, transmit spurs, adjacent channel power, spectrum emission mask). These types of measurements start by tuning to the signal, then they tune away from it and must be able to reset the attenuation value as needed.

Optimize your GSM output RF spectrum switching measurement

For ORFS (switching), setting the break frequency to zero (0) puts the analyzer in a measurement setup where it can use a direct time measurement algorithm, instead of an FFT-based algorithm. This non-FFT approach is faster. (However, remember that your break frequency for ORFS (modulation) measurements must be >400 kHz for valid measurements, so you will need to change the break frequency if you are making both types of measurements.)

Avoid using RFBurst trigger for single burst signals

RFBurst triggering works best when measuring signals with repetitive bursts. For a non-repetitive or single burst signals, use the IF (video) trigger or external trigger, depending on what you have available.

RFBurst triggering depends on its establishment of a valid triggering reference level, based on previous bursts. If you only have a single burst, the peak detection nature of this triggering function, may result in the trigger being done at the wrong level/point generating incorrect data, or it may not trigger at all.

Are you making a single burst measurement?

To get consistent triggering and good data for this type of measurement application, you need to synchronize the triggering of the DUT with the analyzer. You should use the analyzer's internal status system for this.

The first step in this process is to initialize the status register mask to look for the "waiting for trigger" condition (bit 5). Use
`:STATus:OPERation:ENABle 32`

Then, in the measurement loop:

1. `:STATus:OPERation:EVENt?` This query of the operation event register is to clear the current register contents.
2. `:READ:PVT?` initiates a measurement (in this example, for GSM power versus time) using the previous setup. The measurement will then be waiting for the trigger.

Make sure the attenuation is set manually. Do NOT use automatic attenuation as this requires an additional burst to determine the proper attenuation level before the measurement can be made.

3. Create a small loop that will serial poll the instrument for a status byte value of binary 128. Then wait 1 msec (100 ms if the display is left on/enabled) before checking again, to keep the bus traffic down. These two commands are repeated until the condition is set, so we know that the trigger is armed and ready.
4. Trigger your DUT to send the burst.
5. Return the measurement data to your computer.

NOTE This process cannot be done by using with the current VXI plug-n-play driver implementation. You will need to use the above SCPI commands.

When making power measurements on multiple bursts or slots, use CALCulate:DATA<n>:COMPress?

The CALC:DATA:COMP? query is the fastest way to measure power data for multiple bursts/slots. There are two reasons for this: 1. it can be used to measure data across multiple, consecutive slots/frames with just one measurement, instead of a separate measurement on each slot, and 2. it can pre-process and/or decimate the data so that you only return the information that you need which minimizes data transfer to the computer.

For example: let's say you want to do a power measurement for a GSM base station where you generate a repeating frame with 8 different power levels. You can gather all the data with a single CALC:DATA:COMP? acquisition, using the waveform measurement.

With **`CALC:DATA2:COMP? MEAN,9,197,1730`** you can measure the mean power in those bursts. This single command will measure the data across all 8 frames, locate the first slot/burst in each of the frames, calculate the mean power of those bursts, then return the resulting 8 values.

NOTE For later version of firmware (after E4406 A.05.00) you can use equivalent time values for the CALC:DATA<n>:COMP? query. The command would then be **`CALC:DATA2:COMP? MEAN,25us,526us,579.6us,8`**

Let's set up the GSM Waveform measurement:

- **`:CONF:WAV?`** turns on the waveform measurement
- **`:WAV:BAND 300khz`** sets a resolution bandwidth of 300 kHz
- **`:WAV:SWE:TIME 5ms`** sets a sweep time of 5 milliseconds
- **`:WAV:BAND:TYPE FLAT`** selects the flat filter type
- **`:WAV:DEC 4;DEC:STAT ON`** selects a decimation of 4 and turns on

decimation. This reduces the amount of data that needs to be sent since the instrument hardware decimates (throws some away).
- `:INIT` to initiate a measurement and acquire the data
- `CALC:DATA2:COMP? MEAN,25us,526us,579.6us,8` to return the desired data

There are two versions of this command depending on your firmware revision. Earlier revisions require the optional variables be entered in terms of their position in the trace data array. Current instruments allow the variables to be entered in terms of time.

For early firmware revisions you need to know the sample interval. In the waveform measurement it is equal to the aperture value. Query :WAVeform:APERture? to find the sample interval. (Note: the WAV:APER? command always takes decimation into account.) The sample interval (aperture value) is dependent on the settings for resolution bandwidth, filter type, and decimation. See the following table to see how these value relate.

The parameters for this GSM example are:
`MEAN,9,197,1730` (or with later firmware:
`MEAN,25us,526us,579.6us,8`)

- MEAN calculates the mean of the measurement points indicated
- 9 is how many points you want to discard before you look at the data. This allows you to skip over any “unsettled” values at the beginning of the burst. You can calculate this start offset by (25µs/sampleInterval)l
- 197 is the length of the data you want to use. This would be the portion of the burst that you want to find the mean power over. You can calculate this length by (526µs/sampleInterval)
- 1730 is how much data you have before you repeat the process. For this example it’s the time between the start offset point on the burst in the first slot (first frame) to the same spot on the burst in the first slot (second frame). You can calculate this by (576.9µs*N/sampleInterval) where N is the number of data items that you want. In this case it is the number of slots in the frame, N=8.)

Table 2-1 GSM Parameters for 1 Slot/Frame Measurement Requirements

Resolution Bandwidth	Filter Type	Decimation	Aperture	Start	Length	Repeat
500 or 300 kHz	Flat or Gaussian	4 or 1	dependent on settings	24 µsec[a]	526 µsec[a]	576.9 µsec[a]
500 kHz	Gaussian	1	0.2 µsec	124	2630	2884.6
500 kHz	Gaussian	4	0.8 µsec	31	657	721.15
500 kHz	Flat	1	0.4 µsec	61	1315	1442.3

Table 2-1 GSM Parameters for 1 Slot/Frame Measurement Requirements

Resolution Bandwidth	Filter Type	Decimation	Aperture	Start	Length	Repeat
500 kHz	Flat	4	1.6 µsec	15	329	360.575
300 kHz	Gaussian	1	0.2667 µsec	90	1972	2163.1
300 kHz	Gaussian	4	1.07 µsec	22	492	539.16
300 kHz	Flat	1	0.6667 µsec	36	789	865.31
300 kHz	Flat	4	2.667 µsec	9	197	216.33

a. The use of time values is only allowed in firmware versions of A.05.00 and later.

Preventing Local or Remote Interference While Programming

The following SCPI commands can help prevent interference from other users while you are programming the instrument remotely. See the SYSTem subsystem section of the Language Reference chapter for a full description of these commands.

- `:SYSTem:KLOCk 0|1|OFF|ON` locks the transmitter tester's keyboard.
- `:SYSTem:MESSage <string>` enables you to send a message that will appear in status bar at the bottom of the instrument display.

Using the Status Registers

Figure on page 101 shows the E4406A instrument status registers and their hierarchy.

What Status Registers Are

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

The operation and questionable status registers are sets of registers that monitor the overall instrument condition. They are accessed with the STATus:OPERation and STATus:QUEStionable commands in the STATus command subsystem. Each register set is made up of five registers:

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

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

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

Event Register it latches any signal state changes, in the way specified by the filter registers. Bits in the event register are never cleared by signal state changes. Event registers are cleared when read. They are also

cleared by *CLS and by presetting the instrument.

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

The STATus:QUEStionable registers report abnormal operating conditions. The status register hierarchy is:

1. The summary outputs from the six STATus:QUEStionable:<keyword> detail registers are inputs to the STATus:QUEStionable register.
2. The summary output from the STATus:QUEStionable register is an input to the Status Byte Register. See the E4406 Figure on page 101.

The STATus:OPERation register set has no summarized inputs. The inputs to the STATus:OPERation:CONDition register indicate the real time state of the instrument. The STATus:OPERation:EVENt register summary output is an input to the Status Byte Register.

The STATus:OPERation:ENABle register has an additional function in the E4406A. It is ANDed with the STATus:OPERation:CONDition register to determine what the instrument busy state is, that is then interpreted by the *OPC, *OPC? and *WAI commands. If the ANDed result is non-zero the instrument is considered busy.

What Status Register SCPI Commands Are

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

*CLS (clear status) clears the status byte by emptying the error queue and clearing all the event registers.

*ESE, *ESE? (event status enable) sets and queries the bits in the enable register part of the standard event status register.

*ESR? (event status register) queries and clears the event register part of the standard event status register.

*OPC, *OPC? (operation complete) sets the standard event status register to monitor the completion of all commands. The query stops any new commands from being processed until the current processing is complete, then returns a ‘1’.

*PSC, *PSC? (power-on state clear) sets the power-on state so that it clears the service request enable register and the event status enable register at power on.

*SRE, *SRE? (service request enable) sets and queries the value of the service request enable register.

*STB? (status byte) queries the value of the status byte register without erasing its contents.

How to Use the Status Registers

A program often needs to be able to detect and manage error conditions or changes in instrument status. There are two methods you can use to programmatically access the information in status registers:

- **The polling method**
- **The service request (SRQ) method**

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

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

Use the SRQ method when:

— you need time-critical notification of changes
— you are monitoring more than one device which supports SRQs
— you need to have the controller do something else while waiting
— you can't afford the performance penalty inherent to polling

Use polling when:

— your programming language/development environment does not support SRQ interrupts
— you want to write a simple, single-purpose program and don't want the added complexity of setting up an SRQ handler

To monitor a condition:

1. Determine which register contains the bit that reports the condition.
2. Send the unique SCPI query that reads that register.
3. Examine the bit to see if the condition has changed.

You can monitor conditions in different ways.

- Check the current instrument hardware and firmware status.

  Do this by querying the condition registers which continuously monitor status. These registers represent the current state of the

instrument. Bits in a condition register are updated in real time. When the condition monitored by a particular bit becomes true, the bit is set to 1. When the condition becomes false, the bit is reset to 0.

- Monitor a particular condition (bit).

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

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

  — The transition registers are preset to register if the condition goes from 0 to 1 (false to true, or a positive transition).
  — This can be changed so the selected condition is detected if the bit goes from 1 to 0 (true to false, or a negative transition).
  — It can also be set for both types of transitions occurring.
  — Or it can be set for neither transition. If both transition registers are set to 0 for a particular bit position, that bit will not be set in the event register for either type of change.

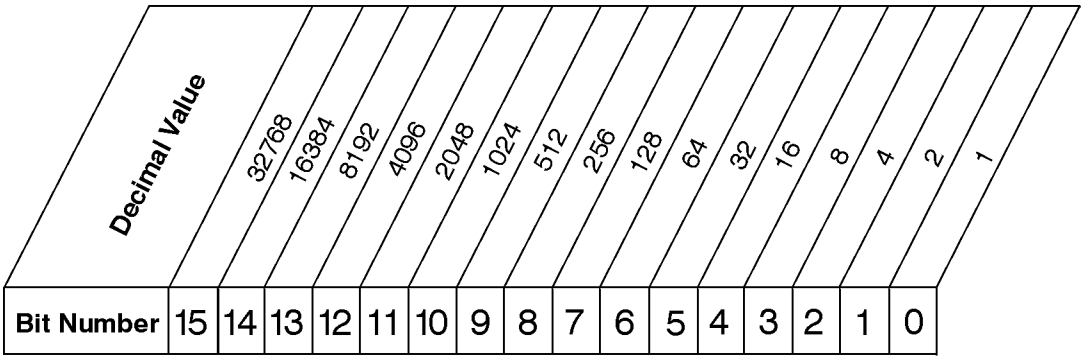
Using a Status Register

Each bit in a register is represented by a numerical value based on its location. See Figure 2-4 below. This number is sent with the command to enable a particular bit. If you want to enable more than one bit, you would send the sum of all the bits that you want to monitor.

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

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

Figure 2-4 Status Register Bit Values



NOTE Bit 15 is not used to report status.

Using the Service Request (SRQ) Method

Your language, bus and programming environment must be able to support SRQ interrupts. (For example, BASIC used with the GPIB.) SRQ is available on SICL LAN, USB, and GPIB. When you monitor a condition with the SRQ method, you must:

1. Determine which bit monitors the condition.
2. Determine how that bit reports to the request service (RQS) bit of the status byte.
3. Send GPIB commands to enable the bit that monitors the condition and to enable the summary bits that report the condition to the RQS bit.
4. Enable the controller to respond to service requests.

When the condition changes, the instrument sets its RQS bit and the GPIB SRQ line. The controller is informed of the change as soon as it occurs. As a result, the time the controller would otherwise have used to monitor the condition can be used to perform other tasks. Your program determines how the controller responds to the SRQ.

Generating a Service Request

To use the SRQ method, you must understand how service requests are generated. Bit 6 of the status byte register is the request service (RQS) bit. The *SRE command is used to configure the RQS bit to report changes in instrument status. When such a change occurs, the RQS bit is set. It is cleared when the status byte register is queried using `*SRE?` (with a serial poll.) It can be queried without erasing the contents with `*STB?`.

When a register set causes a summary bit in the status byte to change from 0 to 1, the instrument can initiate the service request (SRQ) process. However, the process is only initiated if both of the following conditions are true:

- The corresponding bit of the service request enable register is also set to 1.
- The instrument does not have a service request pending. (A service request is considered to be pending between the time the instrument's SRQ process is initiated and the time the controller reads the status byte register.)

The SRQ process sets the GPIB SRQ line true. It also sets the status byte's request service (RQS) bit to 1. Both actions are necessary to inform the controller that the instrument requires service. Setting the

SRQ line only informs the controller that some device on the bus requires service. Setting the RQS bit allows the controller to determine which instrument requires service.

If your program enables the controller to detect and respond to service requests, it should instruct the controller to perform a serial poll when the GPIB SRQ line is set true. Each device on the bus returns the contents of its status byte register in response to this poll. The device whose RQS bit is set to 1 is the device that requested service.

NOTE When you read the instrument's status byte register with a serial poll, the RQS bit is reset to 0. Other bits in the register are not affected.

NOTE If the status register is configured to SRQ on end-of-measurement and the measurement is in continuous mode, then restarting a measurement (INIT command) can cause the measuring bit to pulse low. This causes an SRQ when you have not actually reached the "end-of-measurement" condition. To avoid this:

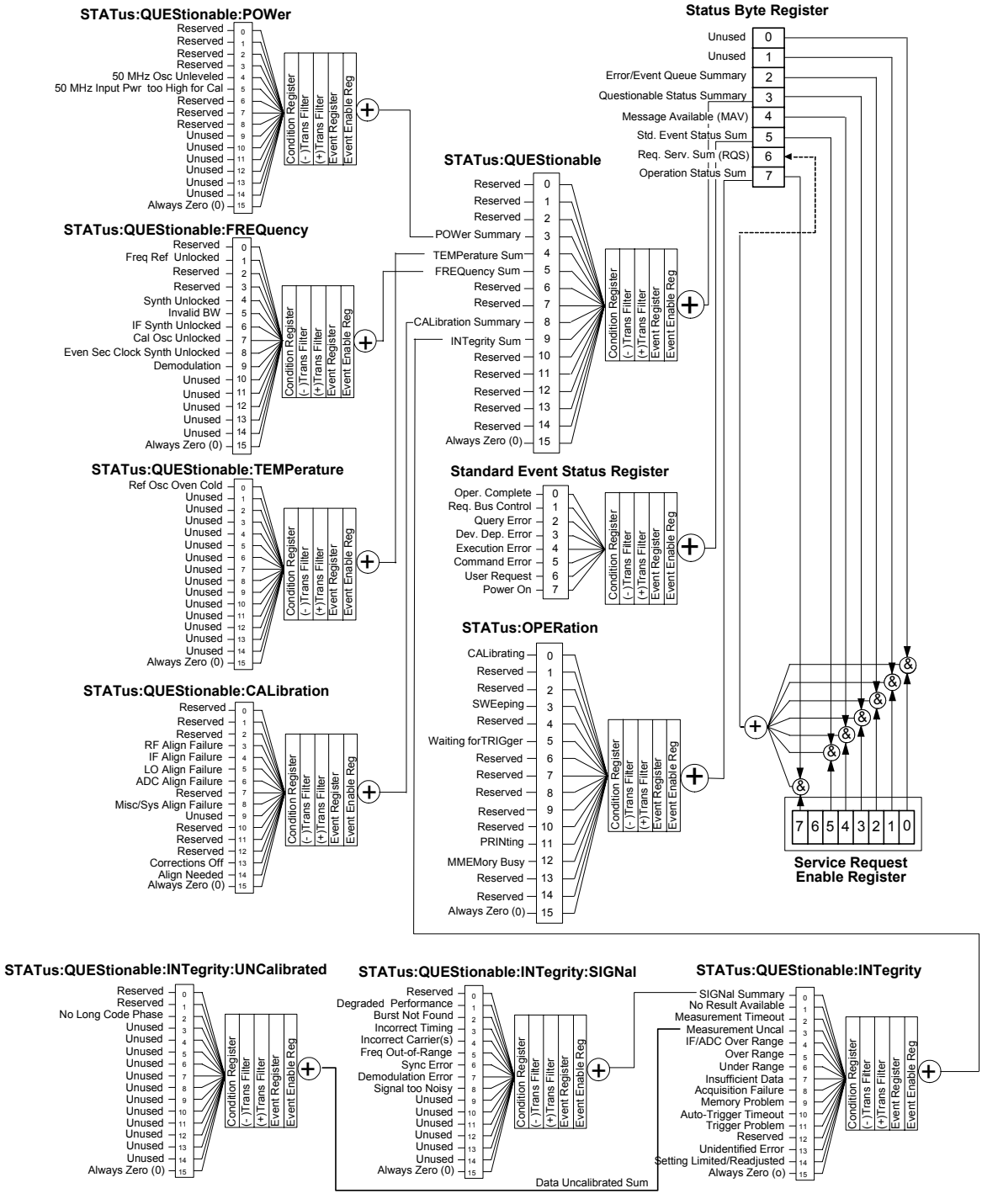
1. Set INITiate:CONTinuous off.
2. Set/enable the status registers.
3. Restart the measurement (send INIT).

E4406A Core Status Register System

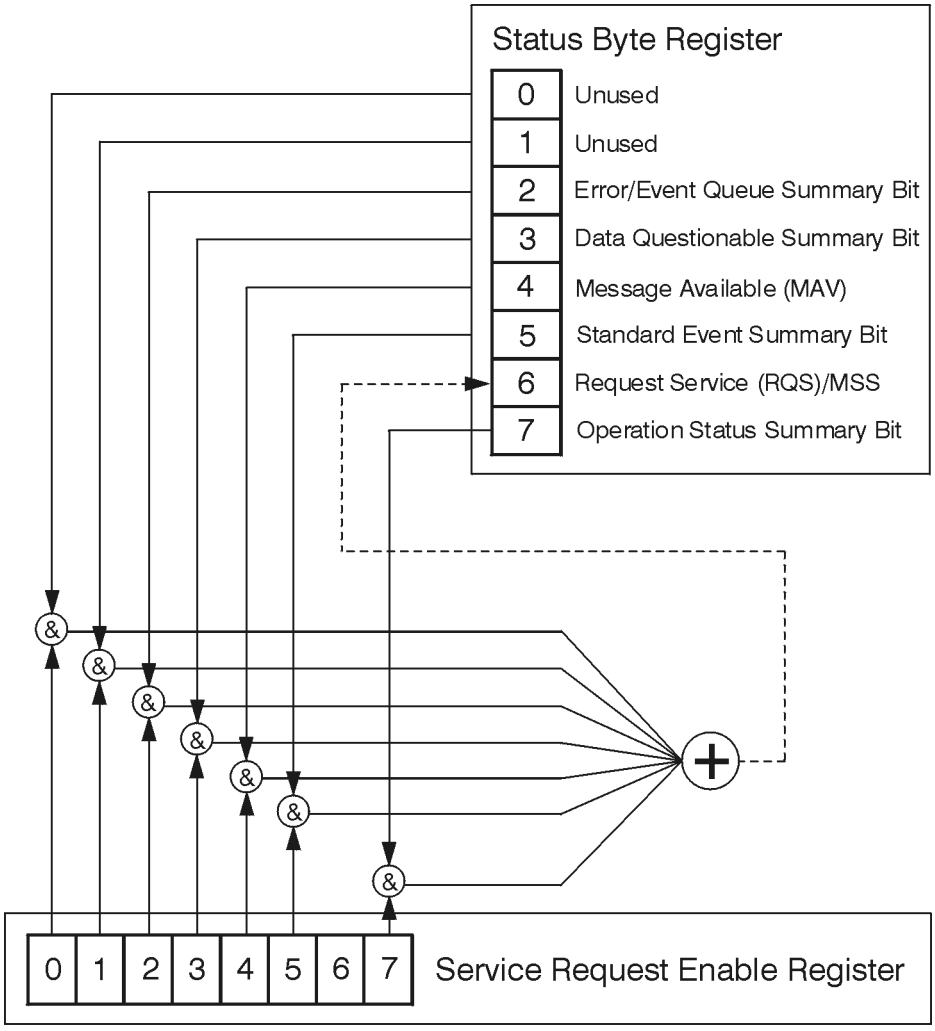
Preset Values

For All Registers: (-) Transition Filter = 0's
(+) Transion Filter = 1's
For STAT:QUES, STAT:OPER, & all OPER:INST:ISUM registers: Event Enable = 0's
For all Other Registers: Event Enable = 1's

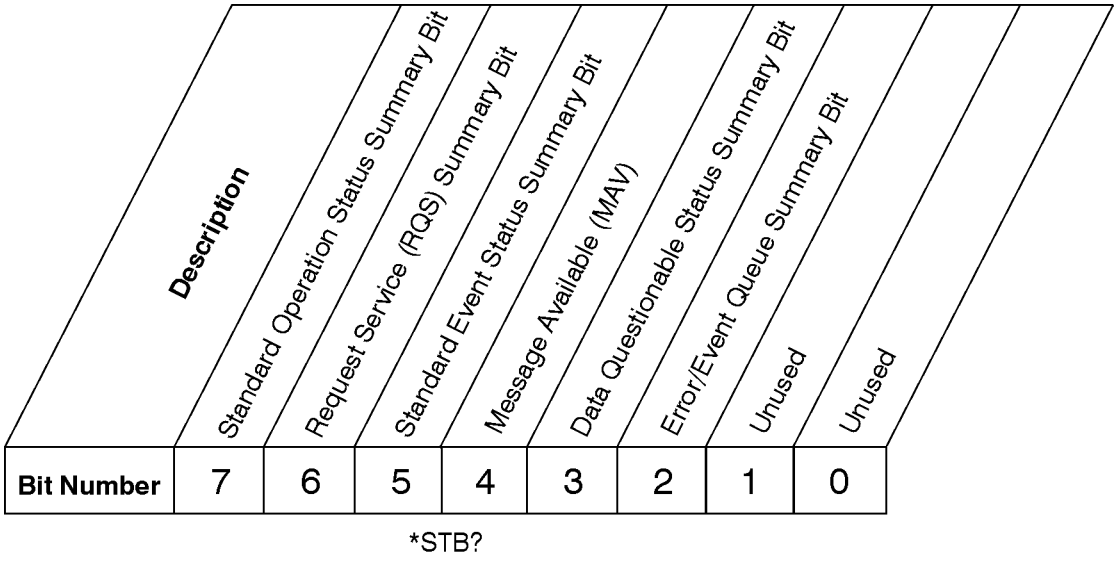
Unused: All unused bits = 0



Status Byte Register



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



*STB?

Status Byte Register

ck725a

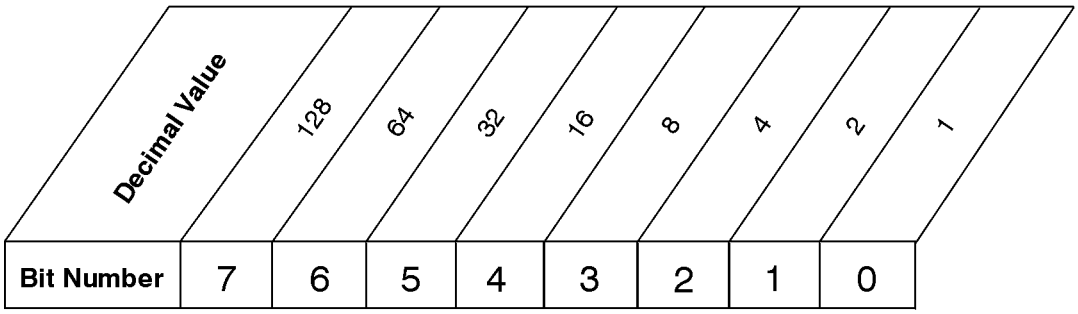
Bit	Description
0, 1	These bits are always set to 0.
2	A 1 in this bit position indicates that the SCPI error queue is not empty which means that it contains at least one error message.
3	A 1 in this bit position indicates that the data questionable summary bit has been set. The data questionable event register can then be read to determine the specific condition that caused this bit to be set.
4	A 1 in this bit position indicates that the instrument has data ready in the output queue. There are no lower status groups that provide input to this bit.
5	A 1 in this bit position indicates that the standard event summary bit has been set. The standard event status register can then be read to determine the specific event that caused this bit to be set.
6	A 1 in this bit position indicates that the instrument has at least one reason to report a status change. This bit is also called the master summary status bit (MSS).
7	A 1 in this bit position indicates that the standard operation summary bit has been set. The standard operation event register can then be read to determine the specific condition that caused this bit to be set.

To query the status byte register, send the command `*STB?` The response will be the *decimal* sum of the bits which are set to 1. For example, if bit number 7 and bit number 3 are set to 1, the decimal sum of the 2 bits is 128 plus 8. So the decimal value 136 is returned. The `*STB` command does not clear the status register

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

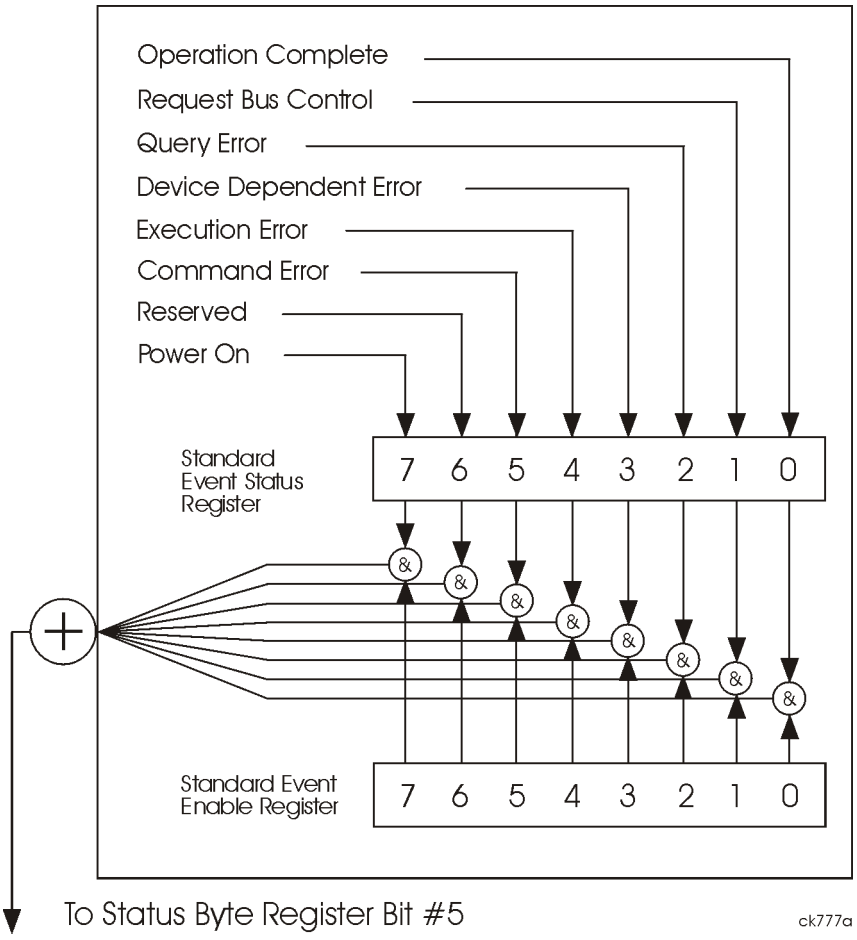
Send the `*SRE <number>` command where `<number>` is the sum of the decimal values of the bits you want to enable plus the decimal value of bit 6. For example, assume that you want to enable bit 7 so that whenever the standard operation status register summary bit is set to 1 it will trigger a service request. Send the command `*SRE 192` (because 192 = 128 + 64). You must always add 64 (the numeric value of RQS bit 6) to your numeric sum when you enable any bits for a service request. The command `*SRE?` returns the decimal value of the sum of the bits previously enabled with the `*SRE <number>` command.

The service request enable register presets to zeros (0).

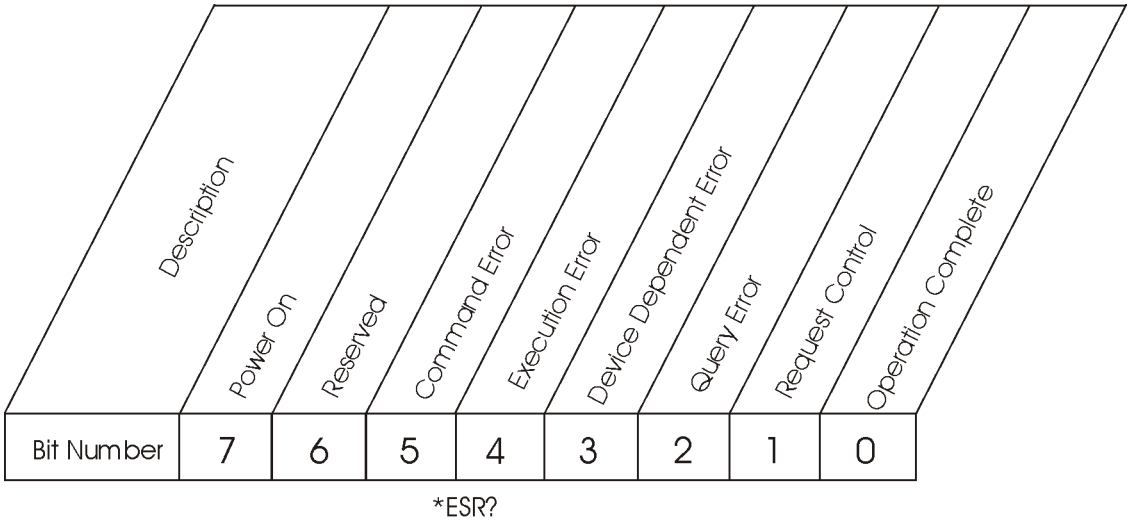


Service Request Enable Register ck726a

Standard Event Status Register



The standard event status register contains the following bits:



Standard Event Status Register

ck727a

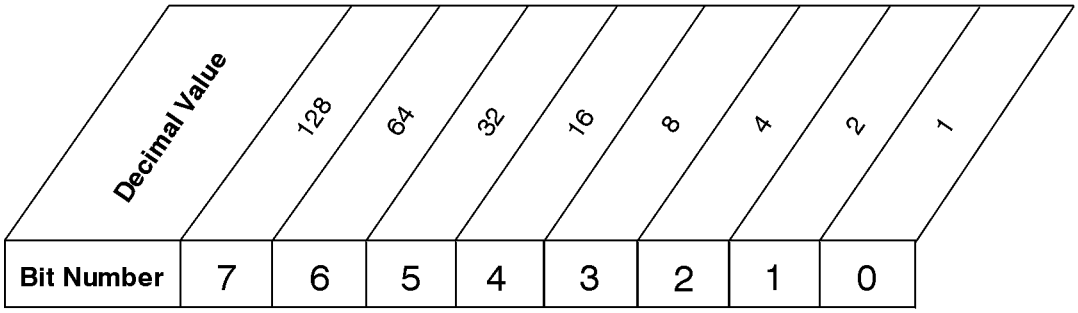
Bit	Description
0	A 1 in this bit position indicates that all pending operations were completed following execution of the `*OPC` command.
1	This bit is always set to 0. (The instrument does not request control.)
2	A 1 in this bit position indicates that a query error has occurred. Query errors have SCPI error numbers from –499 to –400.
3	A 1 in this bit position indicates that a device dependent error has occurred. Device dependent errors have SCPI error numbers from –399 to –300 and 1 to 32767.
4	A 1 in this bit position indicates that an execution error has occurred. Execution errors have SCPI error numbers from –299 to –200.
5	A 1 in this bit position indicates that a command error has occurred. Command errors have SCPI error numbers from –199 to –100.
6	Reserved
7	A 1 in this bit position indicates that the instrument has been turned off and then on.

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

In addition to the standard event status register, the standard event status group also contains a standard event status enable register. This register lets you choose which bits in the standard event status register

will set the summary bit (bit 5 of the status byte register) to 1. Send the *ESE <number> command where <number> is the sum of the decimal values of the bits you want to enable. For example, to enable bit 7 and bit 4 so that whenever either of those bits is set to 1, the standard event status summary bit of the status byte register will be set to 1, send the command *ESE 144 (128 + 16). The command *ESE? returns the decimal value of the sum of the bits previously enabled with the *ESE <number> command.

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



Standard Event Status Enable Register ck728a

Operation and Questionable Status Registers

The operation and questionable status registers are registers that monitor the overall instrument condition. They are accessed with the STATus:OPERation and STATus:QUEStionable commands in the STATus command subsystem. See the Figure on page 101.

Operation Status Register

The operation status register monitors the current instrument measurement state. It checks to see if the instrument is calibrating, sweeping, or waiting for a trigger. For more information see the *OPC? command located in the IEEE Common Commands section.

Bit	Condition	Operation
0	Calibrating	The instrument is busy executing its automatic alignment process
3	Sweeping	The instrument is busy taking a sweep.
5	Waiting for trigger	The instrument is waiting for the trigger conditions to be met, then it will trigger a sweep or measurement.

Bit	Condition	Operation
8	Paused	The instrument is paused (waiting) because you have pressed the **Pause** Meas Control key or sent the INITiate:PAUSe command. Bit is only valid for Modes: Spectrum Analysis, Phase Noise
11	Printing	The instrument is busy sending display data to the external printer.
12	Mass memory busy	The instrument is busy accessing the internal mass memory.

Questionable Status Register

The questionable status register monitors the instrument's condition to see if anything questionable has happened to it. It is looking for anything that might cause an error or a bad measurement like a hardware problem, an out of calibration situation, or a unusual signal. All the bits are summary bits from lower-level event registers.

Bit	Condition	Operation
3	Power summary	The instrument hardware has detected a power unleveled condition.
4	Temperature summary	The instrument is still warming up.
5	Frequency summary	The instrument hardware has detected an unlocked condition or a problem with the external frequency reference.
8	Calibration summary	The instrument has detected a hardware problem while doing the automatic internal alignment process.
9	Integrity summary	The instrument has detected a questionable measurement condition such as: bad timing, bad signal/data, timeout problem, signal overload, or "meas uncal".

Using the LAN to Control the Instrument

Refer to the E4406 User's Guide "Using System Features" chapter for information about configuring the instrument input/output settings from the front panel. Use the SYSTem commands to change settings remotely.

NOTE Remember that in any type of programming using LAN you should avoid constantly opening and closing connections. This uses up processing resources, adds to your system overhead, and can cause problems with asynchronous implementation of successive commands. When you are sending the instrument multiple commands: open the connection, send all the commands, and close the connection.

Using ftp for File Transfers

You can use the instrument LAN connection to transfer files. For example, you can use the ftp functionality to download instrument screen dumps to an external server.

The following is an example of an ftp session from an MSDOS window on a PC:

1. `ftp 141.88.163.118` (enter the instrument IP address, found/set from the front panel by pressing **System, Config I/O**)
2. At the user name prompt, enter: `vsa`
3. At the password prompt, enter: `service`

   You are now in the instrument `/users` directory and can get files from the instrument. The ftp commands in the following steps may not all be available from your controller. To show the ftp commands available on your system, type `help` at the prompt. To end the ftp session, type `quit`.

NOTE Do *not* delete files from this directory. Most of the files are required for instrument operation, and for the operation of optional personality modes.

4. `cd userdir` (change to the directory where data files are saved)
5. `ls` (list all available files, `ls -la` shows file permissions)
6. `bin` (change to the binary file transfer mode)
7. `get myfilename` (enter the file name; the name *is* case sensitive)

   This "gets" (copies) your file. The file is copied to the location you were pointing to when you *started* the ftp process. To query the current location, enter `lcd.` (include the period). To change the current location, enter the desired path/directory location as follows: `lcd C:\my path\mydir`

NOTE To use a web browser for this example, enter: `ftp://vsa:service@141.88.163.118/userdir`

The Standard UNIX FTP Command:

Synopsis `ftp [-g] [-i] [-n] [-v] [server-host] [-B DataSocketBufferSize]`

Description The `ftp` command is used to transfer files using the File Transfer Protocol. `ftp` transfers files over a network connection between a local machine and the remote `server-host`.

Options and Parameters When ftp is invoked with a server-host specified, a connection is opened immediately. Otherwise, ftp waits for user commands.

The following options are supported:

`-g`	disables expansion of shell metacharacters in file and directory names
`-i`	disables prompts during multiple-file operations
`-n`	disables automatic log-in
`-v`	enables verbose output
`-B`	specifies a new `DataSocketBufferSize`
`server-host`	the name or address of the remote host.

This table lists the available user commands.

Table 2-2 `ftp` Commands

Command	Description
ascii	Sets the file transfer type to ASCII.
binary	Sets the file transfer type to binary.
bye	Closes the connection to the host and exits ftp.
cd *remote_directory*	Sets the working directory on the host to *remote_directory*.
delete *remote_file*	Deletes *remote_file* or empty *remote_directory*.
dir [*remote_directory*]	Lists the contents of the specified *remote_directory*. If *remote_directory* is unspecified, the contents of the current remote directory are listed.
get *remote_file* [*local_file*]	Copies *remote_file* to *local_file*. If *local_file* is unspecified, ftp uses the *remote_file* name as the *local_file* name.
help	Provides a list of ftp commands.
help *command*	Provides a brief description of *command*.
image	Sets the file transfer type to binary.
lcd [*local_directory*]	Sets the local working directory to *local_directory*.
ls [*remote_directory*]	Lists the contents of the specified *remote_directory*. If the *remote_directory* is unspecified, the contents of the current remote directory are listed.
mget *remote_file [local_file]*	Copy *remote_file* to the local system. If *local_file* is unspecified, ftp uses the *remote_file* name as the *local_file* name.
mput *local_file [remote_file]*	Copies *local_file* to remote file. If *remote_file* is unspecified, ftp uses the *local_file* name as the *remote_file* name.
put *local_file* [*remote_file*]	Copies *local_file* to *remote file*. If *remote_file* is unspecified, ftp uses the *local_file* name as the *remote_file* name.
quit	Closes the connection to the host and exits ftp.

Using Telnet to Send Commands

Using telnet to send commands to your instrument works in a similar way to communicating over GPIB. You establish a connection with the instrument, and then send or receive information using SCPI commands.

NOTE If you need to control the bus using "device clear" or SRQ's, you can use SICL LAN. SICL LAN provides control of your instrument via IEEE 488.2 GPIB over the LAN. See "Using SICL LAN to Control the Instrument" on page 117. in this chapter.

NOTE STATus bits that are already set when the socket connection is made cannot be read. Only status bit changes that occur after the socket connection is made will returned when the status register is queried.

On unix or PC:

The syntax of the telnet command is:

```
telnet <IP address> <5023>
```

The initial telnet connection message will be displayed and then a SCPI> prompt. At the SCPI prompt, simply enter the desired SCPI commands.

On a PC (with telnet gui that has host/port setting menu):

You would type at the dos prompt

```
telnet
```

NOTE Early versions of Windows XP Telnet will initially only send a LF, not a CRLF. So the telnet port 5023 does not work. You can manually correct this situation by sending the escape sequence and then a CRLF. After connecting to the instrument, type in the telnet window:

- Crtl-] (press the control and] keys simultaneously)
- set crlf <enter key>
- <enter key>
  You should now see the SCPI> prompt and you can continue working.

The Windows XP Service Pack 2 fixes this problem. You can get Service Pack 2 from the Microsoft Windows update website.

Unix Telnet Example:

To connect to the instrument with host name `vsa` and port number `5023`, enter the following command:

```
telnet vsa 5023
```

When you connect to the instrument, it will display a welcome message and a command prompt. The instrument is now ready to accept your SCPI commands. As you type SCPI commands, query results appear on the next line. At any time, you can send a <device clear> by pressing cntrl-c on your keyboard. When you are done, break the telnet connection using your escape character, and type `quit`.

When the instrument responds with the welcome message and the SCPI prompt, you can immediately enter programming (SCPI) commands.

Typical E4406 commands might be:

```
CONF:SPECTRUM
CALC:SPECTRUM:MARK:TRACE SPECTRUM
CALC:SPECTRUM:MARK:MAX
CALC:SPECTRUM:MARK:MAX?
```

The small program above sets the instrument to measure a signal amplitude by placing a marker on the maximum point of the trace, and then querying the instrument for the amplitude of the marker.

You need to press Enter after typing in each command. After pressing Enter on the last line in the example above, the instrument returns the amplitude level of the marker to your computer and displays it on the next line. For example, after typing `(For E4406) CALC:SPEC:MARK:MAX?` and pressing Enter, the computer could display:

```
+1.71000000000E+002
```

When you are done, close the telnet connection. Enter the escape character to get the telnet prompt. The escape character (Ctrl and “]” in this example) does not print.

At the telnet prompt, type `quit` or `close`.

The telnet connection closes and you see your regular prompt.

```
Connection closed.
```

The following example shows a terminal screen using the example commands above.

E4406 Telnet Example:

```
Trying...
Connected to at10.sr.hp.com.
Escape character is '^]'.
Welcome to at10
Hewlett-Packard,E4406A,US38430092,PROTOTYPE

SCPI> *IDN?
Hewlett-Packard,E4406A,US38430092,PROTOTYPE
SCPI> *OPT?
"GSM","CDMA","NADC","IDEN"
SCPI> READ:SPECtrum?
-1.00714661E+002,+9.99620056E+008,+1095,+9.99499207E+008,+9.15527344E+002,+706,-
2.00000000E-007,+2.66666667E-007,+1,+1.88000000E-004,+25
SCPI>
```

NOTE

If your telnet connection is in a mode called “line-by-line,” there is no local echo. This means you will not be able to see the characters you are typing on your computer's display until *after* you press the Enter key.

To remedy this, you need to change your telnet connection to “character-by-character” mode. This can be accomplished in most systems by escaping out of telnet to the `telnet>` prompt and then typing `mode char`. If this does not work, consult your telnet program's documentation for how to change to “character-by-character” mode.

The Standard UNIX TELNET Command:

Synopsis `telnet` [`host` [`port`]]

Description The `telnet` command is used to communicate with another host using the TELNET protocol. When `telnet` is invoked with `host` or `port` arguments, a connection is opened to `host`, and input is sent from the user to `host`.

Options and Parameters `telnet` operates in line-by-line mode or in character-at-a-time mode. In line-by-line mode, typed text is first echoed on the screen. When the line is completed by pressing the **Enter** key, the text line is then sent to `host`. In character-at-a-time mode, text is echoed to the screen and sent to `host` as it is typed.

In some cases, if your telnet connection is in “line-by-line” mode, there is no local echo. This means you will not be able to see the characters you are typing on your computer's display until *after* you press the **Enter** key.

To remedy this, you need to change your telnet connection to “character-by-character” mode. This can be accomplished in most systems by escaping out of telnet to the `telnet>` prompt and then typing `mode char`. Consult your telnet program's documentation for how to change to “character-by-character” mode.

Using Socket LAN to Send Commands

Your instrument implements a sockets Applications Programming Interface (API) compatible with Berkeley sockets, Winsock, and other standard sockets APIs. You can write programs using sockets to control your instrument by sending SCPI commands to a socket connection you create in your program. Refer to Using a Java™ Applet Over Socket LAN in this chapter for example programs using sockets to control the instrument.

Setting Up Your Instrument for Socket Programming

Before you can use socket programming, you must identify your instrument's socket port number. The default is 5025.

1. Press **System, Config I/O, SCPI LAN, Socket Port**.
2. Notice that the port number you will use for your socket connection to the instrument is 5025.

NOTE You may need to enable the termination character attribute when using the VISA libraries for socket communication. If the termchar attribute is disabled, then no termination character is sent with the data and the bus will time out waiting for it. (Set vi_attr_termchar_en)

NOTE STATus bits that are already set when the socket connection is made cannot be read. Only status bit changes that occur after the socket connection is made will returned when the status register is queried.

Troubleshooting help:

You can verify that you can open a socket connection to your instrument by using telnet:

telnet <IP address> 5025

Characters typed from your keyboard won't be echoed from the instrument and the SCPI prompt won't be given. However, you will be able to send commands and query the instrument. For example, you can type `*idn?` and the instrument identification string will be returned.

Using SICL LAN to Control the Instrument

SICL LAN is a LAN protocol using the Standard Instrument Control Library (SICL). It provides control of your instrument over the LAN, using a variety of computing platforms, I/O interfaces, and operating systems. With SICL LAN, you control your remote instrument over the LAN with the same methods you use for a local instrument connected directly to the controller with the GPIB. More information about SICL LAN can be found in the *HP Standard Instrument Control Library* user's guide for HP-UX, part number E2091-90004.

Your instrument implements a SICL LAN *server*. To control the instrument, you need a SICL LAN *client* application running on a computer or workstation that is connected to the instrument over a LAN. Typical applications implementing a SICL LAN client include

- HP/Agilent VEE
- HP/Agilent BASIC
- National Instrument's LabView with HP/Agilent VISA/SICL client drivers

NOTE

The SICL LAN protocol is Agilent's implementation of the VXI-11 Instrument Protocol, defined by the VXIbus Consortium working group.

Older versions of National Instruments' VISA does not support the VXI-11 Instrument Protocol. Contact National Instruments for their latest version.

SICL LAN can be used with Windows 95, Windows 98, Windows NT, and HP-UX.

Your instrument has a SICL LAN server to emulate GPIB over LAN, but it cannot be used to control other externally connected GPIB instruments.

Collecting SICL LAN Set-up Information

Before you set up your instrument as a SICL LAN server, you need some information about your instrument. The "value" of the following parameters is used to set up your VISA/SICL LAN client application:

Emulated GPIB Name — The GPIB name is the name given to a device used to communicate with the instrument. Your instrument is shipped with `gpib7` as its GPIB name. The GPIB name is the same as the remote SICL address.

Emulated GPIB Logical Unit — The logical unit number is a unique integer assigned to the device to be controlled using SICL LAN. Your instrument is shipped with the logical unit number set

to 8.

This can't be change, but you don't care. Numbers 0 through 30, excluding 21, are valid logical unit numbers for your instrument. Logical unit number 21 is used for the instrument's internal emulation mode. (If you are using Agilent VEE and SICL LAN, the logical unit number is limited to the range of 0-8.)

Emulated GPIB Address — The emulated GPIB address (bus address) is assigned to the device to be controlled using SICL LAN. The instrument is shipped with the emulated GPIB address set to 18. The emulated GPIB address, for E4406, will be the same as your current setting of the GPIB address. If you change the GPIB address, the emulated GPIB address will not change until you restart the instrument.

The SICL LAN server uses the GPIB name, GPIB logical unit number, and GPIB address configuration on the SICL LAN client to communicate with the client. You must match these parameters *exactly* (including case) when you set up the SICL LAN client and server.

Configuring Your Instrument as a SICL LAN Server

After you have collected the required information from the SICL LAN client, perform the following steps to set up your instrument as a SICL LAN server:

1. Identify the GPIB name.

   Press **System, Config I/O, SICL Server, Emulated GPIB Name**, and notice that it is **gpib7**.

2. Notice that the **Emulated GPIB Logical Unit** is set to **8**.

3. Notice that the **Emulated GPIB Address** is set to **18**.

Configuring a PC as a SICL LAN Client

The descriptions here are based on Agilent's VISA revision G.02.02, model number 2094G. A copy of Agilent VISA instrument io libraries can be found on Agilent's website:

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

see also

```
http://www.agilent.com/find/vee
```

The VISA User's Guide information on LAN programming may also be useful, see:

```
ftp://ftp.agilent.com/pub/mpusup/pc/binfiles/iop/index.html
```

The following assumes a LAN connection between your computer and your instrument. This will not work for the GPIB to LAN gateway.

1. Install VISA revision G.02.02 or higher.
2. Run I/O configuration.
3. Select LAN Client from the available interface types.
4. Press Configure.
5. Enter an interface name, such as lan1.
6. Enter a logical unit number, such as 7.
7. Select Okay.
8. Select VISA LAN Client from the available interface types.
9. Press Configure.
10. Enter a VISA interface name, such as GPIB1.
11. Enter the host name or IP address of your instrument in the host name field, such as aaa.companyname.com or 137.12.255.255.

NOTE Changing the host name in your instrument does not change your LAN system representation of the host name. You must work through your local system administrator to change the host name on your LAN system and then change it to match in your instrument.

12. Enter a Remote SICL address, such as GPIB7.
13. Set the LAN interface to match the defined LAN client.
14. Select OK.
15. Close I/O Configuration by selecting OK.

Controlling Your Instrument with SICL LAN and HP/Agilent VEE

Before you can use SICL LAN with VEE, you need to set up VISA/SICL LAN I/O drivers for use with your VEE application. Consult your VEE documentation for information how to do this.

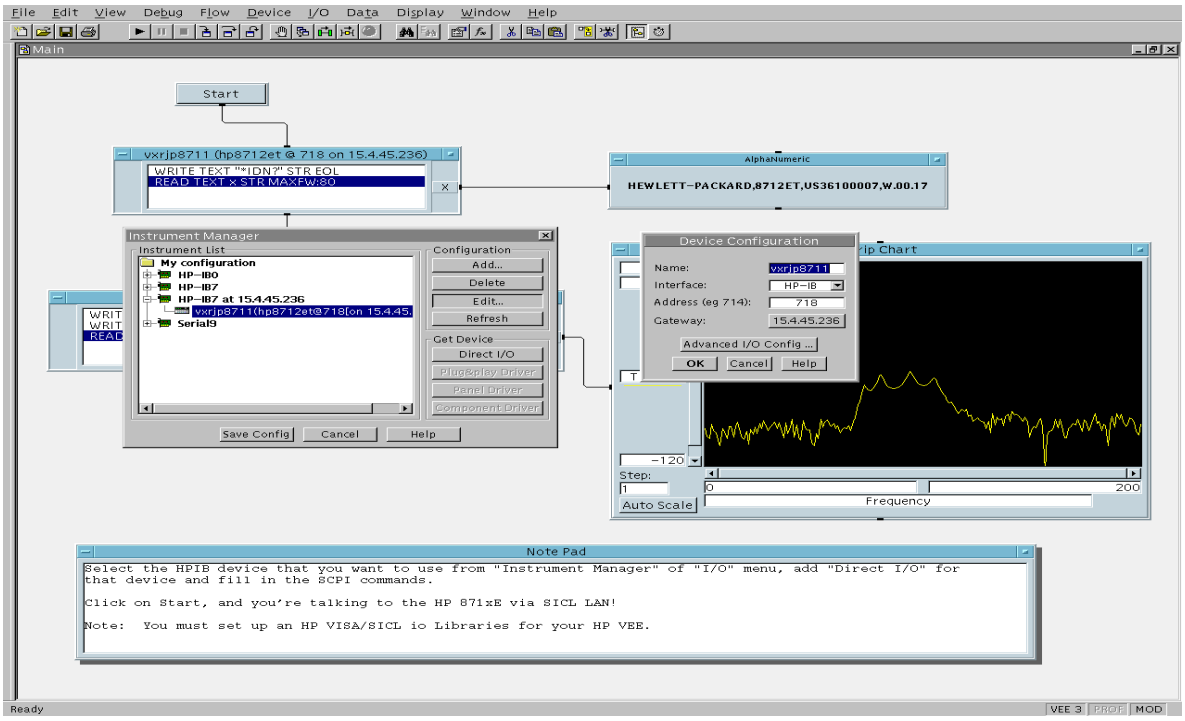
NOTE If you are using Agilent VEE and SICL LAN, the logical unit number is limited to the range of 0-8.

The logical unit number is the same as the interface select code (ISC). VEE reserves ISC values 9-18, and does not allow you to use them for SICL/LAN communications with your instrument. VEE also does not allow any ISC values higher than 18.

After you have the VISA/SICL LAN I/O drivers installed, perform the steps below to set up VEE to control your instrument:

1. On your computer or workstation, select `I/O|Instrument Manager`.
2. Add a new GPIB device with an address of 7XX, where XX is the GPIB device address from your instrument.

Figure 2-5 Adding Your Instrument as a VEE Device



To send SCPI commands to the instrument, select I/O | Instrument Manager, and the GPIB device just added. Select Direct I/O. You can now type SCPI commands in the command window, and they are sent over the LAN to your instrument.

Controlling Your Instrument with SICL LAN and Agilent BASIC for Windows

Before you can use Agilent BASIC for Windows with SICL LAN, you need to set up VISA/SICL LAN I/O drivers for use with your BASIC applications. Consult your BASIC documentation for information how to do this.

To set up SICL LAN for BASIC, add the following statement to your AUTOST program (all on a single line):

```
LOAD BIN "GPIBS;DEV lan[analyzer IP address]:GPIB name TIME 30 ISC 7"
```

Replace `analyzer IP address` with the IP address of your instrument, `GPIP name` with the GPIB name given to your instrument, and `7` with the logical unit number.

For example, the following LOAD statement should be added to your AUTOST program for the parameters listed below:

instrument IP address **191.108.344.225**

instrument GPIB name **inst0**

logical unit number **7**

timeout value (seconds) **30**

LOAD statement (all on a single line)

LOAD BIN “GPIBS;DEV lan[**191.108.344.225**]:**inst0** TIME **30** ISC **7**”

Consult your BASIC documentation to learn how to load the SICL driver for BASIC.

After the SICL driver is loaded, you control your instrument using commands such as the following:

```
OUTPUT 718; "*IDN?"
ENTER 718; S$
```

where 18 is the device address for the instrument.

See the BASIC example program in this chapter for more information.

Controlling Your Instrument with SICL LAN and BASIC for UNIX (Rocky Mountain BASIC)

Before you can use Rocky Mountain Basic (HPRMB) with SICL LAN, you will need to set up the SICL LAN I/O drivers for HPRMB. Consult your system administrator for details.

Create a `.rmbrc` file in your root directory of your UNIX workstation with the following entries:

```
SELECTIVE_OPEN=ON
Interface 8= "lan[analyzer IP address]:GPIB name";NORMAL
```

Replace `analyzer IP address` with the IP address of your instrument, and `GPIB name` with the GPIB name given to your instrument. Also replace the “8” of `Interface 8` with the logical unit number. Consult your HPRMB documentation for the exact syntax.

After your SICL driver is configured correctly on your UNIX workstation, you control your instrument using commands such as the following:

```
OUTPUT 818; "*IDN?"
ENTER 818; S$
```

where 18 is the device address for the instrument.

Using HP/Agilent VEE Over Socket LAN

(There is a VEE example program provided on the documentation CD-ROM.)

(There is a LabView example program provided on the documentation CD-ROM.)

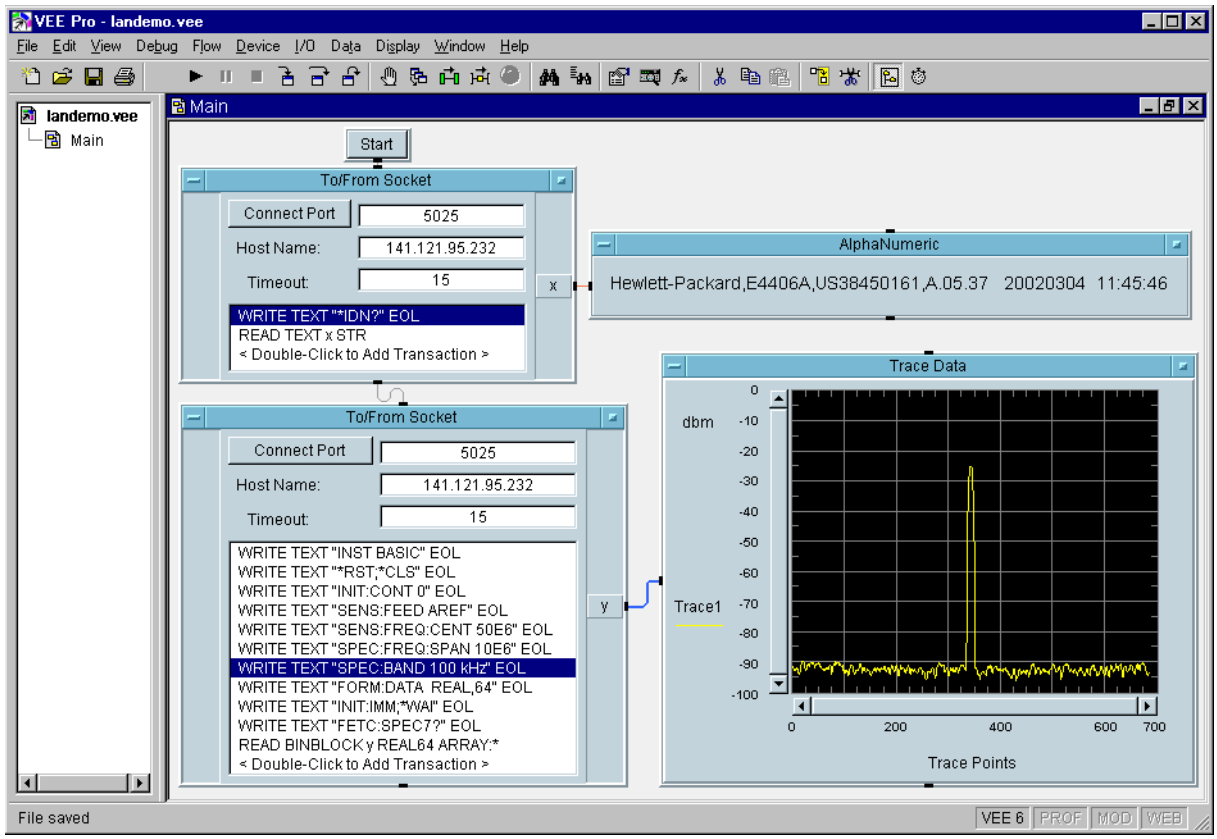
To control your instrument via socket LAN using VEE, click on the VEE menu titled “I/O”. Then select “To/From Socket” and position the I/O object box on the screen. Fill in the following fields:

```
Connect Port:    5025
Host Name:       <your_hostname>
Timeout:         15
```

For faster troubleshooting, you may want to set the timeout to a smaller number. If the host name you enter doesn't work, try using the IP address of your instrument (example: `191.108.43.5`). Using the IP address rather than the hostname may also be faster. See Figure 2-6 on page 124 for an example of an VEE screen.

NOTE Changing the host name in the instrument does not change your LAN system’s representation of the host name. You must work through your local system administrator to change the host name on your LAN system.

Figure 2-6 Sample E4406 VEE Screen



Using a Java™ Applet Over Socket LAN

There is a programming example in the VSA *Programmer's Guide* that demonstrates simple socket programming with Java. It is written in Java programming language, and will compile with Java compilers versions 1.0 and above.

This program is also on the documentation CD ROM that shipped with your product.

Using a C Program Over Socket LAN

The VSA *Programmer's Guide* contains two examples of simple LAN socket programs. They are written in C. One compiles in the HP-UX UNIX environment and one is written for the WIN32 environment.

In UNIX, LAN communication via sockets is very similar to reading or writing a file. The only difference is the openSocket `()` routine, which uses a few network library routines to create the TCP/IP network connection. Once this connection is created, the standard fread `()` and fwrite `()` routines are used for network communication.

In Windows, the routines send `()` and recv `()` *must* be used, because fread `()` and fwrite `()` may not work on sockets.

NOTE You may need to enable the termination character attribute when using the VISA libraries for socket communication. If the termchar attribute is disabled, then no termination character is sent with the data and the bus will time out waiting for it. (Set vi_attr_termchar_en)

General LAN Troubleshooting

- "Troubleshooting the Initial Connection" on page 125
- "Common Problems After a Connection is Made" on page 127
- "Pinging the Instrument from a Computer or Workstation" on page 129
- "EIA/TIA 568B Wiring Information" on page 131

Troubleshooting the Initial Connection

Getting the instrument to work with your network often requires detailed knowledge of your local network software. This section attempts to help you with some common problems. Contact your network administrator for additional assistance.

The instrument LAN interface does not need or include any proprietary driver software. It was designed to operate with common network utilities and drivers.

Either a hardware problem or a software problem can prevent the

instrument's remote file server from communicating over the LAN. The following common problems may be encountered:

Communications Not Established If you have just installed and configured the LAN interface and you have never been able to access the instrument via ftp or telnet, go directly to "Pinging the Instrument from a Computer or Workstation" on page 129.

If you have previously been able to access the instrument via ftp or telnet and now cannot do so, check the following:

- ❏ Has any hardware been added or moved on your network? This includes adding or removing any workstations or peripherals, or changing any cabling.
- ❏ Have software applications been added to the network?
- ❏ Has the functionality been turned off from the front panel? Press **System, Config I/O, SCPI LAN**.
- ❏ Have any configuration files been modified? Pressing **System, Restore Sys Defaults** restores the original factory defaults and you will have to re-set the instrument IP address and host name.
- ❏ Is the upper- and lower-case character usage in your host name consistent?
- ❏ Have any of the following files been deleted or overwritten?

  UNIX:

  — /etc/hosts

  — /etc/inetd.conf

  — /etc/services

  PCs:

  — dependent network files

If you know or suspect that something has changed on your network, consult with your network administrator.

Timeout Errors Timeout errors such as "Device Timeout," "File Timeout," and "Operation Timeout," are symptoms of one or both of the following problems:

— The currently configured timeout limits are too short compared to the time it takes the LAN to complete some operations. This problem may occur during periods of increased LAN traffic.

— The LAN connection has failed, or fails occasionally.

To increase your timeout period, refer to your computer documentation for instructions. Contact your LAN administrator if problems continue.

Packets Routinely Lost If packets are routinely lost, proceed to the troubleshooting section in this chapter relating to your network.

Problems Transferring or Copying Files If you have problems copying files out of or into the instrument, you might be experiencing timeout problems. See the previous section on "Timeout Errors."

Common Problems After a Connection is Made

This section describes common problems you may encounter when using the instrument on a LAN. It assumes you have been able to connect to the instrument in the past. If this is not so, refer to the previous sections first.

NOTE Pressing **Preset** does not affect LAN settings, but pressing **System, Restore Sys Defaults** will reset to the original factory defaults. You will then have to re-set the instrument IP address and other LAN settings in **System, Config I/O**.

NOTE Remember that in any type of programming using LAN you should avoid constantly opening and closing connections. This uses up processing resources, adds to your system overhead, and can cause problems with asynchronous implementation of successive commands. When you are sending the instrument multiple commands: open the connection, send all the commands, and close the connection.

Cannot connect to the analyzer

- If you suspect a bad LAN connection between your computer and instrument, you can verify the network connection by using the ping command described later in this chapter or another similar echo request utility.
- If a bad connection is revealed, try the following solutions:
  - — Make sure the instrument is turned on.
  - — Check the physical connection to the LAN.
  - — Make sure the internet (IP) Address of the instrument is set up correctly in the LAN port setup menu. (Press **System, Config I/O, IP Address**.)
  - — When connecting to your instrument over a closed network (directly through a hub or crossover cable) it may help to set the instrument to its default settings for subnet mask and gateway. (subnet mask: 255.255.0.0, gateway 0.0.0.0)
  - — If the instrument and the computer are on different networks or subnets, make sure the gateway address and subnet mask values are set correctly.

Cannot access the file system via ftp

- If you get a “connection refused” message, try the following solutions:
  - — If the power to the instrument was just turned on, make sure that you wait about 25 seconds before attempting the connection.
- If you get a “connection timed out” message
  - — Verify the LAN connection between your computer and the instrument. Refer to “If you cannot connect to the instrument” earlier in this section.

Cannot telnet to the command parser port

- For a “connection refused” message
  - — Check the telnet port number from the front panel keys.
- For a “connection timed out” or “no response from host” message
  - — Verify the LAN connection between your computer and the instrument. Refer to “If you cannot connect to the instrument” earlier in this section.
- For a “connection refused” or “no response from host” message
  - — If the instrument was just turned on, make sure that you wait about 25 seconds before attempting the connection.

An “operation timed-out” message

- Check the LAN connection between the computer and the instrument. Refer to “If you cannot connect to the instrument” in this section.
- Increase the file time-out value on your PC or workstation.

Cannot access internal web pages or import graphic images when using a point-to-point connection

- Disable the use of proxy servers. You may have to specify this in a number of locations, depending on the operating system and software you are using.
- Disable the use of cached copies of web pages to ensure that you always get a new copy of the instrument’s screen image.

If all else fails

- Contact your network administrator.
- If you still cannot solve the problem, contact an Agilent Service Center for repair information.

Pinging the Instrument from a Computer or Workstation

Verify the communications link between the computer and the instrument remote file server using the ping utility.

From a UNIX workstation, type:

```
ping hostname 64 10
```

where 64 is the packet size, and 10 is the number of packets transmitted.

From a DOS or Windows environment, type:

```
ping hostname 10
```

where 10 is the number of echo requests.

Normal Response for UNIX

A normal response to the ping will be a total of 9, 10, or possibly 11 packets received with a minimal average round-trip time. The minimal average will be different from network to network. LAN traffic will cause the round-trip time to vary widely.

Because the number of packets received depends on your network traffic and integrity, the normal number might be different for your network.

Normal Response for DOS or Windows

A normal response to the ping will be a total of 9, 10, or possibly 11 packets received if 10 echo requests were specified.

Because the number of packets received depends on your network traffic and integrity, the normal number might be different for your network.

Error Messages

If error messages appear, then check the command syntax before continuing with the troubleshooting. If the syntax is correct, then resolve the error messages using your network documentation, or by consulting your network administrator.

If an unknown host error message appears, then check that the host name and IP address for your instrument are correctly entered from the front panel. Press **System, Config I/O**.

No Response No packets received indicates no response from a ping.

If there is no response, try typing in the IP address with the ping command, instead of using the host name. Check that the typed address matches the IP address assigned in the **System, Config I/O** menu, then check the other addresses in the menu.

Check that the host name and IP address are correctly entered in

the node names database.

If you are using a UNIX environment, ping each node along the route between your workstation and the instrument, starting with the your workstation. Ping each gateway, then attempt a ping of the remote file server.

If the instrument still does not respond to ping, then you should suspect a hardware problem with the instrument. To check the instrument performance, refer to "Verify the Instrument Performance" in this chapter.

Intermittent Response If you received 1 to 8 packets back, there is probably a problem with the network. Because the number of packets received depends on your network traffic and integrity, the number might be different for your network.

Use a LAN analyzer or LAN management software to monitor activity and determine where bottlenecks or other problems are occurring. The instrument will still function, but communications over the LAN will be slower.

On a single-client/single-server network, the most likely cause of intermittent response to an echo request is a hardware problem with the LAN module installed in the PC, the cable, or the instrument. To check the instrument, refer to "Verify the Instrument Performance" later in this chapter.

The Standard UNIX PING Command **Synopsis** `ping [-r] [-v] [-o] host [packetsize] [count]`

Description The `ping` command sends an echo request packet to the `host` once per second. Each echo response packet that is returned is listed on the screen, along with the round-trip time of the echo request and echo response.

Options and Parameters `-r` Bypasses the routing tables, and sends the request directly to the host.

`-v`	Reports all packets that are received, including the response packets.
`-o`	Requests information about the network paths taken by the requests and responses.
`host`	The host name or IP address.
`packetsize`	The size of each packet (8 bytes - 4096 bytes).
`count`	The number of packets to send before ending ping ($1-(2^{31}-1)$). If `count` is not specified, `ping` sends packets until interrupted.

EIA/TIA 568B Wiring Information

Table 2-3 Straight-Through Cable (Unshielded-twisted-pair (UTP) cable with RJ-45 connectors)

Standard, Straight-Through Wiring (each end)			
Signal Name	**RJ-45 Pin #**	**Wire Color**	**Pair #**
RX+	1	white/orange	2
RX-	2	orange	
TX+	3	white/green	3
TX-	6	green	
Not Used	4	blue	1
	5	white/blue	
	7	white/brown	4
	8	brown	

Table 2-4 Cross-Over Cable (Unshielded-twisted-pair (UTP) cable with RJ-45 connectors)

Cross-Over Wiring[a]			
Connector A		**Connector B**	
Signal Name	**RJ-45 Pin #**	**RJ-45 Pin #**	**Signal Name**
RX+	1	3	TX+
RX-	2	6	TX-
TX+	3	1	RX+
TX-	6	2	RX-
Not Used	4	4	Not Used
	5	5	
	7	7	
	8	8	

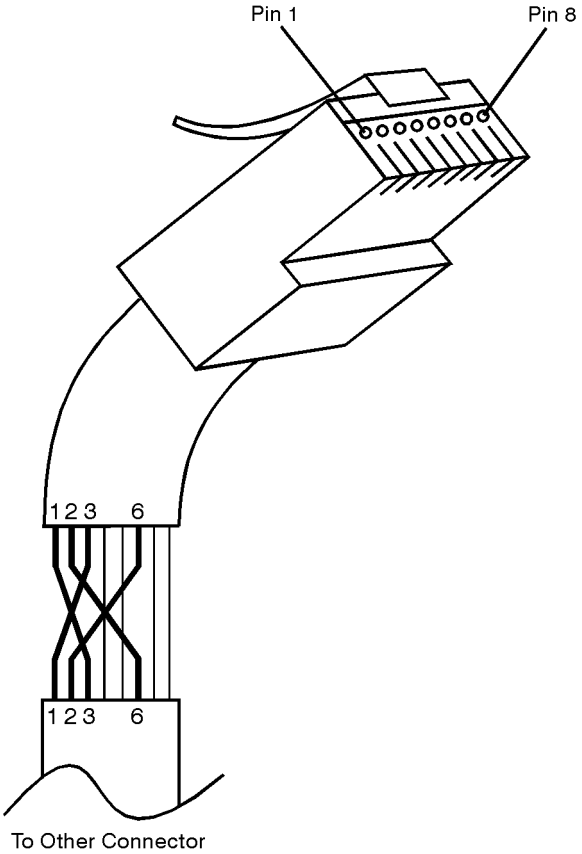
a. Either end of this cable can be used at the instrument or LAN device. The connector names are a convention useful during cable construction only.

This cable can be used to cascade hubs or to make point-to-point connections without a LAN hub.

NOTE A convenient way to make a cross-over adapter is to use two RJ-45 *jacks* wired according to Table 2-4, above. Standard straight-through patch cables can then be used from the instrument to the adapter, and from the adapter to other LAN devices. If you use a special-purpose adapter, you will avoid having a cross-over cable mistaken for a standard, straight-through patch cable.

NOTE Some commercially-available cross-over cables do not implement the cross-over wiring required for your instrument. Please refer to Table 2-4, above, and verify all connections before using cables not made by Agilent Technologies.

Figure 2-7 Cross-Over Patch Cable Wiring (cross-over end)



Programming in C Using the VTL

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

Refer to your C programming language documentation for more details. (This information is taken from the manual "VISA Transition Library", part number E2090-90026.) The following topics are included:

Typical Example Program Contents

The following is a summary of the VTL function calls used in the example programs.

`visa.h` This file is included at the beginning of the file to provide the function prototypes and constants defined by VTL.

`ViSession` The `ViSession` is a VTL data type. Each object that will establish a communication channel must be defined as `ViSession`.

`viOpenDefaultRM` You must first open a session with the default resource manager with the `viOpenDefaultRM` function. This function will initialize the default resource manager and return a pointer to that resource manager session.

`viOpen` This function establishes a communication channel with the device specified. A session identifier that can be used with other VTL functions is returned. This call must be made for each device you will be using.

`viPrintf`
`viScanf` These are the VTL formatted I/O functions that are patterned after those used in the C programming

language. The `viPrintf` call sends the IEEE 488.2 `*RST` command to the instrument and puts it in a known state. The `viPrintf` call is used again to query for the device identification (`*IDN?`). The `viScanf` call is then used to read the results.

`viClose` This function must be used to close each session. When you close a device session, all data structures that had been allocated for the session will be de-allocated. When you close the default manager session, all sessions opened using the default manager session will be closed.

Linking to VTL Libraries

Your application must link to one of the VTL import libraries:

32-bit Version:

`C:\VXIPNP\WIN95\LIB\MSC\VISA32.LIB` for Microsoft compilers

`C:\VXIPNP\WIN95\LIB\BC\VISA32.LIB` for Borland compilers

16-bit Version:

`C:\VXIPNP\WIN\LIB\MSC\VISA.LIB` for Microsoft compilers

`C:\VXIPNP\WIN\LIB\BC\VISA.LIB` for Borland compilers

See the following section, "Compiling and Linking a VTL Program" for information on how to use the VTL run-time libraries.

Compiling and Linking a VTL Program

32-bit Applications

The following is a summary of important compiler-specific considerations for several C/C++ compiler products when developing WIN32 applications.

For Microsoft Visual C++ version 2.0 compilers:

- Select `Project | Update All Dependencies` from the menu.
- Select `Project | Settings` from the menu. Click on the `C/C++` button. Select `Code Generation` from the `Use Run-Time Libraries` list box. VTL requires these definitions for WIN32. Click on `OK` to close the dialog boxes.
- Select `Project | Settings` from the menu. Click on the `Link` button and add `visa32.lib` to the `Object / Library Modules` list box. Optionally, you may add the library directly to your project file. Click on `OK` to close the dialog boxes.
- You may wish to add the include file and library file search paths.

They are set by doing the following:

1. Select `Tools | Options` from the menu.
2. Click on the `Directories` button to set the include file path.
3. Select `Include Files` from the `Show Directories For` list box.
4. Click on the `Add` button and type in the following:
   `C:\VXIPNP\WIN95\INCLUDE`
5. Select `Library Files` from the `Show Directories For` list box.
6. Click on the `Add` button and type in the following:
   `C:\VXIPNP\WIN95\LIB\MSC`

For Borland C++ version 4.0 compilers:

- You may wish to add the include file and library file search paths. They are set under the `Options | Project` menu selection. Double click on `Directories` from the `Topics` list box and add the following:

  ```
  C:\VXIPNP\WIN95\INCLUDE
  C:\VXIPNP\WIN95\LIB\BC
  ```

16-bit Applications

The following is a summary of important compiler-specific considerations for the Windows compiler.

For Microsoft Visual C++ version 1.5:

- To set the memory model, do the following:

  1. Select `Options | Project`.
  2. Click on the `Compiler` button, then select `Memory Model` from the `Category` list.
  3. Click on the `Model` list arrow to display the model options, and select `Large`.
  4. Click on `OK` to close the `Compiler` dialog box.

- You may wish to add the include file and library file search paths. They are set under the `Options | Directories` menu selection:

  ```
  C:\VXIPNP\WIN\INCLUDE
  C:\VXIPNP\WIN\LIB\MSC
  ```

  Otherwise, the library and include files should be explicitly specified in the project file.

Example Program

This example program queries a GPIB device for an identification string and prints the results. Note that you must change the address.

```
/*idn.c - program filename */

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

void main ()
{
     /*Open session to GPIB device at address 18 */
     ViOpenDefaultRM (&defaultRM);
     ViOpen (defaultRM, GPIB0::18::INSTR", VI_NULL,
       VI_NULL, &vi);

     /*Initialize device */
     viPrintf (vi, "*RST\n");

     /*Send an *IDN? string to the device */
     printf (vi, "*IDN?\n");

     /*Read results */
     viScanf (vi, "%t", &buf);

     /*Print results */
     printf ("Instrument identification string: %s\n", buf);

     /* Close sessions */
     viClose (vi);
     viClose (defaultRM);
}
```

Including the VISA Declarations File

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

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

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

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

Opening a Session

A session is a channel of communication. Sessions must first be opened

on the default resource manager, and then for each device you will be using. The following is a summary of sessions that can be opened:

- A **resource manager session** is used to initialize the VISA system. It is a parent session that knows about all the opened sessions. A resource manager session must be opened before any other session can be opened.
- A **device session** is used to communicate with a device on an interface. A device session must be opened for each device you will be using. When you use a device session you can communicate without worrying about the type of interface to which it is connected. This insulation makes applications more robust and portable across interfaces. Typically a device is an instrument, but could be a computer, a plotter, or a printer.

NOTE

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

Device Sessions

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

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

viOpenDefaultRM (*sesn*);

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

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

Your program may have several sessions open at the same time by

creating multiple session identifiers by calling the `viOpen` function multiple times.

The following summarizes the parameters in the previous function calls:

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

The following is an example of opening sessions with a GPIB multimeter and a GPIB-VXI scanner:

```
ViSession defaultRM, dmm, scanner;
.
.
viOpenDefaultRM(&defaultRM);
viOpen (defaultRM, "GPIB0::22::INSTR", VI_NULL,
    VI_NULL, &dmm);
viOpen (defaultRM, "GPIB-VXI0::24::INSTR", VI_NULL,
    VI_NULL, &scanner);
.
.
viClose (scanner);
viClose (dmm);
viClose(defaultRM);
```

The above function first opens a session with the default resource manager. The session returned from the resource manager and a device address is then used to open a session with the GPIB device at address 22. That session will now be identified as **dmm** when using other VTL functions. The session returned from the resource manager is then used again with another device address to open a session with the GPIB-VXI device at primary address 9 and VXI logical address 24. That session will now be identified as **scanner** when using other VTL functions. See the following section for information on addressing particular devices.

Addressing a Session

As seen in the previous section, the *rsrcName* parameter in the `viOpen` function is used to identify a specific device. This parameter is made up of the VTL interface name and the device address. The interface name is determined when you run the VTL Configuration Utility. This name

is usually the interface type followed by a number. The following table illustrates the format of the *rsrcName* for the different interface types:

Interface	Syntax
VXI	VXI [*board*]::*VXI logical address*[::INSTR]
GPIB-VXI	GPIB-VXI [*board*]::*VXI logical address*[::INSTR]
GPIB	GPIB [*board*]::*primary address*[::*secondary address*][::INSTR]

The following describes the parameters used above:

board — This optional parameter is used if you have more than one interface of the same type. The default value for *board* is 0.

VSI logical address — This is the logical address of the VXI instrument.

primary address — This is the primary address of the GPIB device.

secondary address — This optional parameter is the secondary address of the GPIB device. If no secondary address is specified, none is assumed.

INSTR — This is an optional parameter that indicates that you are communicating with a resource that is of type **INSTR**, meaning instrument.

NOTE If you want to be compatible with future releases of VTL and VISA, you must include the INSTR parameter in the syntax.

The following are examples of valid symbolic names:

XI0::24::INSTR — Device at VXI logical address 24 that is of VISA type INSTR.

VXI2::128 — Device at VXI logical address 128, in the third VXI system (VXI2).

GPIB-VXI0::24 — A VXI device at logical address 24. This VXI device is connected via a GPIB-VXI command module.

GPIB0::7::0 — A GPIB device at primary address 7 and secondary address 0 on the GPIB interface.

The following is an example of opening a device session with the GPIB device at primary address23.

```
ViSession defaultRM, vi;

.
.
```

```
viOpenDefaultRM (&defaultRM);
viOpen (defaultRM, "GPIB0::23::INSTR", VI_NULL,VI_NULL,&vi);
.
.
viClose(vi);
viClose (defaultRM);
```

Closing a Session

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

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

Overview of the GPIB Bus

An instrument that is part of a GPIB network is categorized as a listener, talker, or controller, depending on its current function in the network.

Listener	A listener is a device capable of receiving data or commands from other instruments. Any number of instruments in the GPIB network can be listeners simultaneously.
Talker	A talker is a device capable of transmitting data or commands to other instruments. To avoid confusion, a GPIB system allows only one device at a time to be an active talker.
Controller	A controller is an instrument, typically a computer, capable of managing the various GPIB activities. Only one device at a time can be an active controller.

GPIB Command Statements

Command statements form the nucleus of GPIB programming. They are understood by all instruments in the network. When combined with the programming language codes, they provide all management and data communication instructions for the system. Refer to the your programming language manual and your computers I/O programming manual for more information.

The seven fundamental command functions are as follows:

- An abort function that stops all listener/talker activity on the interface bus, and prepares all instruments to receive a new command from the controller. Typically, this is an initialization command used to place the bus in a known starting condition (sometimes called: abort, abortio, reset, halt).
- A remote function that causes an instrument to change from local control to remote control. In remote control, the front panel keys are disabled except for the Local key and the line power switch (sometimes called: remote, resume).
- A local lockout function, that can be used with the remote function, to disable the front panel Local key. With the Local key disabled, only the controller (or a hard reset by the line power switch) can restore local control (sometimes called: local lockout).
- A local function that is the complement to the remote command, causing an instrument to return to local control with a fully enabled front panel (sometimes called: local, resume).

- A clear function that causes all GPIB instruments, or addressed instruments, to assume a cleared condition. The definition of clear is unique for each instrument (sometimes called: clear, reset, control, send).
- An output function that is used to send function commands and data commands from the controller to the addressed instrument (sometimes called: output, control, convert, image, iobuffer, transfer).
- An enter function that is the complement of the output function and is used to transfer data from the addressed instrument to the controller (sometimes called: enter, convert, image, iobuffer, on timeout, set timeout, transfer).

Overview of the Serial (RS-232) Bus

This feature is not implemented.

Serial interface programming techniques are similar to most general I/O applications. Refer to your programming language documentation for information on how to initiate the card and verify the status.

Due to the asynchronous nature of serial I/O operations, special care must be exercised to ensure that data is not lost by sending to another device before the device is ready to receive. Modem line handshaking can he used to help solve this problem. These and other topics are discussed in greater detail in your programming language documentation.

Settings for the Serial Interface

Please refer to the documentation on your computer and I/O to configure the serial bus. Some common serial interface configuration settings are:

Baud Rate to	9600
Bits per character to	8
Parity to	Odd and disabled
Stop bits to	1

Handshake and Baud Rate

To determine hardware operating parameters, you need to know the answer for each of the following questions about the peripheral device:

- Which of the following signal and control lines are actively used during communication with the peripheral?
  - — Data Set Ready (DSR)
  - — Clear to Send (CTS)
- What baud rate is expected by the peripheral?

Character Format Parameters

To define the character format, you must know the requirements of the peripheral device for the following parameters:

- Character Length: Eight data bits are used for each character, excluding start, stop, and parity bits.
- Parity Enable: Parity is disabled (absent) for each character.

- Stop Bits: One stop bit is included with each character.

Modem Line Handshaking

To use modem line handshaking for data transfer you would consider the following tasks:

1. Set Data Terminal Ready and Request-to-Send modem lines to active state.
2. Check Data Set Ready and Clear-to-Send modem lines to be sure they are active.
3. Send information to the interface and thence to the peripheral.
4. After data transfer is complete, clear Data Terminal Ready and Request-to-Send signals.

For ENTER operations:

1. Set Data Terminal Ready line to active state. Leave Request-to-Send inactive.
2. Check Data Set Ready and Data Carrier Detect modem lines to be sure they are active.
3. Input information from the interface as it is received from the peripheral.
4. After the input operation is complete, clear the Data Terminal Ready signal.

Data Transfer Errors

The serial interface can generate several types of errors when certain conditions are encountered while receiving data from the peripheral device. Errors can be generated by any of the following conditions:

- Parity error. The parity bit on an incoming character does not match the parity expected by the receiver. This condition is most commonly caused by line noise.
- Framing error. Start and stop bits do not match the timing expectations of the receiver. This can occur when line noise causes the receiver to miss the start bit or obscures the stop bits.
- Overrun error. Incoming data buffer overrun caused a loss of one or more data characters. This is usually caused when data is received by the interface, but no ENTER statement has been activated to input the information.
- Break received. A BREAK was sent to the interface by the peripheral device. The desktop computer program must be able to properly interpret the meaning of a break and take appropriate action.

3 Programming Examples

- The programming examples were written for use on an IBM compatible PC.
- The programming examples use C, Visual Basic, or VEE programming languages.
- The programming examples use VISA interfaces (GPIB, LAN, or USB).
- Some of the examples use the IVI-COM drivers.

  Interchangeable Virtual Instruments COM (IVI-COM) drivers: Develop system automation software easily and quickly. IVI-COM drivers take full advantage of application development environments such as Visual Studio using Visual Basic, C# or Visual C++ as well as Agilent's Test and Measurement Toolkit. You can now develop application programs that are portable across computer platforms and I/O interfaces. With IVI-COM drivers you do not need to have in depth test instrument knowledge to develop sophisticated measurement software. IVI-COM drivers provide a compatible interface to all. COM environments. The IVI-COM software drivers can be found at the URL:
  http://www.agilent.com/find/ivi-com
- Most of the examples are written in C, Visual Basic, VEE, or LabVIew using the Agilent VISA transition library.

  The Agilent I/O Libraries Suite must be installed and the GPIB card, USB to GPIB interface, or Lan interface USB interface configured. The latest Agilent I/O Libraries Suite is available:
  www.agilent.com/find/iolib
- The STATus subsystem of commands is used to monitor and query hardware status. These hardware registers monitor various events and conditions in the instrument. Details about the use of these commands and registers can be found in the manual/help in the Utility Functions section on the STATus subsystem.

Available Programing Examples

The following examples work with a E4406A VSA Series Transmitter Tester. These examples use one of the following programming languages: Visual Basic® 6, Visual Basic.NET®, MS Excel®, C++, ANSI C, C#.NET, and Agilent VEE Pro.

These examples are available in the "`progexamples`" directory on the Agilent Technologies E4406A VSA Series Transmitter Tester documentation CD-ROM. The file names for each example is listed at the end of the example description. The examples can also be found on the Agilent Technologies, Inc. web site at URL:

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

Programming using Visual Basic® 6, Visual Basic.NET® and MS Excel®:

- *Transfer Screen Images* from your E4406A VSA Series Transmitter Tester using Visual Basic 6

  The example program, screen_e4406a.bas, is written for the E4406A VSA Series Analyzer. It transfers the current screen image on the analyzer over GPIB or LAN and stores the image on your PC in the current directory from which you run the executable as screen.gif.

  File name: screen_e4406a_Dec03.exe

- *Binary Block Trace* data transfer from your E4406A VSA Series Transmitter Tester using Visual Basic 6

  This example program queries the IDN string from the instrument and then reads the trace data in Spectrum Analyzer mode in binary format (Real,32 or Real,64 or Int,32). The data is then stored to a file "`bintrace.txt`". This data transfer method is faster than the default ASCII transfer mode, because less data is sent over the bus.

  File name: bintrace_e4406a_Dec03.exe

- *IQ Trace Data (IVI-Com)* data transfer from your E4406A VSA Series Transmitter Tester using Visual Basic 6

  This example demonstrates how to use the IVI-COM driver in Visual Basic. NET. The program queries the raw I/Q trace data from the Basic Waveform measurement and outputs it to the PC's display.

  File name: VBIviComSA_BasicWaveform.vb

Programming using C++ or ANSI C and C#.NET:

- *Serial Poll for Alignment Complete* using C++ or ANSI C

  The example program demonstrates how to perform an alignment

and use serial polling to determine when the alignment is finished. This allows the user to perform an alignment and not tie up the GPIB bus while waiting for the alignment to complete.

File name: SerAlign.c

Programming using LabView™:

- *Transfer ASCII Trace Data* from my E4406A VSA Series Transmitter Tester using LabView™

  This example program does the following:

  — Opens a VXI 11.3 Lan connection to the instrument

  — Changes the data format to ASCII

  — Initiates a Power vs. Time measurement and reads the results

  — Converts the comma separated ASCII results string into an array of values

  File name: EPVT.exe

SCPI Remote Programming Examples

This section includes examples of how to program the instrument using the instrument SCPI programming commands. Most of the examples are written for a PC, using GPIB. They are written in C using the Agilent VISA transition library.

The Agilent I/O Libraries Suite must be installed and the GPIB card, USB to GPIB interface, or Lan interface USB interface configured. The latest Agilent I/O Libraries Suite is available at the following URL: www.agilent.com/find/iolib

These examples are available on the Agilent Technologies E4406A documentation CD-ROM.

The section "Programming in C Using the VTL" on page 133, includes some basic information about using the C programming language. That information can be used with the examples in this chapter to create your own measurement routines.

Examples are also available showing you how to program the instrument using the VXI plug & play instrument driver that is provided. Examples are included in the on-line documentation in the driver itself. The driver allows you to use several different programming languages including: VEE, LabView, C, C++, and BASIC. The software driver can be found at the URL `http://www.agilent.com/find/vsa`.

The programming examples include:

Using Markers

This is the C programming example Markers.c.

```
/************************************************************************
*Markers.c
*Agilent Technologies 2001
*
*E4406A VSA Series Transmitter Tester using VISA for I/O
*The C program does the following:
*Open session to GPIB device at address 18
*Check opening session success
*set the instrument to Basic Mode
*Preset the instrument
*Set the input port to the internal 50Mhz reference source
*Tune the analyzer to 50MHZ
*Put the analyzer in a single mode
*Zoom the spectrum display
*Trigger a spectrum measurement
*Poll the operation complete query
*Assign marker 1 to the average trace of the spectrum
*Put the marker 1 on the signal peak
*Query the 50 MHz signal amplitude
*Get the 50 MHz signal amplitude
*Assign marker 2 to the average trace of the spectrum
*Assign the marker function NOISE to marker 2
*Position marker 2 on the noise floor
*Query NOISE marker
*Get the NOISE marker reading
*Put the analyzer back to continuous mode
*Calculate the difference between the marker peak and the NOISE marker
*Print result to the standard output
*Close session
************************************************************************/
#include <stdio.h>
#include <stdlib.h>
```

```
#include <math.h>
#include "visa.h"

void main ()
{
/*program variables*/
ViSession defaultRM, viVSA;
ViStatus viStatus= 0;
double dPeakPower= 0;
double dNoiseMarker = 0;
double dResult= 0;
 long lComplete= 0;
 /*open session to GPIB device at address 18 */
viStatus=viOpenDefaultRM (&defaultRM);
viStatus=viOpen (defaultRM, "GPIB0::18::INSTR", VI_NULL,VI_NULL, &viVSA);
/*check opening session sucess*/
if(viStatus)
{
printf("Could not open a session to GPIB device at address 18!\n");
exit(0);
}
    /*set the instrument to Basic Mode*/
viPrintf(viVSA, "INST BASIC\n");
/*Preset the instrument */
 viPrintf(viVSA, "*RST\n");
/*set the input port to the internal 50Mhz reference source*/
viPrintf(viVSA, "SENS:FEED AREF\n");
/*tune the analyzer to 50MHZ*/
viPrintf(viVSA, "SENS:FREQ:CENT 50E6\n");
/*put the analyzer in a single mode*/
viPrintf(viVSA, "INIT:CONT 0\n");
/*zoom the spectrum display*/
viPrintf(viVSA, "DISP:FORM:ZOOM1\n");
/*trigger a spectrum measurement*/
viPrintf(viVSA, "INIT:IMM;*OPC?\n");
/*poll the operation complete query*/
```

```
while (!lComplete)
viScanf (viVSA,"%d",&lComplete);
 /*assign marker 1 to the average trace of the spectrum*/
viPrintf(viVSA, "CALC:SPEC:MARK1:TRAC ASP\n");
/*put the marker 1 on the signal peak*/
viPrintf(viVSA, "CALC:SPEC:MARK1:MAX\n");
/*query the 50 MHz signal amplitude*/
viPrintf(viVSA, "CALC:SPEC:MARK1:Y?\n");
/*get the the 50 MHz signal amplitude*/
viScanf (viVSA,"%lf",&dPeakPower);
/*assign marker 2 to the average trace of the spectrum*/
viPrintf(viVSA, "CALC:SPEC:MARK2:TRAC ASP\n");
/*assign the marker function NOISE to marker 2 */
viPrintf(viVSA, "CALC:SPEC:MARK2:FUNC NOISE\n");
/*position marker 2 on the noise floor*/
viPrintf(viVSA, "CALC:SPEC:MARK2:X 50.2E6\n");
/*query NOISE marker*/
viPrintf(viVSA, "CALC:SPEC:MARK2:FUNC:RES?\n");
/*get the the NOISE marker reading*/
viScanf (viVSA,"%lf",&dNoiseMarker);
/*put the analyzer back to continuous mode*/
viPrintf(viVSA, "INIT:CONT 1\n");
/*calculate the difference between the marker peak and the NOISE marker*/
dResult = fabs(dNoiseMarker - dPeakPower);
/*print result to the standart output*/
printf("The Peak Marker measured = %.2lf dBm\n",dPeakPower);
printf("The Noise Marker at 50.2 MHz measured = %.2lf dBm/Hz\n",dNoiseMarker);
printf("The difference between the Peak and the Noise Floor = %.2lf
dBc/Hz\n\n",dResult);
/* close session */
viClose (viVSA);
viClose (defaultRM);
}
```

Saving Binary Trace Data in an ASCII File

This is the C programming example Trace.c

```
/*************************************************************************
*Trace.c
*Agilent Technologies 2001
*
*E4406A VSA Series Transmitter Tester using VISA for I/O
*This Program shows how to get and save a binary trace data
*
*Set up VSA commands so they can be done FAST (all in one transaction).
*Reset the device and clear status.
*Set the input port to the internal 50MHz reference source.
*Zoom the spectrum display and tune the analyzer to 50MHz.
*Set the ouput format to a binary format.
*Set binary byte order to SWAP (for PC).
*Trigger a spectrum measurement and fetch the trace data.
*Open a session to GPIB device at address 18.
*Get the number of bytes in length of postceeding trace data.
*Put this in sBuffer.
*Put the trace data into sBuffer and convert it from ASCII to integer.
*Calculate the number of points given the number of byte in the trace.
*REAL 64 binary format means each number is represented by 8 bytes.
*Save trace data to an ASCII file.
*Close the session.
*
*************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <memory.h>
#include "visa.h"

void main ()
{
/*program variable*/
```

```
ViSession defaultRM, viVSA;
ViStatus viStatus= 0;
char sBuffer[80]= {0};
FILE *fTraceFile;
long lNumberPoints= 0;
long lNumberBytes= 0;
long lLength= 0;
long i = 0;
unsigned long lBytesRetrieved;
ViReal64 adTraceArray[10240];

char *vsaSetup = /* setup commands for VSA */
"*RST;*CLS;"/* Reset the device and clear status */
":FEED AREF;"/* set the input port to the internal
                        50MHz reference source*/
":DISP:FORM:ZOOM1;"/* zoom the spectrum display*/
":FREQ:CENT 50E6;"/* tune the analyzer to 50MHz*/
":FORM REAL,64;"/* Set the ouput format to a binary format*/
":FORM:BORD SWAP;"/* set the binary byte order to SWAP (for PC)*/
":INIT:IMM;"/* trigger a spectrum measurement*/
":FETC:SPEC7?";/* fetch the spectrum trace data*/

  /*open session to GPIB device at address 18 */
viStatus=viOpenDefaultRM (&defaultRM);
viStatus=viOpen (defaultRM, "GPIB0::18::INSTR", VI_NULL,VI_NULL, &viVSA);
/*check opening session sucess*/
if(viStatus)
{
printf("Could not open a session to GPIB device at address 18!\n");
exit(0);
}

/* Set I/O timeout to five seconds */
 viSetAttribute(viVSA,VI_ATTR_TMO_VALUE,5000);

/* Send setup commands to instrument */
```

```
 viPrintf(viVSA,"%s\n",vsaSetup);

/*print message to the standard output*/
printf("Getting the spectrum trace in binary format...\nPlease wait...\n\n");

/* get number of bytes in length of postceeding trace data
   and put this in sBuffer*/
viRead (viVSA,(ViBuf)sBuffer,2,&lBytesRetrieved);

/* Put the trace data into sBuffer */
viRead (viVSA,(ViBuf)sBuffer,sBuffer[1] - '0',&lBytesRetrieved);

/* append a null to sBuffer */
sBuffer[lBytesRetrieved] = 0;

/* convert sBuffer from ASCII to integer */
lNumberBytes = atoi(sBuffer);

/*calculate the number of points given the number of byte in the trace
REAL 64 binary format means each number is represented by 8 bytes*/
lNumberPoints = lNumberBytes/sizeof(ViReal64);

/*get and save trace in data array */
viRead (viVSA,(ViBuf)adTraceArray,lNumberBytes,&lBytesRetrieved);

    /* read the terminator character and discard */
viRead (viVSA,(ViBuf)sBuffer,1, &lLength);

/* loop until all errors read */
do
{
viPrintf (viVSA,"SYST:ERR?\n");  /* check for errors */
viRead (viVSA,(ViBuf)sBuffer,80,&lLength);/* read back last error message */
sBuffer[lLength] = 0;                     /* append a null to byte count */
printf("%s\n",sBuffer);                   /* print error buffer to display */
} while (sBuffer[1] != '0');
```

```
/*set the analyzer to continuous mode for manual use */
viPrintf(viVSA, "INIT:CONT 1\n");

/*save trace data to an ASCII file*/
fTraceFile=fopen("C:\\Trace.txt","w");
fprintf(fTraceFile,"Trace.exe Output\nAgilent Technologies 2001\n\n");
fprintf(fTraceFile,"List of %d points of the averaged spectrum
trace:\n\n",lNumberPoints);
for (i=0;i<lNumberPoints;i++)
fprintf(fTraceFile,"\tAmplitude of point[%d] = %.2lf dBm\n",i+1,adTraceArray[i]);
fclose(fTraceFile);
/*print message to the standard output*/
printf("The %d trace points were saved to C:\\Trace.txt file\n\n",lNumberPoints);
 /* Close session */
viClose (viVSA);
viClose (defaultRM);
}
```

Saving ASCII Trace Data in an ASCII File

This is the C programming example TraceASC.c

```
/*************************************************************************
*TraceASC.c
*Agilent Technologies 2001
*
*E4406A VSA Series Transmitter Tester using VISA for I/O
*This Program shows how to get and save an ASCII trace
*The C program does the following:
*Open session to GPIB device at address 18
*Check opening session success
*Set the instrument to Basic Mode
*Reset device
*Set the input port to the internal 50MHz reference source
*Zoom the spectrum display
*Tune the analyzer to 50MHz
*Set the analyzer in single mode
*Trigger a spectrum measurement and wait for it to complete
*Query the spectrum trace information
*Save the info trace to buffer
*Query the spectrum trace data
*Save the spectrum trace data to buffer
*Set the analyzer back to continuous mode
*Save trace data to an ASCII file
*Close session
*************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <memory.h>
#include "visa.h"

void main ()
{
/*program variable*/
```

```
ViSession defaultRM, viVSA;
ViStatus viStatus= 0;
char sTraceInfo  [256]= {0};
char sTraceBuffer[1024*100]= {0};
FILE *fTraceFile;
long lComplete= 0;
unsigned long lBytesRetrieved;
  /*open session to GPIB device at address 18 */
viStatus=viOpenDefaultRM (&defaultRM);
viStatus=viOpen (defaultRM, "GPIB0::18::INSTR", VI_NULL,VI_NULL, &viVSA);
/*check opening session sucess*/
if(viStatus)
{
printf("Could not open a session to GPIB device at address 18!\n");
exit(0);
}
/* Set the instrument to basic mode */
 viPrintf(viVSA, "INST BASIC\n");
/* Reset device */
 viPrintf(viVSA, "*RST\n");
/*set the input port to the internal 50MHz reference source*/
viPrintf(viVSA, "SENS:FEED AREF\n");
/*zoom the spectrum display*/
viPrintf(viVSA, "DISP:FORM:ZOOM1\n");
/*tune the analyzer to 50MHz*/
viPrintf(viVSA, "SENS:FREQ:CENT 50E6\n");
/*print message to the standard output*/
printf("Getting the spectrum trace in ASCII format...\nPlease wait...\n\n");
/*set the analyzer in single mode*/
viPrintf(viVSA, "INIT:CONT 0\n");
/*trigger a  spectrum measurement and wait for it to complete*/
viPrintf(viVSA, "INIT:IMM;*WAI\n");
  /*query the spectrum trace information*/
viPrintf(viVSA, "FETCH:SPEC1?\n");
/*save the info trace to buffer*/
viRead (viVSA,(ViBuf)sTraceInfo,256,&lBytesRetrieved);
```

```
  /*query the spectrum trace data*/
viPrintf(viVSA, "FETCH:SPEC7?\n");
/*save the spectrum trace data to buffer*/
viRead (viVSA,(ViBuf)sTraceBuffer,1024*100,&lBytesRetrieved);
/*set the analyzer back to continuous mode*/
viPrintf(viVSA, "INIT:CONT 1\n");
/*save trace data to an ASCII file*/
fTraceFile=fopen("C:\\TraceASC.txt","w");
fprintf(fTraceFile,"TraceASC.exe Output\nAgilent Techonologies 2001\n\n");
fprintf(fTraceFile,"Please refer to the PROGRAMMER'S GUIDE to read about:
FETCH:SPEC[n]\n\n");
 fprintf(fTraceFile,"The trace information: n=1\n----------------------------\n");
fprintf(fTraceFile,sTraceInfo);
fprintf(fTraceFile,"\n\nThe averaged spectrum trace data:
n=7\n----------------------------\n\n");
fprintf(fTraceFile,sTraceBuffer);
 fprintf(fTraceFile,"\n----------------------------\nEnd of the trace data");
fclose(fTraceFile);
/*print message to the standard output*/
printf("The spectrum information was saved to C:\\TraceASC.txt file\n\n");
 /* Close session */
viClose (viVSA);
viClose (defaultRM);
}
```

Saving and Recalling Instrument State Data

This is the C programming example State.c

```
/**************************************************************************
*State.c
*Agilent Technologies 2001
*
*HPE4406A VSA Series Transmitter Tester using VISA for I/O
*This program shows how to save and recall a state of the instrument
*The C program does the following:
*Open session to GPIB device at address 18
*Check opening session success
*Reset the instrument
*Set the input port to the internal 50Mhz reference source
*Zoom the spectrum display
*Tune the analyzer to 50MHZ
*Change the resolution bandwidth
*Change the Y Axis Scale/Div
*Change the display reference level
*Trigger the instrument and wait for it to complete
*Save this state in register 10! Careful this will overwrite register 10
*Display message
*Wait for any key to be pressed
*Reset the instrument
*Set again the input port to the internal 50Mhz reference source
*Display message
*Wait for any key to be pressed
*Recall the state saved in register 10
*Zoom the spectrum display
*Display message
*Wait for any key to be pressed
*Reset the instrument
*Set the instrument to continuous sweep
*Close session
**************************************************************************/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include "visa.h"

void main ()
{
/*program variables*/
ViSession defaultRM, viVSA;
ViStatus viStatus= 0;
 /*open session to GPIB device at address 18 */
viStatus=viOpenDefaultRM (&defaultRM);
viStatus=viOpen (defaultRM, "GPIB0::18::INSTR", VI_NULL,VI_NULL, &viVSA);
/*check opening session sucess*/
if(viStatus)
{
printf("Could not open a session to GPIB device at address 18!\n");
exit(0);
}
/*set the instrument to basic mode*/
 viPrintf(viVSA, "INST BASIC\n");
/*reset the instrument */
 viPrintf(viVSA, "*RST\n");
/*set the input port to the internal 50Mhz reference source*/
viPrintf(viVSA, "SENS:FEED AREF\n");
/*zoom the spectrum display*/
viPrintf(viVSA, "DISP:FORM:ZOOM1\n");
/*tune the analyzer to 50MHZ*/
viPrintf(viVSA, "SENS:FREQ:CENT 50E6\n");
 /*change the resolution bandwidth*/
viPrintf(viVSA, "SENS:SPEC:BAND:RES 100E3\n");
/*change the Y Axis Scale/Div*/
viPrintf(viVSA, "DISP:SPEC:WIND:TRAC:Y:SCAL:PDIV 5\n");
/*Change the display refernece level*/
viPrintf(viVSA, "DISP:SPEC:WIND:TRAC:Y:SCAL:RLEV -15\n");
/*trigger the instrument*/
```

```
viPrintf(viVSA, "INIT:IMM;*WAI\n");
/*save this state in register 10.
!!!Carefull this will overwrite register 10*/
viPrintf(viVSA, "*SAV 10\n");
/*display message*/
printf("HPE4406A Programming example showing *SAV,*RCL SCPI commands\n");
printf("used to save instrument state\n\t\t------------------------");
printf("\n\nThe instrument state has been saved to an internal register\n");
printf("Please observe the display and notice the signal shape\n");
printf("Then press any key to reset the
instrument\a\n\t\t------------------------");
/*wait for any key to be pressed*/
getch();
/*reset the instrument */
 viPrintf(viVSA, "*RST\n");
/*set again the input port to the internal 50Mhz reference source*/
viPrintf(viVSA, "SENS:FEED AREF\n");
/*display message*/
 printf("\n\nThe instrument was reset to the factory default setting\n");
printf("Notice the abscence of the signal on the display\n");
 printf("Press any key to recall the saved
state\a\n\t\t------------------------");
/*wait for any key to be pressed*/
getch();
/*recall the state saved in register 10*/
viPrintf(viVSA, "*RCL 10\n");
/*zoom the spectrum display*/
viPrintf(viVSA, "DISP:FORM:ZOOM1\n");
/*display message*/
printf("\n\nNotice the previous saved instrument settings were restored\n");
 printf("Press any key to terminate the
program\a\n\t\t------------------------\n\n");
/*wait for any key to be pressed*/
getch();
/*reset the instrument */
 viPrintf(viVSA, "*RST;*wai\n");
/*Set the instrument to continuous sweep */
```

```
 viPrintf(viVSA, "INIT:CONT 1\n");
/* close session */
viClose (viVSA);
viClose (defaultRM);
}
```

Performing Alignments and Getting Pass/Fail Results

This is the C programming example Align.c

```
 /************************************************************
*Align.c
*Agilent Technologies 2001
*
*E4406A VSA Series Transmitter Tester using VISA for I/O
*The C program does the following:
*Open session to GPIB device at address 18
*Increase timeout to 75 sec
*Lock out front-panel keypad control
*Reset the analyzer
*Auto-align the analyzer
*Check for alignment success
*Alignment succeeds if query result is zero (0)
*Print success/failure message to standard output
*Set the Analzyer to Continuous Sweep
*Unlock the front-panel keypad
*Reset timeout to 2 sec
*Close session
************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include "visa.h"

void main ()
{
/*program variables*/
ViSession defaultRM, viVSA;
ViStatus viStatus=0;
long lCalStatus=0;
/*open session to GPIB device at address 18 */
viStatus=viOpenDefaultRM (&defaultRM);
```

```
viStatus=viOpen (defaultRM, "GPIB0::18::INSTR", VI_NULL,VI_NULL, &viVSA);
/*check opening session sucess*/
if(viStatus)
{
printf("Could not open a session to GPIB device at address 18!\n");
exit(0);
}
/*increase timeout to 75 sec*/
viSetAttribute(viVSA,VI_ATTR_TMO_VALUE,75000);
/*Lock out the front-panel keypad*/
viPrintf(viVSA, "SYST:KLOCK 1\n");
/*reset the analyzer*/
viPrintf(viVSA, "*RST\n");
/*print message*/
printf("The auto-alignement is in progress...\nPlease wait...\n\n");
 /*auto-align the analyzer*/
viPrintf(viVSA, "CAL?\n");
/*check for alignement success*/
viScanf (viVSA,"%d",&lCalStatus);
/*alignement succeeds if query result is zero(0)*/
if (!lCalStatus)
/*print success message to standard output*/
printf("The analyzer auto-alignement was successful!\n\n");
else
/*print failure message to standard output*/
printf("The analyzer auto-alignement was not successful!\n\n");
 /*Set the Analyzer to Continuous Sweep*/
viPrintf(viVSA, "INIT:CONT 1\n");
/*Unlock the front-panel keypad*/
viPrintf(viVSA, "SYST:KLOCK 0\n");
/*reset timeout to 2 sec*/
viSetAttribute(viVSA,VI_ATTR_TMO_VALUE,3000);
/* Close session */
viClose (viVSA);
viClose (defaultRM);
}
```

Making an ACPR Measurement in cdmaOne (Option BAC)

This is the C programming example ACPR.c

```
/**************************************************************************
*ACPR.c
*Agilent Technologies 2001
*
*E4406A VSA Series Transmitter Tester using VISA for I/O
*This Program shows how to do an ACPR measurement and get the data
*   Note: You can do this measurement in Basic Mode by changing the
*         INST CDMA command to INST BASIC
*   This C program does the following:
*         Open session to GPIB device at address 18
*  check opening session sucess
*     increase timeout to 60 sec
*     print message to the standard output
*  switch to CDMA MODE
*  Reset device
*  set the analyzer in single mode
*  trigger an ACPR measurement and wait for it to complete
*  Get the data into a buffer
*  set the analyzer in continuous mode
*  save the data to a file
*  print message to the standard output
*  Close session
**************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include "visa.h"

void main ()
{
/*program variable*/
ViSession defaultRM, viVSA;
```

```
ViStatus viStatus    = 0;
char sTraceInfo  [1024]= {0};
FILE *fTraceFile;
 unsigned long lBytesRetrieved;
  /*open session to GPIB device at address 18 */
viStatus=viOpenDefaultRM (&defaultRM);
viStatus=viOpen (defaultRM, "GPIB0::18::INSTR", VI_NULL,VI_NULL, &viVSA);
/*check opening session sucess*/
if(viStatus)
{
printf("Could not open a session to GPIB device at address 18!\n");
exit(0);
}
/*increase timeout to 60 sec*/
viSetAttribute(viVSA,VI_ATTR_TMO_VALUE,60000);
/*print message to the standard output*/
printf("Getting ACPR measurement results...\nPlease wait...\n\n");
//switch to CDMA MODE
viPrintf(viVSA, "INST CDMA\n");
/* Reset device */
 viPrintf(viVSA, "*RST\n");
/*set the analyzer in single mode*/
viPrintf(viVSA, "INIT:CONT 0\n");
/*trigger an ACPR measurement*/
viPrintf(viVSA, "READ:ACPR?;*WAI\n");
 /*Get the data into a buffer*/
viRead (viVSA,(ViBuf)sTraceInfo,1024,&lBytesRetrieved);
/*set the analyzer in continuous mode*/
viPrintf(viVSA, "INIT:CONT 1\n");
 /*save the data to a file*/
fTraceFile=fopen("C:\\ACPR.txt","w");
fprintf(fTraceFile,"ACPR.exe Output\nAgilent Technnologies 2001\n\n");
 fprintf(fTraceFile,"The ACPR Measurement
Result\n---------------------------\n");
fprintf(fTraceFile,sTraceInfo);
 fclose(fTraceFile);
```

```
/*print message to the standard output*/
printf("The The ACPR Measurement Result was saved to C:\\ACPR.txt file\n\n");
 /* Close session */
viClose (viVSA);
viClose (defaultRM);
}
```

Making a Power Calibration for a GSM Mobile Handset

This C programming example (powercal.c) can be found on the Documentation CD.

This program uses Basic mode which is optional -B7J- in the PSA Series spectrum analyzers and is standard in the E4406A Vector Signal Analyzer. It uses the Waveform measurement with the `CALC:DATA2:COMP? DME` command to return the power of 75 consecutive GSM/EDGE bursts. The DME (dB Mean) parameter returns the average of the dB trace values. The DME parameter is only available in later version of instrument firmware ≥ A.05.00 for PSA and ≥ A.07.00 for VSA.

This program also demonstrates how to serial poll the “Waiting for Trigger” status bit to determine when to initiate the GSM phone. The data results are placed in an ASCII file (powercal.txt).

The program can also be adapted to perform W-CDMA Downlink Power Control measurements in the code domain power `Symbol Power` view. In essence, you can average any stepped power measurement trace using this method.

Example:

```
/*****************************************************************************
*     powercal.c
*     Agilent Technologies 2003
*
*     This program demonstrates the process of using the Waveform
*     measurement and the CALC:DATA2:COMP? DME command to return the power
*     of 75 consecutive GSM/EDGE bursts.
*     The DME (db Mean) parameter returns the average of the dB trace values.
*
*     This program also demonstrates how to serial poll the "Waiting
*     for Trigger" Status bit to determine when to initiate the GSM phone
*     The data results are placed in an ASCII file, powercal.txt
*
*     This program can also be adapted to perform W-CDMA Downlink PowerControl
*     measurements in the Code Domain Power Symbol Power View.  In essence,
*     you can average any stepped power measurement trace using this method.
```

```
*
*     Instrument Requirements:
*        E444xA with option B7J and firmware version >= A.05.00 or
*        E4406A with firmware version >= A.07.00 or
*
*     Signal Source Setup:
*        Set up GSM/EDGE frame for either 1, 2, 4, or eight slots per frame.
*        When configuring two slots per frame, turn on slots 1 and 5
*        When configuring four slots per frame, turn on slots 1,3,5, and 7.
*        Set frame repeat to Single.
*        Set the signal amplitude to -5 dBm.
*        Set the signal source frequency to 935.2 MHz
*
*     CALC:DATA2:COMP? DME parameters:
*        soffset = 25us (This avoids averaging data points when the burst
*                        is transitioning on.)
*        length = 526us (Period over which the power of the burst is averaged)
*        roffset = 4.6153846 ms / slots per frame (Repitition interval of burst)
*******************************************************************************
****/
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include <math.h>
#include "c:\program files\visa\winnt\include\visa.h"

void main ()
{
/*program variable*/
ViSession defaultRM, viVSA;
ViStatus viStatus= 0;
ViUInt16 stb;
FILE *fDataFile;
long lthrowaway,lbursts;
long lNumberPoints= 0;
long lNumberBytes= 0;
```

```
long lLength     = 0;
long i              = 0;
long lOpc= 0L;
double sweeptime    = 0;
double burstinterval= 0;
unsigned long lBytesRetrieved;
ViReal64 adDataArray[100];
char sBuffer[80]= {0};

char *basicSetup = // measurement setup commands for VSA/PSA
":INST:SEL BASIC;"// Put the instrument in Basic Mode
        "*RST;"// Preset the instrument
"*CLS;"//Clear the status byte
":STAT:OPER:ENAB 32;"       //Enable Status Operation
":DISP:ENAB 0;"// Turn the Display off (improves Speed)
":FORM REAL,64;"// Set the ouput format to binary
":FORM:BORD SWAP;"// set the binary byte order to SWAP (for PC)
":CONF:WAV;"// Changes measurement to Waveform
":INIT:CONT 0;"// Puts instrument in single measurement mode
":CAL:AUTO OFF;"//Turn auto align off
":FREQ:CENTER 935.2MHz;"//Set Center Freq to 935.2MHz
":WAV:ACQ:PACK MED;"//Set DataPacking to Medium
":WAV:BAND:TYPE FLAT;"//Select Flattop RBW Filter
":WAV:DEC:FACT 4;"//Set Decimation Factor to 4
":WAV:DEC:STAT ON;"//Turn Decimation On
":DISP:WAV:WIND1:TRAC:Y:RLEV 5;" //Set referance level to 5 dBm
":WAV:BWID:RES 300kHz;"//Set Res bandwith filter to 300kHz
":POW:RF:ATT 5;"//Set 5dB of internal attenuation
":WAV:ADC:RANG P0;"//Set ADC Range to P0, This is
//necessary to prevent autoranging
":WAV:TRIG:SOUR IF;"//Set Trigger source to IF burst
":TRIG:SEQ:IF:LEV -20;";//Set IF Trig level to -20dB

/*open session to GPIB device at address 18 */
viStatus=viOpenDefaultRM (&defaultRM);
viStatus=viOpen (defaultRM, "GPIB0::18", VI_NULL,VI_NULL,&viVSA);
```

```
/*check opening session sucess*/
if(viStatus)
{
printf("Could not open a session to GPIB device at address 18!\n");
exit(0);
}

/* Set I/O timeout to ten seconds */
 viSetAttribute(viVSA,VI_ATTR_TMO_VALUE,10000);

viClear(viVSA);//send device clear to instrument

/*print message to the standard output*/
printf("Enter number of bursts per frame (1,2,4 or 8): ");
scanf( "%ld",&lbursts);

/* Send setup commands to instrument */
 viPrintf(viVSA,"%s\n",basicSetup);

/* Calculate sweep time and set it*/
burstinterval = 4.6153846 / 1000.00 / lbursts;
sweeptime= burstinterval * 75.0;

viPrintf(viVSA,":WAV:SWE:TIME %fs\n",sweeptime);

/* Clear status event register */
viQueryf(viVSA,"STAT:OPER:EVENT?\n","%ld",&lthrowaway);

/* Initiate the waveform measurement and get the instrument ready
   to calculate the mean RMS I/Q voltage in each burst
   (We will convert these discreate values into Mean dBm Power values) */

viPrintf(viVSA,"INIT:IMM\n");

/* Serial poll the instrument to determine when it is waiting for
```

```
   trigger and the GSM phone can be told to send its 75 bursts. */

while(1)
{
viReadSTB(viVSA,&stb);  //read status byte
if (stb & 128) break;   //look for "waiting for trigger" bit
printf("Waiting on Analyzer...\n");
Sleep (50); // wait 50 ms between each serial poll
}
/*print message to the standard output*/
printf("Analyzer is Ready!\n\nWaiting for phone to trigger...\n\n");

/*Querry for Operation Complete */
viQueryf(viVSA,"*OPC?\n", "%d", &lOpc);

    /*Use the CALC:DATA0:COMP command to return the average power in each burst*/
viPrintf(viVSA,":CALC:DATA2:COMP? DME,25E-6,526E-6,%f\n",burstinterval);

/* get number of bytes in length of postceeding data and put this in sBuffer*/
viRead (viVSA,(ViBuf)sBuffer,2,&lBytesRetrieved);
printf("Getting the burst data in binary format...\n\n");

/* Put the returned data into sBuffer */
viRead (viVSA,(ViBuf)sBuffer,sBuffer[1] - '0',&lBytesRetrieved);

/* append a null to sBuffer */
sBuffer[lBytesRetrieved] = 0;

/* convert sBuffer from ASCII to integer */
lNumberBytes = atoi(sBuffer);

/*calculate the number of returned values given the number of bytes.
      REAL 64 binary data means each number is represented by 8 bytes */
 lNumberPoints = lNumberBytes/sizeof(ViReal64);

/*get and save returned data into an array */
```

```
viRead (viVSA,(ViBuf)adDataArray,lNumberBytes,&lBytesRetrieved);

/* read the terminator character and discard */
viRead (viVSA,(ViBuf)sBuffer,1, &lthrowaway);

/*print message to the standard output*/
printf("Querying instrument to see if any errors in Queue.\n");

/* loop until all errors read */
do
{
   viPrintf (viVSA,"SYST:ERR?\n");           /* check for errors */
   viRead (viVSA,(ViBuf)sBuffer,80,&lLength);/* read back last error message */
   sBuffer[lLength] = 0;                     /* append a null to byte count */
   printf("%s\n",sBuffer);                   /* print error buffer to display */
} while (sBuffer[1] != '0');

/* Turn the Display of the instrument back on */
viPrintf(viVSA,"DISP:ENAB 1\n");

/*save result data to an ASCII file*/
fDataFile=fopen("powercal.txt","w");
fprintf(fDataFile,"powercal.exe Output\nAgilent Technologies 2003\n\n");
fprintf(fDataFile,"Power of %d GSM/EDGE bursts:\n",lNumberPoints);
fprintf(fDataFile,"(%d burst(s) per frame):\n\n",lbursts);
for (i=0;i<lNumberPoints;i++)
{
fprintf(fDataFile,"\tPower of burst[%d] = %.2lf dBm\n",i+1,adDataArray[i]);
}
fclose(fDataFile);
/*print message to the standard output*/
printf("The %d burst powers were saved to powercal.txt file.\n\n",lNumberPoints);

viClose (viVSA);
viClose (defaultRM);
}
```

Using C Programming Over Socket LAN

This is the C programming example socketio.c. It demonstrates simple socket programming. It is written in C, and compiles in the HP-UX UNIX environment, or the WIN32 environment. It is portable to other UNIX environments with only minor changes.

In UNIX, LAN communication via sockets is very similar to reading or writing a file. The only difference is the open Socket() routine, which uses a few network library routines to create the TCP/IP network connection. Once this connection is created, the standard fread() and fwrite() routines are used for network communication.

In Windows, the routines send() and recv() must be used, since fread() and fwrite() may not work on sockets.

The program reads the analyzer's host name from the command line, followed by the SCPI command. It then opens a socket to the analyzer, using port 5025, and sends the command. If the command appears to be a query, the program queries the analyzer for a response, and prints the response.

This example program can also be used as a utility to talk to your analyzer from the command prompt on your UNIX workstation or Windows 95 PC, or from within a script.

This program is also available on your documentation CD ROM.

```
/*************************************************************************
*  $Header: lanio.c,v 1.5 96/10/04 20:29:32 roger Exp $
*  $Revision: 1.5 $
*  $Date: 96/10/04 20:29:32 $
*
*  $Contributor:     LSID, MID $
*
*  $Description:     Functions to talk to an Agilent E4406A transmitter
*                    tester via TCP/IP.  Uses command-line arguments.
*
*                    A TCP/IP connection to port 5025 is established and
*                    the resultant file descriptor is used to "talk" to the
*                    instrument using regular socket I/O mechanisms. $
*
*
*
```

```
*  E4406A Examples:
*
*   Query the center frequency:
*        lanio 15.4.43.5 'sens:freq:cent?'
*
* Query X and Y values of marker 1 and marker 2 (assumes they are on):
*        lanio my4406 'calc:spec:mark1:x?;y?; :calc:spec:mark2:x?;y?'
*
*   Check for errors (gets one error):
*        lanio my4406 'syst:err?'
*
*   Send a list of commands from a file, and number them:
*        cat scpi_cmds | lanio -n my4406
*
*************************************************************************
*
*  This program compiles and runs under
*     - HP-UX 10.20 (UNIX), using HP cc or gcc:
*          + cc -Aa    -O -o lanio  lanio.c
*          + gcc -Wall -O -o lanio  lanio.c
*
*     - Windows 95, using Microsoft Visual C++ 4.0 Standard Edition
*     - Windows NT 3.51, using Microsoft Visual C++ 4.0
*          + Be sure to add  WSOCK32.LIB  to your list of libraries!
*          + Compile both lanio.c and getopt.c
*          + Consider re-naming the files to lanio.cpp and getopt.cpp
*
*  Considerations:
*     - On UNIX systems, file I/O can be used on network sockets.
*       This makes programming very convenient, since routines like
*       getc(), fgets(), fscanf() and fprintf() can be used.  These
*       routines typically use the lower level read() and write() calls.
*
*     - In the Windows environment, file operations such as read(), write(),
*       and close() cannot be assumed to work correctly when applied to
*       sockets.  Instead, the functions send() and recv() MUST be used.
```

```
 */

/* Support both Win32 and HP-UX UNIX environment */
#ifdef _WIN32     /* Visual C++ 4.0 will define this */
#  define WINSOCK
#endif

#ifndef WINSOCK
#  ifndef _HPUX_SOURCE
#  define _HPUX_SOURCE
#  endif
#endif

#include <stdio.h>         /* for fprintf and NULL  */
#include <string.h>        /* for memcpy and memset */
#include <stdlib.h>        /* for malloc(), atol() */
#include <errno.h>         /* for strerror          */

#ifdef WINSOCK

#include <windows.h>

#  ifndef _WINSOCKAPI_
#  include <winsock.h>   // BSD-style socket functions
#  endif

#else /* UNIX with BSD sockets */

#  include <sys/socket.h>    /* for connect and socket*/
#  include <netinet/in.h>    /* for sockaddr_in       */
#  include <netdb.h>         /* for gethostbyname     */

#  define SOCKET_ERROR (-1)
#  define INVALID_SOCKET (-1)

   typedef  int SOCKET;
```

```
#endif /* WINSOCK */

#ifdef WINSOCK
  /* Declared in getopt.c.  See example programs disk. */
  extern char *optarg;
  extern int  optind;
  extern int getopt(int argc, char * const argv[], const char* optstring);
#else
#  include <unistd.h>           /* for getopt(3C) */
#endif

#define COMMAND_ERROR  (1)
#define NO_CMD_ERROR  (0)

#define SCPI_PORT  5025
#define INPUT_BUF_SIZE (64*1024)


/*************************************************************************
 * Display usage
 ************************************************************************/
static void usage(char *basename)
{
    fprintf(stderr,"Usage: %s [-nqu] <hostname> [<command>]\n", basename);
    fprintf(stderr,"       %s [-nqu] <hostname> < stdin\n", basename);
    fprintf(stderr,"  -n, number output lines\n");
    fprintf(stderr,"  -q, quiet; do NOT echo lines\n");
    fprintf(stderr,"  -e, show messages in error queue when done\n");
}



#ifdef WINSOCK
int init_winsock(void)
{
```

```
    WORD wVersionRequested;
    WSADATA wsaData;
    int err;
    wVersionRequested = MAKEWORD(1, 1);
    wVersionRequested = MAKEWORD(2, 0);

    err = WSAStartup(wVersionRequested, &wsaData);

    if (err != 0) {
        /* Tell the user that we couldn't find a useable */
        /* winsock.dll.     */
        fprintf(stderr, "Cannot initialize Winsock 1.1.\n");
        return -1;
    }
    return 0;
}

int close_winsock(void)
{
    WSACleanup();
    return 0;
}
#endif /* WINSOCK */



/*****************************************************************************
 *
 > $Function: openSocket$
 *
 * $Description:  open a TCP/IP socket connection to the instrument $
 *
 * $Parameters:  $
 *    (const char *) hostname . . . . Network name of instrument.
 *                                    This can be in dotted decimal notation.
 *    (int) portNumber  . . . . . . . The TCP/IP port to talk to.
```

```
 *                                    Use 5025 for the SCPI port.
 *
 * $Return:     (int)  . . . . . . . . A file descriptor similar to open(1).$
 *
 * $Errors:     returns -1 if anything goes wrong $
 *
 ******************************************************************************/
SOCKET openSocket(const char *hostname, int portNumber)
{
    struct hostent *hostPtr;
    struct sockaddr_in peeraddr_in;
    SOCKET s;


    memset(&peeraddr_in, 0, sizeof(struct sockaddr_in));

    /*********************************************/
    /* map the desired host name to internal form. */
    /*********************************************/
    hostPtr = gethostbyname(hostname);
    if (hostPtr == NULL)
    {
        fprintf(stderr,"unable to resolve hostname '%s'\n", hostname);
        return INVALID_SOCKET;
    }

    /******************/
    /* create a socket */
    /******************/
    s = socket(AF_INET, SOCK_STREAM, 0);
    if (s == INVALID_SOCKET)
    {
        fprintf(stderr,"unable to create socket to '%s': %s\n",
                hostname, strerror(errno));
        return INVALID_SOCKET;
    }
```

```
    memcpy(&peeraddr_in.sin_addr.s_addr, hostPtr->h_addr, hostPtr->h_length);
    peeraddr_in.sin_family = AF_INET;
    peeraddr_in.sin_port = htons((unsigned short)portNumber);

    if (connect(s, (const struct sockaddr*)&peeraddr_in,
                sizeof(struct sockaddr_in)) == SOCKET_ERROR)
    {
        fprintf(stderr,"unable to create socket to '%s': %s\n",
                hostname, strerror(errno));
        return INVALID_SOCKET;
    }

    return s;
}



/*****************************************************************************
 *
 > $Function: commandInstrument$
 *
 * $Description:  send a SCPI command to the instrument.$
 *
 * $Parameters:  $
 *     (FILE *) . . . . . file pointer associated with TCP/IP socket.
 *     (const char *command)  . . SCPI command string.
 * $Return:  (char *) . . . . . a pointer to the result string.
 *
 * $Errors:   returns 0 if send fails $
 *
 *****************************************************************************/
int commandInstrument(SOCKET sock,
                      const char *command)
{
    int count;
```

```
    /* fprintf(stderr, "Sending \"%s\".\n", command);  */
    if (strchr(command, '\n') == NULL) {
       fprintf(stderr, "Warning: missing newline on command %s.\n", command);
    }

    count = send(sock, command, strlen(command), 0);
    if (count == SOCKET_ERROR) {
        return COMMAND_ERROR;
    }

    return NO_CMD_ERROR;
}


/*******************************************************************************
 * recv_line(): similar to fgets(), but uses recv()
 ******************************************************************************/
char * recv_line(SOCKET sock, char * result, int maxLength)
{
#ifdef WINSOCK
    int cur_length = 0;
    int count;
    char * ptr = result;
    int err = 1;

    while (cur_length < maxLength) {
        /* Get a byte into ptr */
        count = recv(sock, ptr, 1, 0);

        /* If no chars to read, stop. */
        if (count < 1) {
            break;
        }
        cur_length += count;
```

```
        /* If we hit a newline, stop. */
        if (*ptr == '\n') {
            ptr++;
            err = 0;
            break;
        }
        ptr++;

    }

    *ptr = '\0';

    if (err) {
        return NULL;
    } else {
        return result;
    }
#else
    /*************************************************************************
     * Simpler UNIX version, using file I/O.  recv() version works too.
     * This demonstrates how to use file I/O on sockets, in UNIX.
     ************************************************************************/
    FILE * instFile;
    instFile = fdopen(sock, "r+");
    if (instFile == NULL)
    {
        fprintf(stderr, "Unable to create FILE * structure : %s\n",
                strerror(errno));
        exit(2);
    }
    return fgets(result, maxLength, instFile);
#endif
}
```

```
/*****************************************************************************
 *
 > $Function: queryInstrument$
 *
 * $Description:  send a SCPI command to the instrument, return a response.$
 *
 * $Parameters:  $
 *     (FILE *) . . . . . file pointer associated with TCP/IP socket.
 *     (const char *command)  . . SCPI command string.
 *     (char *result) . . . . . where to put the result.
 *     (size_t) maxLength . . . . maximum size of result array in bytes.
 *
 * $Return:  (long) . . . . . The number of bytes in result buffer.
 *
 * $Errors:   returns 0 if anything goes wrong. $
 *
 *****************************************************************************/
long queryInstrument(SOCKET sock,
                     const char *command, char *result, size_t maxLength)
{
    long ch;
    char tmp_buf[8];
    long resultBytes = 0;
    int command_err;
    int count;

    /*********************************************************
     * Send command to analyzer
     *********************************************************/
    command_err = commandInstrument(sock, command);
    if (command_err) return COMMAND_ERROR;



    /*********************************************************
     * Read response from analyzer
     *********************************************************/
```

```
count = recv(sock, tmp_buf, 1, 0); /* read 1 char */
ch = tmp_buf[0];

if ((count < 1) || (ch == EOF)  || (ch == '\n'))
{
    *result = '\0';  /* null terminate result for ascii */
    return 0;
}

/* use a do-while so we can break out */
do
{
    if (ch == '#')
    {
        /* binary data encountered - figure out what it is */
        long numDigits;
        long numBytes = 0;
        /* char length[10]; */

        count = recv(sock, tmp_buf, 1, 0); /* read 1 char */
        ch = tmp_buf[0];
        if ((count < 1) || (ch == EOF)) break; /* End of file */

        if (ch < '0' || ch > '9') break;  /* unexpected char */
        numDigits = ch - '0';

        if (numDigits)
        {
            /* read numDigits bytes into result string. */
            count = recv(sock, result, (int)numDigits, 0);
            result[count] = 0;  /* null terminate */
            numBytes = atol(result);
        }

        if (numBytes)
        {
```

```
        resultBytes = 0;
        /* Loop until we get all the bytes we requested. */
        /* Each call seems to return up to 1457 bytes, on HP-UX 9.05 */
        do {
            int rcount;
            rcount = recv(sock, result, (int)numBytes, 0);
            resultBytes += rcount;
            result      += rcount;  /* Advance pointer */
        } while ( resultBytes < numBytes );

        /*************************************************************
         * For LAN dumps, there is always an extra trailing newline
         * Since there is no EOI line.  For ASCII dumps this is
         * great but for binary dumps, it is not needed.
         *************************************************************/
        if (resultBytes == numBytes)
        {
            char junk;
            count = recv(sock, &junk, 1, 0);
        }
    }
    else
    {
        /* indefinite block ... dump til we can an extra line feed */
        do
        {
            if (recv_line(sock, result, maxLength) == NULL) break;
            if (strlen(result)==1 && *result == '\n') break;
            resultBytes += strlen(result);
            result += strlen(result);
        } while (1);
    }
}
else
{
    /* ASCII response (not a binary block) */
```

```
            *result = (char)ch;
            if (recv_line(sock, result+1, maxLength-1) == NULL) return 0;

            /* REMOVE trailing newline, if present.  And terminate string. */
            resultBytes = strlen(result);
            if (result[resultBytes-1] == '\n') resultBytes -= 1;
            result[resultBytes] = '\0';
        }
    } while (0);

    return resultBytes;
}
```

```
/*****************************************************************************
 *
 > $Function: showErrors$
 *
 * $Description: Query the SCPI error queue, until empty.  Print results. $
 *
 * $Return:  (void)
 *
 *****************************************************************************/
void showErrors(SOCKET sock)
{
    const char * command = "SYST:ERR?\n";
    char result_str[256];

    do {
        queryInstrument(sock, command, result_str, sizeof(result_str)-1);

        /*****************************************************************
         * Typical result_str:
         *     -221,"Settings conflict; Frequency span reduced."
```

```
         *     +0,"No error"
         * Don't bother decoding.
         ****************************************************************/
        if (strncmp(result_str, "+0,", 3) == 0) {
            /* Matched +0,"No error" */
            break;
        }
        puts(result_str);
    } while (1);

}


/*****************************************************************************
 *
 > $Function: isQuery$
 *
 * $Description: Test current SCPI command to see if it a query. $
 *
 * $Return:  (unsigned char) . . . non-zero if command is a query.  0 if not.
 *
 *****************************************************************************/
unsigned char isQuery( char* cmd )
{
    unsigned char q = 0 ;
    char *query ;

    /*******************************************************/
    /* if the command has a '?' in it, use queryInstrument.  */
    /* otherwise, simply send the command.                   */
    /* Actually, we must a little more specific so that      */
    /* marker value queries are treated as commands.         */
    /* Example:  SENS:FREQ:CENT (CALC1:MARK1:X?)             */
    /*******************************************************/
    if ( (query = strchr(cmd,'?')) != NULL)
    {
```

```
        /* Make sure we don't have a marker value query, or
         * any command with a '?' followed by a ')' character.
         * This kind of command is not a query from our point of view.
         * The analyzer does the query internally, and uses the result.
         */
        query++ ;       /* bump past '?' */
        while (*query)
        {
            if (*query == ' ') /* attempt to ignore white spc */
                query++ ;
            else break ;
        }

        if ( *query != ')' )
        {
            q = 1 ;
        }
    }
    return q ;
}



/*****************************************************************************
 *
 > $Function: main$
 *
 * $Description: Read command line arguments, and talk to analyzer.
                 Send query results to stdout. $
 *
 * $Return:  (int) . . . non-zero if an error occurs
 *
 *****************************************************************************/
int main(int argc, char *argv[])
{
```

```
SOCKET instSock;
char *charBuf = (char *) malloc(INPUT_BUF_SIZE);
char *basename;
int chr;
char command[1024];
char *destination;
unsigned char quiet = 0;
unsigned char show_errs = 0;
int number = 0;

basename = strrchr(argv[0], '/');
if (basename != NULL)
    basename++ ;
else
    basename = argv[0];

while ( ( chr = getopt(argc,argv,"qune")) != EOF )
    switch (chr)
    {
        case 'q':  quiet = 1; break;
        case 'n':  number = 1; break ;
        case 'e':  show_errs = 1; break ;
        case 'u':
        case '?':  usage(basename); exit(1) ;
    }

/* now look for hostname and optional <command> */
if (optind < argc)
{
    destination = argv[optind++] ;
    strcpy(command, "");
    if (optind < argc)
    {
        while (optind < argc) {
            /* <hostname> <command> provided; only one command string */
            strcat(command, argv[optind++]);
```

```
                if (optind < argc) {
                    strcat(command, " ");
                } else {
                    strcat(command, "\n");
                }
            }
        }
        else
        {
            /* Only <hostname> provided; input on <stdin> */
            strcpy(command, "");

            if (optind > argc)
            {
                usage(basename);
                exit(1);
            }
        }
    }
    else
    {
        /* no hostname! */
        usage(basename);
        exit(1);
    }

    /*********************************************/
    /* open a socket connection to the instrument */
    /*********************************************/
#ifdef WINSOCK
    if (init_winsock() != 0) {
        exit(1);
    }
#endif /* WINSOCK */

    instSock = openSocket(destination, SCPI_PORT);
```

```
if (instSock == INVALID_SOCKET) {
    fprintf(stderr, "Unable to open socket.\n");
    return 1;
}
/* fprintf(stderr, "Socket opened.\n"); */

if (strlen(command) > 0)
{
    /*********************************************************/
    /* if the command has a '?' in it, use queryInstrument. */
    /* otherwise, simply send the command.                   */
    /*********************************************************/
    if ( isQuery(command) )
    {
        long bufBytes;
        bufBytes = queryInstrument(instSock, command,
                                   charBuf, INPUT_BUF_SIZE);
        if (!quiet)
        {
            fwrite(charBuf, bufBytes, 1, stdout);
            fwrite("\n", 1, 1, stdout) ;
            fflush(stdout);
        }
    }
    else
    {
        commandInstrument(instSock, command);
    }
}
else
{
    /* read a line from <stdin> */
    while ( gets(charBuf) != NULL )
    {
        if ( !strlen(charBuf) )
            continue ;
```

```
if ( *charBuf == '#' || *charBuf == '!' )
    continue ;

strcat(charBuf, "\n");

if (!quiet)
{
    if (number)
    {
        char num[10];
        sprintf(num,"%d: ",number);
        fwrite(num, strlen(num), 1, stdout);
    }
    fwrite(charBuf, strlen(charBuf), 1, stdout) ;
    fflush(stdout);
}

if ( isQuery(charBuf) )
{
    long bufBytes;

    /* Put the query response into the same buffer as the
     * command string appended after the null terminator.
     */
    bufBytes = queryInstrument(instSock, charBuf,
                               charBuf + strlen(charBuf) + 1,
                               INPUT_BUF_SIZE -strlen(charBuf) );
    if (!quiet)
    {
        fwrite("  ", 2, 1, stdout) ;
        fwrite(charBuf + strlen(charBuf)+1, bufBytes, 1, stdout);
        fwrite("\n", 1, 1, stdout) ;
        fflush(stdout);
    }
}
```

```
            else
            {
                commandInstrument(instSock, charBuf);
            }
            if (number) number++;
        }
    }

    if (show_errs) {
        showErrors(instSock);
    }

#ifdef WINSOCK
    closesocket(instSock);
    close_winsock();
#else
    close(instSock);
#endif /* WINSOCK */

    return 0;
}

/* End of lanio.c */
```

Using C Programming Over Socket LAN (Windows NT)

This is the C programming example getopt.c that demonstrates simple socket programming. It is written in C, and compiles in the Windows NT environment.

In Windows, the routines send() and recv() must be used, since fread() and fwrite() may not work on sockets.

The program reads the analyzer's host name from the command line, followed by the SCPI command. It then opens a socket to the analyzer, using port 5025, and sends the command. If the command appears to be a query, the program queries the analyzer for a response, and prints the response.

This example program can also be used as a utility to talk to your analyzer from the command prompt on your Windows NT PC, or from within a script.

```
/*****************************************************************************

getopt(3C)                                                        getopt(3C)



NAME
     getopt - get option letter from argument vector

SYNOPSIS
     int getopt(int argc, char * const argv[], const char *optstring);

     extern char *optarg;
     extern int optind, opterr, optopt;

DESCRIPTION
     getopt returns the next option letter in argv (starting from argv[1])
     that matches a letter in optstring.  optstring is a string of
     recognized option letters; if a letter is followed by a colon, the
     option is expected to have an argument that may or may not be
     separated from it by white space.  optarg is set to point to the start
     of the option argument on return from getopt.
```

```
      getopt places in optind the argv index of the next argument to be
      processed.  The external variable optind is initialized to 1 before
      the first call to the function getopt.

      When all options have been processed (i.e., up to the first non-option
      argument), getopt returns EOF.  The special option -- can be used to
      delimit the end of the options; EOF is returned, and -- is skipped.

 ***************************************************************************/


#include <stdio.h>      /* For NULL, EOF */
#include <string.h>     /* For strchr() */

char    *optarg;        /* Global argument pointer. */
int     optind = 0;     /* Global argv index. */

static char     *scan = NULL;   /* Private scan pointer. */

int getopt( int argc, char * const argv[], const char* optstring)
{
    char c;
    char *posn;

    optarg = NULL;

    if (scan == NULL || *scan == '\0') {
        if (optind == 0)
            optind++;

        if (optind >= argc || argv[optind][0] != '-' || argv[optind][1] == '\0')
            return(EOF);
        if (strcmp(argv[optind], "--")==0) {
            optind++;
            return(EOF);
```

```
        }

        scan = argv[optind]+1;
        optind++;
    }

    c = *scan++;
    posn = strchr(optstring, c);        /* DDP */

    if (posn == NULL || c == ':') {
        fprintf(stderr, "%s: unknown option -%c\n", argv[0], c);
        return('?');
    }

    posn++;
    if (*posn == ':') {
        if (*scan != '\0') {
            optarg = scan;
            scan = NULL;
        } else {
            optarg = argv[optind];
            optind++;
        }
    }

    return(c);
}
```

Using Java Programming Over Socket LAN

This is the Java programming example ScpiDemo.java that demonstrates simple socket programming with Java. It is written in Java programming language, and will compile with Java compilers versions 1.0 and above.

```
import java.awt.*;
import java.io.*;
import java.net.*;
import java.applet.*;

// This is a SCPI Demo to demonstrate how one can communicate with the
// E4406A VSA  with a JAVA capable browser.  This is the
// Main class for the SCPI Demo.  This applet will need Socks.class to
// support the I/O commands and a ScpiDemo.html for a browser to load
// the applet.
// To use this applet, either compile this applet with a Java compiler
// or use the existing compiled classes.  copy ScpiDemo.class,
// Socks.class and ScpiDemo.html to a floppy.  Insert the floppy into
// your instrument.  Load up a browser on your computer and do the
// following:
//     1. Load this URL in your browser:
//        ftp://<Your instrument's IP address or name>/int/ScpiDemo.html
//     2. There should be two text windows show up in the browser:
//        The top one is the SCPI response text area for any response
//        coming back from the instrument.  The bottom one is for you
//        to enter a SCPI command.  Type in a SCPI command and hit enter.
//        If the command expects a response, it will show up in the top
//        window.
public class ScpiDemo extends java.applet.Applet implements Runnable {
    Thread       responseThread;
    Socks        sck;
    URL          appletBase;
    TextField    scpiCommand = new TextField();
    TextArea     scpiResponse = new TextArea(10, 60);
    Panel        southPanel = new Panel();
```

```
Panel        p;

// Initialize the applets
public void init() {

    SetupSockets();
    SetupPanels();

    // Set up font type for both panels
    Font font = new Font("TimesRoman", Font.BOLD,14);
    scpiResponse.setFont(font);
    scpiCommand.setFont(font);
    scpiResponse.appendText("SCPI Demo Program:  Response messages\n");
    scpiResponse.appendText("-----------------------------------------\n");
}

// This routine is called whenever the applet is actived
public void start() {
    // Open the sockets if not already opened
    sck.OpenSockets();
    // Start a response thread
    StartResponseThread(true);
}

// This routine is called whenever the applet is out of scope
// i.e. minize browser
public void stop() {
    // Close all local sockets
    sck.CloseSockets();
    // Kill the response thread
    StartResponseThread(false);
}

// Action for sending out scpi commands
// This routine is called whenever a command is received from the
// SCPI command panel.
```

```
public boolean action(Event evt, Object what) {
    // If this is the correct target
    if (evt.target == scpiCommand) {
        // Get the scpi command
        String str = scpiCommand.getText();
        // Send it out to the Scpi socket
        sck.ScpiWriteLine(str);
        String tempStr = str.toLowerCase();
        // If command str is "syst:err?", don't need to send another one.
        if ( (tempStr.indexOf("syst") == -1) ||
             (tempStr.indexOf("err") == -1)  ) {
            // Query for any error
            sck.ScpiWriteLine("syst:err?");
        }
        return true;
    }
    return false;
}

// Start/Stop a Response thread to display the response strings
private void StartResponseThread(boolean start) {
    if (start) {
        // Start a response thread
        responseThread = new Thread(this);
        responseThread.start();
    }
    else {
        // Kill the response thread
        responseThread = null;
    }
}

// Response thread running
public void run() {
    String str = "";  // Initialize str to null
```

```
    // Clear the error queue before starting the thread
    // in case if there's any error messages from the previous actions
    while ( str.indexOf("No error") == -1 ) {
        sck.ScpiWriteLine("syst:err?");
        str = sck.ScpiReadLine();
    }

    // Start receiving response or error messages
    while(true) {
        str = sck.ScpiReadLine();
        if ( str != null ) {
            // If response messages is "No error", do no display it,
            // replace it with "OK" instead.
            if ( str.equals("+0,\"No error\"") ) {
                str = "OK";
            }
            // Display any response messages in the Response panel
            scpiResponse.appendText(str+"\n");
        }
    }
}

// Set up and open the SCPI sockets
private void SetupSockets() {
    // Get server url
    appletBase = (URL)getCodeBase();
    // Open the sockets
    sck = new Socks(appletBase);
}

// Set up the SCPI command and response panels
private void SetupPanels() {
    // Set up SCPI command panel
    southPanel.setLayout(new GridLayout(1, 1));
    p = new Panel();
    p.setLayout(new BorderLayout());
```

```
        p.add("West", new Label("SCPI command:"));
        p.add("Center", scpiCommand);
        southPanel.add(p);

        // Set up the Response panel
        setLayout(new BorderLayout(2,2));
        add("Center", scpiResponse);
        add("South", southPanel);
    }

}


//  Socks class is responsible for open/close/read/write operations
//  from the predefined socket ports.  For this example program,
//  the only port used is 5025 for the SCPI port.
class Socks extends java.applet.Applet {
    // Socket Info
    // To add a new socket, add a constant here, change MAX_NUM_OF_SOCKETS
    // then, edit the constructor for the new socket.
    public final int SCPI=0;
    private final int MAX_NUM_OF_SOCKETS=1;

    // Port number
    // 5025 is the dedicated port number for E4406A Scpi Port
    private final int SCPI_PORT = 5025;

    // Socket info
    private URL appletBase;
    private Socket[] sock = new Socket[MAX_NUM_OF_SOCKETS];
    private DataInputStream[] sockIn = new DataInputStream[MAX_NUM_OF_SOCKETS];
    private PrintStream[] sockOut = new PrintStream[MAX_NUM_OF_SOCKETS];
    private int[] port = new int[MAX_NUM_OF_SOCKETS];
    private boolean[] sockOpen = new boolean[MAX_NUM_OF_SOCKETS];

    // Constructor
```

```
Socks(URL appletB)
{
    appletBase = appletB;

    // Set up for port array.
    port[SCPI] = SCPI_PORT;

    // Initialize the sock array
    for ( int i = 0; i < MAX_NUM_OF_SOCKETS; i++ ) {
        sock[i] = null;
        sockIn[i] = null;
        sockOut[i] = null;
        sockOpen[i] = false;
    }
}

//***** Sockects open/close routines
// Open the socket(s) if not already opened
public void OpenSockets()
{
    try {
        // Open each socket if possible
        for ( int i = 0; i < MAX_NUM_OF_SOCKETS; i++ ) {
            if ( !sockOpen[i] ) {
                sock[i] = new Socket(appletBase.getHost(),port[i]);
                sockIn[i] = new DataInputStream(sock[i].getInputStream());
                sockOut[i] = new PrintStream(sock[i].getOutputStream());
                if ( (sock[i] != null) && (sockIn[i] != null) &&
                      (sockOut[i] != null) ) {
                    sockOpen[i] = true;
                }
            }
        }
    }
    catch (IOException e) {
        System.out.println("Sock, Open Error "+e.getMessage());
```

```
    }
}

// Close the socket(s) if opened
public void CloseSocket(int s)
{
    try {
        if ( sockOpen[s] == true ) {
            // write blank line to exit servers elegantly
            sockOut[s].println();
            sockOut[s].flush();
            sockIn[s].close();
            sockOut[s].close();
            sock[s].close();
            sockOpen[s] = false;
        }
    }
    catch (IOException e) {
        System.out.println("Sock, Close Error "+e.getMessage());
    }
}

// Close all sockets
public void CloseSockets()
{
    for ( int i=0; i < MAX_NUM_OF_SOCKETS; i++ ) {
        CloseSocket(i);
    }
}

// Return the status of the socket, open or close.
public boolean SockOpen(int s)
{
    return sockOpen[s];
}
```

```
//************* Socket I/O routines.

//*** I/O routines for SCPI socket

// Write an ASCII string with carriage return to SCPI socket
public void ScpiWriteLine(String command)
{
    if ( SockOpen(SCPI) ) {
        sockOut[SCPI].println(command);
        sockOut[SCPI].flush();
    }
}

// Read an ASCII string, terminated with carriage return from SCPI socket
public String ScpiReadLine()
{
    try {
        if ( SockOpen(SCPI) ) {
            return sockIn[SCPI].readLine();
        }
    }
    catch (IOException e) {
        System.out.println("Scpi Read Line Error "+e.getMessage());
    }
    return null;
}

// Read a byte from SCPI socket
public byte ScpiReadByte()
{
    try {
        if ( SockOpen(SCPI) ) {
            return sockIn[SCPI].readByte();
        }
    }
```

```
        catch (IOException e) {
            System.out.println("Scpi Read Byte Error "+e.getMessage());
        }
        return 0;
    }

}
```

Using VEE Over Socket LAN

This is the VEE programming example landemo.vee. It demonstrates simple socket programming using VEE.

The user must have Version K of the Agilent IO libraries installed alone or installed side-by-side with the National Instruments IO libraries. Also, the user must first import the VXI plug and play driver into LabView before running this example. The instrument drivers are available at:

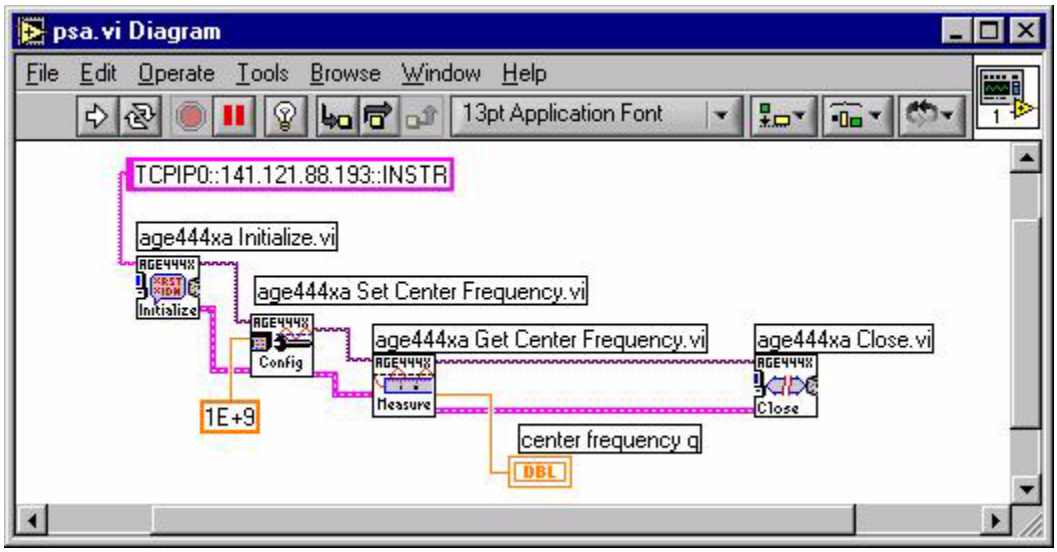
`http://www.agilent.com/find/iolib` (Click on instrument drivers.)

This example:

1. Opens a VXI 11.3 Lan connection to the instrument
2. Sets the Center Frequency to 1 GHz
3. Queries the instrument's center frequency
4. Closes the Lan connection to the instrument

NOTE Substitute your instruments I.P. address for the one used in the example.

Example:



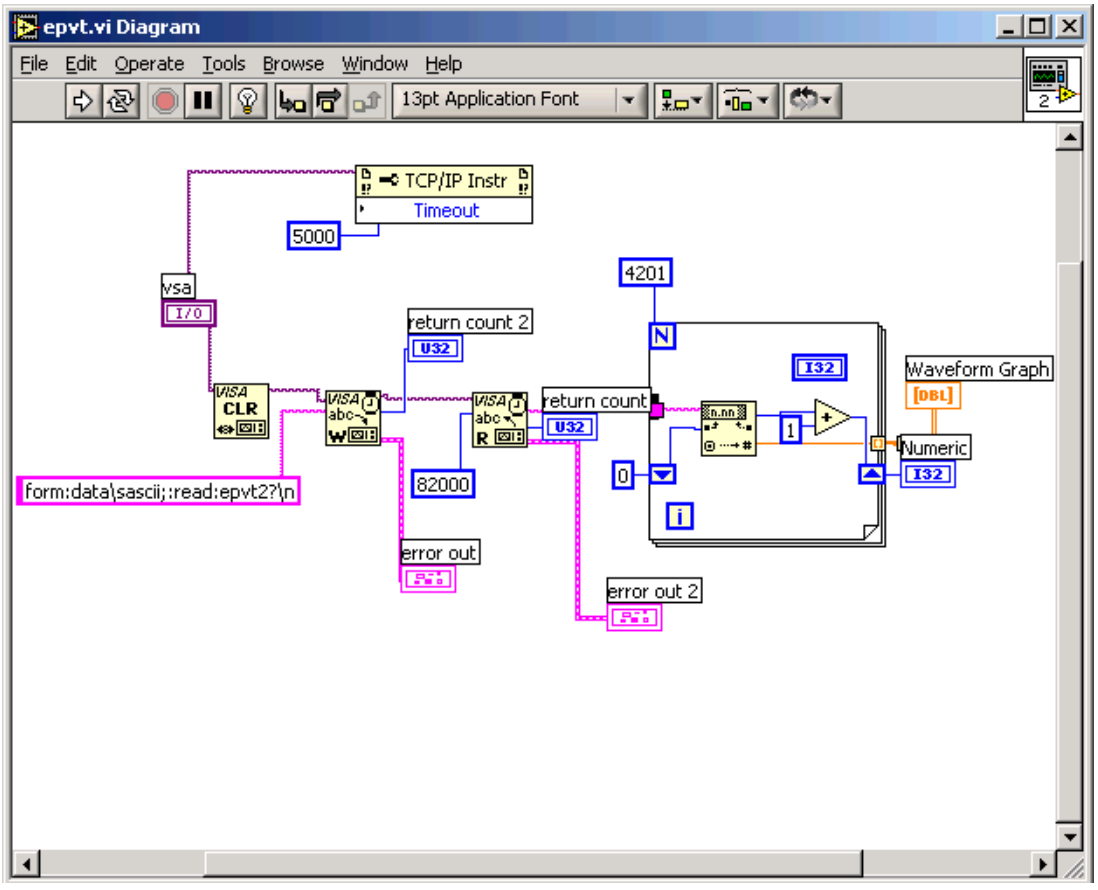
Using LabVIEW® 6 to Make an EDGE GSM Measurement

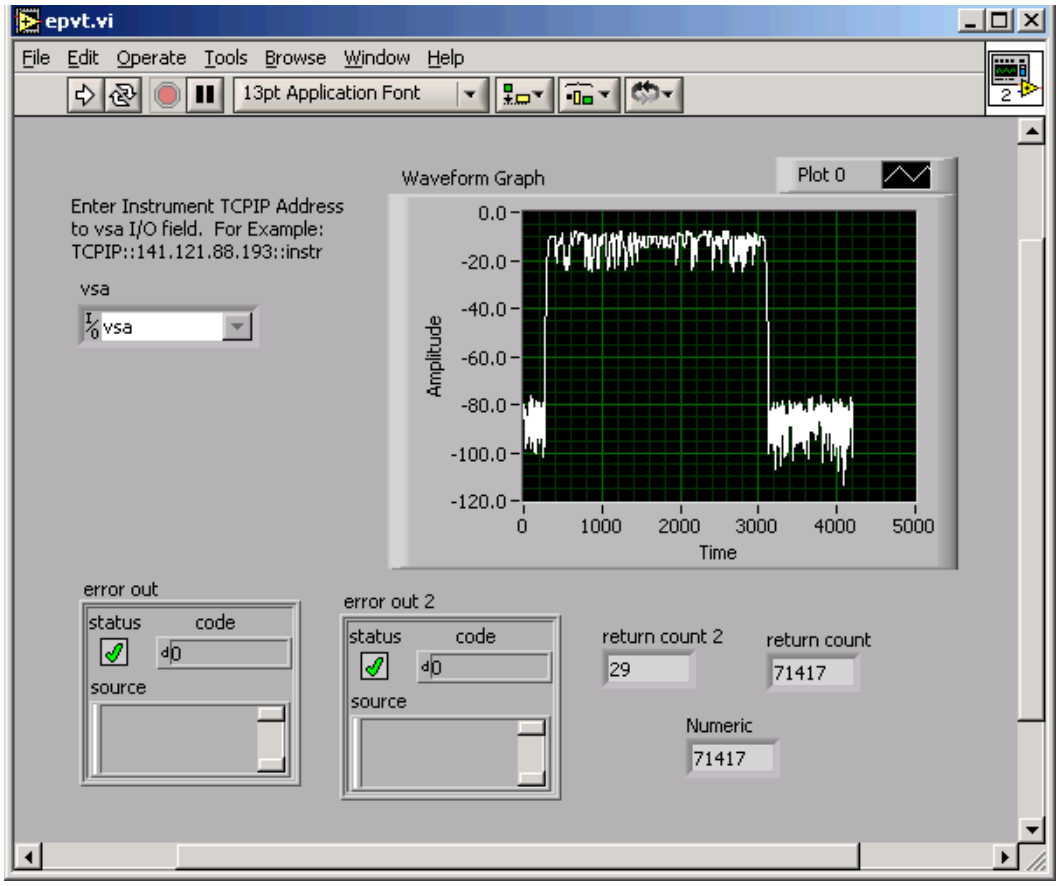
This is a LabVIEW 6 example that uses SCPI commands instead of the instrument driver. It demonstrates reading ASCII trace points of entire EDGE waveform data in the Power Vs. Time measurement over LAN. This program uses the optional GSM/EDGE personality in the PSA Series Spectrum Analyzers and in the E4406A Vector Signal Analyzer. The vi file (epvt.vi) can be found on the Documentation CD.

This example:

1. Opens a VXI 11.3 Lan connection to the instrument
2. Changes the data format to ASCII.
3. Initiates a power vs. time measurement and reads the results.
4. The comma separated ASCII results string is converted to an array of values.

Example:




epvt.vi
File Edit Operate Tools Browse Window Help
13pt Application Font
Waveform Graph
Plot 0
Enter Instrument TCPIP Address
to vsa I/O field. For Example:
TCPIP::141.121.88.193::instr
vsa
vsa
Amplitude
0.0
-20.0
-40.0
-60.0
-80.0
-100.0
-120.0
0
1000
2000
3000
4000
5000
Time
error out
status
code
source
error out 2
status
code
source
return count 2
29
return count
71417
Numeric
71417

Programming Examples

Using Visual Basic® 6 to Capture a Screen Image

This is a Visual Basic example that stores the current screen image on your PC. The program works with the E4406A Vector Signal Analyzer. The bas file (screen.bas) and a compiled executable (screen.exe) can be found on the Documentation CD.

This example:

1. Stores the current screen image on the instrument's flash as C:PICTURE.GIF.
2. Transfers the image over GPIB or LAN and stores it on your PC in the current directory as picture.gif.
3. The file C:PICTURE.GIF is then deleted from the instrument's flash.

NOTE This example uses GPIB address 18 for the spectrum analyzer.

```
'' """"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
''  Copyright © 1999- 2003 Agilent Technologies Inc.  All rights reserved.
''
'' You have a royalty-free right to use, modify, reproduce and distribute
'' the Sample Application Files (and/or any modified version) in any way
'' you find useful, provided that you agree that Agilent Technologies has
'' no warranty, obligations or liability for any Sample Application Files.
''
'' Agilent Technologies provides programming examples for illustration only,
'' This sample program assumes that you are familiar with the programming
'' language being demonstrated and the tools used to create and debug
'' procedures. Agilent Technologies support engineers can help explain the
'' functionality of Agilent Technologies software components and associated
'' commands, but they will not modify these samples to provide added
'' functionality or construct procedures to meet your specific needs.
'' """"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

'' To develop VISA applications in Microsoft Visual Basic, you first need
'' to add the Visual Basic (VB) declaration file in your VB project as a
'' Module. This file contains the VISA function definitions and constant
```

```
'' declarations needed to make VISA calls from Visual Basic.
'' To add this module to your project in VB 6, from the menu, select
'' Project->Add Module, select the 'Existing' tab, and browse to the
'' directory containing the VB Declaration file, select visa32.bas, and
'' press 'Open'.
''
'' The name and location of the VB declaration file depends on which
'' operating system you are using.  Assuming the 'standard' VISA directory
'' of C:\Program Files\VISA or the 'standard' VXIpnp directory of
'' C:\VXIpnp, the visa32.bas file can be located in one of the following:
''
''   \winnt\agvisa\include\visa32.bas - Windows NT/2000/XP
''   \winnt\include\visa32.bas        - Windows NT/2000/XP
''   \win95\include\visa32.bas        - Windows 95/98/Me
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'  screen_e4406a.bas
'  The following example program is written for the E4406A Series Vector
'  Signal Analyzer.  It queries the current screen image on the instrument
'  and stores it on your PC in the current directory as screen.gif.
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Option Explicit

Private Sub Main()
    ' Declare Variables used in the program
    Dim status As Long      'VISA function status return code
    Dim defrm As Long       'Session to Default Resource Manager
    Dim vi As Long          'Session to instrument
    Dim x As Integer        'Loop Variable
    Dim ArrayPtr(1) As Long 'Array of Pointers
    Dim ResultsArray(50000) As Byte 'results array, Big enough to hold a GIF
    Dim length As Long      'Number of bytes returned from instrument
    Dim fnum As Integer     'File Number to used to open file to store data
    Dim isOpen As Boolean   'Boolean flag used to keep track of open file
    Dim headerlength As Long 'length of header

    'Set the default number of bytes that will be contained in the
```

```
'ResultsArray to 50,000 (50kB)
length = 50000

'Set the array of pointers to the addresses of the variables
ArrayPtr(0) = VarPtr(length)
ArrayPtr(1) = VarPtr(ResultsArray(0))

'Delete screen.gif file if it exists
On Error Resume Next
Kill "screen.gif"

On Error GoTo Error_Handler

' Open the default resource manager session
status = viOpenDefaultRM(defrm)

' Open the session.  Note: For E4406A, to use LAN, change the string to
' "TCPIP0::xxx.xxx.xxx.xxx::inst0::INSTR" where xxxxx is the IP address
status = viOpen(defrm, "GPIB0::18::INSTR", 0, 0, vi)
If (status < 0) Then GoTo VisaErrorHandler

' Set the I/O timeout to fifteen seconds
status = viSetAttribute(vi, VI_ATTR_TMO_VALUE, 15000)
If (status < 0) Then GoTo VisaErrorHandler

'Grab the screen image file from the instrument
status = viVQueryf(vi, ":HCOPY:SDUM:DATA?" + Chr$(10), _
         "%#y", ArrayPtr(0))

'Close the vi session and the resource manager session
Call viClose(vi)
Call viClose(defrm)

'Store the results in a text file
fnum = FreeFile()  'Get the next free file number
Open "screen.gif" For Binary As #fnum
```

```
    isOpen = True
    headerlength = 2 + (Chr$(ResultsArray(1)))
    For x = headerlength To length - 2
    Put #fnum, , ResultsArray(x)
    Next x

    ' Intentionally flow into Error Handler to close file
Error_Handler:
    ' Raise the error (if any), but first close the file
    If isOpen Then Close #fnum
    If Err Then Err.Raise Err.Number, , Err.Description
    Exit Sub

VisaErrorHandler:
   Dim strVisaErr As String * 200
   Call viStatusDesc(defrm, status, strVisaErr)
   MsgBox "*** Error : " & strVisaErr, vbExclamation, "VISA Error Message"
   Exit Sub
End Sub
```

Using Visual Basic® 6 to Transfer Binary Trace Data

This is a Visual Basic example that gets binary trace data from the instrument. Binary data transfers are faster than the default ASCII transfer mode, because less data is sent over the bus. This example works with the E4406A Vector Signal Analyzer. The bas file (bintrace.bas) and a compiled executable (bintrace.exe) can be found on the Documentation CD.

This example:

1. Queries the IDN (identification) string from the instrument.
2. While in Basic mode, it reads the trace data in binary format (Real,32 or Real,64 or Int,32).
3. Stores the data is then to a file “bintrace.txt”.

NOTE This example uses GPIB address 18 for the spectrum analyzer.

```
'' """""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
''  Copyright © 1999- 2003 Agilent Technologies Inc.  All rights reserved.
''
'' You have a royalty-free right to use, modify, reproduce and distribute
'' the Sample Application Files (and/or any modified version) in any way
'' you find useful, provided that you agree that Agilent Technologies has
'' no warranty, obligations or liability for any Sample Application Files.
''
'' Agilent Technologies provides programming examples for illustration only,
'' This sample program assumes that you are familiar with the programming
'' language being demonstrated and the tools used to create and debug
'' procedures. Agilent Technologies support engineers can help explain the
'' functionality of Agilent Technologies software components and associated
'' commands, but they will not modify these samples to provide added
'' functionality or construct procedures to meet your specific needs.
'' """""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

'' To develop VISA applications in Microsoft Visual Basic, you first need
'' to add the Visual Basic (VB) declaration file in your VB project as a
'' Module. This file contains the VISA function definitions and constant
```

```
'' declarations needed to make VISA calls from Visual Basic.
'' To add this module to your project in VB 6, from the menu, select
'' Project->Add Module, select the 'Existing' tab, and browse to the
'' directory containing the VB Declaration file, select visa32.bas, and
'' press 'Open'.
''
'' The name and location of the VB declaration file depends on which
'' operating system you are using.  Assuming the 'standard' VISA directory
'' of C:\Program Files\VISA or the 'standard' VXIpnp directory of
'' C:\VXIpnp, the visa32.bas file can be located in one of the following:
''
''   \winnt\agvisa\include\visa32.bas - Windows NT/2000/XP
''   \winnt\include\visa32.bas        - Windows NT/2000/XP
''   \win95\include\visa32.bas        - Windows 95/98/Me

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'  bintrace_e4406a.bas
'  The following example program is written for the E4406A and PSA
'  Spectrum Analyzers.  It queries the IDN string from the instrument
'  and then reads the trace data from the Spectrum measurement in Basic mode
'  in binary format (Real,32 or Real,64 or Int,32).  The data is then stored
'  to a file, "bintrace.txt".
'  Binary transfers are faster than the default ASCII transfer mode,
'  because less data is sent over the bus.
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Option Explicit

Private Sub Main()
    ' Declare Variables used in the program
    Dim status As Long      'VISA function status return code
    Dim defrm As Long       'Session to Default Resource Manager
    Dim vi As Long          'Session to instrument
    Dim strRes As String * 100  'Fixed length string to hold *IDN? Results
    Dim x As Integer        'Loop Variable
    Dim output As String    'output string variable
    Dim ArrayPtr(1) As Long 'Array of Pointers
```

```
Dim ResultsArray(10000) As Single 'trace element array of Real,32 values
'For Real,64 data use Double. For Int,32 data use Long
Dim length As Long      'Number of trace elements return from instrument
Dim fnum As Integer     'File Number to used to open file to store data
Dim isOpen As Boolean   'Boolean flag used to keep track of open file

'Set the default number of trace elements to the ResultsArray size
length = 10000

'Set the array of pointers to the addresses of the variables
ArrayPtr(0) = VarPtr(length)
ArrayPtr(1) = VarPtr(ResultsArray(0))

On Error GoTo Error_Handler

' Open the default resource manager session
status = viOpenDefaultRM(defrm)

' Open the session.  Note: To use LAN, change the string to
' "TCPIP0::xxx.xxx.xxx.xxx::inst0::INTSR" where xxxxx is the IP address
status = viOpen(defrm, "GPIB0::18::INSTR", 0, 0, vi)
If (status < 0) Then GoTo VisaErrorHandler

' Set the I/O timeout to five seconds
status = viSetAttribute(vi, VI_ATTR_TMO_VALUE, 5000)
If (status < 0) Then GoTo VisaErrorHandler

'Ask for the devices's *IDN string.
status = viVPrintf(vi, "*IDN?" + Chr$(10), 0)
If (status < 0) Then GoTo VisaErrorHandler

'Read back the IDN string from the instrument
status = viVScanf(vi, "%t", strRes)
If (status < 0) Then GoTo VisaErrorHandler

'Print the IDN string results in a message box
```

```
MsgBox (strRes)

'Change the instrument mode to Basic
status = viVPrintf(vi, ":INST:SEL BASIC" + Chr$(10), 0)
If (status < 0) Then GoTo VisaErrorHandler

' Set instrument trace data format to 32-bit Real
' Note: For higher precision use 64-bit data, ":FORM REAL,64"
' For faster data transfer for ESA, use ":FORM INT,32"
status = viVPrintf(vi, ":FORM REAL,32" + Chr$(10), 0)
If (status < 0) Then GoTo VisaErrorHandler

'Set Analyzer to single sweep mode
status = viVPrintf(vi, ":INIT:CONT 0" + Chr$(10), 0)
If (status < 0) Then GoTo VisaErrorHandler

'Query the Averaged Spectrum trace from the instrument
'Note: Change the "%#zb" to "%#Zb" for Real,64 data
'      For Int,32 leave the modifier as "%#zb"
status = viVQueryf(vi, ":READ:SPEC7?" + Chr$(10), _
         "%#zb", ArrayPtr(0))

'Close the vi session and the resource manager session
Call viClose(vi)
Call viClose(defrm)

'Print number of elements returned
MsgBox ("Number of trace elements returned = " & length)

'Create a string from the ResultsArray to output to a file
For x = 0 To length - 1
  output = output & ResultsArray(x) & vbCrLf
Next x

'Print Results to the Screen
MsgBox (output)
```

```
    'Store the results in a text file
    fnum = FreeFile()  'Get the next free file number
    Open "bintrace.txt" For Output As #fnum
    isOpen = True
    Print #fnum, output

    ' Intentionally flow into Error Handler to close file
Error_Handler:
    ' Raise the error (if any), but first close the file
    If isOpen Then Close #fnum
    If Err Then Err.Raise Err.Number, , Err.Description
    Exit Sub

VisaErrorHandler:
   Dim strVisaErr As String * 200
   Call viStatusDesc(defrm, status, strVisaErr)
   MsgBox "*** Error : " & strVisaErr, vbExclamation, "VISA Error Message"
   Exit Sub
End Sub
```

Using Visual Basic® .NET with the IVI-Com Driver

This example uses Visual Basic .NET with the IVI-Com driver. It makes a time domain (Waveform) measurement using the Basic mode. Basic mode is standard in the E4406A Vector Signal Analyzer and is optional (B7J) in the PSA Series Spectrum Analyzers. The vb file (vbivicomsa_basicwaveform.vb) and the compiled executable file (vbivicomsa.exe) can be found on the Documentation CD.

```
'************************************************************************
' VBIviComSA_BasicWaveform.vb, August 5, 2003
' This example demonstrates the use of the IVI-COM driver in VB.NET
' through an interop assembly.  The Raw I/Q trace data from the Waveform
' measurement in Basic Mode is queried and printed to the screen.
'
' Requirements:
'  1) E4406A or PSA Series Spectrum Analyzer with Option B7J
'  2) Latest AgilentSa IVI-COM driver
'     You may download it here: http://www.agilent.com/find/inst_drivers
'     This example was tested with version 2.1.0.0 of the driver
'  3) Create a new project and add the References to this module
'     and to the the IVI-COM driver dlls:
'     For .NET, right click on Reference, choose Add Reference
'     and then click on Browse and directly link the DLLs in the directory:
'     C:\Program Files\IVI\Bin\Primary Interop Assemblies
'     Agilent.AgilentSa.Interop.dll
'     Agilent.AgilentSaAppBasic.Interop.dll
'     Agilent.Itl.Interop
'     IviDriverLib.dll
'     IviSpecAnLib.dll
'
' THIS CODE AND INFORMATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY
' KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
' IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
' PARTICULAR PURPOSE.
'
```

```
' Copyright (c) 2003. Agilent Technologies, Inc.
'*************************************************************************
Option Strict On

Imports Agilent.TMFramework
Imports Agilent.AgilentSa.Interop
Imports Ivi.Driver.Interop
Imports System.Runtime.InteropServices

Module ConsoleApp

  Sub Main()

    ' Prompt the user for the address of the instrument
    Dim address As String
    Console.WriteLine("Enter address of the instrument " & vbCrLf & _
    "(ex: GPIB0::18::INSTR or TCPIP0::192.168.100.2::inst0::INSTR):")
    address = Console.ReadLine()

    Try
      ' Create an instance of the driver, connection to the instrument
      ' is not established here, it is done by calling Initialize
      Dim instr As New AgilentSaClass()

      ' Establish the connection to the instrument
      ' Last parameter (DriverSetup) is optional, VB could omit it (but not C#)
      ' Important: Close must be called to release resources used by the driver
      instr.Initialize(address, False, False, "")

      Try
        ' INHERENT CAPABILITIES
        ' Note that it is not necessary to program against the IIviDriver
        ' interface, the same can be achieved by using the class directly
        ' Using the IIviDriver interface gives us interchangeable code
        Dim inherent As IIviDriver = instr
```

```
Dim manufacturer As String
Dim model As String
Dim firmware As String

manufacturer = inherent.Identity.InstrumentManufacturer
model = inherent.Identity.InstrumentModel
firmware = inherent.Identity.InstrumentFirmwareRevision
' Output instrument information to the console
Console.WriteLine("Manufacturer: " + manufacturer)
Console.WriteLine("Model: " + model)
Console.WriteLine("Firmware: " + firmware)

' Reset the instrument
inherent.Utility.Reset()

' INSTRUMENT SPECIFIC
' Using the IAgilentSa interface is not necessary or beneficial
' at the moment, but in the future if other instruments implement
' the IAgilentSa interface, the code that is written to work with
' that interface can be reused without changes, as opposed to code
' that is written against the class object directly
Dim sa As IAgilentSa = instr

' Obtain trace data from the instrument
Dim traceData As Array
sa.Application.Select("Basic")
'sa.Application.Basic.Waveform.Configure()
sa.Application.Basic.Spectrum.Traces.Initiate()
traceData = sa.Application.Basic.Waveform.Traces.Item("RawIQ").Read(10000)

' Output the trace data to the console
Console.WriteLine("Press ENTER to display trace data.")
Console.ReadLine()
Dim traceValue As Double
For Each traceValue In traceData
  Console.WriteLine(traceValue)
```

```
        Next

      Catch ex As Exception
        Console.WriteLine(ex.Message)

      Finally
        ' Close the connection
        instr.Close()

      End Try

    Catch ex As COMException
      Console.WriteLine(ex.Message)

    Catch ex As Exception
      Console.WriteLine(ex.Message)

    End Try

    ' Wait for user input
    Console.WriteLine("Press ENTER to end program.")
    Console.ReadLine()

  End Sub

End Module
```

4 Programming Command Cross References

Functional Sort of SCPI Commands

Function	SCPI Command Subsystems	Remarks
Averaging	SENSe:<measurement>:AVERage	
Bandwidth	SENSe:<measurement>:BWIDth	
Calibration	CALibration	
Channel: setting	SENSe:CHANnel	
Data format	FORMat:DATA	Data types include ASCII and real numbers
Display: Views, Scaling	DISPlay:ENABle: DISPlay:SPECtrum:WINDow DISPlay:WAVeform:WINDow	Different display data views are available for any individual measurement.
Errors	SYSTem:ERRors *CLS, *ESE, *ESE?, *ESR?, *OPC, *OPC? *PSC, *PSC?, *SRE, *SRE?, *STB? STATus:	
Frequency	SENSe:FREQuency	
File type	DISPlay:IMAGe	Image file types include .GIF and .WMF
Input/Output/ Configuration	INPut:IMPedance SYSTem:CONFigure SYStem:COMMunicate	
Markers	CALCulate:<measurement>:MARKer:	Not all measurements: 1. have markers available 2. have all the documented markers, or all the marker functions.
Measurements: control	ABORt INITiate:IMMediate INITiate:CONTinuous INITiate:RESTart	
Measurements: select mode	INSTrument:SELect	Modes include Basic and Service. Other optional modes are: NADC, PDC, GSM, EDGE, cdmaOne, W-CDMA, HSDPA & HSUPA, cdma2000, 1xEVDO, iDEN, WiDEN.

Function	SCPI Command Subsystems	Remarks
Measurements: mode setup	SENSe:CHANnel:TSCode SENSe:CORRection:BTS SENSe:CORRection:BS SENSe:FREQuency:CENTer SENSe:POWer[:RF] SENSe:RADio:CARRier SENSe:RADio:STANdard SENSe:SYNC	Mode setup parameters are used for all the measurements available within that mode. Mode setup parameters persist if you go to a different mode and then return to a previous mode.
Measurements: select measurement	CONFigure:<measurement> FETCh:<measurement> MEASure:<measurement> READ:<measurement> INITiate:<measurement>	Information about the types of data available for a measurement is in MEASure description.
Measurements: measurement setup	SENSe:AVERage: SENSe:BANDwidth: SENSe:FREQuency: SENSe:SWEep: SENSe:TRIGger: TRIGger:	
Preset	SYSTem:PRESet:	
Printing	HCOPy: SYSTem:COMMunicate	
Reference level	DISPlay:WINDow:TRACe	
Save/Recall: display images	DISPlay:IMAGe: HCOPy:IMMediate:	
Save/Recall: instrument states	*SAV *RCL	
Save/Recall: trace data	MEASure:<measurement>[n]? FETCh:<measurement>[n]? FORMat:DATA FORMat:BORDer	Descriptions of the traces available for each measurement are in the MEASure subsystem.
Triggering	TRIGger: SENSe:<measurement>:	
Standards, selection	SENSe:RADio	

Programming Command Compatibility Across Model Numbers

Using Applications in PSA Series vs. VSA E4406A

NOTE This information *only* applies to the application modes: Basic, cdmaOne, cdma2000, 1xEV-DO, W-CDMA, GSM, EDGE, NADC, and PDC.

Command	PSA Series	VSA E4406A: A.04.00	VSA E4406A: A.05.00
*RST	Resets instrument, putting it in continuous measurement mode. Use INIT:CONT OFF to select single measurement mode and INIT:IMM to start one measurement.	Resets instrument, putting it in single measurement mode. One measurement is initiated when the command is sent.	Resets instrument, putting it in single measurement mode. No measurement is initiated when the command is sent. Use INIT:IMM to start one measurement.
CONFigure:<measurement>	Accesses the measurement and sets the instrument settings to the defaults. If you were already in single measurement mode, it takes one measurement and then waits.	Same as PSA. Accesses the measurement and sets the instrument settings to the defaults. If you were already in single measurement mode, it takes one measurement and then waits.	Accesses the measurement and sets the instrument settings to the defaults. If you were already in single measurement mode, it does not initiate a measurement. Use INIT:IMM to make one measurement.
*ESE default	Default is 255 which means that every error/status bit change that has occurred will be returned with a *ESR? query. You must set the value of *ESE to choose only the bits/status that you want returned.	Default is 0 which means that none of the error/status bit changes that have occurred will be returned with a *ESR? query. You must set the value of *ESE to choose the bits/status that you want returned.	Same as VSA A.04.00. Default is 0 which means that none of the error/status bit changes that have occurred will be returned with a *ESR? query. You must set the value of *ESE to choose the bits/status that you want returned.
*LRN	The command is *not* available.	The command is available.	The command is available.

Command	PSA Series	VSA E4406A: A.04.00	VSA E4406A: A.05.00
TRIGger commands	In Spectrum Analysis mode only one value can be set for the trigger's source, delay, level, or polarity. Basic, GSM, EDGE, cdmaOne, cdma2000, W-CDMA, NADC, PDC modes function the same as VSA	You can select a unique trigger source for each mode. Each trigger source can have unique settings for the delay, level, and polarity parameters.	Same as VSA A.04.00. You can select a unique trigger source for each mode. Each trigger source can have unique settings for the delay, level, and polarity parameters.
AUTO ON | OFF control and setting manual values	We recommend that you set a function's automatic state to OFF, before you send it your manual value. Some functions will turn off the automatic mode when you send a specific manual value, but others will not. This also varies with the instrument model.	We recommend that you set a function's automatic state to OFF, before you send it your manual value. Some functions will turn off the automatic mode when you send a specific manual value, but others will not. This also varies with the instrument model.	We recommend that you set a function's automatic state to OFF, before you send it your manual value. Some functions will turn off the automatic mode when you send a specific manual value, but others will not. This also varies with the instrument model.

5 Language Reference

This chapter includes the commands that are common to all of the instrument modes. It also contains the commands unique to the basic and service modes. For commands specific to a measurement mode, like the GSM personality, look in the GSM Programming Commands chapter. Only commands in the current selected mode can be executed.

SCPI Command Subsystems

Common IEEE Commands

These commands are specified in IEEE Standard 488.2-1992, *IEEE Standard Codes, Formats, Protocols and Common Commands for Use with ANSI/IEEE Std 488.1-1987*. New York, NY, 1992.

Numeric values for bit patterns can be entered using decimal or hexidecimal representations. (i.e. 0 to 32767 is equivalent to #H0 to #H7FFF) See the SCPI Basics information about using bit patterns for variable parameters.

Calibration Query

***CAL?**

Performs a full alignment and returns a number indicating the success of the alignment. A zero is returned if the alignment is successful. A one is returned if any part of the alignment fails. The equivalent SCPI command is `CALibrate[:ALL]?`

Front Panel Access: **System**, **Alignments**, **Align All Now**

Clear Status

***CLS**

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

Remarks: See `*STB?`

Standard Event Status Enable

***ESE <number>**

***ESE?**

Selects the desired bits from the standard event status enable register. This register monitors I/O errors and synchronization conditions such as operation complete, request control, query error, device dependent error, execution error, command error and power on. The selected bits are OR'd to become a summary bit (bit 5) in the status byte register which can be queried.

The query returns the state of the standard event status enable

register.

Range: Integer, 0 to 255

Standard Event Status Register Query

***ESR?**

Queries and clears the standard event status event register. (This is a destructive read.)

Range: Integer, 0 to 255

Identification Query

***IDN?**

Returns an instrument identification information string to GPIB. The string will contain the model number, serial number and firmware revision.

The response is organized into four fields separated by commas. The field definitions are as follows:

- Manufacturer
- Model
- Serial number
- Firmware version

For example:

```
AgilentTechnologiesInc,E4406A,US00000040,A.01.42
```

Remarks: An @ in the firmware revision information indicates that it is prototype firmware.

Front Panel Access: **System, Show System**

Instrument State Query

***LRN?**

Returns current instrument state data in a block of defined length. The <state data> is in a machine readable format only. Sending the query returns the following format:

```
SYST:SET #NMMMM<state_data>
```

The following example is a response to *LRN? The actual sizes will vary depending on the instrument state data size.

Example: `:SYST:SET #42016<state data>`

Where: 4 (the N in the preceding query response example) represents the number of digits to follow
Where: 2016 (the MMMM in the preceding query response example) represents the number of bytes that follow in the <state data>.

The state can be changed by sending this block of data to the instrument after removing the size information:

```
:SYST:SET <state data>
```

Operation Complete Command

***OPC**

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

The instrument default is to only wait for completion of the internal self-alignment routines. You must set the STATus:OPERation:EVENt register if you want to look for the completion of additional processes. See *OPC? below.

Key Type: There is no equivalent front-panel key.

Operation Complete Query

***OPC?**

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

The instrument default is to only wait for completion of the internal self-alignment routines. You must set the STATus:OPERation:EVENt register if you want to look for the completion of additional processes such as:

VSA Process	STATus:OPER Register Bit	Byte Value
Calibrating	0	1
Sweeping	3	4
Waiting for trigger	5	16

VSA Process	STATus:OPER Register Bit	Byte Value
Printing	11	1024
Mass memory access (floppy drive)	12	2048

For example, if you want to verify the completion of both calibrating and waiting for trigger set :STAT:OPER:ENAB 17 and monitor any changes.

Key Type: There is no equivalent front-panel key.

Query Instrument Options

***OPT?**

Returns a string of all the installed instrument options. It is a comma separated list such as: BAC,BAH. There are a few options that include more then one mode. An instrument with one of these options will report the option number once for each mode. You would get a response: BAC,BAE,BAE,BAH For an instrument that contains cdmaOne (BAC), NADC (BAE), PDC (BAE), and GSM (BAH).

Recall

***RCL <register>**

This command recalls the instrument state from the specified instrument memory register.

Range: registers are an integer, 0 to 19

Front Panel Access: **File, Recall State**

Reset

***RST**

This command presets the instrument to a factory defined condition that is appropriate for remote programming operation. `*RST` is equivalent to performing the two commands `:SYSTem:PRESet` and `*CLS`. `*RST` does not change the mode and *only* resets the parameters for the current mode.

The `:SYSTem:PRESet` command is equivalent to a front panel **Preset**.

The front panel **Preset** sets instrument parameters to values for good front panel usage in the current mode. The `*RST` and front panel **Preset** will be different. For example, the `*RST` will place the instrument in single sweep while the front panel **Preset** will place the instrument in continuous sweep.

Front Panel Access: **Preset**

Save

`*SAV <register>`

This command saves the instrument state to the specified instrument memory register.

Range: Registers are an integer, 0 to 19

Front Panel Access: **File, Save State**

Service Request Enable

`*SRE <integer>`

`*SRE?`

This command sets the value of the service request enable register.

The query returns the value of the register.

Range: Integer, 0 to 63, or 128 to 191

Read Status Byte Query

`*STB?`

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

Remarks: See `*CLS`

Trigger

`*TRG`

This command triggers the instrument. Use the `:TRIGger[:SEQuence]:SOURce` command to select the trigger

source.

The desired measurement has been selected and is waiting. The command causes the system to exit this “waiting” state and go to the “initiated” state. The trigger system is initiated and completes one full trigger cycle. It returns to the “waiting” state on completion of the trigger cycle. See the MEASure subsystem for more information about controlling the measurement process.

The instrument must be in the single measurement mode. If INIT:CONT ON, then the command is ignored. Depending upon the measurement and the number of averages, there may be multiple data acquisitions, with multiple trigger events, for one full trigger cycle.

Remarks: See also the `:INITiate:IMMediate` command

Front Panel Access: **Restart**

Self Test Query

`*TST?`

This query performs a full self alignment and returns a number indicating the success of the alignment. A zero is returned if the alignment is successful. Same as CAL[:ALL]? and *CAL?

Front Panel Access: **System, Alignments, Align All Now**

Wait-to-Continue

`*WAI`

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

The instrument default is to only wait for completion of the internal self-alignment routines. You must set the STATus:OPERation:EVENt register if you want to look for the completion of additional processes. See the *OPC? command for more information.

Key Type: There is no equivalent front-panel key.

ABORt Subsystem

Abort Command

:ABORt

Stops any sweep or measurement in progress and resets the sweep or trigger system. A measurement refers to any of the measurements found in the **MEASURE** menu.

If :INITiate:CONTinuous is off (single measure), then :INITiate:IMMediate will start a new single measurement.

If :INITiate:CONTinuous is on (continuous measure), a new continuous measurement begins immediately.

The INITiate and/or TRIGger subsystems contain additional related commands.

Front Panel Access: For the continuous measurement mode, the **Restart** key is equivalent to ABORt.

CALCulate Subsystem

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

The SCPI default for data output format is ASCII. The format can be changed to binary with FORMat:DATA which transports faster over the bus.

ACP - Limits

Adjacent Channel Power—Limit Test

:CALCulate:ACP:LIMit:STATe OFF|ON|0|1

:CALCulate:ACP:LIMit:STATe?

Turn limit test on or off.

Factory Preset and *RST: On

Remarks: You must be in Basic, cdmaOne, iDEN mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Limit Test

:CALCulate:ACP:LIMit[:TEST] OFF|ON|0|1

:CALCulate:ACP:LIMit[:TEST]?

Turn limit test on or off.

Factory Preset and *RST: On

Remarks: You must be in the NADC, cdmaOne, or PDC mode to use this command. Use INSTrument:SELect to set the mode.

BbIQ CALCulate Commands

BbIQ in Spectrum - I/Q Marker Query

:CALCulate:SPECtrum:MARKer [1]|2|3|4:IQ?

Reads out current I and Q marker values when spectrum mode is

selected.

Remarks: Implemented for BASIC and W-CDMA modes.

History: Version A.05.00 or later

BbIQ in Waveform - I/Q Marker Query

`:CALCulate:WAVeform:MARKer [1]|2|3|4:IQ?`

Reads out current I and Q marker values when waveform is selected.

Remarks: Implemented for BASIC and W-CDMA modes.

History: Version A.05.00 or later

Test Current Results Against all Limits

`:CALCulate:CLIMits:FAIL?`

Queries the status of the current measurement limit testing. It returns a 0 if the measured results pass when compared with the current limits. It returns a 1 if the measured results fail any limit tests.

Data Query

`:CALCulate:DATA[n]?`

Returns the designated measurement data for the currently selected measurement and sub-opcode.

n = any valid sub-opcode for the current measurement. See "MEASure Group of Commands" on page 301 for information on the data that can be returned for each measurement.

For sub-opcodes that return trace data use the `:CALCulate:DATA[n]:COMPress?` command below.

Calculate/Compress Trace Data Query

`:CALCulate:DATA<n>:COMPress?`
`BLOCk|CFIT|MAXimum|MINimum|MEAN|DMEan|RMS|SAMPle|SDEViation|PPHase [,<soffset>[,<length>[,<roffset>[,<rlimit>]]]]`

Returns compressed data for the specified trace data. The data is returned in the same units as the original trace and only works with the currently selected measurement. The command is used with a sub-opcode <*n*> since measurements usually return several types of

trace data. See the following table for the sub-opcodes for the trace data names that are available in each measurement. For sub-opcodes that return scalar data use the `:CALCulate:DATA[n]?` command above.

This command is used to compress or decimate a long trace to extract and return only the desired data. A typical example would be to acquire N frames of GSM data and return the mean power of the first burst in each frame. The command can also be used to identify the best curve fit for the data.

- BLOCk or block data - returns all the data points from the region of the trace data that you specify. For example, it could be used to return the data points of an input signal over several timeslots, excluding the portions of the trace data that you do not want.
- CFIT or curve fit - applies curve fitting routines to the data. <soffset> and <length> are required to define the data that you want. <roffset> is an optional parameter for the desired order of the curve equation. The query will return the following values: the x-offset (in seconds) and the curve coefficients ((order + 1) values).

MAX, MEAN, MIN, RMS, SAMP, SDEV and PPH return one data value for each specified region (or <length>) of trace data, for as many regions as possible until you run out of trace data (using <roffset> to specify regions). Or they return the number regions you specify (using <rlimit>) ignoring any data beyond that.

- MAXimum - returns the maximum data point for the specified region(s) of trace data. For I/Q trace data, the maximum magnitude of the I/Q pairs is returned.
- MEAN - returns the arithmetic mean of the data point values for the specified region(s) of trace data. See "Mean Value of I/Q Data Points for Specified Region(s)" on page 306. For I/Q trace data, the mean of the magnitudes of the I/Q pairs is returned. See "Mean Value of I/Q Data Pairs for Specified Region(s)" on page 306.

NOTE If the original trace data is in dB, this function returns the arithmetic mean of those log values, not log of the mean power, which is a more useful value.

Equation 5-1 Mean Value of Data Points for Specified Region(s)

$$\text{MEAN} = \frac{1}{n} \sum_{Xi \in \text{ region(s)}} Xi$$

where Xi is a data point value, and n is the number of data points in the specified region(s).

Equation 5-2 Mean Value of I/Q Data Pairs for Specified Region(s)

$$MEAN = \frac{1}{n} \sum_{Xi \in \text{region(s)}} |Xi|$$

where |Xi| is the magnitude of an I/Q pair, and n is the number of I/Q pairs in the specified region(s).

- MINimum - returns the minimum data point for the specified region(s) of trace data For I/Q trace data, the minimum magnitude of the I/Q pairs is returned.
- RMS - returns the arithmetic rms of the data point values for the specified region(s) of trace data. See "RMS Value of Data Points for Specified Region(s)" on page 307.

  For I/Q trace data, the rms of the magnitudes of the I/Q pairs is returned. See "RMS Value of I/Q Data Pairs for Specified Region(s)" on page 307.

NOTE This function is very useful for I/Q trace data. However, if the original trace data is in dB, this function returns the rms of the log values which is not usually needed.

Equation 5-3 RMS Value of Data Points for Specified Region(s)

$$RMS = \sqrt{\frac{1}{n} \sum_{Xi \in \text{region(s)}} Xi^2}$$

where Xi is a data point value, and n is the number of data points in the specified region(s).

Equation 5-4 RMS Value of I/Q Data Pairs for Specified Region(s)

$$RMS = \sqrt{\frac{1}{n} \sum_{Xi \in \text{region(s)}} Xi \, Xi^*}$$

where Xi is the complex value representation of an I/Q pair, Xi its conjugate complex number, and n is the number of I/Q pairs in the specified region(s).*

Once you have the rms value for a region of I/Q trace data, you may want to calculate the mean power. You must convert this rms I/Q value (peak volts) to power in dB.

$$10 \times \log[10 \times (\text{rms value})^2]$$

- SAMPle - returns the first data value for the specified region(s) of trace data. For I/Q trace data, the first I/Q pair is returned.
- SDEViation - returns the arithmetic standard deviation for the data point values for the specified region(s) of trace data. See "Standard Deviation of Data Point Values for Specified Region(s)" on page 307.

  For I/Q trace data, the standard deviation of the magnitudes of the I/Q pairs is returned. See "Standard Deviation of I/Q Data Pair Values for Specified Region(s)" on page 308.

Equation 5-5 Standard Deviation of Data Point Values for Specified Region(s)

$$\text{SDEV} = \sqrt{\frac{1}{n} \sum_{Xi \in \text{region(s)}} (Xi - \overline{X})^2}$$

where Xi is a data point value, X is the arithmetic mean of the data point values for the specified region(s), and n is the number of data points in the specified region(s).

Equation 5-6 Standard Deviation of I/Q Data Pair Values for Specified Region(s)

$$\text{SDEV} = \sqrt{\frac{1}{n} \sum_{Xi \in \text{region(s)}} (|Xi| - \overline{X})^2}$$

where |Xi| is the magnitude of an I/Q pair, X is the mean of the magnitudes for the specified region(s), and n is the number of data points in the specified region(s).

- PPH - returns the pairs of rms power (dBm) and arithmetic mean phase (radian) for every specified region and frequency offset (Hz). The number of pairs is defined by the specified number of regions. Assuming this command can be used for I/Q vector (n=0) in Waveform (time domain) measurement and all parameters are specified by data point in PPH.

  The rms power of the specified region may be expressed as:

  Power = 10 x log [10 x (RMS I/Q value)] + 10.

$$\text{The RMS I/Q value (peak volts)} = \sqrt{\frac{1}{n}\sum_{Xi \in \text{ region}} XiXi^*}$$

where Xi is the complex value representation of an I/Q pair, Xi* its conjugate complex number, and n is the number of I/Q pairs in the specified region.

The arithmetic mean phase of the specified region may be expressed as:

$$\text{Phase} = \frac{1}{n}\sum_{Yi \in \text{ region}} Yi$$

Where Yi is the unwrapped phase of I/Q pair with applying frequency correction and n is the number of I/Q pairs in the specified region.

The frequency correction is made by the frequency offset calculated by the arithmetic mean of every specified region's frequency offset. Each frequency offset is calculated by the least square method against the unwrapped phase of I/Q pair.

Figure 5-1 Sample Trace Data - Constant Envelope

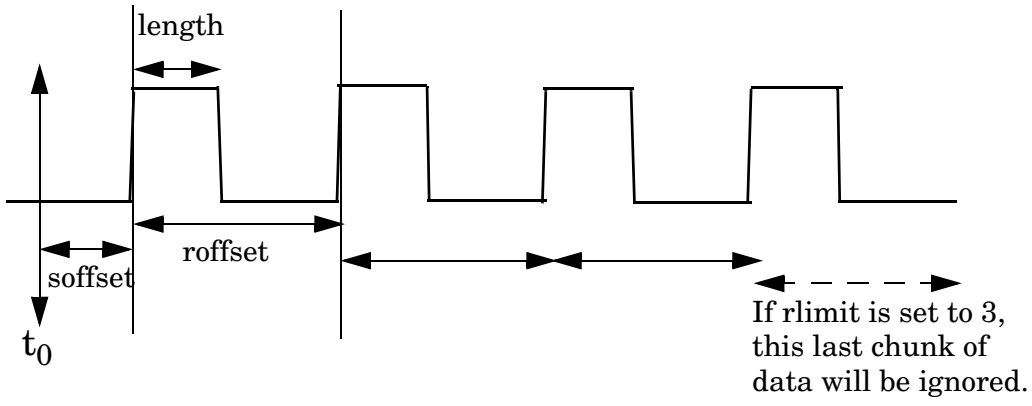
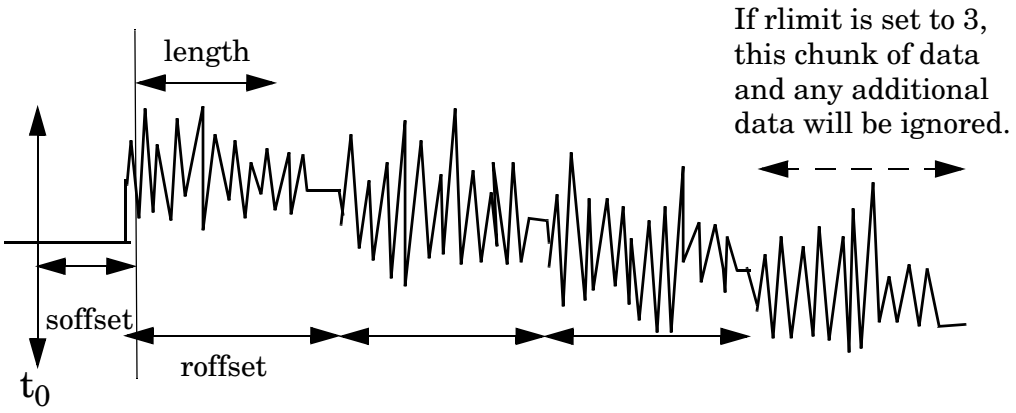


Figure 5-2 Sample Trace Data - Not Constant Envelope



<soffset> - start offset is an optional real number (in seconds). It specifies the amount of data at the beginning of the trace that will be ignored before the decimation process starts. It is the time from the start of the trace to the point where you want to start using the data. The default value is zero.

<length> - is an optional real number (in seconds). It defines how much data will be compressed into one value. This parameter has a default value equal to the current trace length.

<roffset> - repeat offset is an optional real number (in seconds). It defines the beginning of the next field of trace elements to be compressed. This is relative to the beginning of the previous field. This parameter has a default value equal to the <length> variable.

<rlimit> - repeat limit is an optional integer. It specifies the number of data items that you want returned. It will ignore any additional items beyond that number. You can use the Start offset and the Repeat limit to pick out exactly what part of the data you want to use. The default value is all the data.

Example: To query the mean power of a set of GSM bursts:

1. Set the waveform measurement sweep time to acquire at least one burst.
2. Set the triggers such that acquisition happens at a known position relative to a burst.
3. Then query the mean burst levels using, `CALC:DATA2:COMP? MEAN,24e-6,526e-6` (These parameter values correspond to GSM signals, where 526e-6 is the length of the burst in the slot and you just want 1 burst.)

NOTE There is a more detailed example in "Improving Measurement Speed" on page 87.

Remarks: The optional parameters must be entered in the specified order. For example, if you want to specify <length>, you must also specify <soffset>.

This command uses the data in the format specified by FORMat:DATA, returning either binary or ASCII data.

History: Added in revision A.03.00

Changed in revision A.05.00

Measurement	**Available Traces**	**Markers Available?**
ACP - adjacent channel power (Basic, cdmaOne, cdma2000, W-CDMA, iDEN, NADC, PDC modes)	no traces (*n*=0)[a] for I/Q points	no markers
BER - bit error rate (iDEN mode)	no traces (*n*=0)[a] for I/Q data	no markers
CDPower - code domain power (cdmaOne mode)	POWer (*n*=2)[a] TIMing (*n*=3)[a] PHASe (*n*=4)[a] (*n*=0)[a] for I/Q points	yes
CDPower - code domain power (cdma2000, W-CDMA modes)	CDPower (*n*=2)[a] EVM (*n*=5)[a] MERRor (*n*=6)[a] PERRor (*n*=7)[a] SPOWer (*n*=9)[a] CPOWer (*n*=10)[a] (*n*=0)[a] for I/Q points	yes
CHPower - channel power (Basic, cdmaOne, cdma2000, W-CDMA modes)	SPECtrum (*n*=2)[a] (*n*=0)[a] for I/Q points	no markers
CSPur - spurs close (cdmaOne mode)	SPECtrum (*n*=2)[a] ULIMit (*n*=3)[a] (*n*=0)[a] for I/Q points	yes

Measurement	Available Traces	Markers Available?
EEVM - EDGE error vector magnitude (EDGE mode)	EVMerror (*n*=2)[a] MERRor (*n*=3)[a] PERRor (*n*=4)[a] (*n*=0)[a] for I/Q points	yes
EORFspectr - EDGE output RF spectrum (EDGE mode)	RFEMod (*n*=2)[a] RFESwitching (*n*=3)[a] SPEMod (*n*=4)[a] LIMMod (*n*=5)[a] (*n*=0)[a] for I/Q points	yes, only for a single offset yes, only for multiple offsets
EPVTime - EDGE power versus time (EDGE mode)	RFENvelope (*n*=2)[a] UMASk (*n*=3)[a] LMASk (*n*=4)[a] (*n*=0)[a] for I/Q points	yes
ETSPur - EDGE transmit band spurs (EDGE mode)	SPECtrum (*n*=2)[a] ULIMit (*n*=3)[a] (*n*=0)[a] for I/Q points	yes
EVM - error vector magnitude (NADC, PDC modes)	EVM (*n*=2)[a] MERRor (*n*=3)[a] PERRor (*n*=4)[a] (*n*=0)[a] for I/Q points	yes
EVMQpsk - QPSK error vector magnitude (cdma2000, W-CDMA modes)	EVM (*n*=2)[a] MERRor (*n*=3)[a] PERRor (*n*=4)[a] (*n*=0)[a] for I/Q points	yes
IM - intermodulation (cdma2000, W-CDMA modes)	SPECtrum (*n*=2)[a] (*n*=0)[a] for I/Q points	yes
MCPower - multi-carrier power (W-CDMA mode)	no traces (*n*=0)[a] for I/Q points	no markers

Measurement	Available Traces	Markers Available?
OBW - occupied bandwidth (cdmaOne, cdma2000, iDEN, PDC, W-CDMA modes)	no traces (*n*=0)[a] for I/Q points	no markers
ORFSpectrum - output RF spectrum (GSM, EDGE mode)	RFEMod (*n*=2)[a] RFESwitching (*n*=3)[a] SPEMod (*n*=4)[a] LIMMod (*n*=5)[a] (*n*=0)[a] for I/Q points	yes, only for a single offset yes, only for multiple offsets
PFERror - phase and frequency error (GSM, EDGE mode)	PERRor (*n*=2)[a] PFERror (*n*=3)[a] RFENvelope (*n*=4)[a] (*n*=0)[a] for I/Q points	yes
PSTatistic - power statistics CCDF (Basic, cdma2000, W-CDMA modes)	MEASured (*n*=2)[a] GAUSian (*n*=3)[a] REFerence (*n*=4)[a] (*n*=0)[a] for I/Q points	yes
PVTime - power versus time (GSM, EDGE, Service modes)	RFENvelope (*n*=2)[a] UMASk (*n*=3)[a] LMASk (*n*=4)[a] (*n*=0)[a] for I/Q points	yes
RHO - modulation quality (cdmaOne, cdma2000, W-CDMA mode)	(*n*=0)[a] for I/Q points EVM (*n*=2)[a] MERRor (*n*=3)[a] PERRor (*n*=4)[a] (*n*=0)[a] for I/Q points	yes
SEMask - spectrum emissions mask (cdma2000, W-CDMA mode)	SPECtrum (*n*=2)[a] (*n*=0)[a] for I/Q points	yes

Measurement	Available Traces	Markers Available?
TSPur - transmit band spurs (GSM, EDGE mode)	SPECtrum (n=2)[a] ULIMit (n=3)[a] (n=0)[a] for I/Q points	yes
TXPower - transmit power (GSM, EDGE mode)	RFENvelope (n=2)[a] IQ (n=8)[a] (n=0)[a] for I/Q points	yes
SPECtrum - (frequency domain) (all modes)	RFENvelope (n=2)[a] for Service mode IQ (n=3)[a] SPECtrum (n=4)[a] ASPectrum (n=7)[a] (n=0)[a] for I/Q points	yes
WAVEform - (time domain) (all modes)	RFENvelope (n=2)[a] (also for Signal Envelope trace) IQ (n=5)[a] (n=0)[a] for I/Q points	yes

a. The *n* number indicates the sub-opcode that corresponds to this trace. Detailed descriptions of the trace data can be found in the MEASure subsystem documentation by looking up the sub-opcode for the appropriate measurement.

Calculate Peaks of Trace Data

:CALCulate:DATA[n]:PEAKs?
<threshold>,<excursion>[,AMPLitude|FREQuency|TIME]

Returns a list of peaks for the designated trace data *n* for the currently selected measurement. The peaks must meet the requirements of the peak threshold and excursion values.

The command can be used with sub-opcodes (*n*) for any measurement results that are trace data. See the table above. Subopcode *n*=0, raw trace data cannot be searched for peaks. Both real and complex traces can be searched, but complex traces are converted to magnitude in dBm.

Threshold - is the level below which trace data peaks are ignored

Excursion - To be defined as a peak, the signal must rise above the threshold by a minimum amplitude change. Excursion is measured from the lowest point above the threshold (of the rising edge of the peak), to the highest signal point that begins the falling edge.

Amplitude - lists the peaks in order of descending amplitude, so the highest peak is listed first. This is the default peak order listing if the optional parameter is not specified.

Frequency - lists the peaks in order of occurrence, left to right across the x-axis

Time - lists the peaks in order of occurrence, left to right across the x-axis

Example: Select the spectrum measurement.

Use `CALC:DATA4:PEAK? -40,10,FREQ` to identify the peaks above –40 dBm, with excursions of at least 10 dB, in order of increasing frequency.

Query Results: Returns a list of floating-point numbers. The first value in the list is the number of peak points that follow. A peak point consists of two values: a peak amplitude followed by the its corresponding frequency (or time).

If no peaks are found the peak list will consist of only the number of peaks, (0).

The peak list is limited to 100 peaks. Peaks in excess of 100 are ignored.

Remarks: This command uses the data setting specified by the FORMat:DATA command and can return real 32-bit, real 64-bit, or ASCII data. The default data format is ASCII.

History: Added in revision A.03.00 and later

CALCulate:MARKers Subsystem

Markers can be put on your displayed measurement data to supply information about specific points on the data. Some of the things that markers can be used to measure include: precise frequency at a point, minimum or maximum amplitude, and the difference in amplitude or frequency between two points.

When using the marker commands you must specify the measurement in the SCPI command. We recommend that you use the marker commands only on the current measurement. Many marker commands will return invalid results, when used on a measurement that is not

current. (This is true for commands that do more than simply setting or querying an instrument parameter.) No error is reported for these invalid results.

You must make sure that the measurement is completed before trying to query the marker value. Using the MEASure or READ command, before the marker command, forces the measurement to complete before allowing the next command to be executed.

Each measurement has its own instrument state for marker parameters. Therefore, if you exit the measurement, the marker settings in each measurement are saved and are then recalled when you change back to that measurement.

Basic Mode - <measurement> key words

- ACPr - no markers
- CHPower - no markers
- PSTatistic - markers available
- SPECtrum - markers available
- WAVeform - markers available

Service Mode - <measurement> key words

- PVTime - no markers
- SPECtrum - markers available
- WAVeform - markers available

cdmaOne Mode - <measurement> key words

- ACPr - no markers
- CHPower - no markers
- CDPower - markers available
- CSPur - markers available
- RHO - markers available
- SPECtrum - markers available
- WAVeform - markers available

cdma2000 Mode - <measurement> key words

- ACP - no markers
- CDPower - markers available
- CHPower - no markers
- EVMQpsk - markers available
- IM - markers available
- OBW - no markers
- PSTatistic - markers available
- RHO - markers available
- SEMask - markers available
- SPECtrum - markers available
- WAVeform - markers available

EDGE (with GSM) Mode - <measurement> key words

- EEVM - markers available
- EORFspectr - markers available
- EPVTime - no markers
- ORFSpectrum - markers available
- PFERror - markers available
- PVTime - no markers
- SPECtrum - markers available
- TSPur - markers available
- TXPower - no markers
- WAVeform - markers available

GSM Mode - <measurement> key words

- ORFSpectrum - markers available
- PFERror - markers available
- PVTime - no markers
- SPECtrum - markers available
- TSPur - markers available
- TXPower - no markers
- WAVeform - markers available

iDEN Mode - <measurement> key words

- ACP - no markers
- BER - no markers
- OBW - no markers
- SPECtrum - markers available
- WAVeform - markers available

NADC Mode - <measurement> key words

- ACP - no markers
- EVM - markers available
- SPECtrum - markers available
- WAVeform - markers available

PDC Mode - <measurement> key words

- ACP - no markers
- EVM - markers available
- OBW - no markers
- SPECtrum - markers available
- WAVeform - markers available

W-CDMA (3GPP) Mode - <measurement> key words

- ACP - no markers
- CDPower - markers available
- CHPower - no markers
- EVMQpsk - markers available

- IM - markers available
- MCPower - no markers
- OBW - no markers
- PSTatistic - markers available
- RHO - markers available
- SEMask - markers available
- SPECtrum - markers available
- WAVeform - markers available

Example:

Suppose you are using the Spectrum measurement. To position marker 2 at the maximum peak value of the trace that marker 2 is currently on, the command is:

```
:CALCulate:SPECtrum:MARKer2:MAXimum
```

You must make sure that the measurement is completed before trying to query the marker value. Use the MEASure or READ command before using the marker command. This forces the measurement to complete before allowing the next command to be executed.

Markers All Off on All Traces

:CALCulate:<measurement>:MARKer:AOFF

Turns off all markers on all the traces in the specified measurement.

Example: `CALC:SPEC:MARK:AOFF`

Remarks: The keyword for the current measurement must be specified in the command. (Some examples include: SPECtrum, WAVeform)

Front Panel Access: **Marker, More, Marker All Off**

Marker Function

:CALCulate:<measurement>:MARKer[1]|2|3|4:FUNCtion BPOWer|NOISe|OFF

:CALCulate:<measurement>:MARKer[1]|2|3|4:FUNCtion?

Selects the type of marker for the specified marker. A particular measurement may not have all the types of markers that are commonly available.

The marker must have already been assigned to a trace. Use `:CALCulate:<measurement>:MARKer[1]|2|3|4:TRACe` to assign a marker to a particular trace.

Band Power – is the integrated power between the two markers for traces in the frequency domain and is the mean power between the two markers for traces in the time domain.

Noise – is the noise power spectral density in a 1 Hz bandwidth. It is averaged over 32 horizontal trace points.

Off – turns off the marker functions

Example: `CALC:SPEC:MARK3:FUNC Noise`

Remarks: The keyword for the current measurement must be specified in the command. (Some examples include: SPECtrum, WAVeform)

Front Panel Access: **Marker, Marker Function**

Marker Function Result

`:CALCulate:<measurement>:MARKer[1]|2|3|4:FUNCtion:RESult?`

Quires the result of the currently active marker function. The measurement must be completed before querying the marker. A particular measurement may not have all the types of markers available.

The marker must have already been assigned to a trace. Use `:CALCulate:<measurement>:MARKer[1]|2|3|4:TRACe` to assign a marker to a particular trace.

Example: `CALC:SPEC:MARK:FUNC:RES?`

Remarks: The keyword for the current measurement must be specified in the command. (Some examples include: SPECtrum, WAVeform)

Front Panel Access: **Marker, Marker Function**

Marker Peak (Maximum) Search

`:CALCulate:<measurement>:MARKer[1]|2|3|4:MAXimum`

Places the selected marker on the highest point on the trace that is assigned to that particular marker number.

The marker must have already been assigned to a trace. Use `:CALCulate:<measurement>:MARKer[1]|2|3|4:TRACe` to assign a marker to a particular trace.

Example: `CALC:SPEC:MARK1:MAX`

Remarks: The keyword for the current measurement must be specified in the command. (Some examples include: SPECtrum, WAVeform)

Front Panel Access: **Search**

Marker Peak (Minimum) Search

:CALCulate:<measurement>:MARKer[1]|2|3|4:MINimum

Places the selected marker on the lowest point on the trace that is assigned to that particular marker number.

The marker must have already been assigned to a trace. Use :CALCulate:<measurement>:MARKer[1]|2|3|4:TRACe to assign a marker to a particular trace.

Example: `CALC:SPEC:MARK2 MIN`

Remarks: The keyword for the current measurement must be specified in the command. (Some examples include: SPECtrum, WAVeform)

Marker Mode

:CALCulate:<measurement>:MARKer[1]|2|3|4:MODE POSition|DELTa

:CALCulate:<measurement>:MARKer[1]|2|3|4:MODE?

Selects the type of marker to be a normal position-type marker or a delta marker. A specific measurement may not have both types of markers. For example, several measurements only have position markers.

The marker must have already been assigned to a trace. Use :CALCulate:<measurement>:MARKer[1]|2|3|4:TRACe to assign a marker to a particular trace.

Example: `CALC:SPEC:MARK:MODE DELTA`

Remarks: For the delta mode only markers 1 and 2 are valid.

The keyword for the current measurement must be specified in the command. (Some examples include: SPECtrum, WAVeform)

Front Panel Access: **Marker, Marker [Delta]**

Marker On/Off

:CALCulate:<measurement>:MARKer[1]|2|3|4[:STATe] OFF|ON|0|1

:CALCulate:<measurement>:MARKer[1]|2|3|4[:STATe]?

Turns the selected marker on or off.

The marker must have already been assigned to a trace. Use :CALCulate:<measurement>:MARKer[1]|2|3|4:TRACe to assign a marker to a particular trace.

Example: CALC:SPEC:MARK2: on

Remarks: The keyword for the current measurement must be specified in the command. (Some examples include: SPECtrum, AREFerence, WAVeform)

The WAVeform measurement only has two markers available.

Front Panel Access: **Marker, Select** then **Marker Normal** or **Marker On Off**

Marker to Trace

:CALCulate:<measurement>:MARKer[1]|2|3|4:TRACe <trace_name>

:CALCulate:<measurement>:MARKer[1]|2|3|4:TRACe?

Assigns the specified marker to the designated trace. Not all types of measurement data can have markers assigned to them.

Example: With the WAVeform measurement selected, a valid command is CALC:SPEC:MARK2:TRACE rfenvelope.

Range: The names of valid traces are dependent upon the selected measurement. See the following table for the available trace names. The trace name assignment is independent of the marker number.

Remarks: The keyword for the current measurement must be specified in the command. (Some examples include: SPECtrum, WAVeform)

Front Panel

Access: **Marker, Marker Trace**

Measurement	Available Traces	Markers Available?
ACP - adjacent channel power (Basic, cdmaOne, cdma2000, W-CDMA (3GPP), iDEN, NADC, PDC modes)	no traces	no markers
BER - bit error rate (iDEN mode)	no traces	no markers
CDPower - code domain power (cdmaOne mode)	POWer (*n*=2)[a] TIMing (*n*=3)[a] PHASe (*n*=4)[a]	yes
CDPower - code domain power (cdma2000, W-CDMA (3GPP) modes)	CDPower (*n*=2)[a] EVM (*n*=5)[a] MERRor (*n*=6)[a] PERRor (*n*=7)[a] SPOWer (*n*=9)[a] CPOWer (*n*=10)[a]	yes
CHPower - channel power (Basic, cdmaOne, cdma2000, W-CDMA (3GPP) modes)	SPECtrum (*n*=2)[a]	no markers
CSPur - spurs close (cdmaOne mode)	SPECtrum (*n*=2)[a] ULIMit (*n*=3)[a]	yes
EEVM - EDGE error vector magnitude (EDGE mode)	EVMerror (*n*=2)[a] MERRor (*n*=3)[a] PERRor (*n*=4)[a]	yes
EORFspectr - EDGE output RF spectrum (EDGE mode)	RFEMod (*n*=2)[a] RFESwitching (*n*=3)[a] SPEMod (*n*=4)[a] LIMMod (*n*=5)[a]	yes, only for a single offset yes, only for multiple offsets

Measurement	Available Traces	Markers Available?
EPVTime - EDGE power versus time (EDGE mode)	RFENvelope (*n*=2)[a] UMASk (*n*=3)[a] LMASk (*n*=4)[a]	yes
EVM - error vector magnitude (NADC, PDC modes)	EVM (*n*=2)[a] MERRor (*n*=3)[a] PERRor (*n*=4)[a]	yes
EVMQpsk - QPSK error vector magnitude (cdma2000, W-CDMA (3GPP) modes)	EVM (*n*=2)[a] MERRor (*n*=3)[a] PERRor (*n*=4)[a]	yes
IM - intermodulation (cdma2000, W-CDMA (3GPP) modes)	SPECtrum (*n*=2)[a]	yes
MCPower - multi-carrier power (W-CDMA (3GPP) mode)	no traces	no markers
OBW - occupied bandwidth (cdmaOne, cdma2000, iDEN, PDC, W-CDMA (3GPP) modes)	no traces	no markers
ORFSpectrum - output RF spectrum (GSM mode)	RFEMod (*n*=2)[a] RFESwitching (*n*=3)[a] SPEMod (*n*=4)[a] LIMMod (*n*=5)[a]	yes, only for a single offset yes, only for multiple offsets
PFERror - phase and frequency error (GSM mode)	PERRor (*n*=2)[a] PFERror (*n*=3)[a] RFENvelope (*n*=4)[a]	yes
PSTatistic - power statistics CCDF (Basic, cdma2000, W-CDMA (3GPP) modes)	MEASured (*n*=2)[a] GAUSian (*n*=3)[a] REFerence (*n*=4)[a]	yes

Measurement	Available Traces	Markers Available?
PVTime - power versus time (GSM, Service modes)	RFENvelope (*n*=2)[a] UMASk (*n*=3)[a] LMASk (*n*=4)[a]	yes
RHO - modulation quality (cdmaOne, cdma2000, W-CDMA (3GPP) modes)	EVM (*n*=2)[a] MERRor (*n*=3)[a] PERRor (*n*=4)[a]	yes
SEMask - spectrum emissions mask (cdma2000, W-CDMA (3GPP) mode)	SPECtrum (*n*=2)[a]	yes
TSPur - transmit band spurs (GSM mode)	SPECtrum (*n*=2)[a] ULIMit (*n*=3)[a]	yes
TXPower - transmit power (GSM mode)	RFENvelope (*n*=2)[a] IQ (*n*=8)[a]	yes
SPECtrum - (frequency domain) (all modes)	RFENvelope (*n*=2)[a] for Service mode IQ (*n*=3)[a] SPECtrum (*n*=4)[a] ASPectrum (*n*=7)[a]	yes
WAVEform - (time domain) (all modes)	RFENvelope (*n*=2)[a] IQ (*n*=8)[a]	yes

a. The *n* number indicates the sub-opcode that corresponds to this trace. Detailed descriptions of the trace data can be found in the MEASure subsystem documentation by looking up the sub-opcode for the appropriate measurement.

Marker X Value

:CALCulate:<measurement>:MARKer[1]|2|3|4:X <param>

:CALCulate:<measurement>:MARKer[1]|2|3|4:X?

Position the designated marker on its assigned trace at the specified X value. The parameter value is in X-axis units (which is often frequency or time).

The marker must have already been assigned to a trace. Use :CALCulate:<measurement>:MARKer[1]|2|3|4:TRACe to assign a marker to a particular trace.

The query returns the current X value of the designated marker. The measurement must be completed before querying the marker.

Example: CALC:SPEC:MARK2:X 1.2e6 Hz

Default Unit: Matches the units of the trace on which the marker is positioned

Remarks: The keyword for the current measurement must be specified in the command. (Some examples include: SPECtrum, WAVeform)

Front Panel Access: **Marker, <active marker>, RPG**

Marker X Position

:CALCulate:<measurement>:MARKer[1]|2|3|4:X:POSition <integer>

:CALCulate:<measurement>:MARKer[1]|2|3|4:X:POSition?

Position the designated marker on its assigned trace at the specified X position. A trace is composed of a variable number of measurement points. This number changes depending on the current measurement conditions. The current number of points must be identified before using this command to place the marker at a specific location.

The marker must have already been assigned to a trace. Use :CALCulate:<measurement>:MARKer[1]|2|3|4:TRACe to assign a marker to a particular trace.

The query returns the current X position for the designated marker. The measurement must be completed before querying the marker.

Example: CALC:SPEC:MARK:X:POS 500

Range: 0 to a maximum of (3 to 920,000)

Remarks: The keyword for the current measurement must be specified in the command. (Some examples include: SPECtrum, WAVeform)

Front Panel Access: **Marker, <active marker>, RPG**

Marker Readout Y Value

:CALCulate:<measurement>:MARKer[1]|2|3|4:Y?

Readout the current Y value for the designated marker on its assigned trace. The value is in the Y-axis units for the trace (which is often dBm).

The marker must have already been assigned to a trace. Use `:CALCulate:<measurement>:MARKer[1]|2|3|4:TRACe` to assign a marker to a particular trace.

The measurement must be completed before querying the marker.

Example: `CALC:SPEC:MARK1:Y?`

Default Unit: Matches the units of the trace on which the marker is positioned

Remarks: The keyword for the current measurement must be specified in the command. (Some examples include: SPECtrum, WAVeform)

Power Statistic CCDF—Store Reference

`:CALCulate:PSTatistic:STORe:REFerence ON|OFF|1|0`

Store the current measured trace as the user-defined reference trace.

Remarks: You must be in the cdma2000 or W-CDMA (3GPP) mode to use this command. Use INSTrument:SELect to set the mode.

CALibration Subsystem

These commands control the self-alignment and self-diagnostic processes.

Calibration Abort

:CALibration:ABORt

Abort any alignment in progress.

Front Panel Access: **ESC**, when alignment is in progress

Align the ADC Auto-range Threshold

:CALibration:ADC:ARANge

:CALibration:ADC:ARANge?

Align the ADC auto-range thresholds. This same alignment is run as part of the CAL:ALL routine.

The query performs the alignment and returns a zero if the alignment is successful.

Front Panel Access: **System, Alignments, Align subsystem, Align ADC**

Align the ADC Dither Center Frequency

:CALibration:ADC:DITHer

:CALibration:ADC:DITHer?

Align the ADC dithering center frequency. This same alignment is run as part of the CAL:ALL routine.

The query performs the alignment and returns a zero if the alignment is successful.

Front Panel Access: **System, Alignments, Align subsystem, Align ADC**

Align the ADC Offset

:CALibration:ADC:OFFSet

:CALibration:ADC:OFFSet?

Align the six ADC offset DACs. This same alignment is run as part of the CAL:ALL routine.

The query performs the alignment and returns a zero if the alignment is successful.

Front Panel Access: **System, Alignments, Align subsystem, Align ADC**

Align the ADC RAM Gain

:CALibration:ADCRam:GAIN

:CALibration:ADCRam:GAIN?

Align the gain of the six ADC RAM pages. This same alignment is run as part of the CAL:ALL routine.

The query performs the alignment and returns a zero if the alignment is successful.

Front Panel Access: **System, Alignments, Align subsystem, Align ADC???**

Align All Instrument Assemblies

:CALibration[:ALL]

:CALibration[:ALL]?

Performs an alignment of all the assemblies within the instrument.

The query performs a full alignment and returns a number indicating the success of the alignment. A zero is returned if the alignment is successful. A one is returned if any part of the alignment failed.

Front Panel Access: **System, Alignments, Align All Now**

Calibrate the Attenuator

:CALibration:ATTenuator

:CALibration:ATTenuator?

Calculate the gain error of 40 RF attenuator steps. The nominal setting of 10 dB is assumed to have 0 dB error.

The query performs the alignment and returns a zero if the alignment

is successful.

Remarks: A valid service password needs to be entered prior to sending the command.

Front Panel Access: **System, Alignments, Align subsystem, RF**

Automatic Alignment

:CALibration:AUTO OFF|ALERT|ON

:CALibration:AUTO?

Turns the automatic alignment routines on and off. When turned on, they are run once every 5 minutes, or if the ambient temperature changes by 3 degrees.

If alignment is turned off, the instrument may drift out of specification. The alert mode allows you to turn off the automatic alignment, but reminds you to when to run the alignment again. You will get a warning message if 24 hours has expired or the temperature has change by 3 degrees since the last alignment.

Factory Preset and *RST: Alert

Your setting for the auto alignment is persistent and will remain the same even through an instrument power cycle.

Front Panel Access: **System, Alignments, Auto Align**

Calibration Comb Alignment

:CALibration:COMB

:CALibration:COMB?

Aligns the comb frequencies by measuring them relative to the internal 50 MHz reference signal.

The query performs the alignment and returns a zero if the alignment is successful.

Remarks: A valid service password needs to be entered prior to sending the command.

Front Panel Access: **System, Alignments, Align Subsystem, RF**

Turn Background Calibration Corrections Off

:CALibration:CORRections 0|1|OFF|ON

:CALibration:CORRections?

When set to OFF deactivates background flatness and IF gain alignments, for which nominal values are substituted. Several video shift gain corrections are set to zero, including absolute gain err, gain err vs attenuation, and RF flatness err vs frequency. The IF gain DAC is not compensated to adjust for prefilter gain but is set to a nominal value. Typically used to facilitate troubleshooting.

Factory Preset and *RST: ON

Front Panel Access: **System**, **Alignment**, **Corrections**

Calibration Display Detail

:CALibration:DISPlay:LEVel OFF|LOW|HIGH

:CALibration:DISPlay:LEVel?

Controls the amount of detail shown on the display while the alignment routines are running. The routines run faster if the display level is off, so they do not have to update the display.

Off - displays no trace points

Low - displays every 10^{th} trace

High - displays every trace

Factory Preset and *RST: Low

Front Panel Access: **System, Alignments, Visible Align**

Align the Image Filter Circuitry

:CALibration:FILTer:IMAGe

:CALibration:FILTer:IMAGe?

Align the eight image filter tuning DACs.

The query performs the alignment and returns a zero if the alignment is successful.

Remarks: A valid service password needs to be entered prior to sending the command.

Front Panel Access: **System, Diagnostics**

Align the IF Flatness

:CALibration:FLATness:IF

:CALibration:FLATness:IF?

Finds the flatness shape of the current IF setup (prefilter, mgain, natBW). This information is then used for compensating measurements that use FFT functionality, like the spectrum measurement. The alignment is done frequently in the background. This same alignment is run as part of the CAL:ALL routine.

The query performs the alignment and returns a zero if the alignment is successful.

Front Panel Access: Select **Timebase Freq** under **Measure**, then press **Meas Setup, Auto Adjust Now**.

Auto Adjust the Internal 10 MHz Frequency Reference

:CALibration:FREQuency:REFerence:AADJust

Auto adjustment of the internal frequency reference (10 MHz timebase).

Remarks: You must be in the Service mode to use this command. Use INSTrument:SELect.

Requires the current measurement to be timebase frequency. A valid password needs to be entered sometime prior to sending this command. See the timebase frequency measurement for more information.

Front Panel Access: Select **Timebase Freq** under **Measure**, then press **Meas Setup, Auto Adjust Now**.

Align the ADC

:CALibration:GADC

:CALibration:GADC?

Performs the ADC group of alignments. The query returns a 0 if the alignments occurred without problems.

The query performs the alignment and returns a zero if the alignment is successful.

Front Panel Access: **System, Alignments, Align Subsystem, Align ADC**

Align the IF Gain

:CALibration:GAIN:IF

:CALibration:GAIN:IF?

Calculate the curve coefficients for the IF gain DAC.

The query performs the alignment and returns a zero if the alignment is successful.

Front Panel Access: **System, Alignments, Align Subsystem, IF**

Align the Baseband IQ

:CALibration:GIQ

:CALibration:GIQ?

Performs the IQ group of alignments. The query performs the alignment and returns a 0 if the alignment is successful.

Remarks: Implemented for BASIC and W-CDMA modes.

History: Version A.05.00 or later

BbIQ in Spectrum - IQ Common Mode Response Null

:CALibration:IQ:CMR

:CALibration:IQ:CMR?

Forces a common mode response null on I/Q inputs.

Remarks: Implemented for BASIC and W-CDMA modes.

History: Version A.05.00 or later

BbIQ in Spectrum - IQ Flatness Calibration

:CALibration:IQ:FLATness

:CALibration:IQ:FLATness?

Activates a flatness calibration for all I/Q ranges and impedance settings.

Remarks: Implemented for BASIC and W-CDMA modes.

History: Version A.05.00 or later

BbIQ in Spectrum - IQ Offset Calibration

:CALibration:IQ:OFFSet

:CALibration:IQ:OFFSet?

Activates a calibration of the I/Q input offset DAC.

Remarks: Implemented for BASIC and W-CDMA modes.

History: Version A.05.00 or later

Calibrate the Nominal System Gain

:CALibration:GAIN:CSYStem

:CALibration:GAIN:CSYStem?

Calculate the current system gain correction for nominal settings. That is, with 10 dB attenuation, 500 MHz center frequency, 0 dB IF gain and the prefilter off.

Front Panel Access: **System, Alignments, Align Subsystem, IF**

Align the IF

:CALibration:GIF

:CALibration:GIF?

Performs the IF group of alignments. The query returns a 0 if the alignments occurred without problems.

The query performs the alignment and returns a zero if the alignment is successful.

Front Panel Access: **System, Alignments, Align Subsystem, Align IF**

Align the RF

:CALibration:GRF

:CALibration:GRF?

Performs the RF group of alignments. The query returns a 0 if the alignments occurred without problems.

The query performs the alignment and returns a zero if the alignment is successful.

Front Panel Access: **System, Alignments, Align Subsystem, Align RF**

Load the Factory Default Calibration Constants

:CALibration:LOAD:DEFault

Load the factory default alignment data, ignoring the effect of any alignments already done.

Front Panel Access: **System, Alignments, Restore Align Defaults**

Align the Narrow LC Prefilter

:CALibration:PFILter:LC:NARRow

:CALibration:PFILter:LC:NARRow?

Align the narrow LC prefilter. (200 kHz to 1.2 MHz)

The query performs the alignment and returns a zero if the alignment is successful.

Remarks: A valid service password needs to be entered prior to sending the command.

Front Panel Access: **System, Alignments, Align Subsystem, IF**

Align the Wide LC Prefilter

:CALibration:PFILter:LC:WIDE

:CALibration:PFILter:LC:WIDE?

Align the wide LC prefilter. (1.2 MHz to 7.5 MHz)

The query performs the alignment and returns a zero if the alignment

is successful.

Remarks: A valid service password needs to be entered prior to sending the command.

Front Panel Access: **System, Diagnostics**

Align the Narrow Crystal Prefilter

:CALibration:PFILter:XTAL:NARRow

:CALibration:PFILter:XTAL:NARRow?

Align the narrow crystal prefilter. (2.5 kHz to 20 kHz)

The query performs the alignment and returns a zero if the alignment is successful.

Remarks: A valid service password needs to be entered prior to sending the command.

Front Panel Access: Enter service password and press **System, Diagnostics**

Align the Wide Crystal Prefilter

:CALibration:PFILter:XTAL:WIDE

:CALibration:PFILter:XTAL:WIDE?

Align the wide crystal prefilter. (20 kHz to 200 kHz)

The query performs the alignment and returns a zero if the alignment is successful.

Remarks: A valid service password needs to be entered prior to sending the command.

Front Panel Access: Enter service password and press **System, Diagnostics**

Adjust the Level of the 321.4 MHz Alignment Signal

:CALibration:REF321

:CALibration:REF321?

Calculate the curve coefficients for setting the level of the 321.4 MHz alignment signal.

Remarks: A valid service password needs to be entered prior to sending the command.

Front Panel Access: **System, Diagnostics**

50 MHz Reference Alignment Signal

Process	Process Step Description	Command
Both	Attach a 50 MHz signal to the RF input.	
Automatic	Does the entire procedure	CAL:REF50[:DOIT]
Interactive	Enter the interactive mode	CAL:REF50:ENTer
Interactive	Tell the instrument what the external signal's power is. (approx. -25 dBm)	CAL:REF50:AMPL
Interactive	Proceed with the adjustment phase.	CAL:REF50:ANOW
Interactive	Exit from the interactive mode.	CAL:REF50:EXIT
Query	Return the last alignment value of the absolute level of the 50 MHz cal signal.	CAL:REF50:LAST:ALEVel?
Query	Return the last alignment value of the ALC DAC.	CAL:REF50:LAST:ALCDac?

External Signal Power for Internal 50 MHz Amplitude Reference Alignment

`:CALibration:REF50:AMPL <power>`

`:CALibration:REF50:AMPL?`

You must set this value equal to the actual amplitude of the external 50 MHz amplitude reference signal applied to the RF INPUT connector. This is used for aligning the 50 MHz amplitude reference with CAL:REF50.

Preset and *RST: –25.00 dBm

Range: –30 to –20 dBm

Default Unit: dBm

Remarks: You must be in the Service mode to use this command. Use INSTrument:SELect.

A valid service password needs to be entered prior to sending this command.

Front Panel Access: **System, Alignments, Align subsystem, Align 50 MHz Reference**

Internal 50 MHz Amplitude Reference Alignment Control

`:CALibration:REF50:ANOW`

Immediately does the automatic alignment of the internal 50 MHz amplitude reference oscillator. This command is used with the interactive mode of the 50 MHz alignment, i.e. CAL:REF50:ENTer.

Remarks: You must be in the Service mode to use this command. Use INSTrument:SELect.

A valid service password needs to be entered prior to sending this command.

Front Panel Access: **System, Alignments, Align subsystem, Align 50 MHz Reference**

Internal 50 MHz Amplitude Reference Alignment Control

`:CALibration:REF50[:DOIT]`

`:CALibration:REF50[:DOIT]?`

Does automatic alignment of the internal 50 MHz amplitude reference oscillator. You do this by setting an external source to –25.00 dBm and using a power meter to measure the exact value. Then use CAL:REF50:AMPL to input the source amplitude, measured on the power meter. Finally, connect the source to the instrument RF INPUT port and run the adjustment.

Remarks: You must be in the Service mode to use this command. Use INSTrument:SELect.

A valid service password needs to be entered prior to sending this command

Front Panel Access: **System, Alignments, Align subsystem, Align 50 MHz Reference**

Enter Interactive Mode for Internal 50 MHz Amplitude Reference Alignment

:CALibration:REF50:ENTer

Turns on the interactive mode for alignment of the internal 50 MHz amplitude reference signal. Use CAL:REF50:ANOW to do the alignment and CAL:REF50:EXIT to exit the interactive mode.

Remarks: You must be in the Service mode to use this command. Use INSTrument:SELect.

A valid service password needs to be entered prior to sending this command.

Front Panel Access: **System, Alignments, Align subsystem, Align 50 MHz Reference**

Exit Interactive Mode for Internal 50 MHz Amplitude Reference Alignment

:CALibration:REF50:EXIT

Turns off the interactive mode for alignment of the internal 50 MHz amplitude reference signal. Use CAL:REF50:ENTer to turn the mode on and CAL:REF50:ANOW to do the alignment immediately.

Remarks: You must be in the Service mode to use this command. Use INSTrument:SELect.

A valid service password needs to be entered prior to sending the command.

Front Panel Access: **System, Alignments, Align subsystem, Align 50 MHz Reference**

Query the Absolute Level for the 50 MHz Amplitude Reference

:CALibration:REF50:LAST:ALEVel?

Query returns the last value of the absolute level of the 50 MHz reference alignment.

Remarks: You must be in the Service mode to use this command. Use INSTrument:SELect.

A valid service password needs to be entered prior to sending this command.

Front Panel Access: **System, Alignments, Align subsystem, Align 50 MHz**

Reference

Query the ALC DAC Value for the 50 MHz Amplitude Reference

`:CALibration:REF50:LAST:ALCDac?`

Query returns the last value of the ALC DAC of the 50 MHz reference alignment.

Remarks: You must be in the Service mode to use this command. Use INSTrument:SELect.

A valid service password needs to be entered prior to sending this command.

Front Panel Access: **System**, **Alignments, Align subsystem**, **Align 50 MHz Reference**

Select Time Corrections

`:CALibration:TCORrections AUTO|ON|OFF`

Controls time corrections used to compensate for the complex (magnitude and phase) response of the analog and digital IF hardware. When only scalar (magnitude) FFT flatness is required, time corrections take more CPU cycles and so are less efficient than frequency corrections. For demod or other time-based (not FFT) measurements, only time corrections can improve the flatness that results from imperfect IF hardware. When the time correction functionality is set to Auto (the default), the individual measurements activate the corrections when they are needed.

.

NOTE Turning time corrections on or off effects all measurements. Time corrections should be left in Auto unless you have specific reasons for forcing them on or off.

Always return time corrections to Auto.

Factory Preset and *RST: AUTO

Front Panel Access: **System**, **Alignments**, **Time Corr**

Align the Trigger Delay

:CALibration:TRIGger:DELay

:CALibration:TRIGger:DELay?

Align any trigger delays needed. One place that this alignment is used is for the even second clock functionality in cdmaOne mode. This same alignment is run as part of the CAL:ALL routine.

The query performs the alignment and returns a zero if the alignment is successful.

Front Panel Access: **System, Alignments, Align subsystem, Align 50 MHz Reference**

Align the Trigger Interpolator

:CALibration:TRIGger:INTerpolator

:CALibration:TRIGger:INTerpolator?

Align the partial sample trigger interpolator. This same alignment is run as part of the CAL:ALL routine.

The query performs the alignment and returns a zero if the alignment is successful.

Front Panel Access: **System, Alignments, Align subsystem, Align 50 MHz Reference**

Calibration Wait

:CALibration:WAIT

Waits until any alignment procedure that is underway is completed.

CONFigure Subsystem

The CONFigure commands are used with several other commands to control the measurement process. The full set of commands are described in the section "MEASure Group of Commands" on page 301.

Selecting measurements with the CONFigure/FETCh/MEASure/READ commands sets the instrument state to the defaults for that measurement and to make a single measurement. Other commands are available for each measurement to allow you to change: settings, view, limits, etc. Refer to:

SENSe:<measurement>, SENSe:CHANnel, SENSe:CORRection,
SENSe:DEFaults, SENSe:DEViation, SENSe:FREQuency,
SENSe:PACKet, SENSe:POWer, SENSe:RADio, SENSe:SYNC
CALCulate:<measurement>, CALCulate:CLIMits
DISPlay:<measurement>
TRIGger

The INITiate[:IMMediate] or INITiate:RESTart commands will initiate the taking of measurement data without resetting any of the measurement settings that you have changed from their defaults.

Configure the Selected Measurement

`:CONFigure:<measurement>`

A CONFigure command must specify the desired measurement. It will set the instrument settings for that measurements standard defaults, but should not initiate the taking of data. The available measurements are described in the MEASure subsystem.

NOTE

If CONFigure initiates the taking of data, the data should be ignored. Other SCPI commands can be processed immediately after sending CONFigure. You do not need to wait for the CONF command to complete this 'false' data acquisition.

Configure Query

`:CONFigure?`

The CONFigure query returns the name of the current measurement.

DISPlay Subsystem

The DISPlay controls the selection and presentation of textual, graphical, and TRACe information. Within a DISPlay, information may be separated into individual WINDows.

Adjacent Channel Power - View Selection

:DISPlay:ACP:VIEW BGRaph|SPECtrum

:DISPlay:ACP:VIEW?

Select the adjacent channel power measurement display of bar graph or spectrum.

You may want to disable the spectrum trace data part of the measurement so you can increase the speed of the rest of the measurement display. Use SENSe:ACP:SPECtrum:ENABle to turn on or off the spectrum trace. (Basic and cdmaOne modes only)

Factory Preset and *RST: Bar Graph (BGRaph)

Remarks: You must be in the Basic, cdmaOne, cdma2000, W-CDMA (3GPP), NADC or PDC mode to use this command. Use INSTrument:SELect to set the mode.

Front Panel Access: **ACP, View/Trace**

Date and Time Display

:DISPlay:ANNotation:CLOCk:DATE:FORMat MDY|DMY

:DISPlay:ANNotation:CLOCk:DATE:FORMat?

Allows you to set the format for displaying the real-time clock. To set the date time use `:SYSTem:DATE <year>,<month>,<day>`.

Factory Preset and *RST: MDY

Remarks: This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.

Front Panel Access: **System, Time/Date, Date Format MDY DMY**

Date and Time Display

:DISPlay:ANNotation:CLOCk[:STATe] OFF|ON|0|1

:DISPlay:ANNotation:CLOCk[:STATe]?

Turns on and off the display of the date and time on the spectrum analyzer screen. The time and date pertain to all windows.

Factory Preset and *RST: On

This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.

Front Panel Access: **System, Time/Date, Time/Date On Off**

Display Annotation Title Data

:DISPlay:ANNotation:TITLe:DATA <string>

:DISPlay:ANNotation:TITLe:DATA?

Enters the text that will be displayed in the user title area of the display.

Front Panel Access: **Display, Title**

Display, Title, Change Title

Display, Title, Clear Title

Turn the Display On/Off

:DISPlay:ENABle OFF|ON|0|1

:DISPlay:ENABle?

Controls the display. If enable is set to off, the display will appear to “freeze” in its current state. Measurements may run faster since the instrument doesn’t have to update the display after every data acquisition. There is often no need to update the display information when using remote operation. An instrument preset will turn the display back on.

Factory Preset and *RST: On

Remarks: The following key presses will turn display enable back on:

1. If in local, press any key
2. If in remote, press the local (system) key
3. If in local lockout, no key

Front Panel Access: **System**, **Disp Updates** for VSA

Select Display Format

:DISPlay:FORMat:TILE

Selects the viewing format that displays multiple windows of the current measurement data simultaneously. Use DISP:FORM:ZOOM to return the display to a single window.

Front Panel Access: **Zoom** (toggles between Tile and Zoom)

Select Display Format

:DISPlay:FORMat:ZOOM

Selects the viewing format that displays only one window of the current measurement data (the current active window). Use DISP:FORM:TILE to return the display to multiple windows.

Front Panel Access: **Zoom** (toggles between Tile and Zoom)

Spectrum - Y-Axis Scale/Div

:DISPlay:SPECtrum[n]:WINDow[m]:TRACe:Y[:SCALe]:PDIVision <power>

:DISPlay:SPECtrum[n]:WINDow[m]:TRACe:Y[:SCALe]:PDIVision?

Sets the amplitude reference level for the y-axis.

n – selects the view, the default is Spectrum.

— n=1, Spectrum

— n=2, I/Q Waveform

— n=3, numeric data (service mode)

— n=4, RF Envelope (service mode)

m – selects the window within the view. The default is 1.

Factory Preset and *RST: 10 dB per division, for Spectrum

Range: .1 dB to 20 dB per division, for Spectrum

Default Unit: 10 dB per division, for Spectrum

Remarks: May affect input attenuator setting.

To use this command, the appropriate mode should be selected with INSTrument:SELect.

Front Panel Access: When in Spectrum measurement: **Amplitude Y Scale, Scale/Div.**

Spectrum - Y-Axis Reference Level

`:DISPlay:SPECtrum[n]:WINDow[m]:TRACe:Y[:SCALe]:RLEVel <power>`

`:DISPlay:SPECtrum[n]:WINDow[m]:TRACe:Y[:SCALe]:RLEVel?`

Sets the amplitude reference level for the y-axis.

n – selects the view, the default is Spectrum.

— n=1, m=1 Spectrum

— n=1, m=2 I/Q Waveform

— n=1, m=3 numeric data (Service mode)

— n=1, m=4 RF envelope (Service mode)

— n=2, m=1 I Waveform

— n=2, m=2 Q Waveform

— n=3, m=1 I/Q Polar

— n=4, m=1 Linear Spectrum

m – selects the window within the view. The default is 1.

Factory Preset: 0 dBm, for Spectrum

Range: –250 to 250 dBm, for Spectrum

Default Unit: dBm, for Spectrum

Remarks: May affect input attenuator setting.

To use this command, the appropriate mode should be selected with INSTrument:SELect.

Front Panel Access: When in Spectrum measurement: **Amplitude Y Scale, Ref Level**

Turn a Trace Display On/Off

`:DISPlay:TRACe[n][:STATe] OFF|ON|0|1`

`:DISPlay:TRACe[n][:STATe]?`

Controls whether the specified trace is visible or not.

n is a sub-opcode that is valid for the current measurement. See the "MEASure Group of Commands" on page 301 for more information about sub-opcodes.

Factory Preset and *RST: On

Range: The valid traces and their sub-opcodes are dependent upon the selected measurement. See the following table.

The trace name assignment is independent of the window number.

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

Front Panel Access: **Display, Display Traces**

Measurement	Available Traces	Markers Available?
ACP - adjacent channel power (Basic, cdmaOne, cdma2000, W-CDMA (3GPP), iDEN, NADC, PDC modes)	no traces	no markers
BER - bit error rate (iDEN mode)	no traces	no markers
CDPower - code domain power (cdmaOne mode)	POWer (*n*=2)[a] TIMing (*n*=3)[a] PHASe (*n*=4)[a]	yes

Measurement	Available Traces	Markers Available?
CDPower - code domain power (cdma2000, W-CDMA (3GPP) modes)	CDPower (*n*=2)[a] EVM (*n*=5)[a] MERRor (*n*=6)[a] PERRor (*n*=7)[a] SPOWer (*n*=9)[a] CPOWer (*n*=10)[a]	yes
CHPower - channel power (Basic, cdmaOne, cdma2000, W-CDMA (3GPP) modes)	SPECtrum (*n*=2)[a]	no markers
CSPur - spurs close (cdmaOne mode)	SPECtrum (*n*=2)[a] ULIMit (*n*=3)[a]	yes
EEVM - EDGE error vector magnitude (EDGE mode)	EVMerror (*n*=2)[a] MERRor (*n*=3)[a] PERRor (*n*=4)[a]	yes
EORFspectr - EDGE output RF spectrum (EDGE mode)	RFEMod (*n*=2)[a] RFESwitching (*n*=3)[a] SPEMod (*n*=4)[a] LIMMod (*n*=5)[a]	yes, only for a single offset yes, only for multiple offsets
EPVTime - EDGE power versus time (EDGE mode)	RFENvelope (*n*=2)[a] UMASk (*n*=3)[a] LMASk (*n*=4)[a]	yes
EVM - error vector magnitude (NADC, PDC modes)	EVM (*n*=2)[a] MERRor (*n*=3)[a] PERRor (*n*=4)[a]	yes
EVMQpsk - QPSK error vector magnitude (cdma2000, W-CDMA (3GPP) modes)	EVM (*n*=2)[a] MERRor (*n*=3)[a] PERRor (*n*=4)[a]	yes

Measurement	Available Traces	Markers Available?
IM - intermodulation (cdma2000, W-CDMA (3GPP) modes)	SPECtrum (*n*=2)[a]	yes
MCPower - multi-carrier power (W-CDMA (3GPP) mode)	no traces	no markers
OBW - occupied bandwidth (cdmaOne, cdma2000, iDEN, PDC, W-CDMA (3GPP) modes)	no traces	no markers
ORFSpectrum - output RF spectrum (GSM mode)	RFEMod (*n*=2)[a] RFESwitching (*n*=3)[a] SPEMod (*n*=4)[a] LIMMod (*n*=5)[a]	yes, only for a single offset yes, only for multiple offsets
PFERror - phase and frequency error (GSM mode)	PERRor (*n*=2)[a] PFERror (*n*=3)[a] RFENvelope (*n*=4)[a]	yes
PSTatistic - power statistics CCDF (Basic, cdma2000, W-CDMA (3GPP) modes)	MEASured (*n*=2)[a] GAUSian (*n*=3)[a] REFerence (*n*=4)[a]	yes
PVTime - power versus time (GSM, Service modes)	RFENvelope (*n*=2)[a] UMASk (*n*=3)[a] LMASk (*n*=4)[a]	yes
RHO - modulation quality (cdmaOne, cdma2000, W-CDMA (3GPP) modes)	EVM (*n*=2)[a] MERRor (*n*=3)[a] PERRor (*n*=4)[a]	yes
SEMask - spectrum emissions mask (cdma2000, W-CDMA (3GPP) modes)	SPECtrum (*n*=2)[a]	yes
TSPur - transmit band spurs (GSM mode)	SPECtrum (*n*=2)[a] ULIMit (*n*=3)[a]	yes

Measurement	Available Traces	Markers Available?
TXPower - transmit power (GSM mode)	RFENvelope (n=2)[a] IQ (n=8)[a]	yes
SPECtrum - (frequency domain) (all modes)	RFENvelope (n=2)[a] for Service mode IQ (n=3)[a] SPECtrum (n=4)[a] ASPectrum (n=7)[a]	yes
WAVEform - (time domain) (all modes)	RFENvelope (n=2)[a] IQ (n=8)[a]	yes

a. The *n* number indicates the sub-opcode that corresponds to this trace. Detailed descriptions of the trace data can be found in the MEASure subsystem documentation by looking up the sub-opcode for the appropriate measurement.

Waveform - Y-Axis Reference Level

`:DISPlay:WAVeform[n]:WINDow[m]:TRACe:Y[:SCALe]:RLEVel <power>`

`:DISPlay:WAVeform[n]:WINDow[m]:TRACe:Y[:SCALe]:RLEVel?`

Sets the amplitude reference level for the y-axis.

n, selects the view, the default is RF envelope.

n=1, m=1 RF envelope

n=2, m=1 I and Q Waveform

n=3 m=1 I Waveform

n=3, m=2 Q Waveform

n=4, m=1 I/Q Polar

n=5, m=1 Linear Spectrum

m, selects the window within the view. The default is 1.

Factory Preset: 0 dBm, for RF envelope

Range: –250 to 250 dBm, for RF envelope

Default Unit: dBm, for RF envelope

Remarks: May affect input attenuator setting.

To use this command, the appropriate mode should be selected with INSTrument:SELect.

Front Panel Access: When in Waveform measurement: **Amplitude Y Scale, Ref Level**

FETCh Subsystem

The FETCh? commands are used with several other commands to control the measurement process. These commands are described in the section "MEASure Group of Commands" on page 301.

Fetch the Current Measurement Results

:FETCh:<measurement>[n]?

A FETCh? command must specify the desired measurement. It will return the valid results that are currently available, but will not initiate the taking of any new data. You can only fetch results from the measurement that is currently selected. The code number n selects the kind of results that will be returned. The available measurements and data results are described in "MEASure Group of Commands" on page 301.

FORMat Subsystem

The FORMat subsystem sets a data format for transferring numeric and array information.

Byte Order

:FORMat:BORDer NORMal|SWAPped

:FORMat:BORDer?

Selects the binary data byte order for numeric data transfer. In normal mode the most significant byte is sent first. In swapped mode the least significant byte is first. (PCs use the swapped order.) Binary data byte order functionality does not apply to ASCII.

Factory Preset
and *RST: Normal

Numeric Data format

:FORMat[:DATA] ASCii|REAL,32|REAL,64

:FORMat[:DATA]?

This command controls the format of data output, that is, data transfer across any remote port. The REAL and ASCII formats will format trace data in the current amplitude units.

The format of state data cannot be changed. It is always in a machine readable format only.

ASCII - Amplitude values are in ASCII, in amplitude units, separated by commas. ASCII format requires more memory than the binary formats. Therefore, handling large amounts of this type of data, will take more time and storage space.

Real,32 (or 64) - Binary 32-bit, or 64-bit, real values in amplitude unit, in a definite length block. Transfers of real data are done in a binary block format.

A definite length block of data starts with an ASCII header that begins with # and indicates how many additional data points are following in the block. Suppose the header is #512320.

- The first digit in the header (5) tells you how many additional digits/bytes there are in the header.
- The 12320 means 12 thousand, 3 hundred, 20 data bytes follow the

header.

- Divide this number of bytes by your selected data format bytes/point, either 8 (for real 64), or 4 (for real 32). In this example, if you are using real 64 then there are 1540 points in the block.

Factory Preset
and *RST: ASCII

HCOPy Subsystem

The HCOPy subsystem controls the setup of printing to an external device.

Screen Printout Destination

:HCOPy:DESTination FPANel|PRINter

:HCOPy:DESTination?

This command was created to support backward compatibility to early instrument functionality. It is used to specify whether the hardcopy printout goes to the printer or to a destination that is specified from the front-panel key **Print Setup**, **Print To File|Printer**.

Example: HCOP:DEST printer

Factory Preset and *RST: Front panel. This parameter is persistent, which means it retains the value previously selected even through a power cycle.

History: Revision A.04.00 and later

Front Panel Access: **Print Setup, Print To**

Custom Printer Color Capability

:HCOPy:DEVice:COLor NO|YES

:HCOPy:DEVice:COLor?

Specifies whether the printer is color capable, not whether you want to print in color. HCOPY:DEVICE:TYPE CUSTOM must be selected.

Example: HCOP:DEV:COLOR YES

Factory Preset and *RST: Yes. This parameter is persistent, which means it retains the value previously selected even through a power cycle.

History: Revision A.04.00 and later

Front Panel Access: **Print Setup**, (select **Print To** (Printer) and **Printer Type** (Custom)), **Define Custom**

Custom Printer Language

:HCOPy:DEVice:LANGuage PCL3|PCL5

:HCOPy:DEVice:LANGuage?

Specifies the type of printer control language that your custom printer uses. HCOPY:DEVICE:TYPE CUSTOM must be selected.

Example: HCOP:DEV:LANG pcl3

Factory Preset and *RST: PCL3. This parameter is persistent, which means it retains the value previously selected even through a power cycle.

History: Revision A.04.00 and later

Front Panel Access: **Print Setup**, (select **Print To** (Printer) and **Printer Type** (Custom)), **Define Custom**

Printer Type

:HCOPy:DEVice[:TYPE] CUSTom|NONE

:HCOPy:DEVice[:TYPE]?

Set up the printer by selecting the type of printer.

CUSTom - allows you to configure a custom printer if your printer cannot be auto-configured. Use other HCOPy:DEVice commands to specify some of the characteristics of your custom printer. The color and language must be defined for your custom printer. You must select the custom printer type to print hardcopy output.

NONE - tells the instrument that there is no hard copy (printer) device available.

Factory Preset and *RST: NONE - This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.

History: Revision A.04.00 and later

Front Panel Access: **Print Setup**, (select **Print To** (Printer)), **Printer Type**

Color Hard Copy

:HCOPy:IMAGe:COLor[:STATe] OFF|ON|0|1

:HCOPy:IMAGe:COLor[:STATe]?

Selects between color and monochrome mode for hard copy output. You must set HCOP:DEV:COLOR YES before using this command.

Factory Preset and *RST: On. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.

Remarks: Revision A.04.00 and later

Front Panel Access: **Print Setup**, (select **Print To** (Printer)), **Color**

Print a Hard Copy

:HCOPy[:IMMediate]

The entire screen image is output to the printer at the parallel port.

Front Panel Access: **Print**

Form Feed the Print Item

:HCOPy:ITEM:FFEed[:IMMediate]

Sends the printer a command to form feed. No form feed will occur unless the printer has only printed one image of a multi-image printout.

History: Revision A.04.00 and later

Front Panel Access: **Print Setup**, (select **Print To** (Printer)), **More**, **Eject Page**

Page Orientation

:HCOPy:PAGE:ORIentation LANDscape|PORTrait

:HCOPy:PAGE:ORIentation?

Specifies the orientation of the print image.

NOTE Landscape mode is not presently supported for PCL-3 printers.

Factory Preset and *RST: Portrait. This parameter is persistent, which means

that it retains the setting previously selected, even through a power cycle.

History: Revision A.04.00 and later

Front Panel Access: **Print Setup**, (select **Print To** (Printer)), **Orientation**

Number of Items Printed on a Page

:HCOPy:PAGE:PRINts 1|2

:HCOPy:PAGE:PRINts?

Sets the number of display images that should be printed on one sheet of paper, before a form feed is sent.

Factory Preset and *RST: 1. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.

History: Revision A.04.00 and later

Remarks: This must be set to 1 if the paper orientation is landscape.

Front Panel Access: **Print Setup**, (select **Print To** (Printer)), **Prints/Page**

Reprint the Last Image

:HCOPy:REPRint[:IMMediate]

Reprint the most recently printed image.

Example: HCOP:REPR

History: Revision A.04.00 and later

Front Panel Access: **Print Setup** with **Print To** (Printer) selected

Screen Dump Query

:HCOPy:SDUMp:DATA? [GIF]|BMP|WMF

The query returns the current screen image as a file. If the optional file type is not specified it returns GIF type graphic data. The orientation is always portrait and the image is always in color.

The data is formatted as block data where the block of data starts with an ASCII header that indicates how many additional binary data bytes are following in the block. (e.g. #DNNN<binary data>) The binary data is the actual graphics file. To process the block of data you would:

- Read the first header byte #. The # tells you to read the next digit (D). That digit tells you how many additional digits there are in the header. (In the above example D=3.)
- Then read the next D (that is, 3) bytes. The digits NNN tell you the number of bytes of data there are following the header.
- Those final data bytes are the screen image in the requested format. They can be saved as a separate file with a .gif, .bmp or .wmf suffix to use in other applications.

Factory Preset and *RST: GIF

History: Firmware revision A.03.28 and later, changed A.04.00

Screen Dump Image Inverting

:HCOPy:SDUMp:IMAGe NORMal|INVert

:HCOPy:SDUMp:IMAGe?

Controls the trace background color when using the HCOPy:SDUMp:DATA? query.

Normal, is black trace background

Invert, is white trace background

Factory Preset and *RST: Invert

History: Revision A.04.00 and later

Screen Dump to a Printer

:HCOPy:SDUMp[:IMMediate]

The entire screen image is output to the printer (remote interface). The image is always inverted.

History: Revision A.04.00 and later

INITiate Subsystem

The INITiate subsystem is used to control the initiation of the trigger. Refer to the TRIGger and ABORt subsystems for related commands.

Take New Data Acquisition for Selected Measurement

:INITiate:<measurement_name>

This command initiates a trigger cycle for the measurement specified. The available measurement names are described in the MEASure subsystem. It also holds off additional commands on GPIB until the acquisition is complete. So if it is followed by a FETCh command, valid data will be returned.

If your selected measurement is currently active (in the idle state) it triggers the measurement, assuming the trigger conditions are met. Then it completes one trigger cycle. Depending upon the measurement and the number of averages, there may be multiple data acquisitions, with multiple trigger events, for one full trigger cycle.

If your selected measurement is not currently active it will change to the measurement in your INIT:<meas_name> command and initiate a trigger cycle.

Example: INIT:ACP

Continuous or Single Measurements

:INITiate:CONTinuous OFF|ON|0|1

:INITiate:CONTinuous?

Selects whether a trigger is continuously initiated or not. Each trigger initiates a single, complete, measurement operation.

When set to ON another trigger cycle is initiated at the completion of each measurement.

When set to OFF, the trigger system remains in the “idle” state until an INITiate[:IMMediate] command is received. On receiving the INITiate[:IMMediate] command, it will go through a single trigger/measurement cycle, and then return to the “idle” state.

Example: INIT:CONT ON

Factory Preset: On

*RST: Off (recommended for remote operation)

Front Panel

Access: **Meas Control, Measure Cont Single**

Take New Data Acquisitions

:INITiate[:IMMediate]

The instrument must be in the single measurement mode. If INIT:CONT is ON, then the command is ignored. The desired measurement must be selected and waiting. The command causes the system to exit the “waiting” state and go to the “initiated” state.

The trigger system is initiated and completes one full trigger cycle. It returns to the “waiting” state on completion of the trigger cycle. Depending upon the measurement and the number of averages, there may be multiple data acquisitions, with multiple trigger events, for one full trigger cycle.

This command triggers the instrument, if external triggering is the type of trigger event selected. Otherwise, the command is ignored. Use the TRIGer[:SEQuence]:SOURce EXT command to select the external trigger.

Example: INIT:IMM

Remarks: See also the *TRG command and the TRIGger subsystem.

Front Panel Access: **Meas Control, Measure Cont Single**

Restart the Measurement

:INITiate:RESTart

It restarts the current measurement from the “idle” state regardless of its current operating state. It is equivalent to:

INITiate[:IMMediate]

ABORt (for continuous measurement mode)

Example: INIT:REST

Front Panel Access: **Restart**

or

Meas Control, Restart

INPut Subsystem

The INPut subsystem controls the characteristics of all the instrument input ports.

BbIQ - Select Input Impedance

`:INPut:IMPedance:IQ U50|B600|U1M|B1M`

`:INPut:IMPedance:IQ?`

Selects the characteristic input impedance when input port is set to I or Q.

Factory Preset and *RST: U50

Remarks: Implemented for BASIC and W-CDMA modes.
1000000|1E6 sets input impedance to 1 M ohm.

History: Version A.05.00 or later

BbIQ - Select Input Impedance Reference

`:INPut:IMPedance:REFerence Int32 [OHM]`

`:INPut:IMPedance:REFerence?`

Sets the value of the input impedance reference when input port is set to I or Q.

Range: 1 to 10,000,000.

Remarks: Implemented for BASIC and W-CDMA modes.
1000000|1E6 sets input impedance to 1 M ohm.

History: Version A.05.00 or later

BbIQ - Activate IQ Alignment

`:INPut:IQ:ALIGn 0|1|OFF|ON`

`:INPut:IQ:ALIGn?`

Activates or deactivates IQ alignment.

Factory Preset and *RST: Off

Remarks:

History: Version A.05.00 or later

BbIQ - I DC Offset

:INPut:OFFSet:I Float64 [V] -2.5|0|+2.5

:INPut:OFFSet:I?

Sets adjustment to compensate for I voltage bias on signals when I port is selected.

Factory Preset and *RST: 0

Remarks: Implemented for BASIC and W-CDMA modes.

History: Version A.05.00 or later

BbIQ - Q DC Offset

:INPut:OFFSet:Q Float64[V] -2.5|0|+2.5

:INPut:OFFSet:Q?

Sets adjustment to compensate Q voltage bias on signals when Q port is selected.

Factory Preset and *RST: 0

Remarks: Implemented for BASIC and W-CDMA modes.

History: Version A.05.00 or later

INSTrument Subsystem

This subsystem includes commands for querying and selecting instrument measurement (personality option) modes.

Catalog Query

:INSTrument:CATalog[:FULL]?

Returns a comma separated list of strings which contains the names of all the installed applications. These names can only be used with the INST:SELECT command. If the optional keyword FULL is specified, each name is immediately followed by its associated instrument number. These instrument numbers can only be used with the INST:NSELect command.

Example: INST:CAT?

Query response: “GSM”,”CDMA”

Example: INST:CAT:FULL?

Query response: “GSM”3,”CDMA”4

Select Application by Number

:INSTrument:NSELect <integer>

:INSTrument:NSELect?

Select the measurement mode by its instrument number. The actual available choices depends upon which applications are installed in the instrument. These instrument numbers can be obtained with INST:CATalog:FULL?

1 = SERVICE
3 = GSM
4 = CDMA (cdmaOne)
5 = NADC
6 = PDC
8 = BASIC
9 = WCDMA (3GPP W-CDMA with HSDPA/HSUPA)
10 = CDMA2K (cdma2000 with 1xEV-DV)
11 = IDEN
13 = EDGEGSM
15 = CMDA1XEV (1xEV-D0)

NOTE If you are using the status bits and the analyzer mode is changed, the

status bits should be read, and any errors resolved, prior to switching modes. Error conditions that exist prior to switching modes cannot be detected using the condition registers after the mode change. This is true unless they recur after the mode change, although transitions of these conditions can be detected using the event registers.

Changing modes resets all SCPI status registers and mask registers to their power-on defaults. Hence, any event or condition register masks must be re-established after a mode change. Also note that the power up status bit is set by any mode change, since that is the default state after power up.

Example: INST:NSEL 3

Factory Preset and *RST: Persistent state with factory default of 8 (BASIC)

Range: 1 to x, where x depends upon which applications are installed.

Front Panel Access: **Mode**

Select Application

:INSTrument[:SELect]
BASIC|**SERVICE**|**CDMA**|**CDMA2K**|**GSM**|**EDGEGSM**|**IDEN**|**NADC**|**PDC**|**WCDMA**|**ARIBWCDMA**

:INSTrument[:SELect]?

Select the measurement mode. The actual available choices depend upon which modes (measurement applications) are installed in the instrument. A list of the valid choices is returned with the INST:CAT? query.

Once an instrument mode is selected, only the commands that are valid for that mode can be executed.

1 = SERVICE
3 = GSM
4 = CDMA (cdmaOne)
5 = NADC
6 = PDC
8 = BASIC
9 = WCDMA (3GPP W-CDMA with HSDPA/HSUPA)
10 = CDMA2K (cdma2000 with 1xEV-DV)
11 = IDEN
13 = EDGEGSM
15 = CMDA1XEV (1xEV-D0) (E4406/PSA)

NOTE

If you are using the status bits and the analyzer mode is changed, the status bits should be read, and any errors resolved, prior to switching modes. Error conditions that exist prior to switching modes cannot be detected using the condition registers after the mode change. This is true unless they recur after the mode change, although transitions of these conditions can be detected using the event registers.

Changing modes resets all SCPI status registers and mask registers to their power-on defaults. Hence, any event or condition register masks must be re-established after a mode change. Also note that the power up status bit is set by any mode change, since that is the default state after power up.

Example: INST:SEL GSM

Factory Preset and *RST: Persistent state with factory default of Basic mode.

Front Panel Access: **Mode**

MEASure Group of Commands

This group includes the CONFigure, FETCh, MEASure, and READ commands that are used to make measurements and return results. The different commands can be used to provide fine control of the overall measurement process, like changing measurement parameters from their default settings. Most measurements should be done in single measurement mode, rather than measuring continuously.

The SCPI default for data output format is ASCII. The format can be changed to binary with FORMat:DATA which transports faster over the bus.

Measure

For key and remote command information on each measurement, refer to the section which describes the measurement of interest.

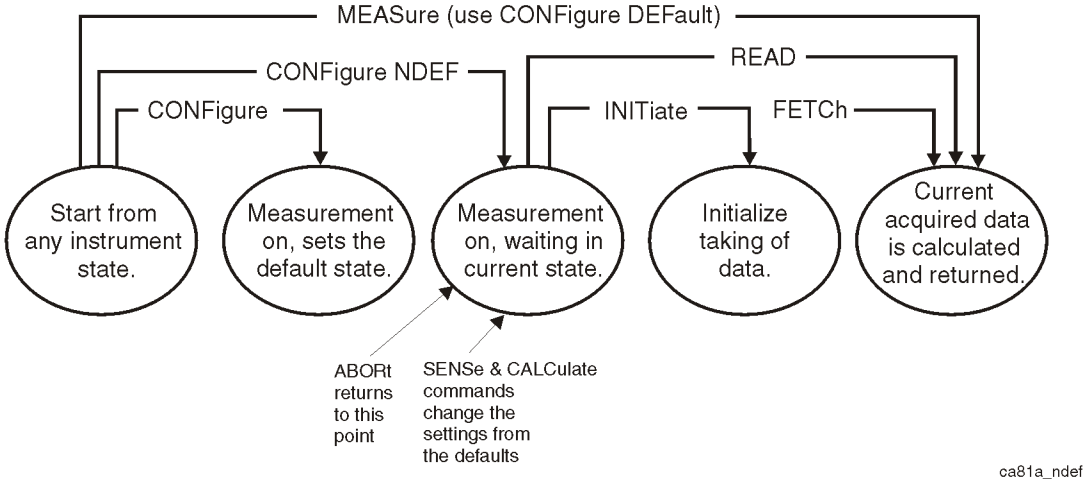
Measurements available under the **Measure** key are specific to the current Mode.

Key Path	Front-panel key
Help Map ID	4008

Command Interactions: MEASure, CONFigure, FETCh, INITiate and READ

Each one-button measurement has a group of commands that work together to make the measurement fast, but flexible.

Figure 5-3 Measurement Group of Commands



Measure Commands:

`:MEASure:<measurement>[n]?`

This is a fast single-command way to make a measurement using the factory default instrument settings. These are the settings and units that conform to the Mode Setup settings (e.g. radio standard) that you have currently selected.

- Stops the current measurement (if any) and sets up the instrument for the specified measurement using the factory defaults
- Initiates the data acquisition for the measurement
- Blocks other SCPI communication, waiting until the measurement is complete before returning results.
- After the data is valid it returns the scalar results, or the trace data, for the specified measurement. The type of data returned may be defined by an [n] value that is sent with the command.

  The scalar measurement results will be returned if the optional [n] value is not included, or is set to 1. If the [n] value is set to a value other than 1, the selected trace data results will be returned. See each command for details of what types of scalar results or trace data results are available.

  ASCII is the default format for the data output. The binary data formats should be used for handling large blocks of data since they are smaller and faster than the ASCII format. Refer to the FORMat:DATA command for more information.

If you need to change some of the measurement parameters from the factory default settings you can set up the measurement with the CONFigure command. Use the commands in the SENSe:<measurement> and CALCulate:<measurement> subsystems to change the settings. Then you can use the READ? command to initiate the measurement and query the results.

If you need to repeatedly make a given measurement with settings other than the factory defaults, you can use the commands in the SENSe:<measurement> and CALCulate:<measurement> subsystems to set up the measurement. Then use the READ? command to initiate the measurement and query results.

Measurement settings persist if you initiate a different measurement and then return to a previous one. Use READ:<measurement>? if you want to use those persistent settings. If you want to go back to the default settings, use MEASure:<measurement>?.

Configure Commands:

`:CONFigure:<measurement>`

This command stops the current measurement (if any) and sets up the instrument for the specified measurement using the factory default instrument settings. It sets the instrument to single measurement mode but should not initiate the taking of measurement data unless INIT:CONTinuous is ON. If you change any measurement settings after using the CONFigure command, the READ command can be used to initiate a measurement without changing the settings back to their defaults.

The CONFigure? query returns the current measurement name.

Fetch Commands:

`:FETCh:<measurement>[n]?`

This command puts selected data from the most recent measurement into the output buffer. Use FETCh if you have already made a good measurement and you want to return several types of data (different [n] values, e.g. both scalars and trace data) from a single measurement. FETCh saves you the time of re-making the measurement. You can only FETCh results from the measurement that is currently active, it will not change to a different measurement. An error is reported if a measurement other than the current one, is specified.

If you need to get new measurement data, use the READ command, which is equivalent to an INITiate followed by a FETCh.

The scalar measurement results will be returned if the optional [n] value is not included, or is set to 1. If the [n] value is set to a value other than 1, the selected trace data results will be returned. See each command for details of what types of scalar results or trace data results are available. The binary data formats should be used for handling large blocks of data since they are smaller and transfer faster then the ASCII format. (FORMat:DATA)

FETCh may be used to return results other than those specified with the original READ or MEASure command that you sent.

INITiate Commands:

`:INITiate:<measurement>`

This command is not available for measurements in all the instrument modes:

- Initiates a trigger cycle for the specified measurement, but does not output any data. You must then use the FETCh<meas> command to return data. If a measurement other than the current one is specified, the instrument will switch to that measurement and then initiate it.

  For example, suppose you have previously initiated the ACP measurement, but now you are running the channel power measurement. If you send INIT:ACP? it will change from channel power to ACP and will initiate an ACP measurement.
- Does not change any of the measurement settings. For example, if you have previously started the ACP measurement and you send INIT:ACP? it will initiate a new ACP measurement using the same instrument settings as the last time ACP was run.
- If your selected measurement is currently active (in the idle state) it triggers the measurement, assuming the trigger conditions are met. Then it completes one trigger cycle. Depending upon the measurement and the number of averages, there may be multiple data acquisitions, with multiple trigger events, for one full trigger cycle. It also holds off additional commands on GPIB until the acquisition is complete.

READ Commands:

`:READ:<measurement>[n]?`

- Does not preset the measurement to the factory default settings. For example, if you have previously initiated the ACP measurement and you send READ:ACP? it will initiate a new measurement using the same instrument settings.
- Initiates the measurement and puts valid data into the output buffer. If a measurement other than the current one is specified, the instrument will switch to that measurement before it initiates the measurement and returns results.

  For example, suppose you have previously initiated the ACP measurement, but now you are running the channel power measurement. Then you send READ:ACP? It will change from channel power back to ACP and, using the previous ACP settings, will initiate the measurement and return results.
- Blocks other SCPI communication, waiting until the measurement is complete before returning the results

  If the optional [n] value is not included, or is set to 1, the scalar measurement results will be returned. If the [n] value is set to a value other than 1, the selected trace data results will be returned. See each command for details of what types of scalar results or trace data results are available. The binary data formats should be used when handling large blocks of data since they are smaller and faster then the ASCII format. (FORMat:DATA)

Current Measurement Query (Remote Command Only)

This command returns the name of the measurement that is currently running.

Mode	All
Remote Command	`:CONFigure?`
Example	`CONF`

Test current results against all limits (Remote Command Only)

Queries the status of the current measurement limit testing. It returns a 0 if the measured results pass when compared with the current limits. It returns a 1 if the measured results fail any limit tests.

Mode	All
Remote Command	`:CALCulate:CLIMits:FAIL?`
Range	0 | 1
Help Map ID	0

Data Query (Remote Command Only)

Returns the designated measurement data for the currently selected measurement and subopcode.

n = any valid subopcode for the current measurement. See the

measurement command results table in each measurement section for information about what data is returned for the subopcodes.

Mode	All
Remote Command	`:CALCulate:DATA[n]? <real>,…`
Notes	The return trace depends on the measurement. In CALCulate:DATA[n], n is any valid subopcode for the current measurement.
Help Map ID	0

Calculate/Compress Trace Data Query (Remote Command Only)

```
:CALCulate:DATA<n>:COMPress?
BLOCk|CFIT|MAXimum|MINimum|MEAN|DMEan|RMS|SAMPle|SDEViation
|PPHase [,<soffset>[,<length>[,<roffset>[,<rlimit>]]]]
```

Returns compressed data for the specified trace data. The data is returned in the same units as the original trace and only works with the currently selected measurement. The command is used with a subopcode <*n*> since measurements usually return several types of trace data. See the following table for the subopcodes for the trace data names that are available in each measurement. For subopcodes that return scalar data use the `:CALCulate:DATA[n]?` command above.

NOTE This description of CALC:DATA:COMP? operation applies to all measurements except Swept SA measurement. See the description in the **Trace/Detector** section for use in Swept SA.

This command is used to compress or decimate a long trace to extract and return only the desired data. A typical example would be to acquire N frames of GSM data and return the mean power of the first burst in each frame. The command can also be used to identify the best curve fit for the data.

- BLOCk or block data - returns all the data points from the region of the trace data that you specify. For example, it could be used to return the data points of an input signal over several timeslots, excluding the portions of the trace data that you do not want.
- CFIT or curve fit - applies curve fitting routines to the data. <soffset> and <length> are required to define the data that you want. <roffset> is an optional parameter for the desired order of the curve equation. The query will return the following values: the x-offset (in seconds) and the curve coefficients ((order + 1) values).

MAX, MEAN, MIN, RMS, SAMP, SDEV and PPH return one data value for each specified region (or <length>) of trace data, for as many regions as possible until you run out of trace data (using <roffset> to specify regions). Or they return the number regions you specify (using

<rlimit>) ignoring any data beyond that.

- MAXimum - returns the maximum data point for the specified region(s) of trace data. For I/Q trace data, the maximum magnitude of the I/Q pairs is returned.
- MEAN - returns the arithmetic mean of the data point values for the specified region(s) of trace data. See "Mean Value of I/Q Data Points for Specified Region(s)" on page 306. For I/Q trace data, the mean of the magnitudes of the I/Q pairs is returned. See "Mean Value of I/Q Data Pairs for Specified Region(s)" on page 306.

  Note: If the original trace data is in dB, this function returns the arithmetic mean of those log values, not log of the mean power, which is a more useful value.

Equation 5-7 Mean Value of I/Q Data Points for Specified Region(s)

$$\text{MEAN} = \frac{1}{n} \sum_{\text{Xi} \in \text{ region(s)}} \text{Xi}$$

where Xi is a data point value, and n is the number of data points in the specified region(s).

Equation 5-8 Mean Value of I/Q Data Pairs for Specified Region(s)

$$\text{MEAN} = \frac{1}{n} \sum_{\text{Xi} \in \text{ region(s)}} |\text{Xi}|$$

where |Xi| is the magnitude of an I/Q pair, and n is the number of I/Q pairs in the specified region(s).

- MINimum - returns the minimum data point for the specified region(s) of trace data For I/Q trace data, the minimum magnitude of the I/Q pairs is returned.
- RMS - returns the arithmetic rms of the data point values for the specified region(s) of trace data. See "RMS Value of Data Points for Specified Region(s)" on page 307.

  For I/Q trace data, the rms of the magnitudes of the I/Q pairs is returned. See "RMS Value of I/Q Data Pairs for Specified Region(s)" on page 307.

Note: This function is very useful for I/Q trace data. However, if the original trace data is in dB, this function returns the rms of the log values which is not usually needed.

Equation 5-9 RMS Value of Data Points for Specified Region(s)

$$RMS = \sqrt{\frac{1}{n} \sum_{Xi \in \text{region(s)}} Xi^2}$$

where Xi is a data point value, and n is the number of data points in the specified region(s).

Equation 5-10 RMS Value of I/Q Data Pairs for Specified Region(s)

$$RMS = \sqrt{\frac{1}{n} \sum_{Xi \in \text{region(s)}} Xi \, Xi^*}$$

where Xi is the complex value representation of an I/Q pair, Xi its conjugate complex number, and n is the number of I/Q pairs in the specified region(s).*

Once you have the rms value for a region of I/Q trace data, you may want to calculate the mean power. You must convert this rms I/Q value (peak volts) to power in dB.

$$10 \times \log[10 \times (\text{rms value})^2]$$

- SAMPle - returns the first data value for the specified region(s) of trace data. For I/Q trace data, the first I/Q pair is returned.
- SDEViation - returns the arithmetic standard deviation for the data point values for the specified region(s) of trace data. See "Standard Deviation of Data Point Values for Specified Region(s)" on page 307.

  For I/Q trace data, the standard deviation of the magnitudes of the I/Q pairs is returned. See "Standard Deviation of I/Q Data Pair Values for Specified Region(s)" on page 308.

Equation 5-11 Standard Deviation of Data Point Values for Specified Region(s)

$$SDEV = \sqrt{\frac{1}{n} \sum_{Xi \in \text{region(s)}} (Xi - \overline{X})^2}$$

where Xi is a data point value, X is the arithmetic mean of the data point values for the specified region(s), and n is the number of data

points in the specified region(s).

Equation 5-12 Standard Deviation of I/Q Data Pair Values for Specified Region(s)

$$SDEV = \sqrt{\frac{1}{n} \sum_{Xi \in \text{region(s)}} (|Xi| - \overline{X})^2}$$

where $|Xi|$ *is the magnitude of an* I/Q *pair,* X *is the mean of the magnitudes for the specified region(s), and* n *is* the number of data points in the specified region(s).

- PPH - returns the pairs of rms power (dBm) and arithmetic mean phase (radian) for every specified region and frequency offset (Hz). The number of pairs is defined by the specified number of regions. Assuming this command can be used for I/Q vector (n=0) in Waveform (time domain) measurement and all parameters are specified by data point in PPH.

  The rms power of the specified region may be expressed as:

  Power = 10 x log [10 x (RMS I/Q value)] + 10

  The RMS I/Q value (peak volts) =

  $$\sqrt{\frac{1}{n} \sum_{Xi \in region} XiXi^*}$$

  where Xi is the complex value representation of an I/Q pair, Xi* its conjugate complex number, and n is the number of I/Q pairs in the specified region.

  The arithmetic mean phase of the specified region may be expressed as:

  $$\text{Phase} = \frac{1}{n} \sum_{Yi \in region} Yi$$

  Where Yi is the unwrapped phase of I/Q pair with applying frequency correction and n is the number of I/Q pairs in the specified region.
  The frequency correction is made by the frequency offset calculated by the arithmetic mean of every specified region's frequency offset. Each frequency offset is calculated by the least square method against the unwrapped phase of I/Q pair.

Figure 5-4 Sample Trace Data - Constant Envelope

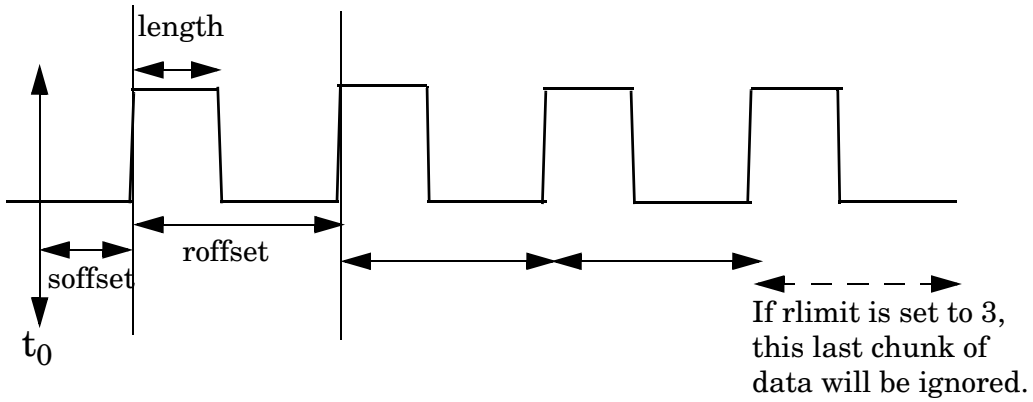
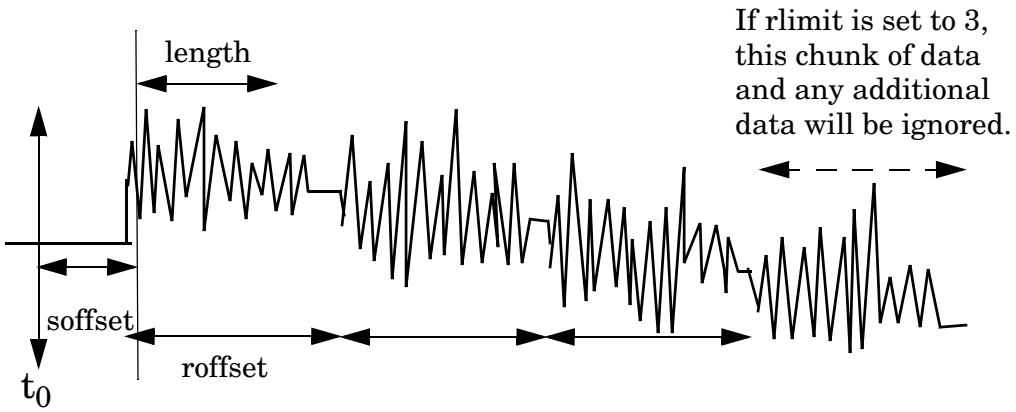


Figure 5-5 Sample Trace Data - Not Constant Envelope



<soffset> - start offset is an optional real number (in seconds). It specifies the amount of data at the beginning of the trace that will be ignored before the decimation process starts. It is the time from the start of the trace to the point where you want to start using the data. The default value is zero.

<length> - is an optional real number (in seconds). It defines how much data will be compressed into one value. This parameter has a default value equal to the current trace length.

<roffset> - repeat offset is an optional real number (in seconds). It defines the beginning of the next field of trace elements to be compressed. This is relative to the beginning of the previous field. This parameter has a default value equal to the <length> variable.

<rlimit> - repeat limit is an optional integer. It specifies the number of data items that you want returned. It will ignore any additional items beyond that number. You can use the Start offset and the Repeat limit to pick out exactly what part of the data you want to use. The default value is all the data.

Example: To query the mean power of a set of GSM bursts:

1. Set the waveform measurement sweep time to acquire at least one burst.
2. Set the triggers such that acquisition happens at a known position relative to a burst.
3. Then query the mean burst levels using, `CALC:DATA2:COMP? MEAN,24e-6,526e-6` (These parameter values correspond to GSM signals, where 526e-6 is the length of the burst in the slot and you just want 1 burst.)

Remarks: The optional parameters must be entered in the specified order. For example, if you want to specify <length>, you must also specify <soffset>.

This command uses the data in the format specified by FORMat:DATA, returning either binary or ASCII data.

Measurement	**Available Traces**	**Markers Available?**
ACP - adjacent channel power (Basic, cdmaOne, cdma2000, W-CDMA, NADC, PDC modes)	no traces (*n*=0)[a] for I/Q points	no markers
CDPower - code domain power (cdmaOne mode)	POWer (*n*=2)[a] TIMing (*n*=3)[a] PHASe (*n*=4)[a] (*n*=0)[a] for I/Q points	yes
CDPower - code domain power (cdma2000, W-CDMA modes)	CDPower (*n*=2)[a] EVM (*n*=5)[a] MERRor (*n*=6)[a] PERRor (*n*=7)[a] SPOWer (*n*=9)[a] CPOWer (*n*=10)[a] (*n*=0)[a] for I/Q points	yes
CHPower - channel power (Basic, cdmaOne, cdma2000, W-CDMA modes)	SPECtrum (*n*=2)[a] (*n*=0)[a] for I/Q points	no markers

Measurement	Available Traces	Markers Available?
CSPur - spurs close (cdmaOne mode)	SPECtrum (n=2)[a] ULIMit (n=3)[a] (n=0)[a] for I/Q points	yes
EEVM - EDGE error vector magnitude (EDGE mode)	EVMerror (n=2)[a] MERRor (n=3)[a] PERRor (n=4)[a] (n=0)[a] for I/Q points	yes
EORFspectr - EDGE output RF spectrum (EDGE mode)	RFEMod (n=2)[a] RFESwitching (n=3)[a] SPEMod (n=4)[a] LIMMod (n=5)[a] (n=0)[a] for I/Q points	yes, only for a single offset yes, only for multiple offsets
EPVTime - EDGE power versus time (EDGE mode)	RFENvelope (n=2)[a] UMASk (n=3)[a] LMASk (n=4)[a] (n=0)[a] for I/Q points	yes
ETSPur - EDGE transmit band spurs (EDGE mode)	SPECtrum (n=2)[a] ULIMit (n=3)[a] (n=0)[a] for I/Q points	yes
EVM - error vector magnitude (NADC, PDC modes)	EVM (n=2)[a] MERRor (n=3)[a] PERRor (n=4)[a] (n=0)[a] for I/Q points	yes
EVMQpsk - QPSK error vector magnitude (cdma2000, W-CDMA modes)	EVM (n=2)[a] MERRor (n=3)[a] PERRor (n=4)[a] (n=0)[a] for I/Q points	yes

Measurement	Available Traces	Markers Available?
IM - intermodulation (cdma2000, W-CDMA modes)	SPECtrum (n=2)[a] (n=0)[a] for I/Q points	yes
MCPower - multi-carrier power (W-CDMA mode)	no traces (n=0)[a] for I/Q points	no markers
OBW - occupied bandwidth (cdmaOne, cdma2000, PDC, W-CDMA modes)	no traces (n=0)[a] for I/Q points	no markers
ORFSpectrum - output RF spectrum (GSM, EDGE mode)	RFEMod (n=2)[a] RFESwitching (n=3)[a] SPEMod (n=4)[a] LIMMod (n=5)[a] (n=0)[a] for I/Q points	yes, only for a single offset yes, only for multiple offsets
PFERror - phase and frequency error (GSM, EDGE mode)	PERRor (n=2)[a] PFERror (n=3)[a] RFENvelope (n=4)[a] (n=0)[a] for I/Q points	yes
PSTatistic - power statistics CCDF (Basic, cdma2000, W-CDMA modes)	MEASured (n=2)[a] GAUSian (n=3)[a] REFerence (n=4)[a] (n=0)[a] for I/Q points	yes
PVTime - power versus time (GSM, EDGE modes)	RFENvelope (n=2)[a] UMASk (n=3)[a] LMASk (n=4)[a] (n=0)[a] for I/Q points	yes

Measurement	Available Traces	Markers Available?
RHO - modulation quality (cdmaOne, cdma2000, W-CDMA mode)	(*n*=0)[a] for I/Q points EVM (*n*=2)[a] MERRor (*n*=3)[a] PERRor (*n*=4)[a] (*n*=0)[a] for I/Q points	yes
SEMask - spectrum emissions mask (cdma2000, W-CDMA mode)	SPECtrum (*n*=2)[a] (*n*=0)[a] for I/Q points	yes
TSPur - transmit band spurs (GSM, EDGE mode)	SPECtrum (*n*=2)[a] ULIMit (*n*=3)[a] (*n*=0)[a] for I/Q points	yes
TXPower - transmit power (GSM, EDGE mode)	RFENvelope (*n*=2)[a] IQ (*n*=8)[a] (*n*=0)[a] for I/Q points	yes
SPECtrum - (frequency domain) (all modes)	IQ (*n*=3)[a] SPECtrum (*n*=4)[a] ASPectrum (*n*=7)[a] (*n*=0)[a] for I/Q points	yes
WAVEform - (time domain) (all modes)	RFENvelope (*n*=2)[a] (also for Signal Envelope trace) IQ (*n*=5)[a] (*n*=0)[a] for I/Q points	yes

a. The *n* number indicates the subopcode that corresponds to this trace. Detailed descriptions of the trace data can be found in the MEASure subsystem documentation by looking up the subopcode for the appropriate measurement.

Calculate peaks of trace data (Remote Command Only)

Returns a list of peaks for the designated trace data n for the currently selected measurement. The peaks must meet the requirements of the peak threshold and excursion values. The command can only be used with specific [n] (subopcode) values, for measurement results that are

trace, or scalar, data. See the remote command section of each measurement for the appropriate subopcodes. Both real and complex traces can be searched, but complex traces are converted to magnitude in dBm. subopcode n=0, is the raw trace data which cannot be searched for peaks. subopcode n=1, is the scaler data which also cannot be searched for peaks.

Mode	All
Remote Command	`:CALCulate:DATA[n]:PEAKs?` `<threshold>,<excursion>[,AMPLitude|FREQuency|TIME]`
Notes	The return trace depends on the measurement.
Help Map ID	0

Adjacent Channel Power Ratio (ACP) Measurement

This measures the total rms power in the specified channel and in 5 offset channels. You must be in Basic, cdmaOne, cdma2000, W-CDMA (3GPP), iDEN, NADC or PDC mode to use these commands. Use INSTrument:SELect to set the mode.

The general functionality of CONFigure, FETCh, MEASure, and READ are described at the beginning of this section. See the SENSe:ACP commands for more measurement related commands.

:CONFigure:ACP

:FETCh:ACP[n]?

:READ:ACP[n]?

:MEASure:ACP[n]?

For Basic mode, a channel frequency and power level can be defined in the command statement to override the default standard setting. A comma must precede the power value as a place holder for the frequency, when no frequency is sent.

History: Added to Basic mode, version A.03.00 or later

Front Panel Access: **Measure**, **ACP** or **ACPR**

After the measurement is selected, press **Restore Meas Defaults** to restore factory defaults.

Measurement Results Available

Measurement Type	n	Results Returned
	0	Returns unprocessed I/Q trace data, as a series of comma-separated trace points, in volts. The I values are listed first in each pair, using the 0 through even-indexed values. The Q values are the odd-indexed values.

Measurement Type	n	Results Returned
	not specified or n=1 NADC and PDC mode	Returns 22 comma-separated scalar results, in the following order: 1. Center frequency – absolute power (dBm) 2. Center frequency – absolute power (W) 3. Negative offset frequency (1) – relative power (dB) 4. Negative offset frequency (1) – absolute power (dBm) 5. Positive offset frequency (1) – relative power (dB) 6. Positive offset frequency (1) – absolute power (dBm) . . . 1. Positive offset frequency (5) – relative power (dB) 2. Positive offset frequency (5) – absolute power (dBm)
	not specified or n=1 iDEN mode	Returns 13 comma-separated scalar results, in the following order: 1. Center frequency – relative power (dB) 2. Center frequency – absolute power (dBm) 3. Lower offset frequency – relative power (dB) 4. Lower offset freq– absolute power (dBm) 5. Upper offset frequency – relative power (dB) 6. Upper offset frequency – absolute power (dBm) 7. Total power (dBm) 8. Offset frequency (Hz) 9. Reference BW (Hz) 10. Offset BW (Hz) 11. Carrier/center frequency (Hz) 12. Frequency span (Hz) 13. Average count

Measurement Type	n	Results Returned
Total power reference	not specified or n=1 Basic, cdmaOne, cdma2000, or W-CDMA (3GPP) mode	Returns 24 comma-separated scalar results, in the following order: 1. Upper adjacent chan center frequency - relative power (dB) 2. Upper adjacent chan center frequency - absolute power (dBm) 3. Lower adjacent chan center frequency - relative power (dB) (same as upper) 4. Lower adjacent chan center frequency - absolute power (dBm) (same as upper) 5. Negative offset frequency (1) - relative power (dB), 6. Negative offset frequency (1) - absolute power (dBm) 7. Positive offset frequency (1) - relative power (dB) 8. Positive offset frequency (1) - absolute power (dBm) . . . 1. Positive offset frequency (5) - relative power (dB) 2. Positive offset frequency (5) - absolute power (dBm)

Measurement Type	n	Results Returned
Power spectral density reference	not specified or n=1 Basic, cdmaOne, cdma2000, or W-CDMA (3GPP) mode	Returns 24 comma-separated scalar results, in the following order: 1. Upper adjacent chan center frequency - relative power (dB) 2. Upper adjacent chan center frequency - absolute power (dBm/Hz) 3. Lower adjacent chan center frequency - relative power (dB) (same as upper) 4. Lower adjacent chan center frequency - absolute power (dBm/Hz) (same as upper) 5. Negative offset frequency (1) - relative power (dB) 6. Negative offset frequency (1) - absolute power (dBm/Hz) 7. Positive offset frequency (1) - relative power (dB) 8. Positive offset frequency (1) - absolute power (dBm/Hz) . . . 1. Positive offset frequency (5) - relative power (dB) 2. Positive offset frequency (5) - absolute power (dBm/Hz)
	2 NADC and PDC mode	Returns 10 comma-separated scalar values of the pass/fail (0=passed, or 1=failed) results determined by testing the absolute power of the offset frequencies: 1. Negative offset frequency (1) absolute power 2. Positive offset frequency (1) absolute power . . . 1. Negative offset frequency (5) absolute power 2. Positive offset frequency (5) absolute power
	2 iDEN mode	Returns 3 comma-separated scalar values of the histogram absolute power trace: 1. Lower offset frequency – absolute power 2. Reference frequency – absolute power 3. Upper offset frequency – absolute power

Measurement Type	n	Results Returned
Total power reference	2 Basic, cdmaOne, cdma2000, or W-CDMA (3GPP) mode	Returns 11 comma-separated scalar values (in dBm) corresponding to the total power histogram display. The values are returned in ascending frequency order: 1. Negative offset frequency (5) 2. Negative offset frequency (4) . . . 1. Center frequency 2. Positive offset frequency (1) . . . 1. Positive offset frequency (5)
	3 NADC and PDC mode	Returns 10 comma-separated scalar values of the pass/fail (0=passed, or 1=failed) results determined by testing the relative power of the offset frequencies: 1. Negative offset frequency (1) relative power 2. Positive offset frequency (1) relative power . . . 1. Negative offset frequency (5) relative power 2. Positive offset frequency (5) relative power
	3 iDEN mode	Returns 3 comma-separated scalar values of the histogram relative power trace: 1. Lower offset frequency – relative power 2. Reference frequency – relative power 3. Upper offset frequency – relative power

Measurement Type	n	Results Returned
Power spectral density reference	3 Basic, cdmaOne, cdma2000, or W-CDMA (3GPP) mode	Returns 11 comma-separated scalar values (in dBm/Hz) corresponding to the power spectral density histogram display. The values are returned in ascending frequency order: 1. Negative offset frequency (5) 2. Negative offset frequency (4) . . . 1. Center frequency 2. Positive offset frequency (1) . . . 1. Positive offset frequency (5)
	4 NADC and PDC mode	Returns the frequency-domain spectrum trace (data array) for the entire frequency range being measured. In order to return spectrum data, the ACP display must be in the spectrum view and you must not turn off the spectrum trace.
	4 iDEN mode	Returns 4 comma-separated absolute power results for the reference and offset channels. 1. Reference channel – absolute power 2. Reference channel – absolute power (duplicate of above) 3. Lower offset channel – absolute power 4. Upper offset channel – absolute power

Measurement Type	n	Results Returned
(For cdma2000 and W-CDMA the data is only available with spectrum display selected)	4 Basic, cdmaOne, cdma2000, or W-CDMA (3GPP) mode	Returns the frequency-domain spectrum trace data for the entire frequency range being measured. With the spectrum view selected (DISPlay:ACP:VIEW SPECtrum) and the spectrum trace on (SENSe:ACP:SPECtrum:ENABle): • In FFT mode (SENSe:ACP:SWEep:TYPE FFT) the number of trace points returned are 343 (cdma2000) or 1715 (W-CDMA). This is with the default span of 5 MHz (cdma2000) or 25 MHz (W-CDMA). The number of points also varies if another offset frequency is set. • In sweep mode (SENSe:ACP:SWEep:TYPE SWEep), the number of trace points returned is 601 (for cdma2000 or W-CDMA) for any span. With bar graph display selected, one point of –999.0 will be returned.
	5 iDEN mode	Returns 4 comma-separated relative power values for the reference and offset channels: 1. Reference channel – relative power 2. Reference channel – relative power (duplicate of above) 3. Lower offset channel – relative power 4. Upper offset channel – relative power
Total power reference	5 Basic, cdmaOne, cdma2000, or W-CDMA (3GPP) mode	Returns 12 comma-separated scalar values (in dBm) of the absolute power of the center and the offset frequencies: 1. Upper adjacent chan center frequency 2. Lower adjacent chan center frequency 3. Negative offset frequency (1) 4. Positive offset frequency (1) . . . 1. Negative offset frequency (5) 2. Positive offset frequency (5)

Measurement Type	n	Results Returned
Power spectral density reference	5 Basic, cdmaOne, cdma2000, or W-CDMA (3GPP) mode	Returns 12 comma-separated scalar values (in dBm/Hz) of the absolute power of the center and the offset frequencies: 1. Upper adjacent chan center frequency 2. Lower adjacent chan center frequency 3. Negative offset frequency (1) 4. Positive offset frequency (1) . . . 1. Negative offset frequency (5) 2. Positive offset frequency (5)
	6 iDEN mode	Returns 4 comma-separated pass/fail test results for the absolute power of the reference and offset channels: 1. Reference channel absolute power pass/fail 2. Reference channel absolute power pass/fail (duplicate of above) 3. Lower offset channel absolute power pass/fail 4. Upper offset channel absolute power pass/fail
Total power reference	6 Basic, cdmaOne, cdma2000, or W-CDMA (3GPP) mode	Returns 12 comma-separated scalar values (total power in dB) of the power relative to the carrier at the center and the offset frequencies: 1. Upper adjacent chan center frequency 2. Lower adjacent chan center frequency 3. Negative offset frequency (1) 4. Positive offset frequency (1) 5. Negative offset frequency (5) . . . 1. Negative offset frequency (5) 2. Positive offset frequency (5)

Measurement Type	n	Results Returned
Power spectral density reference	6 Basic, cdmaOne, cdma2000, or W-CDMA (3GPP) mode	Returns 12 comma-separated scalar values (power spectral density in dB) of the power relative to the carrier at the center and offset frequencies: 1. Upper adjacent chan center frequency 2. Lower adjacent chan center frequency 3. Negative offset frequency (1) 4. Positive offset frequency (1) . . . 1. Negative offset frequency (5) 2. Positive offset frequency (5)
	7 iDEN mode	Returns 4 comma-separated pass/fail test results for the relative power of the reference and offset channels: 1. Reference channel relative power pass/fail 2. Reference channel relative power pass/fail (duplicate of above) 3. Lower offset channel relative power pass/fail 4. Upper offset channel relative power pass/fail
Total power reference	7 Basic, cdmaOne, cdma2000, or W-CDMA (3GPP) mode	Returns 12 comma-separated scalar values of the pass/fail (0=passed, or 1=failed) results determined by testing the absolute power limit of the center and offset frequencies (measured as total power in dB): 1. Upper adjacent chan center frequency 2. Lower adjacent chan center frequency 3. Negative offset frequency (1) 4. Positive offset frequency (1) . . . 1. Negative offset frequency (5) 2. Positive offset frequency (5)

Measurement Type	n	Results Returned
Power spectral density reference	7 Basic, cdmaOne, cdma2000, or W-CDMA (3GPP) mode	Returns 12 comma-separated scalar values of the pass/fail (0=passed, or 1=failed) results determined by testing the absolute power limit of the center and offset frequencies (measured as power spectral density in dB): 1. Upper adjacent chan center frequency 2. Lower adjacent chan center frequency 3. Negative offset frequency (1) 4. Positive offset frequency (1) . . . 1. Negative offset frequency (5) 2. Positive offset frequency (5)
Total power reference	8 Basic, cdmaOne, cdma2000, or W-CDMA (3GPP) mode	Returns 12 comma-separated scalar values of the pass/fail (0=passed, or 1=failed) results determined by testing the power limit relative to the center frequency (measured as total power spectral in dB): 1. Upper adjacent chan center frequency 2. Lower adjacent chan center frequency 3. Negative offset frequency (1) 4. Positive offset frequency (1) . . . 1. Negative offset frequency (5) 2. Positive offset frequency (5)
Power spectral density reference	8 Basic, cdmaOne, cdma2000, or W-CDMA (3GPP) mode	Returns 12 comma-separated scalar values of the pass/fail (0=passed, or 1=failed) results determined by testing the power limit relative to the center frequency (measured as power spectral density in dB): 1. Upper adjacent chan center frequency 2. Lower adjacent chan center frequency 3. Negative offset frequency (1) 4. Positive offset frequency (1) . . . 1. Negative offset frequency (5) 2. Positive offset frequency (5)

50 MHz Amplitude Reference Measurement

This aligns the internal 50 MHz reference signal to an external reference signal that you supply. You must be in the Service mode to use these commands. Use INSTrument:SELect to set the mode.

The general functionality of CONFigure, FETCh, MEASure, and READ are described at the beginning of this section. See the SENSe:AREFerence commands for more measurement related commands.

`:CONFigure:AREFerence`

`:FETCh:AREFerence[n]?`

`:READ:AREFerence[n]?`

`:MEASure:AREFerence[n]?`

Remarks: For auto adjustment of the internal 50 MHz amplitude reference, use CALibration:AMPLitude:REFerence:AADJust command after this measurement has been selected.

Front Panel Access: **Measure, 50 MHz Amptd**

After the measurement is selected, press **Restore Meas Defaults** to restore factory defaults.

Measurement Results Available

n	Results Returned
not specified or n=1	Returns 7 scalar results: 1. RF input average amplitude 2. 50 MHz reference oscillator average amplitude 3. Average amplitude error 4. State (for factory use only) 5. Level (for factory use only) 6. Monitored level (for factory use only) 7. Connector status (for factory use only)
2	RF input amplitude trace data.
3	50 MHz oscillator amplitude trace data
4	Amplitude error strip chart trace data

Channel Power Measurement

This measures the total rms power in a specified integration bandwidth. You must be in the Basic, cdmaOne, cdma2000, or W-CDMA (3GPP) mode to use these commands. Use INSTrument:SELect to set the mode.

The general functionality of CONFigure, FETCh, MEASure, and READ are described at the beginning of this section. See the SENSe:CHPower commands for more measurement related commands.

:CONFigure:CHPower

:FETCh:CHPower[n]?

:READ:CHPower[n]?

:MEASure:CHPower[n]?

History: Added to Basic mode, version A.03.00 or later

Front Panel Access: **Measure, Channel Power**

After the measurement is selected, press **Restore Meas Defaults** to restore factory defaults.

Measurement Results Available

n	Results Returned
0	Returns unprocessed I/Q trace data, as a series of comma-separated trace points, in volts. The I values are listed first in each pair, using the 0 through even-indexed values. The Q values are the odd-indexed values.
not specified or n=1	Returns 2 comma-separated scalar results: 1. **Channel Power** is a floating point number representing the total channel power in the specified integration bandwidth. 2. **PSD** (**Power Spectral Density**) is the power (in dBm/Hz) in the specified integration bandwidth.
2	Returns comma-separated floating point numbers that are the captured trace data of the power (in dBm/resolution BW) of the signal. The frequency span of the captured trace data is specified by the **Span** key.

Power Statistics CCDF Measurement

This is a statistical power measurement of the complimentary cumulative distribution function (CCDF). You must be in the Basic, cdma2000, or W-CDMA (3GPP) mode to use these commands. Use INSTrument:SELect to set the mode.

The general functionality of CONFigure, FETCh, MEASure, and READ are described at the beginning of this section. See the SENSe:PSTat commands for more measurement related commands.

:CONFigure:PSTatistic

:FETCh:PSTatistic[n]?

:READ:PSTatastic[n]?

:MEASure:PSTatastic[n]?

History: Version A.03.00 or later, added in Basic A.04.00

Front Panel Access: **Measure, Power Stat CCDF**

After the measurement is selected, press **Restore Meas Defaults** to restore factory defaults.

Measurement Results Available

n	
0	Returns unprocessed I/Q trace data, as a series of comma-separated trace points, in volts. The I values are listed first in each pair, using the 0 through even-indexed values. The Q values are the odd-indexed values,
not specified or n=1	Returns 10 comma-separated scalar results: 1. Average input power (in dBm) 2. Probability at the average input power level (in %) 3. Power level that has 10% of the power 4. Power level that has 1% of the power 5. Power level that has 0.1% of the power 6. Power level that has 0.01% of the power 7. Power level that has 0.001% of the power 8. Power level that has 0.0001% of the power 9. Peak power (in dB) 10. Count

n	
2	Returns a series of 5001 floating point numbers (in percent) that represent the current measured power stat trace. This is the probability at particular power levels (average power), in the following order: 1. Probability at 0.0 dB power 2. Probability at 0.01 dB power 3. Probability at 0.02 dB power . . . 1. Probability at 49.9 dB power 2. Probability at 50.0 dB power
3	Returns a series of 5001 floating point numbers (in percent) that represent the Gaussian trace. This is the probability at particular power levels (average power), in the following order: 1. Probability at 0.0 dB power 2. Probability at 0.01 dB power 3. Probability at 0.02 dB power . . . 1. Probability at 49.9 dB power 2. Probability at 50.0 dB power
4	Returns a series of 5001 floating point numbers (in percent) that represent the user-definable reference trace. This is the probability at particular power levels (average power), in the following order: 1. Probability at 0.0 dB power 2. Probability at 0.01 dB power 3. Probability at 0.02 dB power . . . 1. Probability at 49.9 dB power 2. Probability at 50.0 dB power

Power vs. Time Measurement

This measures the average power during the “useful part” of the burst comparing the power ramp to required timing mask. You must be in EDGE, GSM or Service mode to use these commands. Use INSTrument:SELect to set the mode.

The general functionality of CONFigure, FETCh, MEASure, and READ are described at the beginning of this section. See the SENSe:PVTime commands for more measurement related commands.

:CONFigure:PVTime

:FETCh:PVTime[n]?

:READ:PVTime[n]?

:MEASure:PVTime[n]?

Front Panel Access: **Measure, Power vs Time**

After the measurement is selected, press **Restore Meas Defaults** to restore factory defaults.

History: Modified in version A.05.00.

Measurement Results Available

n	Results Returned
0	Returns unprocessed I/Q trace data, as a series of comma-separated trace points, in volts. The I values are listed first in each pair, using the 0 through even-indexed values. The Q values are the odd-indexed values.

n	Results Returned
not specified or n=1	Returns the following comma-separated scalar results: 1. **Sample time** is a floating point number that represents the time between samples when using the trace queries (n=0,2,etc.). 2. **Power of single burst** is the mean power (in dBm) across the useful part of the selected burst in the most recently acquired data, or in the last data acquired at the end of a set of averages. If averaging is on, the power is for the last burst. 3. **Power averaged** is the power (in dBm) of N averaged bursts, if averaging is on. The power is averaged across the useful part of the burst. Average *m* is a single burst from the acquired trace. If there are multiple bursts in the acquired trace, only one burst is used for average *m*. This means that N traces are acquired to make the complete average. If averaging is off, the value of **power averaged** is the same as the **power single burst** value. 4. **Number of samples** is the number of data points in the captured signal. This number is useful when performing a query on the signal (i.e. when n=0,2,etc.). 5. **Start point of the useful part of the burst** is the index of the data point at the start of the useful part of the burst 6. **Stop point of the useful part of the burst** is the index of the data point at the end of the useful part of the burst 7. Index of the data point where T_0 occurred. 8. **Burst width of the useful part of the burst** is the width of the burst measured at –3 dB below the mean power in the useful part of the burst. 9. **Maximum value** is the maximum value of the most recently acquired data (in dBm). 10. **Minimum value** is the minimum value of the most recently acquired data (in dBm). 11. **Burst search threshold** is the value (in dBm) of the threshold where a valid burst is identified, after the data has been acquired. 12. **IQ point delta** is the number of data points offset that are internally applied to the useful data in traces *n*=2,3,4. You must apply this correction value to find the actual location of the **Start**, **Stop**, or T_0 values.

n	Results Returned
2	Returns comma-separated trace points of the entire captured I/Q trace data. These data points are floating point numbers representing the power of the signal (in dBm). There are N data points, where N is the **number of samples**. The period between the samples is defined by the **sample time**.
3	Returns comma-separated points representing the upper mask (in dBm).
4	Returns comma-separated points representing the lower mask (in dBm).
7	Returns power level values for the 8 slots in the current frame (in dBm).
8, only available when averaging is set to both maximum and minimum	Returns comma-separated trace points of the minimum waveform data. These data points are floating point numbers representing the power of the signal (in dBm). There are N data points, where N is the **number of samples**. The period between the samples is defined by the **sample time**. Use SENSe:PVT:AVERage:TYPE MXMinimum to set averaging to max and min. Use n=2 to return the corresponding maximum trace.

Sensor Measurement

This checks the output of three sensors in the RF and IF circuitry. You must be in the Service mode to use these commands. Use INSTrument:SELect to set the mode.

The general functionality of CONFigure, FETCh, MEASure, and READ are described at the beginning of this section.

:CONFigure:SENSors

:FETCh:SENSors[n]?

:READ:SENSors[n]?

:MEASure:SENSors[n]?

Front Panel Access: With Service Mode selected, **Measure, Sensors**

Measurement Results Available

n	Results Returned
0	Not valid
not specified or n=1	Returns the following comma-separated scalar results: 1. **IF signal amplitude** is the ADC value for the detected 21.4 MHz IF signal at the input to the analog IF. 2. **Calibration Oscillator Level** is a floating point number (is not implemented, currently returns a zero). 3. **RF temperature** is a floating point number for the current temperature in the RF section (in degrees Celsius).

Spectrum (Frequency Domain) Measurement

This measures the amplitude of your input signal with respect to the frequency. It provides spectrum analysis capability using FFT (fast Fourier transform) measurement techniques. You must be in the Basic, cdmaOne, cdma2000, 1xEV-DO, W-CDMA, GSM (w/EDGE), NADC, or PDC mode to use these commands. Use INSTrument:SELect, to select the mode.

The general functionality of CONFigure, FETCh, MEASure, and READ are described at the beginning of this section. See the SENSe:SPECtrum commands for more measurement related commands.

`:CONFigure:SPECtrum`

`:FETCh:SPECtrum[n]?`

`:READ:SPECtrum[n]?`

`:MEASure:SPECtrum[n]?`

Front Panel Access: **Measure, Spectrum (Freq Domain)**

After the measurement is selected, press **Restore Meas Defaults** to restore factory defaults.

Measurement Results Available

n	Results Returned
0	Returns unprocessed I/Q trace data, as a series of comma-separated trace points, in volts. The I values are listed first in each pair, using the 0 and even-indexed values. The Q values are the odd-indexed values.

n	Results Returned
not specified or n=1	Returns the following comma-separated scalar results: 1. **FFT peak** is the FFT peak amplitude. 2. **FFT frequency** is the FFT frequency of the peak amplitude. 3. **FFT points** is the Number of points in the FFT spectrum. 4. **First FFT frequency** is the frequency of the first FFT point of the spectrum. 5. **FFT spacing** is the frequency spacing between the FFT points of the spectrum. 6. **Time domain points** is the number of points in the time domain trace used for the FFT. The number of points doubles if the data is complex instead of real. See the time domain scaler description below. 7. **First time point** is the time of the first time domain point, where time zero is the trigger event. (Also called, trigger error offset.) 8. **Time spacing** is the time spacing between the time domain points. The time spacing value doubles if the data is complex instead of real. See the time domain scaler description below. 9. **Time domain** returns a 1 if time domain is complex (I/Q) and complex data will be returned. It returns a 0 if the data is real (raw ADC samples). When this value is 1 rather than 0 (complex vs. real data), the time domain points and the time spacing scalers both increase by a factor of two. 10. **Scan time** is the total scan time of the time domain trace used for the FFT. The total scan time = (time spacing) X (time domain points – 1) 11. **Current average count** is the current number of data measurements that have already been combined, in the averaging calculation.
2, **Service** mode only	Returns the trace data of the log-magnitude versus time. (That is, the RF envelope.)
3	Returns the I and Q trace data. It is represented by I and Q pairs (in volts) versus time.
4	Returns spectrum trace data. That is, the trace of log-magnitude versus frequency. (The trace is computed using a FFT.)
5, **Service** mode only	Returns the averaged trace data of log-magnitude versus time. (That is, the RF envelope.)

n	Results Returned
6	Not used.
7	Returns the averaged spectrum trace data. That is, the trace of the averaged log-magnitude versus frequency.
8	Not used.
9, **Service** mode only	Returns a trace containing the shape of the FFT window.
10, **Service** mode only	Returns trace data of the phase of the FFT versus frequency.
11, cdma2000, 1xEV-DO, W-CDMA, Basic modes only	Linear Spectrum data.
12, cdma2000, 1xEV-DO, W-CDMA, Basic modes only	Average Linear Spectrum data.

Timebase Frequency Measurement

The general functionality of CONFigure, FETCh, MEASure, and READ are described at the beginning of this section. See the SENSe:TBFRequency commands for more measurement related commands.

You must be in the Service mode to use these commands. Use INSTrument:SELect to set the mode.

`:CONFigure:TBFRequency`

`:FETCh:TBFRequency[n]?`

`:READ:TBFRequency[n]?`

`:MEASure:TBFRequency[n]?`

Remarks: For auto adjustment of the internal frequency reference (10 MHz timebase), use the CALibration:FREQuency:REFerence:AADJust command after this measurement has been selected.

Front Panel Access: **Measure, Timebase Freq**

After the measurement is selected, press **Restore Meas Defaults** to restore factory defaults.

Measurement Results Available

n	Results Returned
0	Not valid
not specified or n=1	Returns 3 scalar results: 1. RF input average amplitude 2. Average frequency error 3. Adjustment in process (returns 1 if an adjustment is being performed, returns 0 if no adjustment is in process)
2	Frequency error stripchart trace data.

Waveform (Time Domain) Measurement

This measures the power in your input signal with respect to time and is equivalent to zero-span operation in a traditional spectrum analyzer. You must be in the Basic, cdmaOne, cdma2000, 1xEV-DO, W-CDMA, GSM (w/EDGE), NADC, or PDC mode to use these commands. Use INSTrument:SELect, to select the mode.

The general functionality of CONFigure, FETCh, MEASure, and READ are described at the beginning of this section. See the SENSe:WAVeform commands for more measurement related commands.

`:CONFigure:WAVeform`

`:FETCh:WAVeform[n]?`

`:READ:WAVeform[n]?`

`:MEASure:WAVeform[n]?`

Front Panel Access: **Measure, Waveform (Time Domain)**

After the measurement is selected, press **Restore Meas Defaults** to restore factory defaults.

Measurement Results Available

n	Results Returned
0	Returns unprocessed I/Q trace data, as a series of comma-separated trace points, in volts. The I values are listed first in each pair, using the 0 and even-indexed values. The Q values are the odd-indexed values.

n	Results Returned
not specified or n=1	Returns the following comma-separated scalar results: 1. **Sample time** is a floating point number representing the time between samples when using the trace queries (n=0,2,etc). 2. **Mean power** is the mean power (in dBm). This is either the power across the entire trace, or the power between markers if the markers are enabled. If averaging is on, the power is for the latest acquisition. 3. **Mean power averaged** is the power (in dBm) for N averages, if averaging is on. This is either the power across the entire trace, or the power between markers if the markers are enabled. If averaging is on, the power is for the latest acquisition. If averaging is off, the value of the mean power averaged is the same as the value of the mean power. 4. **Number of samples** is the number of data points in the captured signal. This number is useful when performing a query on the signal (i.e. when n=0,2,etc.). 5. **Peak-to-mean ratio** has units of dB. This is the ratio of the maximum signal level to the mean power. Valid values are only obtained with averaging turned off. If averaging is on, the peak-to-mean ratio is calculated using the highest peak value, rather than the displayed average peak value. 6. **Maximum value** is the maximum of the most recently acquired data (in dBm). 7. **Minimum value** is the minimum of the most recently acquired data (in dBm).
2	Returns comma-separated trace points of the entire captured trace data. These data points are floating point numbers representing the power of the signal (in dBm). There are N data points, where N is the **number of samples**. The period between the samples is defined by the **sample time**.
3	Linear Spectrum data.
4	Phase.
5	I/Q Time.

MEMory Subsystem

The purpose of the MEMory subsystem is to manage instrument memory. This specifically excludes memory used for mass storage which is defined in the MMEMory Subsystem.

Install Application

`:MEMory:INSTall:APPLication <filename>`

Installs the specified application from an external drive to the instrument. Each application allows you to make a specific set of measurements easily and accurately. Installation requires a 12-character license key that you received with your application. The license key number is unique to the option and instrument serial number. If it cannot be located, contact your local Agilent Technologies, Inc. Sales and Service office to re-obtain the information. (Have the instrument model number, option and serial number available.)

Front Panel Access: **System, Uninstall**

Un-install Application

`:MEMory:UNINstall:APPLication <filename>`

Uninstalls (deletes) the specified application from the instrument memory. Re-installation of these programs requires a license key that can be found in the documentation. It can also be found in the **System**, **Options** information screen. Please make a note of this number as it will be needed later to re-install the application.

Front Panel Access: **System, Uninstall**

MMEMory Subsystem

The purpose of the MMEMory subsystem is to provide access to mass storage devices such as internal or external disk drives. Any part of memory that is treated as a device will be in the MMEMory subsystem.

If mass storage is not specified in the filename, the default mass storage specified in the MSIS command will be used.

Memory Available or In-Use

:MMEMory:FREE?

Queries the memory for optional application modes, like option BAH (GSM mode) or option BAE (NADC/PDC mode). The query returns two values, the memory currently in use and the free memory. The sum of the two values is the total instrument memory.

History: Revision A.03.00 or later

Front Panel Access: **System, File System**

Select a Memory Device

:MMEMory:MSIS A|[C]

:MMEMory:MSIS?

Selects a default mass storage device which is used by all MMEMory commands.

The query returns the default mass storage device.

A is the 3.5 inch floppy disk

C is the internal memory

Example: MMEM:MSIS C

History: Added in version A.04.00 and later

Front Panel Access: **Print Setup, Print To File, File Location**

Store a Screen Image in a Graphic File

:MMEMory:STORe:SCReen[:IMMediate] <filename>

The :MMEMory:STORe:SCReen[:IMMediate] command will write the

screen image to a file regardless of what the front panel **Print Setup, Print To** key function is set to. Screen files are always saved in color with an orientation of portrait.

The <filename> variable is composed of:
[<device>:]<name>[.<extension>] where:

<filename> is a string that must be enclosed in single (') or double (") quotes.

<device> must be A or C. Upper or lower case is acceptable. If device is not specified the default is set by MMEM:MSIS.

<name> must be 1 to 8 characters in length and consist only of the characters a...z, A...Z and 0...9 (no underscore). If a name is not specified the default is screen1.

<.extension> must be .gif|.bmp|.wmf. (Note the lower case.) If a file type extension is not specified the default is set by MMEM:STORE:SCREEN:FILE:TYPE

Example: MMEM:STOR:SCR “C:myscreen.gif”

Remarks: When writing to A, <name> can be any valid DOS-compatible name.

When writing to C, <name> must be screen1...screen6. (Note the lower case.)

If you write a file to C any existing screen file with the same name will be replaced, regardless of the extensions. For example, file screen3.gif will replace file screen3.bmp

History: VSA - Added in version A.04.00 and later

Front Panel Access: **Print Setup, Print To File**

Print

Screen File Type

`:MMEMory:STORe:SCReen:FILE[:TYPE] GIF|BMP|WMF`

Sets the default file type for the :MMEMory:STORe:SCReen command.

Factory Preset and *RST: GIF. The file type setting is persistent. It stays at the last user-selected setting even through a power cycle.

Default: GIF

History: Added in version A.04.00 and later

Front Panel Access: **Print Setup, Print To File, File Type**

Screen Image Background

`:MMEMory:STORe:SCReen:IMAGe NORMal|INVert`

`:MMEMory:STORe:SCReen:IMAGe?`

Selects the background color of trace data windows when writing to a file.

NORMal background is black.

INVert background is white.

Factory Preset and *RST: The image setting is persistent. It stays at the last user-selected setting even through a power cycle.

Default: Invert

History: Added in version A.04.00 and later

Front Panel Access: **Print Setup, Print To File, Image**

READ Subsystem

The READ? commands are used with several other commands and are documented in the section on the "MEASure Group of Commands" on page 301.

Initiate and Read Measurement Data

:READ:<measurement>[n]?

A READ? query must specify the desired measurement. It will cause a measurement to occur without changing any of the current settings and will return any valid results. The code number n selects the kind of results that will be returned. The available measurements and data results are described in "MEASure Group of Commands" on page 301.

SENSe Subsystem

Sets the instrument state parameters so that you can measure the input signal.

The SCPI default for data output format is ASCII. The format can be changed to binary with FORMat:DATA which transports faster over the bus.

Adjacent Channel Power Measurement

Commands for querying the adjacent channel power measurement results and for setting to the default values are found in "MEASure Group of Commands" on page 301. The equivalent front-panel keys for the parameters described in the following commands, are found under the **Meas Setup** key, after the **ACP** or **ACPR** measurement has been selected from the **MEASURE** key menu.

Adjacent Channel Power—Average Count

```
[:SENSe]:ACP:AVERage:COUNt <integer>
```

```
[:SENSe]:ACP:AVERage:COUNt?
```

Set the number of data acquisitions that will be platform averaged. After the specified number of average counts, the average mode (termination control) setting determines the average action.

Factory Preset and *RST: 10 for cdma2000, W-CDMA (3GPP)

20 for Basic, cdmaOne, iDEN

Range: 1 to 10,000

Remarks: Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Averaging State

```
[:SENSe]:ACP:AVERage[:STATe] OFF|ON|0|1
```

```
[:SENSe]:ACP:AVERage[:STATe]?
```

Turn average on or off.

Factory Preset and *RST: On

Off for iDEN mode

Remarks: Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Averaging Termination Control

[:SENSe]:ACP:AVERage:TCONtrol EXPonential|REPeat

[:SENSe]:ACP:AVERage:TCONtrol?

Select the type of termination control used for averaging. This determines the averaging action after the specified number of data acquisitions (average count) is reached.

EXPonential – Each successive data acquisition after the average count is reached, is exponentially weighted and combined with the existing average.

REPeat – After reaching the average count, the averaging is reset and a new average is started.

Factory Preset and *RST: REPeat for basic, cdmaOne, cdma2000, W-CDMA (3GPP)

EXPonential for NADC, PDC, iDEN

Remarks: Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Type of Carrier Averaging

[:SENSe]:ACP:AVERage:TYPE MAXimum|RMS

[:SENSe]:ACP:AVERage:TYPE?

Selects the type of averaging to be used for the measurement of the carrier.

Factory Preset and *RST: RMS

Remarks: You must be in Basic, cdmaOne mode to use this command. Use INSTrument:SELect to set the mode.

History: Revision A.03.00 or later, in cdmaOne revision A.04.00

Front Panel Access: **Meas Setup, Avg Mode**

Adjacent Channel Power—Carrier Channel BW

Basic, cdmaOne, iDEN mode

[:SENSe]:ACP:BANDwidth|BWIDth:INTegration <freq>

[:SENSe]:ACP:BANDwidth|BWIDth:INTegration?

cdma2000, W-CMDA (3GPP) mode

[:SENSe]:ACP:BANDwidth[n]|BWIDth[n]:INTegration <freq>

[:SENSe]:ACP:BANDwidth[n]|BWIDth[n]:INTegration?

cdmaOne mode

[:SENSe]:ACP:BANDwidth[n]|BWIDth[n]:INTegration[m] <freq>

[:SENSe]:ACP:BANDwidth[n]|BWIDth[n]:INTegration[m]?

Set the Integration bandwidth that will be used for the main (carrier) channel.

BANDwidth[n]|BWIDth[n]:

m=1 is base station and 2 is mobiles. The default is base station (1).

INTegration[n]:

cdmaOne mode m=1 is cellular bands and 2 is pcs bands. The default is cellular.

Factory Preset and *RST:

Mode	**Format (Modulation Standard)**		
Basic	1.23 MHz		
cdmaOne	1.23 MHz		
iDEN	18 kHz		
cdma2000	1.23 MHz		
W-CDMA (3GPP)	3.84 MHz		

Range: 300 Hz to 20 MHz for Basic, cdmaOne, cdma2000, W-CDMA (3GPP) mode

1 kHz to 5 MHz for iDEN

Default Unit: Hz

Remarks: With measurement type set at (TPR) total power reference, 1.40 MHz is sometimes used. Using 1.23 MHz will give a power that is very nearly identical to the 1.40 MHz value, and using 1.23 MHz will also yield the correct power spectral density with measurement type set at (PSD) reference. However, a setting of 1.40 MHz will not give the correct results with measurement type set at PSD reference.

You must be in Basic, cdmaOne, cdma2000, W-CDMA (3GPP), iDEN mode to use this command. Use

INSTrument:SELect to set the mode.

Adjacent Channel Power—Dynamic Range

[:SENSe]:ACP:DRANge HIGH|NORMal|MODified

[:SENSe]:ACP:DRANge?

Select a dynamic range optimization.

High - chooses settings that provide better dynamic range (better signal to noise ratio) at the expense of longer measurement times. This is a better choice for CDMA signals with multiple carriers turned on at the same time.

Normal - lets the measurement automatically choose settings that trade off dynamic range for faster measurement speed. This is a good choice for making CDMA measurements on a signal with only one carrier turned on at a time.

Modified- is not a customer settable option. This choice is automatically selected depending on your selection of other related settings in the advanced measurement setup, like the number of FFT segments.

Factory Preset and *RST:	NORMal
Remarks:	You must be in the cdmaOne mode to use this command. Use INSTrument:SELect to set the mode.
History:	Added revision A.04.00 or later

Adjacent Channel Power—Fast Mode ADC Range

[:SENSe]:ACP:FAST:OFFSet:ADC:RANGe
AUTO|APEak|APLock|M6|P0|P6|P12|P18|P24

[:SENSe]:ACP:FAST:OFFSet:ADC:RANGe?

Select the range for the gain-ranging that is done in front of the ADC when the [:SENSe]:ACP:SWEep:TYPE is set to Fast. This is an advanced control that normally does not need to be changed. If you are measuring a CW signal, see the description below.

- Auto - sets the ADC range automatically. For most FFT measurements, the auto feature should not be selected. An exception is when measuring a signal which is “bursty,” in which case the auto feature can maximize the time domain dynamic range, if FFT results are less important to you than time domain results.
- Auto Peak (APEak) - sets the ADC range automatically to the peak

signal level. The auto peak feature is a compromise that works well for both CW and burst signals.

- Auto Peak Lock (APLock) - holds the ADC range automatically at the peak signal level. The auto peak lock feature is more stable than the auto peak feature for CW signals, but should not be used for “bursty” signals.
- M6 - sets an ADC range that subtracts 6 dB of fixed gain across the range manually. Manual ranging is best for CW signals.
- P0, P6, P12, P18, or P24 - selects ADC ranges that add 0, 6, 12, 18, or 24 dB of fixed gain across the range manually. Manual ranging is best for CW signals.

Factory Preset and *RST: Auto Peak (APEak)

Remarks: You must be in the W-CDMA (3GPP) mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Fast Mode Relative Attenuation

[:SENSe]:ACP:FAST:OFFSet:RATTenuation <float>

[:SENSe]:ACP:FAST:OFFSet:RATTenuation?

Sets a relative amount of attenuation for the measurements at the offset channels when the [:SENSe]:ACP:SWEep:TYPE is set to Fast. This attenuation is always specified relative to the attenuation that is required to measure the carrier channel. Since the offset channel power is lower than the carrier channel power, less attenuation is required to measure the offset channels and wider dynamic range for the measurement is available.

Factory Preset and *RST: 0

Range: –40.00 to 0.00 dB

Remarks: You must be in the W-CDMA (3GPP) mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Root Raised Cosine Filter Alpha

[:SENSe]:ACP:FILTer[:RRC]:ALPHa <numeric>

[:SENSe]:ACP:FILTer[:RRC]:ALPHa?

Set the alpha value of the Root Raised Cosine (RRC) filter.

Factory Preset and *RST: 0.22

Range: 0.01 to 0.5

Remarks: You must be in the W-CDMA (3GPP) mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Root Raised Cosine Filter Control

[:SENSe]:ACP:FILTer[:RRC][:STATe] OFF|ON|0|1

[:SENSe]:ACP:FILTer[:RRC][:STATe]?

Turn the Root Raised Cosine (RRC) filter on or off.

Factory Preset and *RST: On

Remarks: You must be in the W-CDMA (3GPP) mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Reference Channel FFT Segments

[:SENSe]:ACP:FFTSegment <integer>

[:SENSe]:ACP:FFTSegment?

Selects the number of FFT segments used in making the measurement of the reference channel (carrier). In automatic mode the measurement optimizes the number of FFT segments required for the shortest measurement time. The minimum number of segments required to make a measurement is set by your desired measurement bandwidth. Selecting more than the minimum number of segments will give you more dynamic range for making the measurement, but the measurement will take longer to execute.

To use this command you must first set SENSe:ACP:FFTS:AUTO to off.

Factory Preset and *RST: 1

Range: 1 to 12

Remarks: You must be in Basic, cdmaOne mode to use this command. Use INSTrument:SELect to set the mode.

History: Revision A.03.00 or later, in cdmaOne revision A.04.00

Adjacent Channel Power—Reference Channel FFT Segments State

[:SENSe]:ACP:FFTSegment:AUTO OFF|ON|0|1

[:SENSe]:ACP:FFTSegment:AUTO?

The automatic mode selects the optimum number of FFT segments to measure the reference channel (carrier), while making the fastest possible measurement.

Factory Preset and *RST: ON

Remarks: You must be in Basic, cdmaOne mode to use this command. Use INSTrument:SELect to set the mode.

History: Revision A.03.00 or later, in cdmaOne revision A.04.00

Adjacent Channel Power—Frequency Span Query

[:SENSe]:ACP:FREQuency:SPAN?

Returns the span of the spectrum view.

Remarks: You must be in Basic, cdmaOne mode to use this command. Use INSTrument:SELect to set the mode.

MEAS | READ | FETC:ACP4? returns the frequency-domain spectrum trace data for the entire frequency range being measured.

History: Revision A.05.00 or later

Adjacent Channel Power—Offset Frequency Absolute Limit

[:SENSe]:ACP:LIST:ALIMit
<abs_powr>,<abs_powr>,<abs_powr>,<abs_powr>,<abs_powr>

[:SENSe]:ACP:LIST:ALIMit?

Set the absolute limit on offset frequencies relative to the carrier. You can turn off (not use) specific offsets with the [:SENSe]:ACP:LIST:STATe command.

Factory Preset and *RST:

Mode	Offset A	Offset B	Offset C	Offset D	Offset E
NADC	0 dBm	0 dBm	–13 dBm	0 dBm	0 dBm
PDC	0 dBm	0 dBm	0 dBm	0 dBm	0 dBm

Range: –200 to 50 dBm

Remarks: You must be in the NADC, cdmaOne, or PDC mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Offset Frequency

[:SENSe]:ACP:LIST[:FREQuency]
<f_offset>,<f_offset>,<f_offset>,<f_offset>,<f_offset>

[:SENSe]:ACP:LIST[:FREQuency]?

Define the offset frequencies. You can turn off (not use) specific offsets with the [:SENSe]:ACP:LIST:STATe command.

Factory Preset and *RST:

Mode	Offset A	Offset B	Offset C	Offset D	Offset E
NADC	30 kHz	60 kHz	90 kHz	120 kHz	0 Hz
PDC	50 kHz	100 kHz	0 kHz	0 kHz	0 kHz

Range: 10 Hz to 45 MHz

0 to 200 kHz

Default Unit: Hz

Remarks: You must be in the NADC, cdmaOne, or PDC mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Offset Frequency Power Mode

[:SENSe]:ACP:LIST:POWer
INTeg|PEAK,INTeg|PEAK,INTeg|PEAK,INTeg|PEAK,INTeg|PEAK

[:SENSe]:ACP:LIST:POWer?

Define the power measurement mode for each of the offset frequencies. You can turn off (not use) specific offsets with the SENS:ACP:LIST:STATe command.

Factory Preset and *RST: INTeg, INTeg, INTeg, INTeg, INTeg

Remarks: You must be in the NADC mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Offset Frequency Relative Limit

[:SENSe]:ACP:LIST:RLIMit
<rel_powr>,<rel_powr>,<rel_powr>,<rel_powr>,<rel_powr>

[:SENSe]:ACP:LIST:RLIMit?

Set the relative limit on offset frequencies. You can turn off (not use)

specific offsets with the SENS:ACP:LIST:STATe command.

Factory Preset
and *RST: –45 dB

Mode	Offset A	Offset B	Offset C	Offset D	Offset E
NADC	–26 dB	–45 dB	–45 dB	0 dB	0 dB
PDC	–45 dB	–60 dB	0 dB	0 dB	0 dB

Range: –200 to 50 dB

Remarks: You must be in the NADC, cdmaOne, or PDC mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Offset Frequency Control

[:SENSe]:ACP:LIST:STATe OFF|ON|0|1, OFF|ON|0|1, OFF|ON|0|1, OFF|ON|0|1, OFF|ON|0|1

[:SENSe]:ACP:LIST:STATe?

Turn measurement on or off for the custom offset frequencies.

Factory Preset
and *RST:

Mode	Offset A	Offset B	Offset C	Offset D	Offset E
NADC	ON	ON	ON	OFF	OFF
PDC	ON	ON	OFF	OFF	OFF

Remarks: You must be in the NADC, cdmaOne, or PDC mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Offset Frequency Test Mode

[:SENSe]:ACP:LIST:TEST ABSolute|AND|RELative|OR, ABSolute|AND|RELative|OR, ABSolute|AND|RELative|OR, ABSolute|AND|RELative|OR, ABSolute|AND|RELative|OR

[:SENSe]:ACP:LIST:TEST?

Define the type of testing to be done for the five custom offset frequencies. You can turn off (not use) specific offsets with the SENS:ACP:LIST:STATe command.

Factory Preset

and *RST: RELative, RELative, OR, AND, AND for NADC, PDC mode

Remarks: You must be in the NADC, cdmaOne, or PDC mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Absolute Amplitude Limits

iDEN mode

[:SENSe]:ACP:OFFSet:ABSolute <power>

[:SENSe]:ACP:OFFSet:ABSolute?

Basic, cdmaOne

[:SENSe]:ACP:OFFSet:LIST:ABSolute
<power>,<power>,<power>,<power>,<power>

[:SENSe]:ACP:OFFSet:LIST:ABSolute?

cdma2000, W-CDMA (3GPP) mode

[:SENSe]:ACP:OFFSet[n]:LIST:ABSolute
<power>,<power>,<power>,<power>,<power>

[:SENSe]:ACP:OFFSet[n]:LIST:ABSolute?

Sets the absolute amplitude levels to test against for each of the custom offsets. The list must contain five (5) entries. If there is more than one offset, the offset closest to the carrier channel is the first one in the list. [:SENSe]:ACP:OFFSet[n]:LIST[m]:TEST selects the type of testing to be done at each offset.

You can turn off (not use) specific offsets with the [:SENSe]:ACP:OFFSet[n]:LIST:STATe command.

The query returns five (5) real numbers that are the current absolute amplitude test limits.

Offset[n] n=1 is base station and 2 is mobiles. The default is base station (1).

List[m]

cdmaOne mode m=1 is cellular bands and 2 is pcs bands. The default is cellular.

Factory Preset and *RST:

Mode	Variant	Offset A	Offset B	Offset C	Offset D	Offset E
Basic		0 dBm	0 dBm	0 dBm	0 dBm	0 dBm

Mode	Variant	Offset A	Offset B	Offset C	Offset D	Offset E
cdmaOne	BS cellular	0 dBm	0 dBm	0 dBm	0 dBm	0 dBm
	BS pcs	0 dBm	–13 dBm	–13 dBm	0 dBm	0 dBm
	MS cellular	0 dBm	0 dBm	0 dBm	0 dBm	0 dBm
	MS pcs	0 dBm	–13 dBm	–13 dBm	0 dBm	0 dBm
cdma2000		50 dBm	50 dBm	50 dBm	50 dBm	50 dBm
W-CDMA (3GPP)		50 dBm	50 dBm	50 dBm	50 dBm	50 dBm
iDEN		0 dBm	n/a	n/a	n/a	n/a

Range: –200.0 dBm to 50.0 dBm

Default Unit: dBm

Remarks: You must be in Basic, cdmaOne, cdma2000, W-CDMA (3GPP), or iDEN mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Type of Offset Averaging

`[:SENSe]:ACP:OFFSet:LIST:AVERage:TYPE MAXimum|RMS`

`[:SENSe]:ACP:OFFSet:LIST:AVERage:TYPE?`

Selects the type of averaging to be used for the measurement at each offset. You can turn off (not use) specific offsets with the SENS:ACP:OFFSet:LIST:STATe command.

Factory Preset and *RST:

Mode	Offset A	Offset B	Offset C	Offset D	Offset E
Basic & cdmaOne	RMS	RMS	RMS	RMS	RMS

Remarks: You must be in Basic, cdmaOne mode to use this command. Use INSTrument:SELect to set the mode.

History: Revision A.03.00 or later, in cdmaOne revision A.04.00

Adjacent Channel Power—Define Resolution Bandwidth List

iDEN mode

`[:SENSe]:ACP:OFFSet:BANDwidth|BWIDth <res_bw>`

[:SENSe]:ACP:OFFSet:BANDwidth|BWIDth?

Basic mode

[:SENSe]:ACP:OFFSet:LIST:BANDwidth|BWIDth
<res_bw>,<res_bw>,<res_bw>,<res_bw>,<res_bw>

[:SENSe]:ACP:OFFSet:LIST:BANDwidth|BWIDth?

cdma2000, W-CDMA (3GPP) mode

[:SENSe]:ACP:OFFSet[n]:LIST:BANDwidth|BWIDth
<res_bw>,<res_bw>,<res_bw>,<res_bw>,<res_bw>

[:SENSe]:ACP:OFFSet[n]:LIST:BANDwidth|BWIDth?

cdmaOne mode

[:SENSe]:ACP:OFFSet[n]:LIST[n]:BANDwidth|BWIDth
<res_bw>,<res_bw>,<res_bw>,<res_bw>,<res_bw>

[:SENSe]:ACP:OFFSet[n]:LIST[n]:BANDwidth|BWIDth?

Define the custom resolution bandwidth(s) for the adjacent channel power testing. If there is more than one bandwidth, the list must contain five (5) entries. Each resolution bandwidth in the list corresponds to an offset frequency in the list defined by [:SENSe]:ACP:OFFSet[n]:LIST[n][:FREQuency]. You can turn off (not use) specific offsets with the [:SENSe]:ACP:OFFSet[n]:LIST[n]:STATe command.

Offset[n] n=1 is base station and 2 is mobiles. The default is base station (1).

List[n]

cdmaOne mode n=1 is cellular bands and 2 is pcs bands. The default is cellular.

Factory Preset and *RST:

Mode	Variant	Offset A	Offset B	Offset C	Offset D	Offset E
iDEN		10 kHz	n/a	n/a	n/a	n/a
Basic		30 kHz	30 kHz	30 kHz	30 kHz	30 kHz
cdmaOne	BS cellular	30 kHz	30 kHz	30 kHz	30 kHz	30 kHz
	BS pcs	30 kHz	12.5 kHz	1 MHz	30 kHz	30 kHz
	MS cellular	30 kHz	30 kHz	30 kHz	30 kHz	30 kHz
	MS pcs	30 kHz	12.5 kHz	1 MHz	30 kHz	30 kHz
cdma2000		30 kHz	30 kHz	30 kHz	30 kHz	30 kHz

Mode	Variant	Offset A	Offset B	Offset C	Offset D	Offset E
W-CDMA (3GPP)		3.84 MHz	3.84 MHz	3.84 MHz	3.84 MHz	3.84 MHz

Range: 300 Hz to 20 MHz for cdmaOne, Basic, cdma2000, or W-CDMA (3GPP) mode

1 kHz to 5 MHz for iDEN mode

Default Unit: Hz

Remarks: You must be in Basic, cdmaOne, cdma2000, W-CDMA (3GPP), or iDEN mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—FFT Segments

```
[:SENSe]:ACP:OFFSet:LIST:FFTSegment
<integer>,<integer>,<integer>,<integer>,<integer>
```

```
[:SENSe]:ACP:OFFSet:LIST:FFTSegment?
```

Selects the number of FFT segments used in making the measurement. In automatic mode the measurement optimizes the number of FFT segments required for the shortest measurement time. The minimum number of segments required to make a measurement is set by your desired measurement bandwidth. Selecting more than the minimum number of segments will give you more dynamic range for making the measurement, but the measurement will take longer to execute.

Factory Preset and *RST:

Mode	Offset A	Offset B	Offset C	Offset D	Offset E
Basic & cdmaOne	1	1	1	1	1

Range: 1 to 12

Remarks: You must be in Basic, cdmaOne mode to use this command. Use INSTrument:SELect to set the mode.

History: Revision A.03.00 or later, in cdmaOne revision A.04.00

Adjacent Channel Power—Automatic FFT Segments

```
[:SENSe]:ACP:OFFSet:LIST:FFTSegment:AUTO OFF|ON|0|1,
OFF|ON|0|1, OFF|ON|0|1, OFF|ON|0|1, OFF|ON|0|1
```

[:SENSe]:ACP:OFFSet:LIST:FFTSegment:AUTO?

The automatic mode selects the optimum number of FFT segments to make the fastest possible measurement.

Factory Preset and *RST:

Mode	Offset A	Offset B	Offset C	Offset D	Offset E
Basic & cdmaOne	ON	ON	ON	ON	ON

Remarks: You must be in Basic mode to use this command. Use INSTrument:SELect to set the mode.

History: Revision A.03.00 or later

Adjacent Channel Power—Define Offset Frequency List

iDEN mode

[:SENSe]:ACP:OFFSet[:FREQuency] <f_offset>

[:SENSe]:ACP:OFFSet[:FREQuency]?

Basic mode, cdmaOne

[:SENSe]:ACP:OFFSet:LIST[:FREQuency]
<f_offset>,<f_offset>,<f_offset>,<f_offset>,<f_offset>

[:SENSe]:ACP:OFFSet:LIST[:FREQuency]?

cdma2000, W-CDMA (3GPP) mode

[:SENSe]:ACP:OFFSet[n]:LIST[:FREQuency]
<f_offset>,<f_offset>,<f_offset>,<f_offset>,<f_offset>

[:SENSe]:ACP:OFFSet[n]:LIST[:FREQuency]?

cdmaOne mode

[:SENSe]:ACP:OFFSet[n]:LIST[n][:FREQuency]
<f_offset>,<f_offset>,<f_offset>,<f_offset>,<f_offset>

[:SENSe]:ACP:OFFSet[n]:LIST[n][:FREQuency]?

Define the custom set of offset frequencies at which the switching transient spectrum part of the ACP measurement will be made. The list contains five (5) entries for offset frequencies. Each offset frequency in the list corresponds to a reference bandwidth in the bandwidth list.

An offset frequency of zero turns the display of the measurement for that offset off, but the measurement is still made and reported. You can turn off (not use) specific offsets with the [:SENSe]:ACP:OFFSet:LIST:STATe command.

Offset[n] n=1 is base station and 2 is mobiles. The default is base station (1).

List[n]

cdmaOne mode n=1 is cellular bands and 2 is pcs bands. The default is cellular.

Factory Preset and *RST:

Mode	Variant	Offset A	Offset B	Offset C	Offset D	Offset E
iDEN		25 kHz	n/a	n/a	n/a	n/a
Basic		750 kHz	1.98 MHz	0 Hz	0 Hz	0 Hz
cdmaOne	BS cellular	750 kHz	1.98 MHz	0 Hz	0 Hz	0 Hz
	BS pcs	885 kHz	1.25625 MHz	2.75 MHz	0 Hz	0 Hz
	MS cellular	885 kHz	1.98 MHz	0 Hz	0 Hz	0 Hz
	MS pcs	885 kHz	1.25625 MHz	2.75 MHz	0 Hz	0 Hz
cdma2000	BTS	750 kHz	1.98 MHz	0 Hz	0 Hz	0 Hz
	MS	885 kHz	1.98 MHz	0 Hz	0 Hz	0 Hz
W-CDMA (3GPP)		5 MHz	10 MHz	15 MHz	20 MHz	25 MHz

Range: 0 Hz to 20 MHz for iDEN, Basic

0 Hz to 45 MHz for cdmaOne

0 Hz to 100 MHz for cdma2000, W-CDMA (3GPP)

Default Unit: Hz

Remarks: You must be in Basic, cdmaOne, cdma2000, W-CDMA (3GPP), or iDEN mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Number of Measured Points

```
[:SENSe]:ACP:OFFSet:LIST:POINts
<integer>,<integer>,<integer>,<integer>,<integer>
```

```
[:SENSe]:ACP:OFFSet:LIST:POINts?
```

Selects the number of data points. The automatic mode chooses the optimum number of points for the fastest measurement time with acceptable repeatability. The minimum number of points that could be

used is determined by the sweep time and the sampling rate. You can increase the length of the measured time record (capture more of the burst) by increasing the number of points, but the measurement will take longer. Use `[:SENSe]:ACP:POINts` to set the number of points used for measuring the reference channel.

Factory Preset and *RST:

Mode	Offset A	Offset B	Offset C	Offset D	Offset E
Basic & cdmaOne	1024	1024	1024	1024	1024

Range: 64 to 65536

Remarks: The fastest measurement times are obtained when the number of points measured is 2^n.

You must be in Basic, cdmaOne mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Automatic Measurement Points

`[:SENSe]:ACP:OFFSet:LIST:POINts:AUTO OFF|ON|0|1, OFF|ON|0|1, OFF|ON|0|1, OFF|ON|0|1, OFF|ON|0|1`

`[:SENSe]:ACP:OFFSet:LIST:POINts:AUTO?`

Automatically selects the number of points for the optimum measurement speed.

Factory Preset and *RST:

Mode	Offset A	Offset B	Offset C	Offset D	Offset E
Basic & cdmaOne	ON	ON	ON	ON	ON

Remarks: You must be in Basic or cdmaOne mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Relative Attenuation

`[:SENSe]:ACP:OFFSet:LIST:RATTenuation <rel_powr>,<rel_powr>,<rel_powr>,<rel_powr>,<rel_powr>`

`[:SENSe]:ACP:OFFSet:LIST:RATTenuation?`

Sets a relative amount of attenuation for the measurements made at

your offsets. The amount of attenuation is always specified relative to the attenuation that is required to measure the carrier channel. Since the offset channel power is lower than the carrier channel power, less attenuation is required to measure the offset channel and you get wider dynamic range for the measurement.

You can turn off (not use) specific offsets with the SENS:ACP:OFFSet:LIST:STATe command.

Factory Preset and *RST:

Mode	Offset A	Offset B	Offset C	Offset D	Offset E
Basic & cdmaOne	0 dB	0 dB	0 dB	0 dB	0 dB

Range: –40 to 0 dB, but this relative attenuation cannot exceed the absolute attenuation range of 0 to 40 dB.

Default Unit: dB

Remarks: Remember that the attenuation that you specify is always relative to the amount of attenuation used for the carrier channel. Selecting negative attenuation means that you want less attenuation used. For example, if the measurement must use 20 dB of attenuation for the carrier measurement and you want to use 12 dB less attenuation for the first offset, you would send the value –12 dB.

You must be in Basic or cdmaOne mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Relative Attenuation Control

`[:SENSe]:ACP:OFFSet:LIST:RATTenuation:AUTO OFF|ON|0|1`

`[:SENSe]:ACP:OFFSet:LIST:RATTenuation:AUTO?`

Automatically sets a relative attenuation to make measurements with the optimum dynamic range at the current carrier channel power.

You can turn off (not use) specific offsets with the SENS:ACP:OFFSet:LIST:STATe command.

Factory Preset and *RST: ON

Remarks: You must be in Basic or cdmaOne mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Amplitude Limits Relative to the Carrier

iDEN mode

```
[:SENSe]:ACP:OFFSet:RCARrier <rel_power>
```

```
[:SENSe]:ACP:OFFSet:RCARrier?
```

Basic mode, cdmaOne

```
[:SENSe]:ACP:OFFSet:LIST:RCARrier
<rel_power>,<rel_power>,<rel_power>,<rel_power>,<rel_power>
```

```
[:SENSe]:ACP:OFFSet:LIST:RCARrier?
```

cdma2000, W-CDMA (3GPP) mode

```
[:SENSe]:ACP:OFFSet[n]:LIST:RCARrier
<rel_power>,<rel_power>,<rel_power>,<rel_power>,<rel_power>
```

```
[:SENSe]:ACP:OFFSet[n]:LIST:RCARrier?
```

cdmaOne mode

```
[:SENSe]:ACP:OFFSet[n]:LIST[n]:RCARrier
<rel_power>,<rel_power>,<rel_power>,<rel_power>,<rel_power>
```

```
[:SENSe]:ACP:OFFSet[n]:LIST[n]:RCARrier?
```

Sets the amplitude levels to test against for any custom offsets. This amplitude level is relative to the carrier amplitude. If multiple offsets are available, the list contains five (5) entries. The offset closest to the carrier channel is the first one in the list. [:SENSe]:ACP:OFFSet[n]:LIST[n]:TEST selects the type of testing to be done at each offset.

You can turn off (not use) specific offsets with the [:SENSe]:ACP:OFFSet[n]:LIST[n]:STATe command.

The query returns five (5) real numbers that are the current amplitude test limits, relative to the carrier, for each offset.

Offset[n] n=1 is base station and 2 is mobiles. The default is base station (1).

List[n]

cdmaOne mode n=1 is cellular bands and 2 is pcs bands. The default is cellular.

Factory Preset and *RST:

Mode	Variant	Offset A	Offset B	Offset C	Offset D	Offset E
iDEN		0 dBc	n/a	n/a	n/a	n/a

Mode	Variant	Offset A	Offset B	Offset C	Offset D	Offset E
Basic		–45 dBc	–60 dBc	0 dBc	0 dBc	0 dBc
cdmaOne	BS cellular	–45 dBc	–60 dBc	0 dBc	0 dBc	0 dBc
	BS pcs	–45 dBc	0 dBc	0 dBc	0 dBc	0 dBc
	MS cellular	–42 dBc	–54 dBc	0 dBc	0 dBc	0 dBc
	MS pcs	–42 dBc	0 dBc	0 dBc	0 dBc	0 dBc
cdma2000		0 dBc	0 dBc	0 dBc	0 dBc	0 dBc
W-CDMA (3GPP)	BTS	–44.2 dBc	–49.2 dBc	–49.2 dBc	–49.2 dBc	–44.2 dBc
	MS	–32.2 dBc	–42.2 dBc	–42.2 dBc	–42.2 dBc	–42.2 dBc

Range: –150.0 dB to 50.0 dB for cdmaOne, cdma2000, W-CDMA (3GPP), Basic

–200.0 dB to 50.0 dB for iDEN

Default Unit: dB

Remarks: You must be in Basic, cdmaOne, cdma2000, W-CDMA (3GPP), or iDEN mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Amplitude Limits Relative to the Power Spectral Density

iDEN mode

```
[:SENSe]:ACP:OFFSet:RPSDensity <rel_power>
```

```
[:SENSe]:ACP:OFFSet:RPSDensity?
```

Basic mode, cdmaOne

```
[:SENSe]:ACP:OFFSet:LIST:RPSDensity
<rel_power>,<rel_power>,<rel_power>,<rel_power>,<rel_power>
```

```
[:SENSe]:ACP:OFFSet:LIST:RPSDensity?
```

cdma2000, W-CDMA (3GPP) mode

```
[:SENSe]:ACP:OFFSet[n]:LIST:RPSDensity
<rel_power>,<rel_power>,<rel_power>,<rel_power>,<rel_power>
```

```
[:SENSe]:ACP:OFFSet[n]:LIST:RPSDensity?
```

cdmaOne mode

```
[:SENSe]:ACP:OFFSet[n]:LIST[n]:RPSDensity
<rel_power>,<rel_power>,<rel_power>,<rel_power>,<rel_power>
```

[:SENSe]:ACP:OFFSet[n]:LIST[n]:RPSDensity?

Sets the amplitude levels to test against for any custom offsets. This amplitude level is relative to the power spectral density. If multiple offsets are available, the list contains five (5) entries. The offset closest to the carrier channel is the first one in the list. [:SENSe]:ACP:OFFSet[n]:LIST[n]:TEST selects the type of testing to be done at each offset.

You can turn off (not use) specific offsets with the [:SENSe]:ACP:OFFSet[n]:LIST:STATe command.

The query returns five (5) real numbers that are the current amplitude test limits, relative to the power spectral density, for each offset.

Offset[n] n=1 is base station and 2 is mobiles. The default is base station (1).

List[n]

cdmaOne mode n=1 is cellular bands and 2 is pcs bands. The default is cellular.

Factory Preset and *RST:

Mode	Variant	Offset A	Offset B	Offset C	Offset D	Offset E
iDEN		0 dB	n/a	n/a	n/a	n/a
Basic		–28.87 dB	–43.87 dB	0 dB	0 dB	0 dB
cdmaOne	BS cellular	–28.87 dB	–43.87 dB	0 dB	0 dB	0 dB
	BS pcs	–28.87 dB	0 dB	0 dB	0 dB	0 dB
	MS cellular	–25.87 dB	–37.87 dB	0 dB	0 dB	0 dB
	MS pcs	–25.87 dB	0 dB	0 dB	0 dB	0 dB
cdma2000		0 dB	0 dB	0 dB	0 dB	0 dB
W-CDMA (3GPP)	BTS	–44.2 dBc	–49.2 dBc	–49.2 dBc	–49.2 dBc	–44.2 dBc
	MS	–32.2 dBc	–42.2 dBc	–42.2 dBc	–42.2 dBc	–42.2 dBc

Range: –150.0 dB to 50.0 dB for cdmaOne, Basic, cdma2000, W-CDMA (3GPP)

–200.0 dB to 50.0 dB for iDEN

Default Unit: dB

Remarks: You must be in Basic, cdmaOne, cdma2000, W-CDMA (3GPP), or iDEN mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Select Sideband

[:SENSe]:ACP:OFFSet:LIST:SIDE BOTH|NEGative|POSitive, BOTH|NEGative|POSitive, BOTH|NEGative|POSitive, BOTH|NEGative|POSitive, BOTH|NEGative|POSitive

[:SENSe]:ACP:OFFSet:LIST:SIDE?

Selects which sideband will be measured. You can turn off (not use) specific offsets with the SENS:ACP:OFFSet:LIST:STATe command.

Factory Preset
and *RST:

Mode	Offset A	Offset B	Offset C	Offset D	Offset E
Basic & cdmaOne	BOTH	BOTH	BOTH	BOTH	BOTH

Remarks: You must be in Basic or cdmaOne mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Control Offset Frequency List

Basic mode, cdmaOne

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

[:SENSe]:ACP:OFFSet:LIST:STATe?

cdma2000, W-CDMA (3GPP) mode

[:SENSe]:ACP:OFFSet[n]:LIST:STATe OFF|ON|0|1, OFF|ON|0|1, OFF|ON|0|1, OFF|ON|0|1, OFF|ON|0|1

[:SENSe]:ACP:OFFSet[n]:LIST:STATe?

cdmaOne mode

[:SENSe]:ACP:OFFSet[n]:LIST[n]:STATe OFF|ON|0|1, OFF|ON|0|1, OFF|ON|0|1, OFF|ON|0|1, OFF|ON|0|1

[:SENSe]:ACP:OFFSet[n]:LIST[n]:STATe?

Selects whether testing is to be done at the custom offset frequencies. The measured powers are tested against the absolute values defined with [:SENSe]:ACP:OFFSet[n]:LIST[n]:ABSolute, or the relative values defined with [:SENSe]:ACP:OFFSet[n]:LIST[n]:RPSDensity and [:SENSe]:ACP:OFFSet[n]:LIST[n]:RCARier.

Offset[n] n=1 is base station and 2 is mobiles. The default is base station (1).

List[n]

cdmaOne mode n=1 is cellular bands and 2 is pcs bands. The default is cellular.

Factory Preset and *RST:

Mode	Variant	Offset A	Offset B	Offset C	Offset D	Offset E
Basic		On	On	On	On	On
cdmaOne	BS cellular	On	On	On	On	On
	BS pcs	On	On	On	On	On
	MS cellular	On	On	On	On	On
	MS pcs	On	On	On	On	On
cdma2000		On	On	Off	Off	Off
W-CDMA (3GPP)		On	On	Off	Off	Off

Remarks: You must be in Basic, cdmaOne, cdma2000, or W-CDMA (3GPP) mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Sweep Time

[:SENSe]:ACP:OFFSet:LIST:SWEep:TIME <seconds>,<seconds>,<seconds>,<seconds>,<seconds>

[:SENSe]:ACP:OFFSet:LIST:SWEep:TIME?

Selects a specific sweep time. If you increase the sweep time, you increase the length of the time data captured and the number of points measured. You might need to specify a specific sweep speed to accommodate a specific condition in your transmitter. For example, you may have a burst signal and need to measure an exact portion of the burst.

Selecting a specific sweep time may result in a long measurement time since the resulting number of data points my not be the optimum 2^n. Use `[:SENSe]:ACP:SWEep:TIME` to set the number of points used for measuring the reference channel.

You can turn off (not use) specific offsets with the SENS:ACP:OFFSet:LIST:STATe command.

Factory Preset

and *RST:

Mode	Offset A	Offset B	Offset C	Offset D	Offset E
Basic & cdmaOne	11.20 ms	11.20 ms	11.20 ms	11.20 ms	11.20 ms

Range: 1 µs to 50 ms

Default Unit: seconds

Remarks: You must be in Basic, cdmaOne mode to use this command. Use INSTrument:SELect to set the mode.

History: Revision A.03.00 or later, in cdmaOne revision A.04.00

Adjacent Channel Power—Automatic Sweep Time

[:SENSe]:ACP:OFFSet:LIST:SWEep:TIME:AUTO OFF|ON|0|1, OFF|ON|0|1, OFF|ON|0|1, OFF|ON|0|1, OFF|ON|0|1

[:SENSe]:ACP:OFFSet:LIST:SWEep:TIME:AUTO?

Sets the sweep time to be automatically coupled for the fastest measurement time. You can turn off (not use) specific offsets with the SENS:ACP:OFFSet:LIST:STATe command.

Factory Preset
and *RST:

Mode	Offset A	Offset B	Offset C	Offset D	Offset E
Basic & cdmaOne	On	On	On	On	On

Remarks: You must be in Basic, cdmaOne mode to use this command. Use INSTrument:SELect to set the mode.

History: Revision A.03.00 or later, in cdmaOne revision A.04.00

Adjacent Channel Power—Define Type of Offset Frequency List

iDEN mode

[:SENSe]:ACP:OFFSet:TEST ABSolute|AND|OR|RELative

[:SENSe]:ACP:OFFSet:TEST?

Basic mode, cdmaOne

[:SENSe]:ACP:OFFSet:LIST:TEST ABSolute|AND|OR|RELative, ABSolute|AND|OR|RELative, ABSolute|AND|OR|RELative,

ABSolute|AND|OR|RELative, ABSolute|AND|OR|RELative

[:SENSe]:ACP:OFFSet:LIST:TEST?

cdma2000, W-CDMA (3GPP) mode

[:SENSe]:ACP:OFFSet[n]:LIST:TEST ABSolute|AND|OR|RELative, ABSolute|AND|OR|RELative, ABSolute|AND|OR|RELative, ABSolute|AND|OR|RELative, ABSolute|AND|OR|RELative, ABSolute|AND|OR|RELative

[:SENSe]:ACP:OFFSet[n]:LIST:TEST?

cdmaOne mode

[:SENSe]:ACP:OFFSet[n]:LIST[n]:TEST BSolute|AND|OR|RELative, ABSolute|AND|OR|RELative, ABSolute|AND|OR|RELative, ABSolute|AND|OR|RELative, ABSolute|AND|OR|RELative

[:SENSe]:ACP:OFFSet[n]:LIST[n]:TEST?

Defines the type of testing to be done at any custom offset frequencies. The measured powers are tested against the absolute values defined with [:SENSe]:ACP:OFFSet[n]:LIST[n]:ABSolute, or the relative values defined with [:SENSe]:ACP:OFFSet[n]:LIST[n]:RPSDensity and [:SENSe]:ACP:OFFSet[n]:LIST[n]:RCARrier.

You can turn off (not use) specific offsets with the [:SENS]:ACP:OFFSet[n]:LIST[n]:STATe command.

Offset[n] n=1 is base station and 2 is mobiles. The default is base station (1).

List[n]

cdmaOne mode n=1 is cellular bands and 2 is pcs bands. The default is cellular.

The types of testing that can be done for each offset include:

- Absolute - Test the absolute power measurement. If it fails, then return a failure for the measurement at this offset.
- And - Test both the absolute power measurement and the power relative to the carrier. If they both fail, then return a failure for the measurement at this offset.
- Or - Test both the absolute power measurement and the power relative to the carrier. If either one fails, then return a failure for the measurement at this offset.
- Relative - Test the power relative to the carrier. If it fails, then return a failure for the measurement at this offset.
- OFF - Turns the power test off.

Factory Preset

and *RST:

Mode	Variant	Offset A	Offset B	Offset C	Offset D	Offset E
iDEN		REL	n/a	n/a	n/a	n/a
Basic		REL	REL	REL	REL	REL
cdmaOne	BS cellular	REL	REL	REL	REL	REL
	BS pcs	REL	ABS	ABS	REL	REL
	MS cellular	REL	REL	REL	REL	REL
	MS pcs	REL	ABS	ABS	REL	REL
cdma2000		REL	REL	REL	REL	REL
W-CDMA (3GPP)		REL	REL	REL	REL	REL

Remarks: You must be in Basic, cdmaOne, cdma2000, W-CDMA (3GPP), or iDEN mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Number of Measured Points

`[:SENSe]:ACP:POINts <integer>`

`[:SENSe]:ACP:POINts?`

Selects the number of data points used to measure the reference (carrier) channel. The automatic mode chooses the optimum number of points for the fastest measurement time with acceptable repeatability. The minimum number of points that could be used is determined by the sweep time and the sampling rate.

You can increase the length of the measured time record (capture more of the burst) by increasing the number of points, but the measurement will take longer. Use `[:SENSe]:ACP:OFFSet:LIST:POINts` to set the number of points used for measuring the offset channels.

Factory Preset and *RST: 1024

Remarks: The fastest measurement times are obtained when the number of points measured is 2^n.

You must be in Basic, cdmaOne mode to use this command. Use INSTrument:SELect to set the mode.

Range: 64 to 65536

Adjacent Channel Power—Automatic Measurement Points

[:SENSe]:ACP:POINts:AUTO OFF|ON|0|1

[:SENSe]:ACP:POINts:AUTO?

Automatically selects the number of points for the optimum measurement speed.

Factory Preset and *RST: ON

Remarks: You must be in Basic, cdmaOne mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Spectrum Trace Control

[:SENSe]:ACP:SPECtrum:ENABle OFF|ON|0|1

[:SENSe]:ACP:SPECtrum:ENABle?

Turns on/off the measurement of the spectrum trace data when the spectrum view is selected. (Select the view with DISPlay:ACP:VIEW.) You may want to disable the spectrum trace data part of the measurement so you can increase the speed of the rest of the measurement data.

Factory Preset and *RST: ON

Remarks: You must be in Basic, cdmaOne, iDEN mode to use this command. Use INSTrument:SELect to set the mode.

History: Revision A.03.27 or later, in cdmaOne revision A.04.00

Adjacent Channel Power—Sweep Mode Resolution Bandwidth

[:SENSe]:ACP:SWEep:BANDwidth|BWIDth[:RESolution] <freq>

[:SENSe]:ACP:SWEep:BANDwidth|BWIDth[:RESolution]?

Sets the resolution bandwidth when using the spectrum analyzer type sweep mode. See [:SENSe]:ACP:SWEep:TYPE.

Factory Preset and *RST: Auto coupled.

Range: 1.0 kHz to 1.0 MHz

Resolution: 1.0 kHz

Step Size: 1.0 kHz

Default Unit: Hz

Remarks: You must be in the cdmaOne cdma2000, or W-CDMA (3GPP) mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Sweep Mode Resolution BW Control

[:SENSe]:ACP:SWEep:BANDwidth|BWIDth[:RESolution]:AUTO OFF|ON|0|1

[:SENSe]:ACP:SWEep:BANDwidth|BWIDth[:RESolution]:AUTO?

Sets the resolution bandwidth to automatic, when using the spectrum analyzer type sweep mode. See `[:SENSe]:ACP:SWEep:TYPE`.

Factory Preset and *RST: ON

Remarks: You must be in the cdmaOne cdma2000, or W-CDMA (3GPP) mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Sweep Mode Detection

[:SENSe]:ACP:SWEep:DETector[:FUNCtion] AAVerage|POSitive

[:SENSe]:ACP:SWEep:DETector[:FUNCtion]?

Selects the detector type when using the sweep mode. See `[:SENSe]:ACP:SWEep:TYPE`.

Absolute average (AAVerage) - the absolute average power in each frequency is measured across the spectrum

Positive - the positive peak power in each frequency is measured across the spectrum

Factory Preset and *RST: POSitive

Remarks: You must be in the cdma2000, or W-CDMA (3GPP) mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Sweep Time

[:SENSe]:ACP:SWEep:TIME <seconds>

[:SENSe]:ACP:SWEep:TIME?

Selects a specific sweep time used to measure the reference (carrier) channel. If you increase the sweep time, you increase the length of the

time data captured and the number of points measured. You might need to specify a specific sweep speed to accommodate a specific condition in your transmitter. For example, you may have a burst signal and need to measure an exact portion of the burst.

Selecting a specific sweep time may result in a long measurement time since the resulting number of data points my not be the optimum 2^n. Use `[:SENSe]:ACP:OFFSet:LIST:SWEep:TIME` to set the number of points used for measuring the offset channels for Basic and cdmaOne.

For cdma2000 and W-CDMA, this command sets the sweep time when using the sweep mode. See `[:SENSe]:ACP:SWEep:TYPE`.

Factory Preset and *RST: 625 µs (1 slot) for W-CDMA (3GPP)

1.25 ms for cdma2000

11.20 ms for Basic, cdmaOne

Range: 500 µs to 10 ms

1 µs to 50 ms for Basic, cdmaOne

Default Unit: seconds

Remarks: You must be in the Basic, cdmaOne, cdma2000, or W-CDMA (3GPP) mode to use this command. Use INSTrument:SELect to set the mode.

History: Added to Basic revision A.03.00, to cdmaOne revision A.04.00

Adjacent Channel Power—Automatic Sweep Time

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

`[:SENSe]:ACP:SWEep:TIME:AUTO?`

Sets the sweep time to be automatically coupled for the fastest measurement time.

Factory Preset and *RST: ON

Remarks: You must be in Basic, cdmaOne mode to use this command. Use INSTrument:SELect to set the mode.

History: Revision A.03.00 or later, in cdmaOne revision A.04.00

Adjacent Channel Power—Sweep Type

W-CDMA (3GPP) mode

[:SENSe]:ACP:SWEep:TYPE FAST|FFT|SWEep

[:SENSe]:ACP:SWEep:TYPE?

cdma2000 mode

[:SENSe]:ACP:SWEep:TYPE FFT|SWEep

[:SENSe]:ACP:SWEep:TYPE?

Selects the type of sweeping.

Fast *(W-CDMA (3GPP) mode only)* - the data acquisition is made with the wide channel integration bandwidth and the time-domain data is divided into the narrow data to apply FFT. This mode is faster than the FFT mode but less accurate in power levels.

FFT - the data acquisition is made with the narrow channel integration bandwidth and apply fast Fourier transform (FFT) to convert to the frequency domain data.

Sweep - the measurement is made by the swept spectrum method like the traditional swept frequency spectrum analysis to have better correlation to the input signal with a high crest factor (peak/average ratio). This mode may take a longer time than the FFT mode. See `[:SENSe]:ACP:SWEep:DETector[:FUNCtion]`.

Factory Preset and *RST: FFT

Remarks: You must be in the cdma2000, or W-CDMA (3GPP) mode to use this command. Use INSTrument:SELect to set the mode.

Adjacent Channel Power—Trigger Source

[:SENSe]:ACP:TRIGger:SOURce
EXTernal[1]|EXTernal2|FRAMe|IF|IMMediate|RFBurst

[:SENSe]:ACP:TRIGger:SOURce?

Select the trigger source used to control the data acquisitions.

EXTernal 1 – front panel external trigger input

EXTernal 2 – rear panel external trigger input

FRAMe – internal frame trigger from front panel input

IF – internal IF envelope (video) trigger

IMMediate – the next data acquisition is immediately taken, capturing the signal asynchronously (also called free run).

RFBurst – wideband RF burst envelope trigger that has automatic level control for periodic burst signals.

Factory Preset and *RST: IMMediate for BS

RFBurst for MS

Remarks: You must be in Basic, cdmaOne, iDEN, NADC, or PDC mode to use this command. Use INSTrument:SELect to set the mode.

In Basic mode, for offset frequencies >12.5 MHz, the external triggers will be a more reliable trigger source than RF burst. Also, you can use the Waveform measurement to set up trigger delay.

Adjacent Channel Power—Power Reference

[:SENSe]:ACP:TYPE PSDRef|TPRef

[:SENSe]:ACP:TYPE?

Selects the measurement type. This allows you to make absolute and relative power measurements of either total power or the power normalized to the measurement bandwidth.

Power Spectral Density Reference (PSDRef) - the power spectral density is used as the power reference

Total Power Reference (TPRef) - the total power is used as the power reference

Factory Preset and *RST: Total power reference (TPRef)

Remarks: You must be in the Basic, cdmaOne, cdma2000, W-CDMA (3GPP), NADC, or PDC mode to use this command. Use INSTrument:SELect to set the mode.

BbIQ Commands

Select I/Q Power Range

[:SENSe]:POWer:IQ:RANGe[:UPPer]<Float 64>{DBM|DBMV|W

[:SENSe]:POWer:IQ:RANGe[:UPPer]?

Selects maximum total power expected from unit under test at test port when I or Q port is selected.

Range: For 50 Ohms:

13.0, 7.0. 1.0, -5.1 [DBM]
60.0, 54.0, 48.0, 41.9 [DBMV]

.02, .005, .0013, .00031 [W]

For 600 Ohms:

2.2, -3.8. -9.8, -15.8 [DBM]
60.0, 54.0, 48.0, 41.9 [DBMV]
.0017, .00042, .0001, .000026 [W]

Values for 1 M Ohm vary according to selected reference impedance.

Remarks: Implemented for BASIC and W-CDMA modes.

History: Version A.05.00 or later

Select I/Q Voltage Range

`[:SENSe]:VOLTage:IQ:RANGe[:UPPer]<Float 64> [V]`

`[:SENSe]:VOLTage:IQ:RANGe[:UPPer]?`

Selects upper voltage range when I or Q port is selected. This setting helps set the gain which is generated in the variable gain block of the BbIQ board to improve dynamic range.

Range: 1.0, 0.5, 0.025, 0.125[V]

Remarks: Implemented for BASIC and W-CDMA modes.

History: Version A.05.00 or later

Channel Commands

Select the ARFCN—Absolute RF Channel Number

`[:SENSe]:CHANnel:ARFCn|RFCHannel <integer>`

`[:SENSe]:CHANnel:ARFCn|RFCHannel?`

Set the analyzer to a frequency that corresponds to the ARFCN (Absolute RF Channel Number).

Factory Preset and *RST: 38

Range: 0 to 124, and 975 to 1023 for E-GSM

1 to 124 for P-GSM

0 to 124, and 955 to 1023 for R-GSM

512 to 885 for DCS1800

512 to 810 for PCS1900

259 to 293 for GSM450

306 to 340 for GSM480

438 to 511 for GSM700

128 to 251 for GSM850

Remarks: You must be in the EDGE(w/GSM), GSM mode to use this command. Use INSTrument:SELect to set the mode.

Global to the current mode.

History: Version A.03.00 or later

Front Panel Access: **FREQUENCY Channel, ARFCN**

Select the Lowest ARFCN

[:SENSe]:CHANnel:ARFCn|RFCHannel:BOTTom

Set the analyzer to the frequency of the lowest ARFCN (Absolute RF Channel Number) of the selected radio band.

Factory Preset and *RST: 975 for E-GSM

1 for P-GSM

955 for R-GSM

512 for DCS1800

512 PCS1900

259 GSM450

306 GSM480

438 GSM700

128 GSM850

Remarks: You must be in the EDGE(w/GSM), GSM mode to use this command. Use INSTrument:SELect to set the mode.

Global to the current mode.

History: Version A.03.00 or later

Front Panel Access: **FREQUENCY Channel, BMT Freq**

Select the Middle ARFCN

[:SENSe]:CHANnel:ARFCn|RFCHannel:MIDDle

Set the analyzer to the frequency of the middle ARFCN (Absolute RF Channel Number) of the selected radio band.

Factory Preset and *RST:
38 for E-GSM
63 for P-GSM
28 for R-GSM
699 for DCS1800
661 for PCS1900
276 for GSM450
323 for GSM480
474 for GSM 700
189 for GSM850

Remarks: You must be in the EDGE(w/GSM), GSM mode to use this command. Use INSTrument:SELect to set the mode.

Global to the current mode.

History: Version A.03.00 or later

Front Panel Access: **FREQUENCY Channel, BMT Freq**

Select the Highest ARFCN

`[:SENSe]:CHANnel:ARFCn|RFCHannel:TOP`

Set the analyzer to the frequency of the highest ARFCN (Absolute RF Channel Number) of the selected radio band.

Factory Preset and *RST:
124 for E-GSM
124 for P-GSM
124 for R-GSM
885 for DCS1800
810 for PCS1900
293 for GSM450
340 for GSM480
511 for GSM700
251 for GSM850

Remarks: You must be in the EDGE(w/GSM), GSM mode to use this command. Use INSTrument:SELect to set the mode.

Global to the current mode.

History: Version A.03.00 or later

Front Panel Access: **FREQUENCY Channel, BMT Freq**

Burst Type

[:SENSe]:CHANnel:BURSt TCH|CCH

[:SENSe]:CHANnel:BURSt?

Set the burst type for mobile station testing.

Traffic Channel (TCH) – burst for traffic channel

Control Channel (CCH) – burst for control channel

Factory Preset and *RST: TCH

Remarks: The command is only applicable for mobile station testing, device = MS.

You must be in the NADC or PDC mode to use this command. Use INSTrument:SELect to set the mode.

Channel Burst Type

[:SENSe]:CHANnel:BURSt NORMal|SYNC|ACCess

[:SENSe]:CHANnel:BURSt?

Set the burst type that the analyzer will search for and to which it will sync. This only applies with normal burst selected.

NORMal: Traffic Channel (TCH) and Control Channel (CCH)

SYNC: Synchronization Channel (SCH)

ACCess: Random Access Channel (RACH)

Remarks: Global to the current mode.

You must be in the EDGE(w/GSM), GSM mode to use this command. Use INSTrument:SELect to set the mode.

Front Panel Access: **FREQUENCY Channel, Burst Type**

Digital Demod PN Offset

[:SENSe]:CHANnel:PNOFfset <integer>

[:SENSe]:CHANnel:PNOFfset?

Set the PN offset number for the base station being tested.

Factory Preset and *RST: 0

Range: 0 to 511

Default Unit: None

Remarks: Global to the current mode.

You must be in the cdmaOne mode to use this command. Use INSTrument:SELect to set the mode.

Front Panel Access: **FREQUENCY Channel**, **PN Offset**

or

Mode Setup, **Demod**, **PN Offset**

Time Slot number

[:SENSe]:CHANnel:SLOT <integer>

[:SENSe]:CHANnel:SLOT?

Select the slot number that you want to measure.

In GSM mode the measurement frame is divided into the eight expected measurement timeslots.

Factory Preset and *RST: 0 for GSM, PDC mode

1 for NADC mode

Range: 0 to 5 for PDC mode

1 to 6 for NADC mode

0 to 7 for GSM mode

Remarks: You must be in EDGE(w/GSM), GSM, NADC, PDC mode to use this command. Use INSTrument:SELect to set the mode.

Front Panel Access: **Mode Setup, Radio, Frequency Hopping Repetition Factor**

Time Slot Auto

[:SENSe]:CHANnel:SLOT:AUTO OFF|ON|0|1

[:SENSe]:CHANnel:SLOT:AUTO?

Select auto or manual control for slot searching. The feature is only supported in external and frame trigger source modes. In external trigger mode when timeslot is set on, the demodulation measurement is made on the nth timeslot specified by the external trigger point + n timeslots, where n is the selected timeslot value 0 to 7. In frame trigger mode when timeslot is set on, then demodulation measurement is only made on the nth timeslot specified by bit 0 of frame reference burst + n timeslots, where n is the selected timeslot value 0 to 7 and where the frame reference burst is specified by Ref Burst and Ref TSC (Std) combination.

Factory Preset and *RST: ON, for NADC, PDC mode

OFF, for GSM mode

Remarks: The command is only applicable for mobile station testing, device = MS.

You must be in EDGE(w/GSM), GSM, NADC, PDC mode to use this command. Use INSTrument:SELect to set the mode.

History: Added GSM mode, version A.03.00 or later

Training Sequence Code (TSC)

[:SENSe]:CHANnel:TSCode <integer>

[:SENSe]:CHANnel:TSCode?

Set the training sequence code to search for, with normal burst selected and TSC auto set to off.

Factory Preset and *RST: 0

Range: 0 to 7

Remarks: Global to the current mode.

You must be in the EDGE(w/GSM), GSM mode to use this command. Use INSTrument:SELect to set the mode.

History: Version A.03.00 or later

Front Panel Access: **FREQUENCY Channel, TSC (Std)**

Training Sequence Code (TSC) Auto

`[:SENSe]:CHANnel:TSCode:AUTO OFF|ON|0|1`

`[:SENSe]:CHANnel:TSCode:AUTO?`

Select auto or manual control for training sequence code (TSC) search. With auto on, the measurement is made on the first burst found to have one of the valid TSCs in the range 0 to 7 (i.e. normal bursts only). With auto off, the measurement is made on the 1st burst found to have the selected TSC.

Factory Preset and *RST: AUTO

Remarks: Global to the current mode.

You must be in the EDGE(w/GSM), GSM mode to use this command. Use INSTrument:SELect to set the mode.

Front Panel Access: **FREQUENCY Channel, TSC (Std)**

Channel Power Measurement

Commands for querying the channel power measurement results and for setting to the default values are found in "MEASure Group of Commands" on page 301. The equivalent front-panel keys for the parameters described in the following commands, are found under the **Meas Setup** key, after the **Channel Power** measurement has been selected from the **MEASURE** key menu. CHPower used instead of the more std-compliant CPOWer, as that syntax was already used for Carrier Power measurement (but has since been renamed).

Channel Power—Average Count

`[:SENSe]:CHPower:AVERage:COUNt <integer>`

`[:SENSe]:CHPower:AVERage:COUNt?`

Set the number of data acquisitions that will be averaged. After the specified number of average counts, the averaging mode (terminal control) setting determines the averaging action.

Factory Preset and *RST: 20

200, for W-CDMA

Range: 1 to 10,000

Remarks: You must be in the cdmaOne, cdma2000, W-CDMA (3GPP), or Basic mode to use this command. Use INSTrument:SELect to set the mode.

Channel Power—Averaging State

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

`[:SENSe]:CHPower:AVERage[:STATe]?`

Turn averaging on or off.

Factory Preset and *RST: ON

Remarks: You must be in the cdmaOne, cdma2000, W-CDMA (3GPP), or Basic mode to use this command. Use INSTrument:SELect to set the mode.

Channel Power—Averaging Termination Control

`[:SENSe]:CHPower:AVERage:TCONtrol EXPonential|REPeat`

`[:SENSe]:CHPower:AVERage:TCONtrol?`

Select the type of termination control used for the averaging function. This determines the averaging action after the specified number of data acquisitions (average count) is reached.

EXPonential - Each successive data acquisition after the average count is reached, is exponentially weighted and combined with the existing average.

REPeat - After reaching the average count, the averaging is reset and a new average is started.

Factory Preset and *RST: REPeat

Remarks: You must be in the cdmaOne, cdma2000, W-CDMA (3GPP), or Basic mode to use this command. Use INSTrument:SELect to set the mode.

Channel Power—Integration BW

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

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

Set the Integration BW (IBW) that will be used.

Factory Preset and *RST: 1.23 MHz for Basic, cdmaOne, cdma2000

5.0 MHz for W-CDMA (3GPP)

Range: 1 kHz to 10 MHz

Default Unit: Hz

Remarks: You must be in the cdmaOne, cdma2000, W-CDMA (3GPP), or Basic mode to use this command. Use INSTrument:SELect to set the mode.

Channel Power—Span

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

[:SENSe]:CHPower:FREQuency:SPAN?

Set the frequency span that will be used.

Factory Preset and *RST: 2.0 MHz for Basic, cdmaOne, cdma2000

6.0 MHz for W-CDMA (3GPP)

Range: Dependent on the current setting of the channel power integration bandwidth

Default Unit: Hz

Remarks: You must be in the cdmaOne, cdma2000, W-CDMA (3GPP), or Basic mode to use this command. Use INSTrument:SELect to set the mode.

Channel Power—Data Points

[:SENSe]:CHPower:POINts <integer>

[:SENSe]:CHPower:POINts?

Set the number of data points that will be used. Changing this will change the time record length and resolution BW that are used.

Factory Preset and *RST: 512

Range: 64 to 32768, in a 2^n sequence

Remarks: You must be in the cdmaOne, cdma2000, W-CDMA (3GPP), or Basic mode to use this command. Use INSTrument:SELect to set the mode.

Channel Power—Data Points Auto

[:SENSe]:CHPower:POINts:AUTO OFF|ON|0|1

[:SENSe]:CHPower:POINts:AUTO?

Select auto or manual control of the data points. This is an advanced control that normally does not need to be changed. Setting this to a value other than the factory default, may cause invalid measurement results.

OFF - the Data Points is uncoupled from the Integration BW.

ON - couples the Data Points to the Integration BW.

Factory Preset and *RST: ON

Remarks: You must be in the cdmaOne, cdma2000, W-CDMA (3GPP), or Basic mode to use this command. Use INSTrument:SELect to set the mode.

Channel Power—Sweep Time

[:SENSe]:CHPower:SWEep:TIME <time>

[:SENSe]:CHPower:SWEep:TIME?

Sets the sweep time when using the sweep mode.

Factory Preset and *RST: 68.27 µs

17.07 µs for W-CDMA (3GPP)

Range: 1 µs to 50 ms

Default Unit: seconds

Remarks: You must be in Basic, cdmaOne, cdma2000, or W-CDMA (3GPP) mode to use this command. Use INSTrument:SELect to set the mode.

History: Version A.03.00 and later

Channel Power—Sweep Time

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

[:SENSe]:CHPower:SWEep:TIME:AUTO?

Selects the automatic sweep time, optimizing the measurement.

Factory Preset and *RST: ON

Remarks: You must be in Basic, cdmaOne, cdma2000, or W-CDMA (3GPP) mode to use this command. Use INSTrument:SELect to set the mode.

History: Version A.03.00 and later

Channel Power—Trigger Source

[:SENSe]:CHPower:TRIGger:SOURce
EXTernal[1]|EXTernal2|IMMediate

[:SENSe]:CHPower:TRIGger:SOURce?

Select the trigger source used to control the data acquisitions. This is an Advanced control that normally does not need to be changed.

EXTernal 1 - front panel external trigger input

EXTernal 2 - rear panel external trigger input

IMMediate - the next data acquisition is immediately taken (also called Free Run).

Factory Preset and *RST: IMMediate

Remarks: You must be in the cdmaOne, cdma2000, W-CDMA (3GPP), or Basic mode to use this command. Use INSTrument:SELect to set the mode.

Signal Corrections Commands

Correction for RF Port External Attenuation

[:SENSe]:CORRection[:RF]:LOSS <rel_power>

[:SENSe]:CORRection[:RF]:LOSS?

Set the correction equal to the external attenuation used when measuring the device under test.

Factory Preset and *RST: 0 dB

Range: -50 to +50 dB

Default Unit: dB

Remarks: You must be in the Basic mode to use this command. Use INSTrument:SELect to set the mode.

Value is global to Basic mode.

Front Panel Access: **Input, Ext Atten**

Select the Input Signal

[:SENSe]:FEED RF|IQ|IONLy|QONLy|AREFerence|IFALign

[:SENSe]:FEED?

Selects the input signal. The default input signal is taken from the front panel RF input port. For calibration and testing purposes the input signal can be taken from an internal 321.4 MHz IF alignment signal or an internal 50 MHz amplitude reference source.

If the baseband IQ option (Option B7C) is installed, I and Q input ports are added to the front panel. The I and Q ports accept the in-phase and quadrature components of the IQ signal, respectively. The input signal can be taken from either or both ports.

RF selects the signal from the front panel RF INPUT port.

IQ selects the combined signals from the front panel optional I and Q input ports.

IONLy selects the signal from the front panel optional I input port.

QONLy selects the signal from the front panel optional Q input port.

AREFerence selects the internal 50 MHz amplitude reference signal.

IFALign selects the internal, 321.4 MHz, IF alignment signal.

Factory Preset and *RST: RF

Front Panel Access: **Input, Input Port**

History: VSA modified in A.05.00 version

Select the Input Signal

[:SENSe]:FEED RF|IQ|IONLy|QONLy|AREFerence|IFALign

[:SENSe]:FEED?

Selects the input signal. The default input signal is taken from the front panel RF input port. For calibration and testing purposes the input signal can be taken from an internal 321.4 MHz IF alignment signal or an internal 50 MHz amplitude reference source.

If the baseband IQ option (Option B7C) is installed, I and Q input ports are added to the front panel. The I and Q ports accept the in-phase and quadrature components of the IQ signal, respectively. The input signal can be taken from either or both ports.

RF selects the signal from the front panel RF INPUT port.

IQ selects the combined signals from the front panel optional I and Q input ports.

IONLy selects the signal from the front panel optional I input port.

QONLy selects the signal from the front panel optional Q input port.

IFALign selects the internal, 321.4 MHz, IF alignment signal.

AREFerence selects the internal 50 MHz amplitude reference signal.

Factory Preset and *RST: RF

Front Panel Access: **Input, Input Port**

History: VSA modified in A.05.00 version

Frequency Commands

Center Frequency

[:SENSe]:FREQuency:CENTer <freq>

[:SENSe]:FREQuency:CENTer?

Set the center frequency.

Factory Preset and *RST: 1.0 GHz

942.6 MHz for GSM, EDGE

806.0 MHz for iDEN

Range: 1.0 kHz to 4.3214 GHz

Default Unit: Hz

Front Panel Access: **FREQUENCY/Channel, Center Freq**

Center Frequency Step Size

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

`[:SENSe]:FREQuency:CENTer:STEP[:INCRement]?`

Specifies the center frequency step size.

Factory Preset and *RST:

5.0 MHz

1.25 MHz for cdma2000

Range: 1.0 kHz to 1.0 GHz, in 10 kHz steps

Default Unit: Hz

History: Version A.03.00 or later

Front Panel Access: **FREQUENCY/Channel, CF Stepl**

RF Power Commands

RF Port Input Attenuation

[:SENSe]:POWer[:RF]:ATTenuation <rel_power>

[:SENSe]:POWer[:RF]:ATTenuation?

Set the RF input attenuator. This value is set at its auto value if input attenuation is set to auto.

Factory Preset and *RST: 0 dB

12 dB for iDEN

Range: 0 to 40 dB

Default Unit: dB

Front Panel Access: **Input, Input Atten**

RF Port Input Attenuator Auto

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

[:SENSe]:POWer[:RF]:ATTenuation:AUTO?

Select the RF input attenuator range to be set either automatically or manually.

ON - Input attenuation is automatically set as determined by the reference level setting.

OFF - Input attenuation is manually set.

Front Panel Access: **Input/Output (or Input), Input Atten**

RF Port Power Range Auto

[:SENSe]:POWer[:RF]:RANGe:AUTO OFF|ON|0|1

[:SENSe]:POWer[:RF]:RANGe:AUTO?

Select the RF port power range to be set either automatically or manually.

ON - power range is automatically set as determined by the actual measured power level at the start of a measurement.

OFF - power range is manually set

Factory Preset and *RST: ON

Remarks: You must be in the cdmaOne, EDGE(w/GSM), GSM, NADC, PDC, cdma2000, or W-CDMA (3GPP) mode to use this command. Use INSTrument:SELect to set the mode.

Front Panel Access: **Input, Max Total Pwr (at UUT)**

RF Port Power Range Maximum Total Power

`[:SENSe]:POWer[:RF]:RANGe[:UPPer] <power>`

`[:SENSe]:POWer[:RF]:RANGe[:UPPer]?`

Set the maximum expected total power level at the radio unit under test. This value is ignored if RF port power range is set to auto. External attenuation required above 30 dBm.

Factory Preset and *RST: –15.0 dBm

Range: –100.0 to 80.0 dBm for EDGE, GSM

–100.0 to 27.7 dBm for cdmaOne, iDEN

–200.0 to 50.0 dBm for NADC, PDC

–200.0 to 100.0 dBm for cdma2000, W-CDMA (3GPP)

Default Unit: dBm

Remarks: Global to the current mode. This is coupled to the RF input attenuation

You must be in the Service, cdmaOne, EDGE(w/GSM), GSM, NADC, PDC, cdma2000, or W-CDMA (3GPP) mode to use this command. Use INSTrument:SELect to set the mode.

Front Panel Access: **Input, Max Total Pwr (at UUT)**

Power Statistics CCDF Measurement

Commands for querying the statistical power measurement of the complimentary cumulative distribution function (CCDF) measurement results and for setting to the default values are found in "MEASure Group of Commands" on page 301. The equivalent front-panel keys for the parameters described in the following commands, are found under the **Meas Setup** key, after the **Power Stat CCDF** measurement has been selected from the **MEASURE** key menu.

History: Added PSTatistic to Basic Mode version A.04.00

Power Statistics CCDF—Channel Bandwidth

[:SENSe]:PSTatistic:BANDwidth|BWIDth <freq>

[:SENSe]:PSTatistic:BANDwidth|BWIDth?

Set the bandwidth that will be used for acquiring the signal.

Factory Preset and *RST: 5.0 MHz

Range: 10.0 kHz to 6.7 MHz

Resolution: 0.1 kHz

Step: 1.0 kHz

Default Unit: Hz

Remarks: You must be in the Basic, cdma2000, or W-CDMA (3GPP) mode to use this command. Use INSTrument:SELect to set the mode.

Power Statistics CCDF—Sample Counts

[:SENSe]:PSTatistic:COUNts <integer>

[:SENSe]:PSTatistic:COUNts?

Set the counts. Measurement stops when the sample counts reach this value.

Factory Preset and *RST: 10,000,000

Range: 1,000 to 2,000,000,000

Unit: counts

Remarks: You must be in the Basic, cdma2000, or W-CDMA (3GPP) mode to use this command. Use INSTrument:SELect to set the mode.

Power Statistics CCDF—Sweep Time

[:SENSe]:PSTatistic:SWEep:TIME <time>

[:SENSe]:PSTatistic:SWEep:TIME?

Set the length of measurement interval that will be used.

Factory Preset and *RST: 1.0 ms

Range: 0.1 ms to 10 ms

Resolution: 0.001 ms

Step: 0.001 ms

Default Unit: seconds

Remarks: You must be in the Basic, cdma2000, or W-CDMA (3GPP) mode to use this command. Use INSTrument:SELect to set the mode.

Power Statistics CCDF—Trigger Source

[:SENSe]:PSTatistic:TRIGger:SOURce
EXTernal[1]|EXTernal2|FRAMe|IF|IMMediate|RFBurst

[:SENSe]:PSTatistic:TRIGger:SOURce?

Set the trigger source used to control the data acquisitions.

EXTernal 1 - front panel external trigger input

EXTernal 2 - rear panel external trigger input

FRAMe - uses the internal frame timer, which has been synchronized to the selected burst sync.

IF - internal IF envelope (video) trigger

IMMediate - the next data acquisition is immediately taken, capturing the signal asynchronously (also called Free Run).

RFBurst - wideband RF burst envelope trigger that has automatic level control for periodic burst signals.

Factory Preset and *RST: IMMediate

Remarks: You must be in the Basic, cdma2000, or W-CDMA (3GPP) mode to use this command. Use INSTrument:SELect to set the mode.

Power vs. Time Measurement

Commands for querying the power versus time measurement results and for setting to the default values are found in "MEASure Group of Commands" on page 301. The equivalent front-panel keys for the parameters described in the following commands, are found under the **Meas Setup** key, after the **Power vs Time** measurement has been selected from the **MEASURE** key menu.

Power vs. Time—Number of Bursts Averaged

```
[:SENSe]:PVTime:AVERage:COUNt <integer>
```

```
[:SENSe]:PVTime:AVERage:COUNt?
```

Set the number of bursts that will be averaged. After the specified number of bursts (average counts), the averaging mode (terminal control) setting determines the averaging action.

Factory Preset and *RST: 15

Range: 1 to 10,000

Remarks: You must be in the EDGE(w/GSM), GSM or Service mode to use this command. Use INSTrument:SELect to set the mode.

Power vs. Time—Averaging State

```
[:SENSe]:PVTime:AVERage[:STATe] OFF|ON|0|1
```

```
[:SENSe]:PVTime:AVERage[:STATe]?
```

Turn averaging on or off.

Factory Preset and *RST: OFF

Remarks: You must be in the EDGE(w/GSM), GSM or Service mode to use this command. Use INSTrument:SELect to set the mode.

Power vs. Time—Averaging Mode

```
[:SENSe]:PVTime:AVERage:TCONtrol EXPonential|REPeat
```

```
[:SENSe]:PVTime:AVERage:TCONtrol?
```

Select the type of termination control used for the averaging function. This specifies the averaging action after the specified number of bursts

(average count) is reached.

EXPonential - Each successive data acquisition after the average count is reached is exponentially weighted and combined with the existing average.

REPeat - After reaching the average count, the averaging is reset and a new average is started.

Factory Preset and *RST: EXPonential

Remarks: You must be in the EDGE(w/GSM), GSM or Service mode to use this command. Use INSTrument:SELect to set the mode.

Power vs. Time—Averaging Type

[:SENSe]:PVTime:AVERage:TYPE LOG|MAXimum|MINimum|MXMinimum|RMS

[:SENSe]:PVTime:AVERage:TYPE?

Select the type of averaging to be performed.

LOG - The log of the power is averaged. (This is also known as video averaging.)

MAXimum - The maximum values are retained.

MINimum - The minimum values are retained.

MXMinimum - Both the maximum and the minimum values are retained.

RMS - The power is averaged, providing the rms of the voltage.

Factory Preset and *RST: RMS

Remarks: You must be in the EDGE(w/GSM), GSM or Service mode to use this command. Use INSTrument:SELect to set the mode.

Power vs. Time—Resolution BW

[:SENSe]:PVTime:BANDwidth|BWIDth[:RESolution] <freq>

[:SENSe]:PVTime:BANDwidth|BWIDth[:RESolution]?

Set the resolution BW. This is an advanced control that normally does not need to be changed. Setting this to a value other than the factory default, may cause invalid measurement results.

Factory Preset and *RST: 500 kHz

Range: 1 kHz to 5 MHz

Default Unit: Hz

Remarks: You must be in the EDGE(w/GSM), GSM or Service mode to use this command. Use INSTrument:SELect to set the mode.

Power vs. Time—RBW Filter Type

[:SENSe]:PVTime:BANDwidth|BWIDth[:RESolution]:TYPE FLATtop|GAUSsian

[:SENSe]:PVTime:BANDwidth|BWIDth[:RESolution]:TYPE?

Select the type of resolution BW filter. This is an advanced control that normally does not need to be changed. Setting this to a value other than the factory default, may cause invalid measurement results.

FLATtop - a filter with a flat amplitude response, which provides the best amplitude accuracy.

GAUSsian - a filter with Gaussian characteristics, which provides the best pulse response.

Factory Preset and *RST: GAUSsian

Remarks: You must be in the EDGE(w/GSM), GSM or Service mode to use this command. Use INSTrument:SELect to set the mode.

Power vs. Time—Sweep Time

[:SENSe]:PVTime:SWEep:TIME <integer>

[:SENSe]:PVTime:SWEep:TIME?

Set the number of slots which are used in each data acquisition. Each slot is approximately equal to 570 ms. The measurement is made for a small additional amount of time (about 130 µs) in order to view the burst edges.

Factory Preset and *RST: 1

Range: 1 to 50 (for resolution BW = 500 kHz)

Remarks: You must be in the EDGE(w/GSM), GSM or Service mode to use this command. Use INSTrument:SELect to

set the mode.

Power vs. Time—Trigger Source

[:SENSe]:PVTime:TRIGger:SOURce EXTernal[1]|EXTernal2|FRAMe|IF|IMMediate|RFBurst

[:SENSe]:PVTime:TRIGger:SOURce?

Select the trigger source used to control the data acquisitions.

EXTernal 1 - front panel external trigger input

EXTernal 2 - rear panel external trigger input

FRAMe - uses the internal frame timer, which has been synchronized to the selected burst sync.

IF - internal IF envelope (video) trigger

IMMediate - the next data acquisition is immediately taken, capturing the signal asynchronously (also called Free Run).

RFBurst - wideband RF burst envelope trigger that has automatic level control for periodic burst signals.

Factory Preset and *RST: RFBurst if the RF Burst Hardware (option B7E) has been installed

EXTernal if option B7E has not been installed

Remarks: You must be in the EDGE(w/GSM), GSM or Service mode to use this command. Use INSTrument:SELect to set the mode.

Reference Oscillator Commands

Reference Oscillator External Frequency

[:SENSe]:ROSCillator:EXTernal:FREQuency <frequency>

[:SENSe]:ROSCillator:EXTernal:FREQuency?

Specify to the frequency of the external reference being supplied to the instrument. Switch to the external reference with ROSC:SOUR.

Preset and *RST: Value remains at last user selected value (persistent)

Factory default, 10 MHz

Range: 1 MHz to 30 MHz, with 1 Hz steps

Default Unit: Hz

Remarks: Global to system

Front Panel Access: **System, Reference, Ref Oscillator**

Reference Oscillator Rear Panel Output

[:SENSe]:ROSCillator:OUTPut[:STATe] OFF|ON|0|1

[:SENSe]:ROSCillator:OUTPut?

Turn on and off the 10 MHz frequency reference signal going to the rear panel.

Preset and *RST: Persistent State with factory default of On

Remarks: Global to system. Was SENS:ROSC:REAR

Front Panel Access: **System, Reference, 10 MHz Out**

Reference Oscillator Source

[:SENSe]:ROSCillator:SOURce INTernal|EXTernal

[:SENSe]:ROSCillator:SOURce?

Select the reference oscillator (time base) source. Use ROSC:EXT:FREQ to tell the instrument the frequency of the external reference.

INTernal - uses internally generated 10 MHz reference signal

EXTernal - uses the signal at the rear panel external reference input port.

Preset and *RST:	Persistent State with factory default of Internal
Remarks:	Global to system.
Front Panel Access:	**System, Reference, Ref Oscillator**

Spectrum (Frequency-Domain) Measurement

Commands for querying the spectrum measurement results and for setting to the default values are found in "MEASure Group of Commands" on page 301. The equivalent front-panel keys for the parameters described in the following commands, are found under the **Meas Setup** key, after the **Spectrum (Freq Domain)** measurement has been selected from the **MEASURE** key menu.

Spectrum—Data Acquisition Packing

```
[:SENSe]:SPECtrum:ACQuisition:PACKing
AUTO|LONG|MEDium|SHORt

[:SENSe]:SPECtrum:ACQuisition:PACKing?
```

Select the amount of data acquisition packing. This is an advanced control that normally does not need to be changed.

Factory Preset and *RST: AUTO

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

Spectrum—ADC Dither

```
[:SENSe]:SPECtrum:ADC:DITHer[:STATe] AUTO|ON|OFF|2|1|0

[:SENSe]:SPECtrum:ADC:DITHer[:STATe]?
```

Turn the ADC dither on or off. This is an advanced control that normally does not need to be changed. The "ADC dither" refers to the introduction of noise to the digitized steps of the analog-to-digital converter; the result is an improvement in amplitude accuracy.

Factory Preset and *RST: AUTO

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

Spectrum—ADC Range

12-bit ADC

```
[:SENSe]:SPECtrum:ADC:RANGe
AUTO|APEak|APLock|M6|P0|P6|P12|P18|P24
```

14-bit ADC

```
[:SENSe]:SPECtrum:ADC:RANGe
AUTO|APEak|APLock|NONE|P0|P6|P12|P18
```

[:SENSe]:SPECtrum:ADC:RANGe?

Select the range for the gain-ranging that is done in front of the ADC. This is an advanced control that normally does not need to be changed. Auto peak ranging is the default for this measurement. If you are measuring a CW signal please see the description below.

- AUTO - automatic range

  For FFT spectrums - auto ranging should not be not be used. An exception to this would be if you know that your signal is “bursty”. Then you might use auto to maximize the time domain dynamic range as long as you are not very interested in the FFT data.

- Auto Peak (APEak) - automatically peak the range

  For CW signals, the default of auto-peak ranging can be used, but a better FFT measurement of the signal can be made by selecting one of the manual ranges that are available: M6, P0 - P24.
  Auto peaking can cause the ADC range gain to move monotonically down during the data capture. This movement should have negligible effect on the FFT spectrum, but selecting a manual range removes this possibility. Note that if the CW signal being measured is close to the auto-ranging threshold, the noise floor may shift as much as 6 dB from sweep to sweep.

- Auto Peak Lock (APLock) - automatically peak lock the range

  For CW signals, auto-peak lock ranging may be used. It will find the best ADC measurement range for this particular signal and will not move the range as auto-peak can. Note that if the CW signal being measured is close to the auto-ranging threshold, the noise floor may shift as much as 6 dB from sweep to sweep.
  For “bursty” signals, auto-peak lock ranging should not be used. The measurement will fail to operate, since the wrong (locked) ADC range will be chosen often and overloads will occur in the ADC.

- NONE - (14-bit ADC E4406A) turns off any auto-ranging without making any changes to the current setting.
- M6 - (12-bit ADC E4406A) manually selects an ADC range that subtracts 6 dB of fixed gain across the range. Manual ranging is best for CW signals.
- P0 to 24 - manually selects ADC ranges that add 0 to 24 dB of fixed gain across the range. Manual ranging is best for CW signals.

Factory Preset and *RST: APEak

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

Spectrum—Average Clear

[:SENSe]:SPECtrum:AVERage:CLEar

The average data is cleared and the average counter is reset.

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

Spectrum—Number of Averages

[:SENSe]:SPECtrum:AVERage:COUNt <integer>

[:SENSe]:SPECtrum:AVERage:COUNt?

Set the number of 'sweeps' that will be averaged. After the specified number of 'sweeps' (average counts), the averaging mode (terminal control) setting determines the averaging action.

Factory Preset and *RST: 25

Range: 1 to 10,000

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

Spectrum—Averaging State

[:SENSe]:SPECtrum:AVERage[:STATe] OFF|ON|0|1

[:SENSe]:SPECtrum:AVERage[:STATe]?

Turn averaging on or off.

Factory Preset and *RST: ON

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

Spectrum—Averaging Mode

[:SENSe]:SPECtrum:AVERage:TCONtrol EXPonential|REPeat

[:SENSe]:SPECtrum:AVERage:TCONtrol?

Select the type of termination control used for the averaging function. This determines the averaging action after the specified number of 'sweeps' (average count) is reached.

EXPonential - Each successive data acquisition after the average count is reached, is exponentially weighted and combined with the existing average.

REPeat - After reaching the average count, the averaging is reset and a new average is started.

Factory Preset and *RST: EXPonential

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

Spectrum—Averaging Type

[:SENSe]:SPECtrum:AVERage:TYPE LOG|MAXimum|MINimum|RMS|SCALar

[:SENSe]:SPECtrum:AVERage:TYPE?

Select the type of averaging.

LOG – The log of the power is averaged. (This is also known as video averaging.)

MAXimum – The maximum values are retained.

MINimum – The minimum values are retained.

RMS – The power is averaged, providing the rms of the voltage.

SCALar – The voltage is averaged.

Factory Preset and *RST: LOG

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

Spectrum— Select Pre-FFT Bandwidth

[:SENSe]:SPECtrum:BANDwidth|BWIDth:IF:AUTO OFF|ON|0|1

[:SENSe]:SPECtrum:BANDwidth|BWIDth:IF:AUTO?

Select auto or manual control of the pre-FFT BW.

Factory Preset and *RST: AUTO, 1.55 MHz

Front Panel Access: **Measure**, **Spectrum**, **Meas Setup**, **More**, **Advanced**, **Pre-FFT BW**.

Spectrum — IF Flatness Corrections

`[:SENSe]:SPECtrum:BANDwidth|BWIDth:IF:FLATness OFF|ON|0|1`

`[:SENSe]:SPECtrum:BANDwidth|BWIDth:IF:FLATness?`

Turns IF flatness corrections on and off.

Factory Preset and *RST: ON

Front Panel Access: **Measure**, **Spectrum**, **Meas Setup**, **More**, **Advanced**, **Pre-FFT BW**

Spectrum—Pre-ADC Bandpass Filter

`[:SENSe]:SPECtrum:BANDwidth|BWIDth:PADC OFF|ON|0|1`

`[:SENSe]:SPECtrum:BANDwidth|BWIDth:PADC?`

Turn the pre-ADC bandpass filter on or off. This is an advanced control that normally does not need to be changed.

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

Spectrum—Pre-FFT BW

`[:SENSe]:SPECtrum:BANDwidth|BWIDth:PFFT[:SIZE] <freq>`

`[:SENSe]:SPECtrum:BANDwidth|BWIDth:PFFT[:SIZE]?`

Set the pre-FFT bandwidth. This is an advanced control that normally does not need to be changed.

Frequency span, resolution bandwidth, and the pre-FFT bandwidth settings are normally coupled. If you are not auto-coupled, there can be combinations of these settings that are not valid.

Factory Preset and *RST: 1.55 MHz

1.25 MHz for cdmaOne

155.0 kHz, for iDEN mode

Range: 1 Hz to 10.0 MHz

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

Spectrum—Pre-FFT BW Filter Type

[:SENSe]:SPECtrum:BANDwidth|BWIDth:PFFT:TYPE FLAT|GAUSsian

[:SENSe]:SPECtrum:BANDwidth|BWIDth:PFFT:TYPE?

Select the type of pre-FFT filter that is used. This is an advanced control that normally does not need to be changed.

Flat top (FLAT)- a filter with a flat amplitude response, which provides the best amplitude accuracy.

GAUSsian - a filter with Gaussian characteristics, which provides the best pulse response.

Factory Preset and *RST: FLAT

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

Spectrum—Resolution BW

[:SENSe]:SPECtrum:BANDwidth|BWIDth[:RESolution] <freq>

[:SENSe]:SPECtrum:BANDwidth|BWIDth[:RESolution]?

Set the resolution bandwidth for the FFT. This is the bandwidth used for resolving the FFT measurement. It is not the pre-FFT bandwidth. This value is ignored if the function is auto-coupled.

Frequency span, resolution bandwidth, and the pre-FFT bandwidth settings are normally coupled. If you are not auto-coupled, there can be combinations of these settings that are not valid.

Factory Preset and *RST: 20.0 kHz

250.0 Hz, for iDEN mode

Range: 0.10 Hz to 3.0 MHz

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

Spectrum—Resolution BW Auto

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

[:SENSe]:SPECtrum:BANDwidth|BWIDth[:RESolution]:AUTO?

Select auto or manual control of the resolution BW. The automatic mode couples the resolution bandwidth setting to the frequency span.

Factory Preset and *RST: ON

OFF, for iDEN mode

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

Decimation of Spectrum Display

[:SENSe]:SPECtrum:DECimate[:FACTor] <integer>

[:SENSe]:SPECtrum:DECimate[:FACTor]?

Sets the amount of data decimation done by the hardware and/or the software. Decimation by n keeps every nth sample, throwing away each of the remaining samples in the group of n. For example, decimation by 3 keeps every third sample, throwing away the two in between. Similarly, decimation by 5 keeps every fifth sample, throwing away the four in between.

Using zero (0) decimation selects the automatic mode. The measurement will then automatically choose decimation by “1” or “2” as is appropriate for the bandwidth being used.

This is an advanced control that normally does not need to be changed.

Factory Preset and *RST: 0

Range: 0 to 1,000, where 0 sets the function to automatic

Remarks:

History: Version A.02.00 or later

Spectrum—FFT Length

[:SENSe]:SPECtrum:FFT:LENGth <integer>

[:SENSe]:SPECtrum:FFT:LENGth?

Set the FFT length. This value is only used if length control is set to manual. The value must be greater than or equal to the window length value. Any amount greater than the window length is implemented by zero-padding. This is an advanced control that normally does not need to be changed.

Factory Preset and *RST: 1024

Range: min, depends on the current setting of the spectrum window length

max, 1,048,576

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

History: Short form changed from LENgth to LENGth, A.03.00

Spectrum—FFT Length Auto

[:SENSe]:SPECtrum:FFT:LENGth:AUTO OFF|ON|0|1

[:SENSe]:SPECtrum:FFT:LENGth:AUTO?

Select auto or manual control of the FFT and window lengths.

This is an advanced control that normally does not need to be changed.

On - the window lengths are coupled to resolution bandwidth, window type (FFT), pre-FFT bandwidth (sample rate) and SENSe:SPECtrum:FFT:RBWPoints.

Off - lets you set SENSe:SPECtrum:FFT:LENGth and SENSe:SPECtrum:FFT:WINDow:LENGth.

Factory Preset and *RST: ON

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

History: Short form changed from LENgth to LENGth, A.03.00

Spectrum—FFT Minimum Points in Resolution BW

[:SENSe]:SPECtrum:FFT:RBWPoints <real>

[:SENSe]:SPECtrum:FFT:RBWPoints?

Set the minimum number of data points that will be used inside the resolution bandwidth. The value is ignored if length control is set to manual. This is an advanced control that normally does not need to be changed.

Factory Preset and *RST: 1.30

Range: 0.1 to 100

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

Spectrum—Window Delay (Remote Command Only)

[:SENSe]:SPECtrum:FFT:WINDow:DELay <real>

[:SENSe]:SPECtrum:FFT:WINDow:DELay?

Set the FFT window delay to move the FFT window from its nominal position of being centered within the time capture. This function is not available from the front panel. It is an advanced control that normally does not need to be changed.

Factory Preset and *RST: 0

Range: –10.0 to +10.0 s

Default Unit: seconds

Remarks: To use this command, the Service mode must be selected with INSTrument:SELect. In Service mode, it is possible to get an acquisition time that is longer than the window time so that this function can be used.

Spectrum—Window Length

[:SENSe]:SPECtrum:FFT:WINDow:LENGth <integer>

[:SENSe]:SPECtrum:FFT:WINDow:LENGth?

Set the FFT window length. This value is only used if length control is set to manual. This is an advanced control that normally does not need to be changed.

Factory Preset and *RST: 706

Range: 8 to 1,048,576

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

History: Short form changed from LENgth to LENGth, A.03.00

Spectrum—FFT Window

[:SENSe]:SPECtrum:FFT:WINDow[:TYPE]
BH4Tap|BLACkman|FLATtop|GAUSsian|HAMMing|HANNing|KB70|KB90|KB110|UNIForm

[:SENSe]:SPECtrum:FFT:WINDow[:TYPE]?

Select the FFT window type.

BH4Tap - Blackman Harris with 4 taps

BLACkman - Blackman

FLATtop - flat top, the default (for high amplitude accuracy)

GAUSsian - Gaussian with alpha of 3.5

HAMMing - Hamming

HANNing - Hanning

KB70, 90, and 110 - Kaiser Bessel with sidelobes at –70, –90, or –110 dBc

UNIForm - no window is used. (This is the unity response.)

Factory Preset and *RST: FLATtop

Remarks: This selection affects the acquisition point quantity and the FFT size, based on the resolution bandwidth selected.

To use this command, the appropriate mode should be selected with INSTrument:SELect.

Spectrum—Frequency Span

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

`[:SENSe]:SPECtrum:FREQuency:SPAN?`

Set the frequency span to be measured.

Factory Preset and *RST: 1.0 MHz

100.0 kHz for iDEN mode

Range: 10 Hz to 10.0 MHz (15 MHz when Service mode is selected)

Default Unit: Hz

Remarks: The actual measured span will generally be slightly wider due to the finite resolution of the FFT.

To use this command, the appropriate mode should be selected with INSTrument:SELect.

Spectrum—Sweep (Acquisition) Time

`[:SENSe]:SPECtrum:SWEep:TIME[:VALue] <time>`

`[:SENSe]:SPECtrum:SWEep:TIME?`

Set the sweep (measurement acquisition) time. It is used to specify the length of the time capture record. If the specified value is less than the

capture time required for the specified span and resolution bandwidth, the value is ignored. The value is set at its auto value when auto is selected. This is an advanced control that normally does not need to be changed.

Factory Preset and *RST: 188.0 µs

Range: 100 ns to 10 s

Default Unit: seconds

Remarks: You must be in the Service mode to use this command. Use INSTrument:SELect to set the mode.

This command only effects the RF envelope trace.

Spectrum—Sweep (Acquisition) Time Auto

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

[:SENSe]:SPECtrum:SWEep:TIME:AUTO

Select auto or manual control of the sweep (acquisition) time. This is an advanced control that normally does not need to be changed.

AUTO - couples the Sweep Time to the Frequency Span and Resolution BW

Manual - the Sweep Time is uncoupled from the Frequency Span and Resolution BW.

Factory Preset and *RST: AUTO

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

Spectrum—Trigger Source

[:SENSe]:SPECtrum:TRIGger:SOURce
EXTernal[1]|EXTernal2|FRAMe|IF|LINE|IMMediate|RFBurst

[:SENSe]:SPECtrum:TRIGger:SOURce?

Select the trigger source used to control the data acquisitions.

EXTernal1 - front panel external trigger input

EXTernal2 - rear panel external trigger input

FRAMe - internal frame timer from front panel input

IF - internal IF envelope (video) trigger

LINE - internal line trigger

IMMediate - the next data acquisition is immediately taken (also called free run)

RFBurst - wideband RF burst envelope trigger that has automatic level control for periodic burst signals

Factory Preset and *RST: IMMediate (free run)

RFBurst, for GSM, iDEN mode

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

Waveform (Time-Domain) Measurement

Commands for querying the waveform measurement results and for setting to the default values are found in "MEASure Group of Commands" on page 301. The equivalent front-panel keys for the parameters described in the following commands, are found under the **Meas Setup** key, after the **Waveform (Time Domain)** measurement has been selected from the **MEASURE** key menu.

Waveform—Data Acquisition Packing

`[:SENSe]:WAVeform:ACQuistion:PACKing AUTO|LONG|MEDium|SHORt`

`[:SENSe]:WAVeform:ACQuistion:PACKing?`

This is an advanced control that normally does not need to be changed.

Factory Preset and *RST: AUTO

Remarks: You must be in the Service mode to use this command. Use INSTrument:SELect to set the mode.

Waveform—ADC Dither State

`[:SENSe]:WAVeform:ADC:DITHer[:STATe] |OFF|ON|0|1`

`[:SENSe]:WAVeform:ADC:DITHer[:STATe]?`

Turn the ADC dither on or off. This is an advanced control that normally does not need to be changed. The "ADC dither" refers to the introduction of noise to the digitized steps of the analog-to-digital converter; the result is an improvement in amplitude accuracy.

Factory Preset and *RST: OFF

Remarks: You must be in the Service mode to use this command. Use INSTrument:SELect to set the mode.

Waveform—Pre-ADC Bandpass Filter

[:SENSe]:WAVeform:ADC:FILTer[:STATe] OFF|ON|0|1

[:SENSe]:WAVeform:ADC:FILTer[:STATe]?

Turn the pre-ADC bandpass filter on or off. This is an Advanced control that normally does not need to be changed.

Preset: OFF

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

Waveform—ADC Range

12-bit ADC
[:SENSe]:WAVeform:ADC:RANGe
AUTO|APEak|APLock|GROund|M6|P0|P6|P12|P18|P24

PSA and 14-bit ADC
[:SENSe]:WAVeform:ADC:RANGe
AUTO|APEak|APLock|GROund|NONE|P0|P6|P12|P18

[:SENSe]:WAVeform:ADC:RANGe?

Select the range for the gain-ranging that is done in front of the ADC. This is an Advanced control that normally does not need to be changed.

AUTO - automatic range

Auto Peak (APEak) - automatically peak the range

Auto Peak Lock (APLock)- automatically peak lock the range

GROund - ground

NONE - (14-bit ADC E4406A) turn off auto-ranging without making any changes to the current setting.

M6 - (12-bit ADC E4406A) subtracts 6 dB of fixed gain across the range

P0 to P18 - (14-bit ADC E4406A) adds 0 to 18 dB of fixed gain across the range

P0 to 24 - adds 0 to 24 dB of fixed gain across the range

Factory Preset and *RST: AUTO

Remarks: To use this command, the appropriate mode should be

selected with INSTrument:SELect.

Waveform - Query Aperture Setting

[:SENSe]:WAVeform:APERture?

Returns the waveform sample period (aperture) based on current resolution bandwidth, filter type, and decimation factor. Sample rate is the reciprocal of period.

Remarks: To use this command the appropriate mode should be selected with INSTrument:SELect.

History: Version A.05.00 or later

Waveform—Number of Averages

[:SENSe]:WAVeform:AVERage:COUNt <integer>

[:SENSe]:WAVeform:AVERage:COUNt?

Set the number of sweeps that will be averaged. After the specified number of sweeps (average counts), the averaging mode (terminal control) setting determines the averaging action.

Factory Preset and *RST: 10

Range: 1 to 10,000

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

Waveform—Averaging State

[:SENSe]:WAVeform:AVERage[:STATe] OFF|ON|0|1

[:SENSe]:WAVeform:AVERage[:STATe]?

Turn averaging on or off.

Factory Preset and *RST: OFF

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

Waveform—Averaging Mode

[:SENSe]:WAVeform:AVERage:TCONtrol EXPonential|REPeat

[:SENSe]:WAVeform:AVERage:TCONtrol?

Select the type of termination control used for the averaging function. This determines the averaging action after the specified number of 'sweeps' (average count) is reached.

EXPonential - Each successive data acquisition after the average count is reached, is exponentially weighted and combined with the existing average.

REPeat - After reaching the average count, the averaging is reset and a new average is started.

Factory Preset and *RST: EXPonential

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

Waveform—Averaging Type

[:SENSe]:WAVeform:AVERage:TYPE LOG|MAXimum|MINimum|RMS|SCALar

[:SENSe]:WAVeform:AVERage:TYPE?

Select the type of averaging.

LOG - The log of the power is averaged. (This is also known as video averaging.)

MAXimum - The maximum values are retained.

MINimum - The minimum values are retained.

RMS - The power is averaged, providing the rms of the voltage.

Factory Preset and *RST: RMS

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

Waveform—Resolution BW

[:SENSe]:WAVeform:BANDwidth|BWIDth[:RESolution] <freq>

[:SENSe]:WAVeform:BANDwidth|BWIDth[:RESolution]?

Set the resolution bandwidth. This value is ignored if the function is auto-coupled.

Factory Preset and *RST: 100.0 kHz for NADC, PDC, cdma2000, W-CDMA

(3GPP), basic, service
500.0 kHz for GSM
2.0 MHz for cdmaOne

Range: 1.0 kHz to 8.0 MHz when SENSe:WAV:BWID:RES:TYPE GAUSsian

100 mHz to 10.0 MHz when SENSe:WAV:BWID:RES:TYPE FLATtop

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

Bandwidths > 6.7 MHz will require a slight increase in measurement time.

Waveform - Query Actual Resolution Bandwidth

[:SENSe]:WAVeform:BANDwidth:RESolution]:ACTual?

Due to memory constraints the actual resolution bandwidth value may vary from the value entered by the user. For most applications the resulting difference in value is inconsequential but for some it is necessary to know the actual value; this query retrieves the actual resolution bandwidth value.

Remarks: To use this command the appropriate mode should be selected with INSTrument:SELect.

History: Version A.05.00 or later

Waveform—Resolution BW Filter Type

[:SENSe]:WAVeform:BANDwidth|BWIDth[:RESolution]:TYPE FLATtop|GAUSsian

[:SENSe]:WAVeform:BANDwidth|BWIDth[:RESolution]:TYPE?

Select the type of Resolution BW filter that is used. This is an Advanced control that normally does not need to be changed.

FLATtop - a filter with a flat amplitude response, which provides the best amplitude accuracy.

GAUSsian - a filter with Gaussian characteristics, which provides the best pulse response.

Factory Preset and *RST: GAUSsian

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

Waveform—Decimation of Waveform Display

[:SENSe]:WAVeform:DECimate[:FACTor] <integer>

[:SENSe]:WAVeform:DECimate[:FACTor]?

Set the amount of data decimation done on the IQ data stream. For example, if 4 is selected, three out of every four data points will be thrown away. So every 4th data point will be kept.

Factory Preset and *RST: 1

Range: 1 to 4

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

Waveform—Control Decimation of Waveform Display

[:SENSe]:WAVeform:DECimate:STATe OFF|ON|0|1

[:SENSe]:WAVeform:DECimate:STATe?

Set the amount of data decimation done by the hardware in order to decrease the number of acquired points in a long capture time. This is the amount of data that the measurement ignores.

Factory Preset and *RST: OFF

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

Waveform—Sweep (Acquisition) Time

[:SENSe]:WAVeform:SWEep:TIME <time>

[:SENSe]:WAVeform:SWEep:TIME?

Set the measurement acquisition time. It is used to specify the length of the time capture record.

Factory Preset and *RST: 2.0 ms

10.0 ms, for NADC, PDC

15.0 ms, for iDEN mode

Range: 1 µs to 100 s

Default Unit: seconds

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

Waveform—Trigger Source

[:SENSe]:WAVeform:TRIGger:SOURce EXTernal[1]|EXTernal2|FRAMe|IF|IMMediate|LINE|RFBurst

[:SENSe]:WAVeform:TRIGger:SOURce?

Select the trigger source used to control the data acquisitions.

EXTernal 1 - front panel external trigger input

EXTernal 2 - rear panel external trigger input

FRAMe - internal frame timer from front panel input

IF - internal IF envelope (video) trigger

IMMediate - the next data acquisition is immediately taken (also called free run)

LINE - internal line trigger

RFBurst - wideband RF burst envelope trigger that has automatic level control for periodic burst signals

Factory Preset and *RST: IMMediate (free run), for Basic, cdmaOne, NADC, PDC mode

RFBurst, for GSM, iDEN mode

Remarks: To use this command, the appropriate mode should be selected with INSTrument:SELect.

SERVice Subsystem

Provides SCPI access for the calibration manager.

Numeric values for bit patterns can be entered using decimal or hexidecimal representations. (i.e. 0 to 32767 is equivalent to #H0 to #H7FFF) See the SCPI Basics information about using bit patterns for variable parameters.

Prepare Calibration Files for Access

:SERVice[:PRODuction]:CALibrate:BEGin

Locks all of the calibration files for memory accesses.

Remarks: No query.

Load Default Calibration Data to NRAM

:SERVice[:PRODuction]:CALibrate:DEFault <cal_fid>

Loads the specified calibration data from EEROM to NRAM, initializing the alignment data to the factory defaults.

Range: cal_fid, corresponds to the Calibrate file ID

Remarks: No query.

Unlock Calibration Files

:SERVice[:PRODuction]:CALibrate:END

Unlocks all of the calibration files.

Remarks: info

Store Calibration Data in EEROM

:SERVice[:PRODuction]:CALibrate:STORe <cal_fid>

Stores the specified calibration data into EEROM. The data will survive power cycles and will be reloaded into NRAM on power up.

Range: cal_fid, corresponds to the calibration data file ID

Remarks: No query.

STATus Subsystem

The STATus subsystem controls the SCPI-defined instrument-status reporting structures. Each status register has a set of five commands used for querying or masking that particular register.

Numeric values for bit patterns can be entered using decimal or hexidecimal representations. (i.e. 0 to 32767 is equivalent to #H0 to #H7FFF) See the SCPI Basics information about using bit patterns for variable parameters.

Operation Register

Operation Condition Query

```
:STATus:OPERation:CONDition?
```

This query returns the decimal value of the sum of the bits in the Status Operation Condition register.

NOTE The data in this register is continuously updated and reflects the current conditions.

Key Type: There is no equivalent front-panel key.

Operation Enable

```
:STATus:OPERation:ENABle <integer>
```

```
:STATus:OPERation:ENABle?
```

This command determines what bits in the Operation Event register, will set the Operation Status Summary bit (bit 7) in the Status Byte Register. The variable <number> is the sum of the decimal values of the bits you want to enable.

NOTE The preset condition is to have all bits in this enable register set to 0. To have any Operation Events reported to the Status Byte Register, one or more bits need to be set to 1.

Key Type: There is no equivalent front-panel key.

Factory Preset and *RST: 1

Range: 0 to 32767

Operation Event Query

:STATus:OPERation[:EVENt]?

This query returns the decimal value of the sum of the bits in the Operation Event register.

NOTE

The register requires that the associated PTR or NTR filters be set before a condition register bit can set a bit in the event register.

The data in this register is latched until it is queried. Once queried, the register is cleared.

Key Type: There is no equivalent front-panel key.

Operation Negative Transition

:STATus:OPERation:NTRansition <integer>

:STATus:OPERation:NTRansition?

This command determines what bits in the Operation Condition register will set the corresponding bit in the Operation Event register when the condition register bit has a negative transition (1 to 0). The variable <number> is the sum of the decimal values of the bits that you want to enable.

Key Type: There is no equivalent front-panel key.

Factory Preset and *RST: 0

Range: 0 to 32767

Operation Positive Transition

:STATus:OPERation:PTRansition <integer>

:STATus:OPERation:PTRansition?

This command determines what bits in the Operation Condition register will set the corresponding bit in the Operation Event register when the condition register bit has a positive transition (0 to 1). The variable <number> is the sum of the decimal values of the bits that you want to enable.

Key Type: There is no equivalent front-panel key.

Factory Preset and *RST: 32767 (all 1s)

Range: 0 to 32767

Preset the Status Byte

:STATus:PRESet

Sets bits in most of the enable and transition registers to their default state. It presets all the Transition Filters, Enable Registers, and the Error/Event Queue Enable. It has no effect on Event Registers, Error/Event QUEue, IEEE 488.2 ESE, and SRE Registers as described in IEEE Standard 488.2-1992, *IEEE Standard Codes, Formats, Protocols and Common Commands for Use with ANSI/IEEE Std 488.1-1987*. New York, NY, 1992.

Key Type: There is no equivalent front-panel key.

Questionable Register

Questionable Condition

:STATus:QUEStionable:CONDition?

This query returns the decimal value of the sum of the bits in the Questionable Condition register.

NOTE The data in this register is continuously updated and reflects the current conditions.

Key Type: There is no equivalent front-panel key.

Questionable Enable

:STATus:QUEStionable:ENABle <number>

:STATus:QUEStionable:ENABle?

This command determines what bits in the Questionable Event register will set the Questionable Status Summary bit (bit3) in the Status Byte Register. The variable <number> is the sum of the decimal values of the bits you want to enable.

NOTE The preset condition is all bits in this enable register set to 0. To have any Questionable Events reported to the Status Byte Register, one or more bits need to be set to 1. It is recommended that all bits be enabled in this register. The Status Byte Event Register should be queried after each measurement to check the Questionable Status Summary (bit 3). If it is equal to 1, a condition during the test may have made the test results invalid. If it is equal to 0, this indicates that no hardware

problem or measurement problem was detected by the analyzer.

Key Type: There is no equivalent front-panel key.

Factory Preset and *RST: 0

Range: 0 to 32767

Questionable Event Query

:STATus:QUEStionable[:EVENt]?

This query returns the decimal value of the sum of the bits in the Questionable Event register.

NOTE

The register requires that the associated PTR or NTR filters be set before a condition register bit can set a bit in the event register.

The data in this register is latched until it is queried. Once queried, the register is cleared.

Key Type: There is no equivalent front-panel key.

Questionable Negative Transition

:STATus:QUEStionable:NTRansition <number>

:STATus:QUEStionable:NTRansition?

This command determines what bits in the Questionable Condition register will set the corresponding bit in the Questionable Event register when the condition register bit has a negative transition (1 to 0). The variable <number> is the sum of the decimal values of the bits that you want to enable.

Key Type: There is no equivalent front-panel key.

Factory Preset and *RST: 0

Range: 0 to 32767

Questionable Positive Transition

:STATus:QUEStionable:PTRansition <number>

:STATus:QUEStionable:PTRansition?

This command determines what bits in the Questionable Condition register will set the corresponding bit in the Questionable Event

register when the condition register bit has a positive transition (0 to 1). The variable <number> is the sum of the decimal values of the bits that you want to enable.

Key Type: There is no equivalent front-panel key.

Factory Preset and *RST: 32767 (all 1s)

Range: 0 to 32767

Questionable Calibration Register

Questionable Calibration Condition

:STATus:QUEStionable:CALibration:CONDition?

This query returns the decimal value of the sum of the bits in the Questionable Calibration Condition register.

NOTE The data in this register is continuously updated and reflects the current conditions.

Key Type: There is no equivalent front-panel key.

Questionable Calibration Enable

:STATus:QUEStionable:CALibration:ENABle <number>

:STATus:QUEStionable:CALibration:ENABle?

This command determines what bits in the Questionable Calibration Condition Register will set bits in the Questionable Calibration Event register, which also sets the Calibration Summary bit (bit 8) in the Questionable Register. The variable <number> is the sum of the decimal values of the bits you want to enable.

Key Type: There is no equivalent front-panel key.

Example: STAT:QUES:CAL:ENABLE 16384 could be used if you have turned off the automatic alignment and you want to query if an alignment is needed.

Factory Preset and *RST: 32767 (all 1s)

Range: 0 to 32767

Questionable Calibration Event Query

:STATus:QUEStionable:CALibration[:EVENt]?

This query returns the decimal value of the sum of the bits in the Questionable Calibration Event register.

NOTE

The register requires that the associated PTR or NTR filters be set before a condition register bit can set a bit in the event register.

The data in this register is latched until it is queried. Once queried, the register is cleared.

Key Type: There is no equivalent front-panel key.

Questionable Calibration Negative Transition

:STATus:QUEStionable:CALibration:NTRansition <number>

:STATus:QUEStionable:CALibration:NTRansition?

This command determines what bits in the Questionable Calibration Condition register will set the corresponding bit in the Questionable Calibration Event register when the condition register bit has a negative transition (1 to 0). The variable <number> is the sum of the decimal values of the bits that you want to enable.

Key Type: There is no equivalent front-panel key.

Factory Preset and *RST: 0

Range: 0 to 32767

Questionable Calibration Positive Transition

:STATus:QUEStionable:CALibration:PTRansition <number>

:STATus:QUEStionable:CALibration:PTRansition?

This command determines what bits in the Questionable Calibration Condition register will set the corresponding bit in the Questionable Calibration Event register when the condition register bit has a positive transition (0 to 1). The variable <number> is the sum of the decimal values of the bits that you want to enable.

Key Type: There is no equivalent front-panel key.

Factory Preset and *RST: 32767 (all 1s)

Range: 0 to 32767

Questionable Frequency Register

Questionable Frequency Condition

:STATus:QUEStionable:FREQuency:CONDition?

This query returns the decimal value of the sum of the bits in the Questionable Frequency Condition register.

NOTE The data in this register is continuously updated and reflects the current conditions.

Key Type: There is no equivalent front-panel key.

Questionable Frequency Enable

:STATus:QUEStionable:FREQuency:ENABle <number>

:STATus:QUEStionable:FREQuency:ENABle?

This command determines what bits in the Questionable Frequency Condition Register will set bits in the Questionable Frequency Event register, which also sets the Frequency Summary bit (bit 5) in the Questionable Register. The variable <number> is the sum of the decimal values of the bits you want to enable.

Key Type: There is no equivalent front-panel key.

Factory Preset and *RST: 32767 (all 1s)

Range: 0 to 32767

Questionable Frequency Event Query

:STATus:QUEStionable:FREQuency[:EVENt]?

This query returns the decimal value of the sum of the bits in the Questionable Frequency Event register.

NOTE The register requires that the associated PTR or NTR filters be set before a condition register bit can set a bit in the event register.

The data in this register is latched until it is queried. Once queried, the register is cleared.

Key Type: There is no equivalent front-panel key.

Questionable Frequency Negative Transition

```
:STATus:QUEStionable:FREQuency:NTRansition <number>
```

```
:STATus:QUEStionable:FREQuency:NTRansition?
```

This command determines what bits in the Questionable Frequency Condition register will set the corresponding bit in the Questionable Frequency Event register when the condition register bit has a negative transition (1 to 0). The variable <number> is the sum of the decimal values of the bits that you want to enable.

Key Type: There is no equivalent front-panel key.

Factory Preset and *RST: 0

Range: 0 to 32767

Questionable Frequency Positive Transition

```
:STATus:QUEStionable:FREQuency:PTRansition <number>
```

```
:STATus:QUEStionable:FREQuency:PTRansition?
```

This command determines what bits in the Questionable Frequency Condition register will set the corresponding bit in the Questionable Frequency Event register when the condition register bit has a positive transition (0 to 1). The variable <number> is the sum of the decimal values of the bits that you want to enable.

Key Type: There is no equivalent front-panel key.

Factory Preset and *RST: 32767 (all 1s)

Range: 0 to 32767

Questionable Integrity Register

Questionable Integrity Condition

```
:STATus:QUEStionable:INTegrity:CONDition?
```

This query returns the decimal value of the sum of the bits in the Questionable Integrity Condition register.

NOTE The data in this register is continuously updated and reflects the current conditions.

Key Type: There is no equivalent front-panel key.

Questionable Integrity Enable

:STATus:QUEStionable:INTegrity:ENABle <number>

:STATus:QUEStionable:INTegrity:ENABle?

This command determines what bits in the Questionable Integrity Condition Register will set bits in the Questionable Integrity Event register, which also sets the Integrity Summary bit (bit 9) in the Questionable Register. The variable <number> is the sum of the decimal values of the bits you want to enable.

Key Type: There is no equivalent front-panel key.

Factory Preset and *RST: 32767 (all 1s)

Range: 0 to 32767

Questionable Integrity Event Query

:STATus:QUEStionable:INTegrity[:EVENt]?

This query returns the decimal value of the sum of the bits in the Questionable Integrity Event register.

NOTE

The register requires that the associated PTR or NTR filters be set before a condition register bit can set a bit in the event register.

The data in this register is latched until it is queried. Once queried, the register is cleared.

Key Type: There is no equivalent front-panel key.

Questionable Integrity Negative Transition

:STATus:QUEStionable:INTegrity:NTRansition <number>

:STATus:QUEStionable:INTegrity:NTRansition?

This command determines what bits in the Questionable Integrity Condition register will set the corresponding bit in the Questionable Integrity Event register when the condition register bit has a negative transition (1 to 0)
The variable <number> is the sum of the decimal values of the bits that you want to enable.

Key Type: There is no equivalent front-panel key.

Factory Preset

and *RST: 0

Range: 0 to 32767

Questionable Integrity Positive Transition

```
:STATus:QUEStionable:INTegrity:PTRansition <number>
```

```
:STATus:QUEStionable:INTegrity:PTRansition?
```

This command determines what bits in the Questionable Integrity Condition register will set the corresponding bit in the Questionable Integrity Event register when the condition register bit has a positive transition (0 to 1). The variable <number> is the sum of the decimal values of the bits that you want to enable.

Key Type: There is no equivalent front-panel key.

Factory Preset and *RST: 32767 (all 1s)

Range: 0 to 32767

Questionable Integrity Signal Register

Questionable Integrity Signal Condition

```
:STATus:QUEStionable:INTegrity:SIGNal:CONDition?
```

This query returns the decimal value of the sum of the bits in the Questionable Integrity Signal Condition register.

NOTE The data in this register is continuously updated and reflects the current conditions.

Key Type: There is no equivalent front-panel key.

Questionable Integrity Signal Enable

```
:STATus:QUEStionable:INTegrity:SIGNal:ENABle <number>
```

```
:STATus:QUEStionable:INTegrity:SIGNal:ENABle?
```

This command determines what bits in the Questionable Integrity Signal Condition Register will set bits in the Questionable Integrity Signal Event register, which also sets the Integrity Summary bit (bit 9) in the Questionable Register. The variable <number> is the sum of the decimal values of the bits you want to enable.

Key Type: There is no equivalent front-panel key.

Factory Preset and *RST: 32767 (all 1s)

Range: 0 to 32767

Questionable Integrity Signal Event Query

```
:STATus:QUEStionable:INTegrity:SIGNal[:EVENt]?
```

This query returns the decimal value of the sum of the bits in the Questionable Integrity Signal Event register.

NOTE

The register requires that the associated PTR or NTR filters be set before a condition register bit can set a bit in the event register.

The data in this register is latched until it is queried. Once queried, the register is cleared.

Key Type: There is no equivalent front-panel key.

Questionable Integrity Signal Negative Transition

```
:STATus:QUEStionable:INTegrity:SIGNal:NTRansition <number>
```

```
:STATus:QUEStionable:INTegrity:SIGNal:NTRansition?
```

This command determines what bits in the Questionable Integrity Signal Condition register will set the corresponding bit in the Questionable Integrity Signal Event register when the condition register bit has a negative transition (1 to 0). The variable <number> is the sum of the decimal values of the bits that you want to enable.

Key Type: There is no equivalent front-panel key.

Factory Preset and *RST: 0

Range: 0 to 32767

Questionable Integrity Signal Positive Transition

```
:STATus:QUEStionable:INTegrity:SIGNal:PTRansition <number>
```

```
:STATus:QUEStionable:INTegrity:SIGNal:PTRansition?
```

This command determines what bits in the Questionable Integrity Signal Condition register will set the corresponding bit in the Questionable Integrity Signal Event register when the condition register bit has a positive transition (0 to 1). The variable <number> is the sum of the decimal values of the bits that you want to enable.

Key Type: There is no equivalent front-panel key.

Factory Preset and *RST: 32767 (all 1s)

Range: 0 to 32767

Questionable Integrity Uncalibrated Register

Questionable Integrity Uncalibrated Condition

:STATus:QUEStionable:INTegrity:UNCalibrated:CONDition?

This query returns the decimal value of the sum of the bits in the Questionable Integrity Uncalibrated Condition register.

NOTE The data in this register is continuously updated and reflects the current conditions.

Key Type: There is no equivalent front-panel key.

Questionable Integrity Uncalibrated Enable

:STATus:QUEStionable:INTegrity:UNCalibrated:ENABle

:STATus:QUEStionable:INTegrity:UNCalibrated:ENABle?

This command determines which bits in the Questionable Integrity Uncalibrated Condition Register will set bits in the Questionable Integrity Uncalibrated Event register, which also sets the Data Uncalibrated Summary bit (bit 3) in the Questionable Integrity Register. The variable <number> is the sum of the decimal values of the bits you want to enable.

Key Type: There is no equivalent front-panel key.

Factory Preset and *RST: 32767 (all 1s)

Range: 0 to 32767

Questionable Integrity Uncalibrated Event Query

:STATus:QUEStionable:INTegrity:UNCalibrated[:EVENt]?

This query returns the decimal value of the sum of the bits in the Questionable Integrity Uncalibrated Event register.

NOTE The register requires that the associated PTR or NTR filters be set

before a condition register bit can set a bit in the event register.

The data in this register is latched until it is queried. Once queried, the register is cleared.

Key Type: There is no equivalent front-panel key.

Questionable Integrity Uncalibrated Negative Transition

:STATus:QUEStionable:INTegrity:UNCalibrated:NTRansition <number>

:STATus:QUEStionable:INTegrity:UNCalibrated:NTRansition?

This command determines which bits in the Questionable Integrity Uncalibrated Condition register will set the corresponding bit in the Questionable Integrity Uncalibrated Event register when the condition register bit has a negative transition (1 to 0). The variable <number> is the sum of the decimal values of the bits that you want to enable.

Key Type: There is no equivalent front-panel key.

Factory Preset and *RST: 0

Range: 0 to 32767

Questionable Integrity Uncalibrated Positive Transition

:STATus:QUEStionable:INTegrity:UNCalibrated:PTRansition <number>

:STATus:QUEStionable:INTegrity:UNCalibrated:PTRansition?

This command determines which bits in the Questionable Integrity Uncalibrated Condition register will set the corresponding bit in the Questionable Integrity Uncalibrated Event register when the condition register bit has a positive transition (0 to 1). The variable <number> is the sum of the decimal values of the bits that you want to enable.

Key Type: There is no equivalent front-panel key.

Factory Preset and *RST: 32767 (all 1s)

Range: 0 to 32767

Questionable Power Register

Questionable Power Condition

:STATus:QUEStionable:POWer:CONDition?

This query returns the decimal value of the sum of the bits in the Questionable Power Condition register.

NOTE The data in this register is continuously updated and reflects the current conditions.

Key Type: There is no equivalent front-panel key.

Questionable Power Enable

:STATus:QUEStionable:POWer:ENABle <number>

:STATus:QUEStionable:POWer:ENABle?

This command determines what bits in the Questionable Power Condition Register will set bits in the Questionable Power Event register, which also sets the Power Summary bit (bit 3) in the Questionable Register. The variable <number> is the sum of the decimal values of the bits you want to enable.

Key Type: There is no equivalent front-panel key.

Factory Preset and *RST: 32767 (all 1s)

Range: 0 to 32767

Questionable Power Event Query

:STATus:QUEStionable:POWer[:EVENt]?

This query returns the decimal value of the sum of the bits in the Questionable Power Event register.

NOTE The register requires that the associated PTR or NTR filters be set before a condition register bit can set a bit in the event register.

The data in this register is latched until it is queried. Once queried, the register is cleared.

Key Type: There is no equivalent front-panel key.

Questionable Power Negative Transition

```
:STATus:QUEStionable:POWer:NTRansition <number>
```

```
:STATus:QUEStionable:POWer:NTRansition?
```

This command determines what bits in the Questionable Power Condition register will set the corresponding bit in the Questionable Power Event register when the condition register bit has a negative transition (1 to 0). The variable <number> is the sum of the decimal values of the bits that you want to enable.

Key Type: There is no equivalent front-panel key.

Factory Preset and *RST: 0

Range: 0 to 32767

Questionable Power Positive Transition

```
:STATus:QUEStionable:POWer:PTRansition <number>
```

```
:STATus:QUEStionable:POWer:PTRansition?>
```

This command determines what bits in the Questionable Power Condition register will set the corresponding bit in the Questionable Power Event register when the condition register bit has a positive transition (0 to 1). The variable <number> is the sum of the decimal values of the bits that you want to enable.

Key Type: There is no equivalent front-panel key.

Factory Preset and *RST: 32767 (all 1s)

Range: 0 to 32767

Questionable Temperature Register

Questionable Temperature Condition

```
:STATus:QUEStionable:TEMPerature:CONDition?
```

This query returns the decimal value of the sum of the bits in the Questionable Temperature Condition register.

NOTE The data in this register is continuously updated and reflects the current conditions.

Key Type: There is no equivalent front-panel key.

Questionable Temperature Enable

:STATus:QUEStionable:TEMPerature:ENABle <number>

:STATus:QUEStionable:TEMPerature:ENABle?

This command determines what bits in the Questionable Temperature Condition Register will set bits in the Questionable Temperature Event register, which also sets the Temperature Summary bit (bit 4) in the Questionable Register. The variable <number> is the sum of the decimal values of the bits you want to enable.

Key Type: There is no equivalent front-panel key.

Factory Preset and *RST: 32767 (all 1s)

Range: 0 to 32767

Questionable Temperature Event Query

:STATus:QUEStionable:TEMPerature[:EVENt]?

This query returns the decimal value of the sum of the bits in the Questionable Temperature Event register.

NOTE

The register requires that the associated PTR or NTR filters be set before a condition register bit can set a bit in the event register.

The data in this register is latched until it is queried. Once queried, the register is cleared

Key Type: There is no equivalent front-panel key.

Questionable Temperature Negative Transition

:STATus:QUEStionable:TEMPerature:NTRansition <number>

:STATus:QUEStionable:TEMPerature:NTRansition?

This command determines what bits in the Questionable Temperature Condition register will set the corresponding bit in the Questionable Temperature Event register when the condition register bit has a negative transition (1 to 0). The variable <number> is the sum of the decimal values of the bits that you want to enable.

Key Type: There is no equivalent front-panel key.

Factory Preset and *RST: 0

Range: 0 to 32767

Questionable Temperature Positive Transition

:STATus:QUEStionable:TEMPerature:PTRansition <number>

:STATus:QUEStionable:TEMPerature:PTRansition?

This command determines what bits in the Questionable Temperature Condition register will set the corresponding bit in the Questionable Temperature Event register when the condition register bit has a positive transition (0 to 1). The variable <number> is the sum of the decimal values of the bits that you want to enable.

Key Type: There is no equivalent front-panel key.

Factory Preset and *RST: 32767 (all 1s)

Range: 0 to 32767

SYSTem Subsystem

This subsystem is used to set the controls and parameters associated with the overall system communication. These are functions that are not related to instrument performance. Examples include functions for performing general housekeeping and functions related to setting global configurations.

GPIB Address

`:SYSTem:COMMunicate:GPIB[:SELF]:ADDRess <integer>`

`:SYSTem:COMMunicate:GPIB[:SELF]:ADDRess?`

Sets and queries the GPIB address.

Example: SYST:COMM:GPIB:ADDR 18

Factory Preset and *RST: The factory default is 18.

This function is persistent which means that it stays at the setting previously selected, even through a power cycle.

Range: Integer, 0 to 30

Example: SYST:COMM:GIPB:ADDRESS 18

Front Panel Access: **System, Config I/O, GPIB Addr**

LAN IP Address with Host Name

`:SYSTem:COMMunicate:LAN[:SELF]:IP <string>`

`:SYSTem:COMMunicate:LAN[:SELF]:IP?`

Set the IP (internet protocol) address, domain name and node name for the instrument.

<string> is a string that contains: <IP address> <host name> as shown in the following example:

```
141.4.402.222 sigan
```

where: 141.4.402.222, is the IP address and sigan, is the host name.

Example: SYST:COMM:LAN:IP “22.121.44.45 analyz”

Front Panel Access: **System, Config I/O, Config LAN**

Options Configuration Query

:SYSTem:CONFigure?

The query returns the current options configuration information. It will return the following type of information:

#3764Model Number: E4406ASerial Number: US38330068

Host Id: E566DD69

Firmware Revision: A.05.07

Firmware Date: 20010327

STD SERVICE A.05.07 Standard ok Installed

BAH GSM A.05.07 9C8B6AABF2BE ok Installed

BAC CDMA A.05.07 7FA587C8ECC1 ok Installed

BAE NADC A.05.07 859981C2E0C7 ok Installed

#3764 - is the block data header. See FORMat:DATA for more details
ok / none - is the license key status. ok means the license key is in memory. See SYST:LKEY command. The hexadecimal number in the preceding column is the license key itself. The option firmware must also be installed in memory.
Installed / Not Installed - indicates whether the option is installed/stored in the memory of the instrument. Use the firmware installation process for this. See www.agilent.com/find/vsa for more information.

Example: SYST:CONF?

Front Panel Access: **System, Show System**

Hardware Configuration Default

:SYSTem:CONFigure:DEFault

Resets all instrument functions to the factory defaults, including the persistent functions. Persistent functions are system settings that stay at their current settings even through instrument power-on, such as I/O bus addresses and preset preferences.

Example: SYST:CONF:DEF

Front Panel Access: **System, Restore Sys Defaults**

System Configuration Query

:SYSTem:CONFigure[:SYSTem]?

Returns a block of data listing the current option configuration information as on the **Show System** screen. For more information about how to use block data see the FORMat:DATA command or the Programming Fundamentals: SCPI Language Basics discussion on arbitrary length block data. The query returns the following type of information:

#3764Model Number: E4406ASerial Number: US38330068

Host Id: E566DD69

Firmware Revision: A.05.07

Firmware Date: 20010327

STD SERVICE A.05.07 Standard ok Installed

BAH GSM A.05.07 9C8B6AABF2BE ok Installed

BAC CDMA A.05.07 7FA587C8ECC1 ok Installed

BAE NADC A.05.07 859981C2E0C7 ok Installed

#3764 - is the block data header. See FORMat:DATA for more details
ok / none - is the license key status. ok means the license key is in memory. See SYST:LKEY command. The hexadecimal number in the preceding column is the license key itself. The option firmware must also be installed in memory.
Installed / Not Installed - indicates whether the option is installed/stored in the memory of the instrument. Use the firmware installation process for this. See www.agilent.com/find/vsa for more information.

Example: SYST:CONF?

Front Panel Access: **System, Show System**

Set Date

:SYSTem:DATE <year>,<month>,<day>

:SYSTem:DATE?

Sets the date of the real-time clock of the instrument.

Year - is a 4-digit integer

Month - is an integer from 1 to 12

Day - is an integer from 1 to 31 (depending on the month)

Example: SYST:DAT 2001,4,15

Front Panel Access: **System, Time/Date, Set Date**

Error Information Query

:SYSTem:ERRor[:NEXT]?

This command queries the earliest entry in the error queue and then deletes that entry. It can be used to continuously monitor the error queue for the occurrence of an error.

*CLS clears the entire error queue.

Example: SYST:ERR?

Front Panel Access: **System, Show Errors**

Locate SCPI Command Errors

:SYSTem:ERRor:VERBose OFF|ON|0|1

:SYSTem:ERRor:VERBose?

Adds additional information to the error messages returned by the SYSTem:ERRor? command. It indicates which SCPI command was executing when the error occurred and what about that command was unacceptable.

<error number>,"<error message>;<annotated SCPI command>"

Example: First set SYST:ERR:VERBOSE ON

If the command SENSe:FREQuently:CENTer 942.6MHz is sent, then sending SYST:ERR? returns:

```
-113,"Undefined header;SENSe:FREQuently:<Err>CENTer 942.6MHz $<NL>"
```

The <Err> shown after FREQuently shows you the spelling error. (The $<NL> is the typical representation for the command terminator.

If the command SENSe:FREQuency:CENTer 942.6Sec is sent, then sending SYST:ERR? returns:

```
-113,"Invalid suffix;SENSe:FREQuency:CENTer 942.6Sec<Err> $<NL>"
```

The <Err> shown after Sec shows you the invalid suffix.

Factory Preset and *RST: Off. This parameter is persistent, which means that it retains the setting previously selected, even through a

power cycle.

Remarks: The verbose SCPI error debugging state is global to all the SCPI interfaces.

History: Added version A.04.00

Front Panel Access: System, Show Errors, Verbose

Exit Main Firmware for Upgrade

:SYSTem:EXIT

Exit the main firmware to allow the firmware to be upgraded.

Example: SYST:EXIT

Front Panel Access: **System, Install, Exit Main Firmware**

Host Identification Query

:SYSTem:HID?

Returns a string that contains the host identification. This ID is required in order to obtain the license key that enables a new application (mode) or option.

Example: SYST:HID?

Front Panel Access: **System, Show System**

Keyboard Lock

:SYSTem:KLOCk OFF|ON|0|1

:SYSTem:KLOCk?

Disables the instrument keyboard to prevent local input when instrument is controlled remotely. An annunciator reading “Klock” alerts the local user that the keyboard is locked. Or you can display a system message using SYSTem:MESSage.

Example: SYST:CONF?

History: Added revision A.05.00

License Key for Installing New Applications

:SYSTem:LKEY <'option'>,<'license key'>

:SYSTem:LKEY? <'option'>

Enter the license key required for installing the specified new application (mode) or option. The query returns a string that contains the license key for a specified application or option that is already installed in the instrument. The license key will also be returned if the application is not currently in memory, but had been installed at some previous time.

Option – is a string that is the 3-character designation for the desired option. For example: BAC is the option for cdmaOne.

License key – is a 12 character alphanumeric string given to you with your option.

Example: `SYST:LKEY 'BAC','123A456B789C'`

Remarks: The license key is unique to the specific option installed in a particular instrument.

Front Panel Access: **System, Install, License Key**

Delete a License Key

:SYSTem:LKEY:DELete <'application option'>,<'license key'>

Allows you to delete the license key, for the selected application, from instrument memory.

NOTE If the license key is deleted, you will be unable to reload or update the application in instrument memory without re-entering the license key. The license key only works with one particular instrument serial number.

<application> - is a string that is the same as one of the enumerated items used in the INSTrument[:SELect] command.

<license key> - is a 12 character alphanumeric string given to you with your application

Front Panel Access: **None**

Remote Message

:SYSTem:MESSage <string>

Enables remote user to send message that will appear in the Status Bar at bottom of the instrument display. New message will overwrite any previous message. Message will remain until removed by use of :SYSTem:MESSage:OFF.

The SYSTem:KLOCk command will lock out the front-panel keys.

Example: `:SYSTem:MESSage "Instrument currently in use remotely by Ted in R+D"`

Remarks: Message appears as green text against a black background to differentiate it from internally generated messages which appear as white text against a black background.

History: Added revision A.05.00

Remote Message Turned Off

:SYSTem:MESSage:OFF

Removes any system message from the Status Bar at the bottom of the instrument display. A message can be displayed using the :SYSTem:MESSage command.

Example: `:SYSTem:MESSage:OFF`

History: Added revision A.05.00

Service Password

:SYSTem:PASSword[:CENable]<integer>

Enables access to the service functions by means of the password.

Front Panel Access: **System, Show System, Service Password**

Preset

:SYSTem:PRESet

Returns the instrument to a set of defined conditions. This command does not change any persistent parameters.

Front Panel

Access: **Preset**

Preset Type

Preset and *RST: Factory - This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.

Remarks: `:SYST:PRES:USER:SAVE` defines the *user preset*.

Example: `SYST:PRES:TYPE FACT`

Front Panel Access: **System, Pwr On/Preset, Preset Factory User**

Set Time

`:SYSTem:TIME <hour>,<min>,<sec>`

`:SYSTem:TIME?`

Sets the time of the real-time clock of the instrument.

Hour must be an integer from 0 to 23.

Minute must be an integer from 0 to 59.

Second must be an integer from 0 to 59.

Front Panel Access: **System, Time/Date, Set Time**

Adjust Time

`:SYSTem:TIME:ADJust <seconds>`

Adjust the instruments internal time by the value entered.

Range: Larger than you should ever need

Example: `SYST:TIME:ADJ 3600` will advance the time one hour.

`SYST:TIME:ADJ -86400` will back the date up one day, without changing the time of day (minutes or seconds).

History: In revision A.02.00 and later

Default Unit: seconds

SCPI Version Query

:SYSTem:VERSion?

Returns the SCPI version number with which the instrument complies.

TRIGger Subsystem

The Trigger Subsystem is used to set the controls and parameters associated with triggering the data acquisitions. Other trigger-related commands are found in the INITiate and ABORt subsystems.

The trigger parameters are global within the selected Mode. The commands in the TRIGger subsystem set up the way the triggers function, but selection of the trigger source is made from each measurement. There is a separate trigger source command in the SENSe:<meas> subsystem for each measurement. The equivalent front-panel keys for the parameters described in the following commands, can be found under the **Mode Setup, Trigger** key.

Automatic Trigger Control

:TRIGger[:SEQuence]:AUTO:STATe OFF|ON|0|1

:TRIGger[:SEQuence]:AUTO:STATe?

Turns the automatic trigger function on and off. This function causes a trigger to occur if the designated time has elapsed and no trigger occurred. It can be used with unpredictable trigger sources, like external or burst, to make sure a measurement is initiated even if a trigger doesn't occur. Use TRIGger[:SEQuence]:AUTO[:TIME] to set the time limit.

Factory Preset and *RST — Off for cdma2000, W-CDMA (3GPP), NADC, and PDC

Front Panel Access — **Mode Setup, Trigger, Auto Trig**

Automatic Trigger Time

:TRIGger[:SEQuence]:AUTO[:TIME] <time>

:TRIGger[:SEQuence]:AUTO[:TIME]?

After the measurement is activated the instrument will take a data acquisition immediately upon receiving a signal from the selected trigger source. If no trigger signal is received by the end of the time specified in this command, a data acquisition is taken anyway. TRIGger[:SEQuence]:AUTO:STATE must be on.

Factory Preset and *RST: — 100.0 ms

Range: — 1.0 ms to 1000.0 s

0.0 to 1000.0 s for cdma2000, W-CDMA (3GPP)

Default Unit: seconds

External Trigger Delay

:TRIGger[:SEQuence]:EXTernal[1]|2:DELay <time>

:TRIGger[:SEQuence]:EXTernal[1]|2:DELay?

Set the trigger delay when using an external trigger. Set the trigger value to zero (0) seconds to turn off the delay.

EXT or EXT1 is the front panel trigger input

EXT2 is the rear panel trigger input

Factory Preset and *RST: 0.0 s

Range: –500.0 ms to 500.0 ms

–100.0 ms to 500.0 ms for cdma2000, W-CDMA (3GPP)

Default Unit: seconds

Front Panel Access: **Mode Setup, Trigger, Ext Rear (or Ext Front), Delay**

External Trigger Level

:TRIGger[:SEQuence]:EXTernal[1]|2:LEVel <voltage>

:TRIGger[:SEQuence]:EXTernal[1]|2:LEVel?

Set the trigger level when using an external trigger input.

EXT or EXT1 is the front panel trigger input

EXT2 is the rear panel trigger input

Factory Preset and *RST: 2.0 V

Range: –5.0 to +5.0 V

Default Unit: volts

Front Panel Access: **Mode Setup, Trigger, Ext Rear, Level**

Mode Setup, Trigger, Ext Front, Level

External Trigger Slope

:TRIGger[:SEQuence]:EXTernal[1]|2:SLOPe NEGative|POSitive

:TRIGger[:SEQuence]:EXTernal[1]|2:SLOPe?

Sets the trigger slope when using an external trigger input.

EXT or EXT1 is the front panel trigger input

EXT2 is the rear panel trigger input

Factory Preset and *RST: Positive

Front Panel Access: **Mode Setup, Trigger, Ext Rear (or Ext Front), Slope**

Frame Trigger Adjust

:TRIGger[:SEQuence]:FRAMe:ADJust <time>

Lets you advance the phase of the frame trigger by the specified amount. It does not change the period of the trigger waveform. If the command is sent multiple times, it advances the phase of the frame trigger more each time it is sent.

Factory Preset and *RST: 0.0 s

Range: 0.0 to 10.0 s

Default Unit: seconds

Front Panel Access: None

Frame Trigger Period

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

:TRIGger[:SEQuence]:FRAMe:PERiod?

Set the frame period that you want when using the external frame timer trigger. If the traffic rate is changed, the value of the frame period is initialized to the preset value.

Factory Preset and *RST: 250.0 µs for Basic, cdmaOne

4.615383 ms, for GSM

26.666667 ms for cdma2000

10.0 ms (1 radio frame) for W-CDMA (3GPP)

90.0 ms for iDEN

20.0 ms with rate = full for NADC, PDC

40.0 ms with rate = half for NADC, PDC

Range: 0.0 ms to 559.0 ms for Basic, cdmaOne, GSM, cdma2000, W-CDMA (3GPP)

1.0 ms to 559.0 ms for iDEN, NADC, PDC

Default Unit: seconds

Front Panel Access: **Mode Setup, Trigger, Frame Timer, Period**

Frame Trigger Sync Mode

:TRIGger[:SEQuence]:FRAMe:SYNC EXTFront|EXTRear|OFF

:TRIGger[:SEQuence]:FRAMe:SYNC?

Selects the input port location for the external frame trigger that you are using.

Factory Preset and *RST: Off

Remarks: You must be in the Basic, cdmaOne, EDGE (w/GSM), GSM, iDEN, NADC, PDC, Service mode to use this command. Use INSTrument:SELect to set the mode.

Front Panel Access: **Mode Setup, Trigger, Frame Timer, Sync Source**

Frame Trigger Synchronization Offset

:TRIGger[:SEQuence]:FRAMe:SYNC:OFFSet <time>

:TRIGger[:SEQuence]:FRAMe:SYNC:OFFSet?

Lets you adjust the frame triggering with respect to the external trigger input that you are using.

Factory Preset and *RST: 0.0 s

Range: 0.0 to 10.0 s

Default Unit: seconds

Remarks: You must be in the Basic, cdmaOne, EDGE (w/GSM), GSM, iDEN, NADC, PDC, Service mode to use this command. Use INSTrument:SELect to set the mode.

History: Revision A.03.27 or later

Front Panel Access: **Mode Setup, Trigger, Frame Timer, Offset**

Trigger Holdoff

`:TRIGger[:SEQuence]:HOLDoff <time>`

`:TRIGger[:SEQuence]:HOLDoff?`

Set the holdoff time between triggers. After a trigger, another trigger will not be allowed until the holdoff time expires. This parameter affects all trigger sources.

Factory Preset and *RST: 0.0 s

20.0 ms for iDEN

10.0 ms for NADC or PDC

Range: 0.0 to 500.0 ms

Default Unit: seconds

Front Panel Access: **Mode Setup, Trigger, Trig Holdoff**

Video (IF) Trigger Delay

`:TRIGger[:SEQuence]:IF:DELay <time>`

`:TRIGger[:SEQuence]:IF:DELay?`

Set the trigger delay when using the IF (video) trigger (after the Resolution BW filter).

Factory Preset and *RST: 0.0 s

Range: –500.0 ms to 500.0 ms

–100.0 ms to 500.0 ms for cdma2000, W-CDMA (3GPP)

Default Unit: seconds

Front Panel Access: **Mode Setup, Trigger, Video (IF Envlp), Delay**

Video (IF) Trigger Level

:TRIGger[:SEQuence]:IF:LEVel <power>

:TRIGger[:SEQuence]:IF:LEVel?

Set the trigger level when using the IF (video) trigger.

Factory Preset and *RST: –6.0 dBm for cdmaOne, GSM, Basic, Service, cdma2000, W-CDMA (3GPP)

–20.0 dBm for iDEN

–30.0 dBm for NADC, PDC

Range: –200.0 to 50.0 dBm

Default Unit: dBm

Front Panel Access: **Mode Setup, Trigger, Video (IF Envlp), Level**

Video (IF) Trigger Slope

:TRIGger[:SEQuence]:IF:SLOPe NEGative|POSitive

:TRIGger[:SEQuence]:IF:SLOPe?

Sets the trigger slope when using the IF (video) trigger.

Factory Preset and *RST: Positive

Front Panel Access: **Mode Setup, Trigger, Video (IF Envlp), Slope**

RF Burst Trigger Delay

:TRIGger[:SEQuence]:RFBurst:DELay <time>

:TRIGger[:SEQuence]:RFBurst:DELay?

Set the trigger delay when using the RF burst (wideband) trigger.

Factory Preset and *RST: 0.0 s

Range: –500.0 ms to 500.0 ms

–100.0 ms to 500.0 ms for cdma2000, or W-CDMA (3GPP)

Default Unit: seconds

Front Panel Access: **Mode Setup, Trigger, RF Burst, Delay**

RF Burst Trigger Level

:TRIGger[:SEQuence]:RFBurst:LEVel <rel_power>

:TRIGger[:SEQuence]:RFBurst:LEVel?

Set the trigger level when using the RF Burst (wideband) Trigger. The value is relative to the peak of the signal. RF Burst is also known as RF Envelope.

Factory Preset and *RST: –6.0 dB

Range: –25.0 to 0.0 dB

–200.0 to 0.0 dB for NADC, PDC

Default Unit: dB

Front Panel Access: **Mode Setup**, **Trigger, RF Burst, Peak Level**

RF Burst Trigger Slope

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

:TRIGger[:SEQuence]:RFBurst:SLOPe?

Set the trigger slope when using the RF Burst (wideband) Trigger.

Factory Preset and *RST: Positive

Remarks: You must be in the cdmaOne, cdma2000, or W-CDMA (3GPP) mode to use this command. Use :INSTrument:SELect to set the mode.

Front Panel Access: **Mode Setup, Trigger, RF Burst, Slope**

Index

Index

Index

Index

Index