Programmer's Guide

Agilent Technologies E4406A VSA Series Transmitter Tester

Agilent Technologies

Manufacturing Part Number: E4406-90135

Printed in USA

February 2000

Safety Information

The following safety notes are used throughout this manual. Familiarize yourself with each of the notes and its 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.

WARNING **This is a Safety Class 1 Product (provided with a protective earthing ground incorporated in the power cord). The mains plug shall only be inserted 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 **These servicing instructions are for use by qualified personnel only. To avoid electrical shock, do not perform any servicing unless you are qualified to do so.**

WARNING **The power cord is connected to internal capacitors that may remain live for 5 seconds after disconnecting the plug from its power supply.**

Warranty

This Agilent Technologies instrument product is warranted against defects in material and workmanship for a period of one year from date of shipment. During the warranty period, Agilent Technologies Company will, at its option, either repair or replace products which prove to be defective.

For warranty service or repair, this product must be returned to a service facility designated by Agilent Technologies. Buyer shall prepay shipping charges to Agilent Technologies and Agilent Technologies shall pay shipping charges to return the product to Buyer. However, Buyer shall pay all shipping charges, duties, and taxes for products returned to Agilent Technologies from another country.

Agilent Technologies warrants that its software and firmware designated by Agilent Technologies for use with an instrument will execute its programming instructions when properly installed on that instrument. Agilent Technologies does not warrant that the operation of the instrument, or software, or firmware will be uninterrupted or error-free.

LIMITATION OF WARRANTY

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by Buyer, Buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside of the environmental specifications for the product, or improper site preparation or maintenance.

NO OTHER WARRANTY IS EXPRESSED OR IMPLIED. AGILENT TECHNOLOGIES SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

EXCLUSIVE REMEDIES

THE REMEDIES PROVIDED HEREIN ARE BUYER’S SOLE AND EXCLUSIVE REMEDIES. AGILENT TECHNOLOGIES SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER BASED ON CONTRACT, TORT, OR ANY OTHER LEGAL THEORY.

Contents

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 35.

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
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 17, if your measurement mode is not installed.

The SYSTem:HELP:HEADers? command provides a list of all the commands available in the mode you have currently selected. The programming commands for each optional mode are documented separately. The specific measurements available in a particular mode are indicated below.

Table 1-1 Available Modes and Measurements

Modes	**Measurement Keywords**
Basic - standard	• ACP - adjacent channel power measurement • CHPower - channel power measurement • SPECtrum - spectrum (frequency domain) measurement • WAVeform - waveform (time domain) measurement
cdmaOne - Option BAC	• 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
cdma2000 - Option B78	• ACP - adjacent channel power ratio measurement • CHPower - channel power measurement • PSTATistic - power statistics (CCDF) measurement • EVMQpsk - QPSK error vector magnitude measurement • RHO - rho (waveform quality) measurement • SPECtrum - spectrum (frequency domain) measurement • WAVeform - waveform (time domain) measurement

Table 1-1 Available Modes and Measurements

Modes	Measurement Keywords
W-CDMA - Option BAF	• 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 - rho (waveform quality) measurement • SPECtrum - spectrum (frequency domain) measurement • WAVeform - waveform (time domain) measurement
GSM - Option BAH	• ORFSpectrum - output RF spectrum measurement • PFERror - phase and frequency error measurement • PVTime - power versus time measurement • SPECtrum - spectrum (frequency domain) measurement • TXPower - transmit power measurement • WAVeform - waveform (time domain) measurement
PDC and NADC - Option BAE	• 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	• 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	• 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 follow a two step process.

1. The measurement personality firmware must be installed into the instrument. (See the supplied installation instructions.)
2. A license key number must be entered which enables the measurement personality to run. (Refer to the “License Key Numbers” section below.)

Adding additional measurement personalities requires purchasing a retrofit kit for the desired option. The retrofit kit includes the measurement personality firmware, usually supplied on a zip disk. The license key certificate, included in the kit, contains the license key number. Every retrofit kit will have installation instructions.

The installation instructions require you to know three pieces of information about your instrument: the amount of memory installed, the Host ID, and the instrument serial number.

Required information:	Key Path:
Instrument Memory: ________________	**System**, **File System** (the amount of memory in your instrument will be the sum of the Used memory and the Free memory)
Host ID: ________________	**System**, **Show System**, **Host ID**
Instrument Serial Number: ________________	**System**, **Show System**, **Serial Number**

The **Exit Main Firmware** key is used during the firmware installation process. This key is only for use when you want to update firmware using a LAN connection. The **Exit Main Firmware** key halts the operation of the resident firmware code so you can install an updated version of firmware using a LAN connection. Instructions for loading future firmware updates are available at the following URL: **www.agilent.com/find/vsa/**

Available Personality Options

The option designation consists of three characters, as shown in the

Option column of the table below.

Available Personality Options[a]	**Option**
GSM with EDGE, measurement personality	**BAH**
cdmaOne measurement personality	**BAC**
NADC, PDC measurement personalities	**BAE**
iDEN measurement personality	**HN1**
W-CDMA measurement personality	**BAF**
cdma2000 measurement personality	**B78**

a. As of the print date of this measurement guide.

License Key Numbers

The measurement personality you have purchased with your instrument has been installed and enabled at the factory. With the purchase of the measurement personality, and with any future purchase of a new personality, you will receive a unique license key number. The license key number is a hexadecimal number that is for your specific measurement personality and instrument serial number. The license key enables you to install, or reactivate any personality you have purchased.

Follow these steps to locate the unique license key number for the measurement personality that has come installed in your instrument:

1. Press **System**, **More (1 of 3)**, **More (2 of 3)**, **Install**, **Choose Option.** When you press the **Choose Option** key the alpha editor will be activated. Use the alpha editor to enter the letters (upper-case) and the front-panel numeric keyboard to enter the numbers (if required) for the personality option that has been installed in the instrument.

2. Press the **Done** key on the alpha editor menu. The unique license key number for your instrument will now appear on the **License Key** softkey.

You will want to keep a copy of your license key number in a secure location. Please enter your license key numbers in the box provided below for future reference. If you should lose your license key number, call your nearest Agilent Technologies service or sales office for assistance.

License Key Numbers for Instrument with Serial # ________
For Option______________ the license key number is ____________________
For Option______________ the license key number is ____________________
For Option______________ the license key number is ____________________
For Option______________ the license key number is ____________________
For Option______________ the license key number is ____________________
For Option______________ the license key number is ____________________

If you purchase an option later, you will receive a certificate which displays the unique license key number that you will need to install that option.

NOTE You will need to use a license key number only if you purchase an additional measurement personality, or if you want to reactivate a measurement personality that has been deactivated.

Installing a License Key Number

NOTE Follow this procedure to reinstall a license key number which has been deleted during the uninstall process, or lost due to a memory failure.

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

1. Press **System**, **More(1 of 3)**, **More(2 of 3)**, **Install**, **Choose Option**. Pressing the **Choose Option** key will activate the alpha editor menu. Use the alpha editor to enter the letters (upper-case) and the front-panel numeric keyboard to enter the numbers (if required) for the option designation, then press the **Done** key. As you enter the option, you will see your entry in the active function area of the display.

2. Press **License Key**. Entering the license key number will require entry of both letters and numbers. Use the alpha editor to enter letters. Use the front-panel numeric keyboard to enter numbers. You will see your entry in the active function area of the display. When you have completed entering the license key number, press the **Done** key.

3. Press the **Install Now** key after you have entered the active license key number and the personality option. When pressed, a message may appear in the function area of the display which reads, “`Insert disk and power cycle the instrument`”. Disregard this message. Press the **No** key only if you wish to cancel the installation process. If you want to proceed with the installation, press the **Yes** key and cycle the instrument power off and then on.

Using the Uninstall Key

The following procedure removes the license key number for the selected option. This will make the option unavailable for use, and the message “`Application Not Licensed`” will appear in the Status/Info bar at the bottom of the display. Please write down the 12-digit license key number for the option before proceeding. If that measurement personality is to be used at a later date you will need the license key number to reactivate the personality firmware.

NOTE Using the **Uninstall** key does not remove the personality 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 located at the URL: **www.agilent.com/find/vsa/**

1. Press **System**, **More(1 of 3)**, **More(2 of 3)**, **Uninstall**, **Choose Option**. Pressing the **Choose Option** key will activate the alpha editor menu. Use the alpha editor to enter the letters (upper-case) and the front-panel numeric keyboard to enter the numbers (if required) for the option, then press the **Done** key. As you enter the option, you will see your entry in the active function area of the display.

2. Press the **Uninstall Now** key after you have entered the personality option. Press the **No** key only if you wish to cancel the uninstall process. Press the **Yes** key if you want to continue the uninstall process.

3. Cycle the instrument power off and then on to complete the uninstall process.

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.

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/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>:AVER) • Bandwidth (SENSe:<meas>:BWID) • 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.
- In the program, execute an instrument preset (*RST) and select

single-sweep mode (INITiate:CONTinuous OFF) before setting other instrument functions.

- Select the instrument mode with INST:SELect. Set the mode setup for things like your desired communications standard, channel frequency and triggering.
- Use the MEASure group of commands, described in Chapter 5 , "Language Reference". 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 35.

File Naming Rules

File names for storing states, traces, limit lines or amplitude correction data files in the analyzer should follow the pc conventions as indicated below:

- 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 type them.
- They can contain only the letters A through Z, the number 0 through 9, and the following special characters:

Character[a]	**Description**
_	underscore
^	carat
$	dollar sign
~	tilde
!	exclamation point
#	number sign
%	percent sign
&	ampersand

Character[a]	Description
-	hyphen
{}	braces
@	at sign
‘	single quotation mark
’	apostrophe
()	parenthesis

a. No other characters are valid.

- They cannot contain spaces, commas, backslashes, or periods (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 34399A RS-232 Adapter Kit.
HP/Agilent 34399A RS-232 Adapter Kit	This kit includes four adapters to go from DB9 female cable (34398A) to PC/printer DB25 male or female, or to modem DB9 female or DB25 female.

Figure 1-1 HP/Agilent 24542U Cable

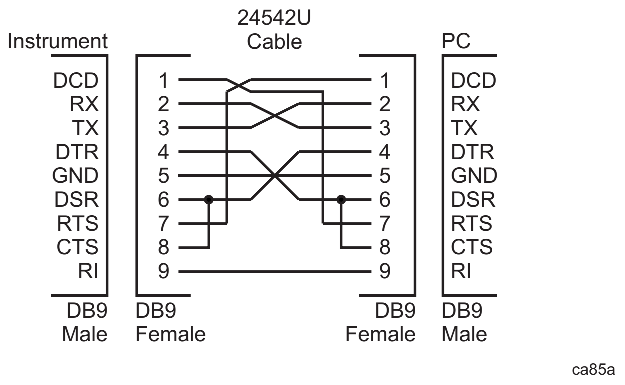


Figure 1-2 HP/Agilent F1047-80002 Cable

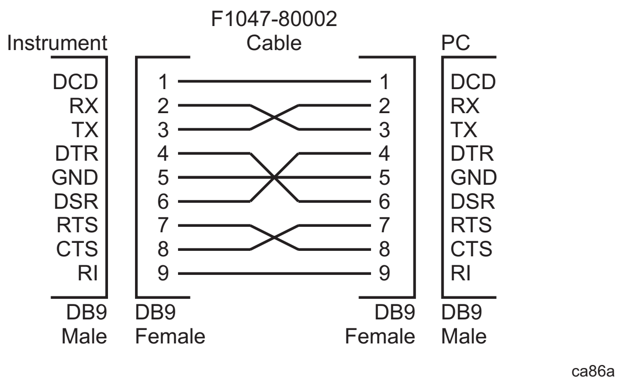


Figure 1-3 HP/Agilent 24542G/H Cable

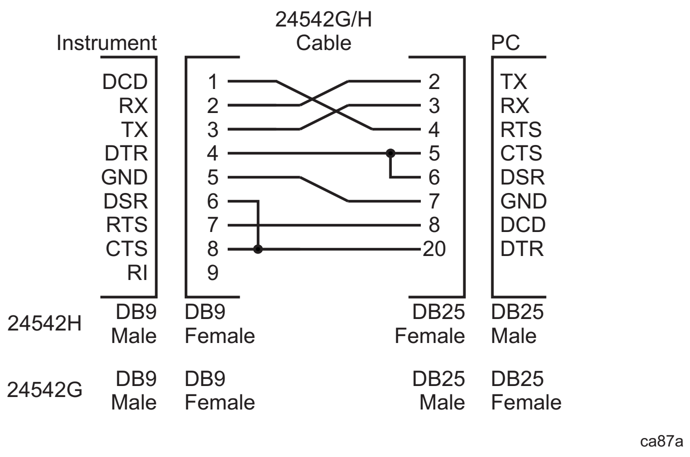


Figure 1-4 HP/Agilent 92219J Cable

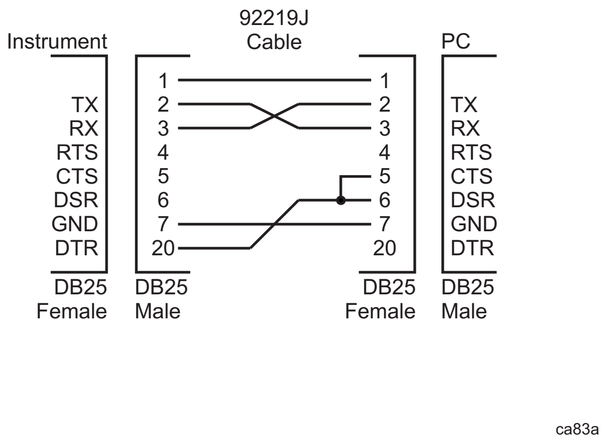


Figure 1-5 **HP/Agilent 13242G Cable**

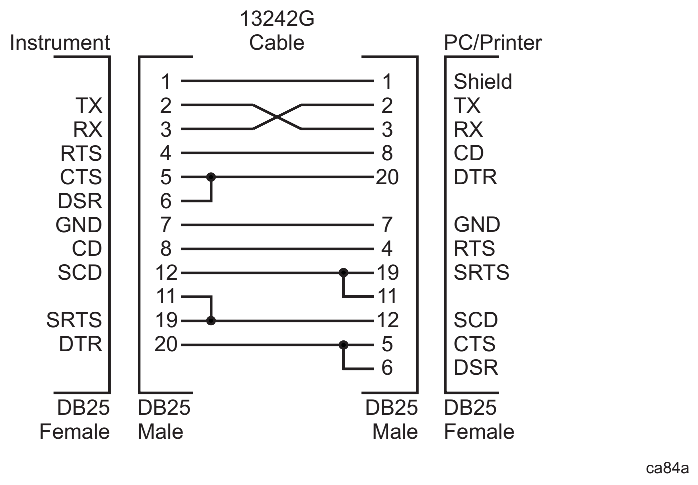


Figure 1-6 **HP/Agilent 24542M Modem Cable**

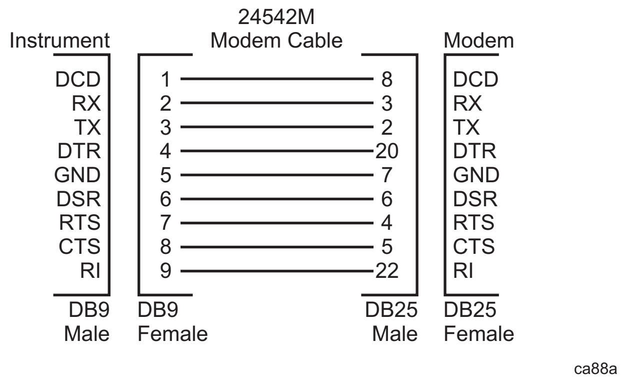


Figure 1-7 HP/Agilent C2913A/C2914A Cable

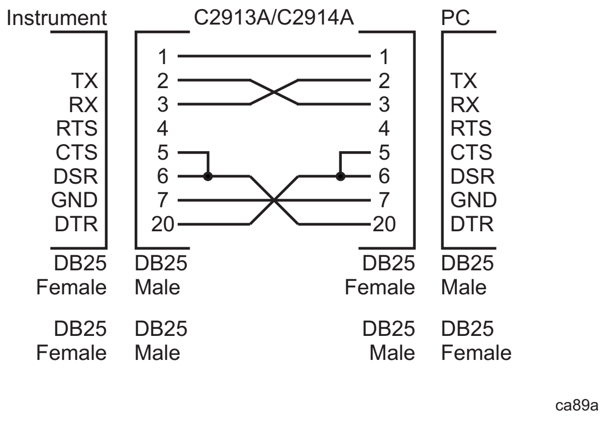


Figure 1-8 Mouse Adapter (typical)

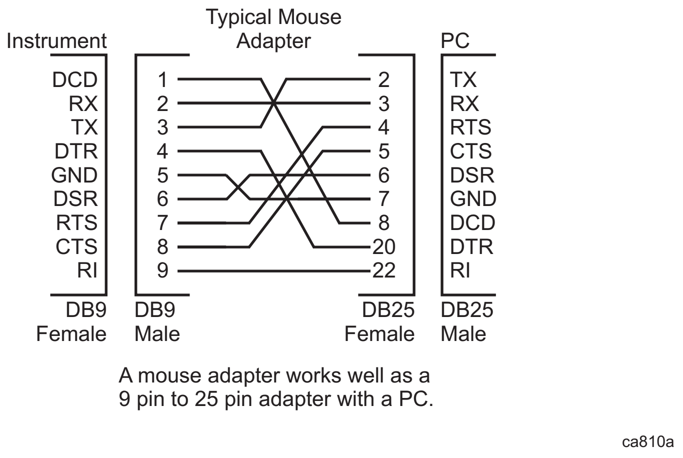


Figure 1-9 HP/Agilent 24542U Cable with 5181-6641 Adapter

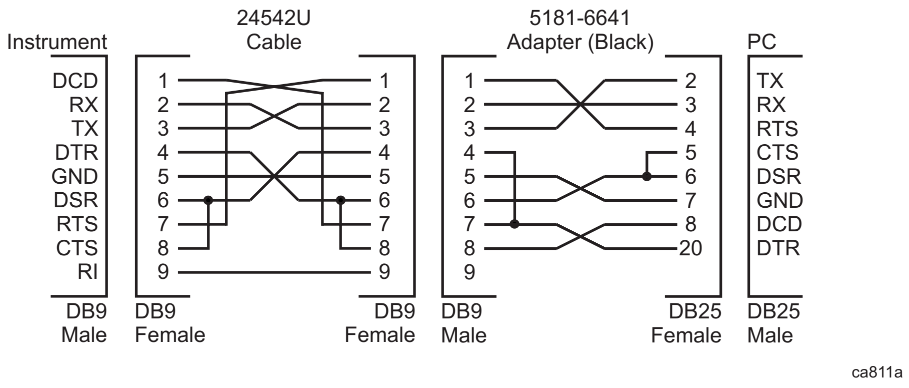


Figure 1-10 HP/Agilent 24542U Cable with 5181-6640 Adapter

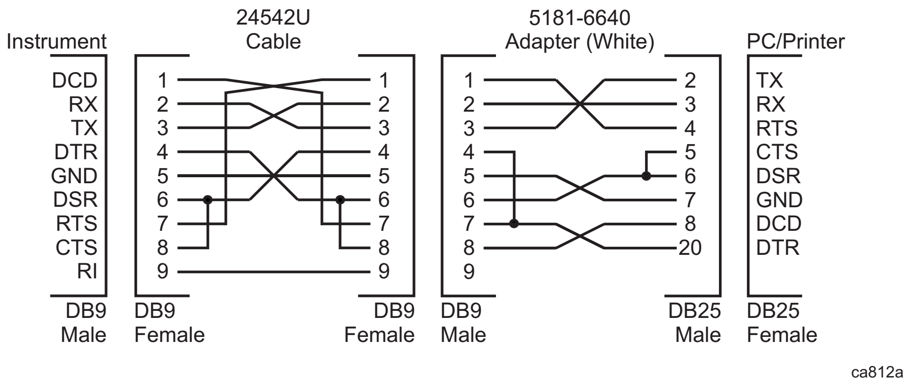


Figure 1-11 HP/Agilent 24542U Cable with 5181-6642 Adapter

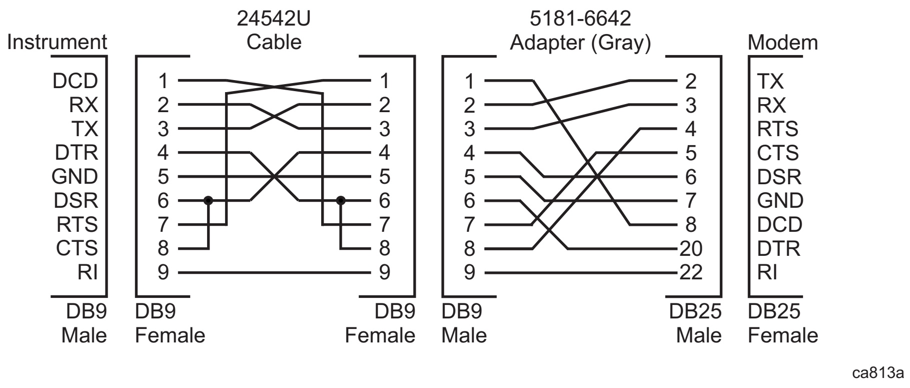


Figure 1-12 HP/Agilent 24542U Cable with 5181-6639 Adapter

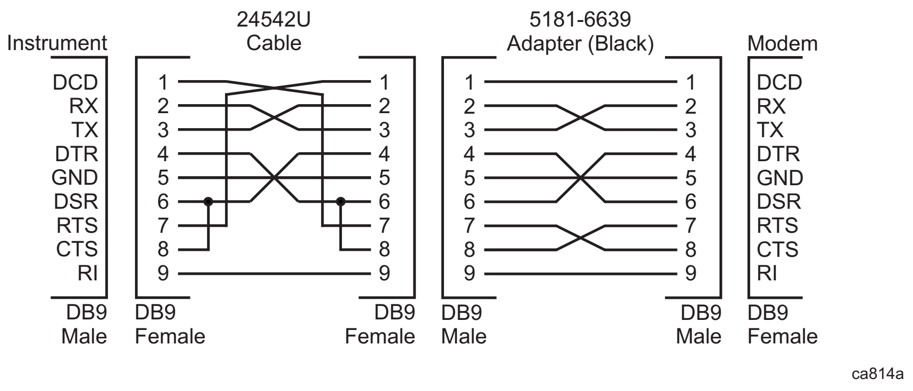


Figure 1-13 HP/Agilent F1047-80002 Cable with 5181-6641 Adapter

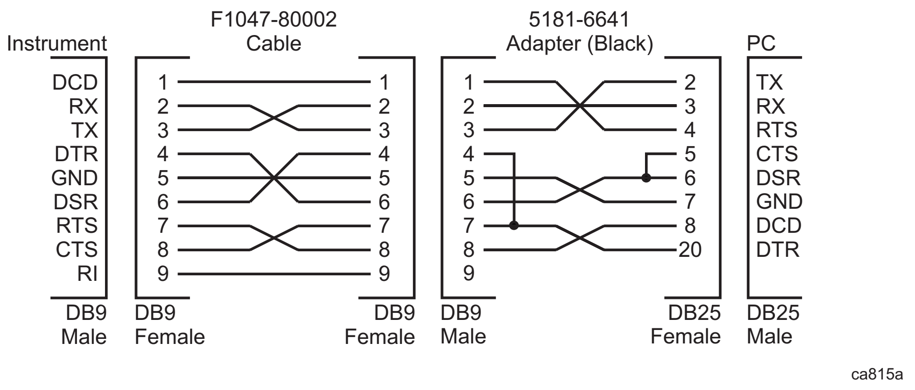


Figure 1-14 HP/Agilent F1047-80002 Cable with 5181-6640 Adapter

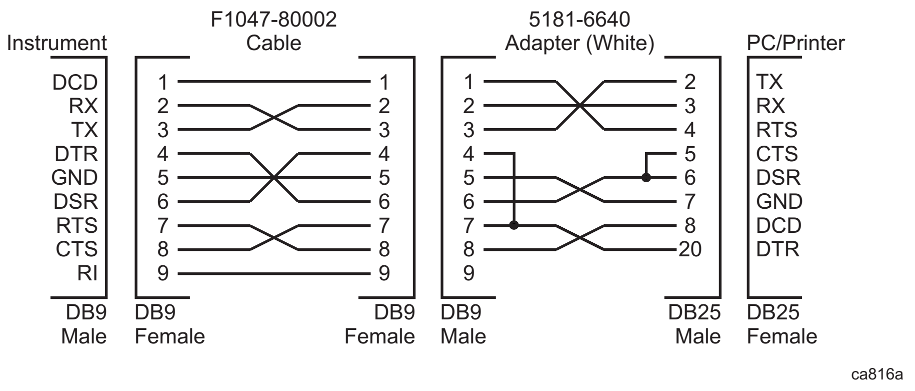


Figure 1-15 HP/Agilent F1047-80002 Cable with 5181-6642 Adapter

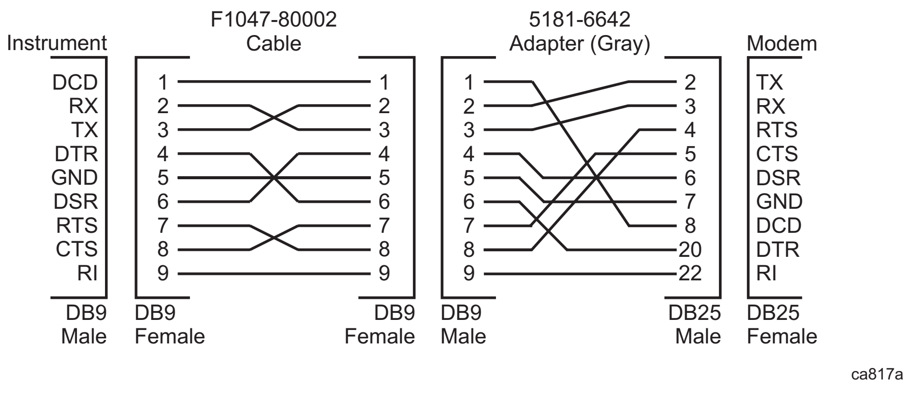
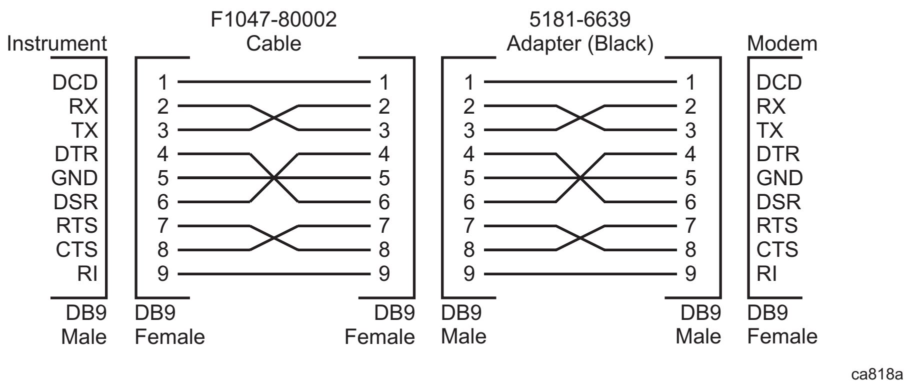


Figure 1-16 HP/Agilent F1047-80002 Cable with 5181-6639 Adapter



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

The various LAN settings can be queried and set from the front panel key menus found by pressing **System, Config I/O** and then pressing the appropriate keys. Configuration of the LAN can only be done from the front panel. There are no equivalent remote commands. 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”.

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 don’t accidentally send commands to your analyzer in the middle of the program execution.

Pressing **Preset** will not change the LAN configuration settings, since they are persistent, they will stay at the last user setting. However, you can return the instrument to it’s 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**.

2 Programming Fundamentals

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 detailed information.

Topics covered in this chapter include:

- "Creating Valid Commands" on page 35
- "Command Key Words and Syntax" on page 36
- "Special Characters in Commands" on page 37
- "Parameters in Commands" on page 37
- "Putting Multiple Commands on the Same Line" on page 40

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.

There is also a book available from Agilent Technologies, *SCPI—Standard Commands for Programmable Instruments*, 1998.

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`

Command Syntax	Sample Valid Commands
`MEASure:SPECtrum[n]? [<freq>[,<level>[,<freq(span)>]]]`	The last command below returns different results then the command above it. The number 3, in the command, causes this. See the command description for more information. • `MEAS:SPEC?` • `Meas:spec?` • `meas:spec3?`
`[SENSe:]CPOWer:AVERage:TCONtrol EXPonential|NORMal|REPeat`	• `CPOW:AVER:TCON EXP` • `CPOWER:aver:tcon Normal`
`INITiate:CONTinuous ON|OFF|1|0`	The sample commands below are identical. • `INIT:CONT ON` • `init:continuous 1`

Command Key Words and Syntax

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

Example: `TRIGger:LEVel:EXT1 2.5V`

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

Example: `Trig:LEVEL:EXT1` 2 is the same as `trigger:Lev:ext1 2v`.

NOTE The command `TRIGG:level:EXT1` is not valid because `trigg` is neither the short, nor the long form of the command. Only the short and long forms of the key words are allowed in valid commands.

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: `TRIG:SOURCE EXT|INT|LINE` The choices are ext, int, and line. `TRIG:SOURCE INT` is one possible command choice.
	A vertical stroke between **key words** indicates identical effects exist for several key words. Only one of these key words is used at a time. The command functions the same for either key word.	Command: `SENSe:BANDwidth|:BWIDth:OFFSet` Two identical commands are: `SENSE:BWIDTH:OFFSET` `SENSE:BAND:OFFSET`
[ ]	Key words in square brackets are optional when composing the command. These implied key words will be executed even if they are omitted.	Command: `[SENSe:]BANDwidth[:RESolution]:AUTO` The following commands are all valid and have identical effects: `bandwidth:auto` `bandwidth:resolution:auto` `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, `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 {level}` A valid command might be `measure:bw 3dB 60dB`

Parameters in Commands

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

ON|OFF|0|1 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, and queries of the parameter always return a numeric value of 0 or 1.
Key Word	The parameter key words that are allowed for a particular command are defined in the command description and are separated with a vertical slash.
Units	Numerical variables may include units. The valid units for a command depends on the variable type being used. See the following variable descriptions. If no units are sent, the indicated default units will be used. Units can follow the numerical value with, or without, a space.
Variable	A variable can be entered in exponential format as well as standard numeric format. The appropriate range of the variable and it's optional units, are defined in the command description.

In addition to these values, the following key words may also be used in commands where they are applicable.

DEFault - resets the parameter to its default value.

UP - increments the parameter.

DOWN - decrements the parameter.

The numeric value associated with DEFault can be queried by adding the key word to the command in its query form. The key word must be entered following the question mark.

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

Variable Parameters

<freq> <bandwidth>	A frequency parameter is a positive rational number followed by optional units. The default unit is Hz. Acceptable units include: HZ, KHZ, MHZ, GHZ.
<time> <seconds>	A time parameter is a rational number followed by optional units. The default units are seconds. Acceptable units include: S, MS, US.
<voltage>	A voltage parameter is a rational number followed by optional units. The default units are V. Acceptable units include: Volts, V, MV, UV.
<power> <ampl>	A power parameter is a rational number followed by

optional units. The default units are dBm. Acceptable units include: DBM, DBMV, W.

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

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

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

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

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

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> A trace parameter 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.

<bit_pattern> A bit pattern parameter is a series of bits. These can be sent in binary format or as a single hexadecimal number. There are no units.

<arbitrary block data> This parameter type consists of a series of rational numbers. The first number sent in the block is an integer indicating how many numbers are in the series. There are no units.

For example, suppose the header is 512320.

- The first byte in the header (5) tells you how many additional 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/point), either 8 (for real 64), or 4 (for real 32). If you're using real 64, 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.

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:

```
INPut
     :ATTenuation

TRIGger
     :AUTO
     :HOLDoff
     :LEVel
          :ADC
         [:EXTernal]
          :BURSt

[SENSe]
     :FREQuency
     :POWer
          :ATTenuation
         [:LEVel]
              [:IMMediate]
                   [:AMPLitude]
                    :OFFSet
                    :HIGH
                    :LOW
               :TRIGgered
```

Bad Command	**Good Command**
`IN:ATT 30dB`	`INP:ATT 30dB`
The short form of INPUT is INP, not IN.	

Bad Command	Good Command
`FREQ 30MHz;ATT 20dBm`	`FREQ 30MHz;POW:ATT 20dBm`
The ATT command is in the same (SENSE) subsystem as FREQ, but executing the FREQ command puts you back at the SENSE level. You must specify POWER to get to the ATT command.	
`FREQ 30MHz;POW:ATT 20dB:LOW 3dBm`	`FREQ 30MHz;POW:ATT 20dB;LOW 3dBm`
ATT and LOW are both in the same (SENSE) subsystem but they are two separate commands and need a semicolon to separate them.	
`INP:ATT 6dB;:TRIG:POWER:ATT 6 dB`	`INP:ATT 6dB;:POWER:ATT 6 dB`
POWER:ATT is in the SENSE subsystem, not the TRIGGER subsystem.	
`FREQ 10MHz;TRIG:HOLD 10ms`	`FREQ 10MHz;:TRIG:HOLD 10ms`
The FREQ command and the HOLDOFF command are not in the same subsystem, so the command that follows must be preceded by a colon.	
`POW?:FREQ?`	`POW?;FREQ?`
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.	
`INP:ATT -5dB;:FREQ 10MHz`	`INP:ATT 5dB;:FREQ 10MHz`
Attenuation should not be a negative value.	

Using the Instrument Status Registers

You can determine the state of certain instrument hardware and firmware events and conditions by programming the status register system. The Figure on page 49 shows all the instrument status registers and their hierarchy. The IEEE common (*) commands access the higher-level summary registers. To access the status information from specific registers you would use the STATus commands in Chapter 5 , “Language Reference.”

- “What are the Status Registers?” on page 43
- “Why Would You Use the Status Registers?” on page 45
- “Using a Status Register” on page 47
- “Using the Service Request (SRQ) Method” on page 47
- “Overall Status Register System” on page 49
- “Standard Event Status Register” on page 53
- “Operation and Questionable Status Registers” on page 56

What are the Status Registers?

The status system is comprised of multiple registers which 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.

Most status registers are actually a family of five registers:

Condition	A condition register continuously monitors the hardware and firmware status of the instrument. There is no latching or buffering for a condition register. It is updated in real time.
Negative Transition	A negative transition register specifies the bits in the condition register that will set corresponding bits in the event register when the condition bit changes from 1 to 0.
Positive Transition	A positive transition register specifies the bits in the condition register that will set corresponding bits in the event register when the condition bit changes from 0 to 1.
Event	An event register latches transition events from the condition register as specified by the positive and negative transition filters. Bits in the event register are latched, and once set, they remain set until cleared by either querying the register contents or sending the `*CLS` command.
Event Enable	An enable register specifies the bits in the event register that can generate a summary bit. Summary bits are, in turn, used by the next higher register.

What are the Status Register SCPI Commands

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

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

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

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

*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’.

*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.

Why Would You Use the Status Registers?

Your 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 analyzer 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 for the occurrence of a particular condition (bit).

  Once you have enabled a bit, using the event enable register, the instrument will monitor that particular bit. If the bit becomes true 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, which clears all event registers.

- Monitor any changes in a particular condition (bit).

  Once you have enabled a bit, the instrument will monitor it for a change in its condition. The transition registers are preset to register the conditions going from 0 to 1, positive transitions. This can be changed so that the selected bit is detected if it goes from true to false (negative transition), or if either transition occurs.

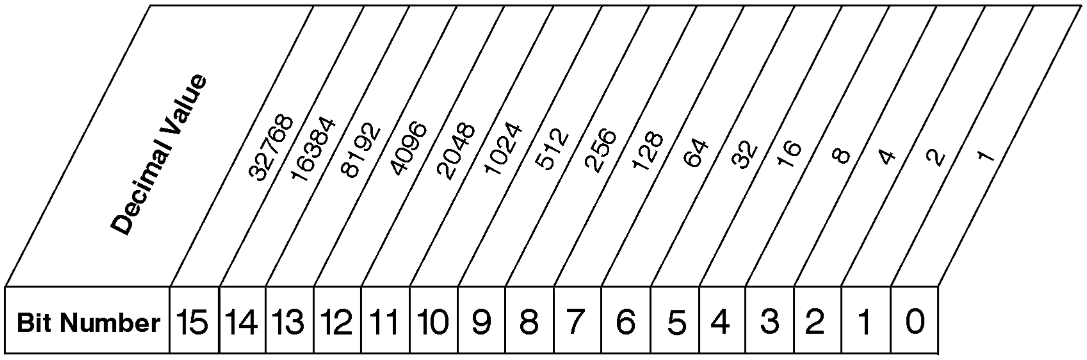
Using a Status Register

Each bit in a register is represented by a numerical value based on its location. See Figure 2-1 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 are interested in.

For example, to enable bit 0 and bit 6 of standard event status register, you would send the command *ESE 65 (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 the bit 7 is true, bit 3 is true and bit 2 is true.

Figure 2-1 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.) 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. The time the controller would otherwise have used to monitor the condition can now 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 RQS bit is set whenever something (that it has been configured to report using *SRE) changes. 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.

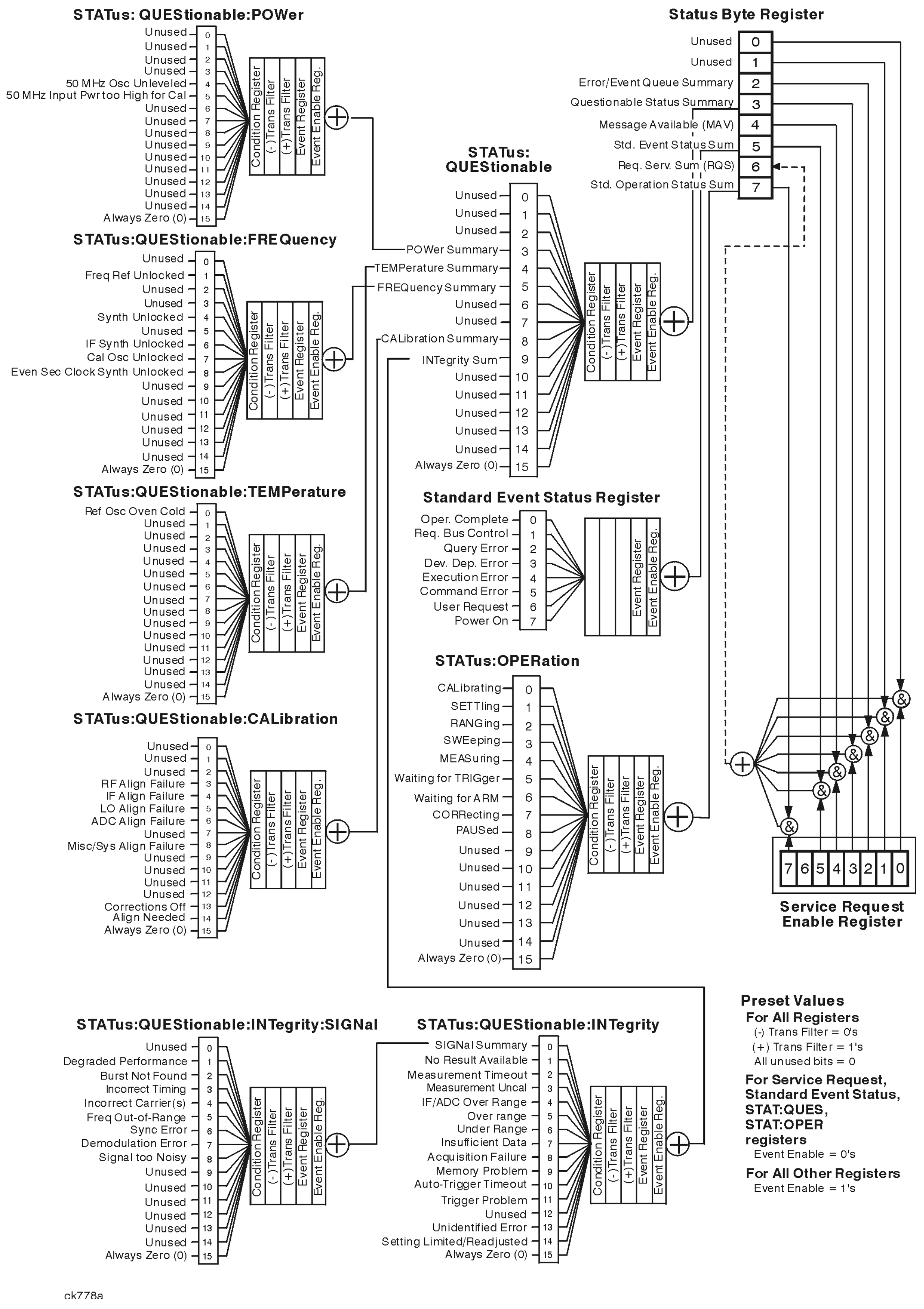
If your program enables the controller to detect and respond to service requests, it should instruct the controller to perform a serial poll when the GPIB SRQ line is set true. Each device on the bus returns the contents of its status byte register in response to this poll. The device whose RQS bit is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

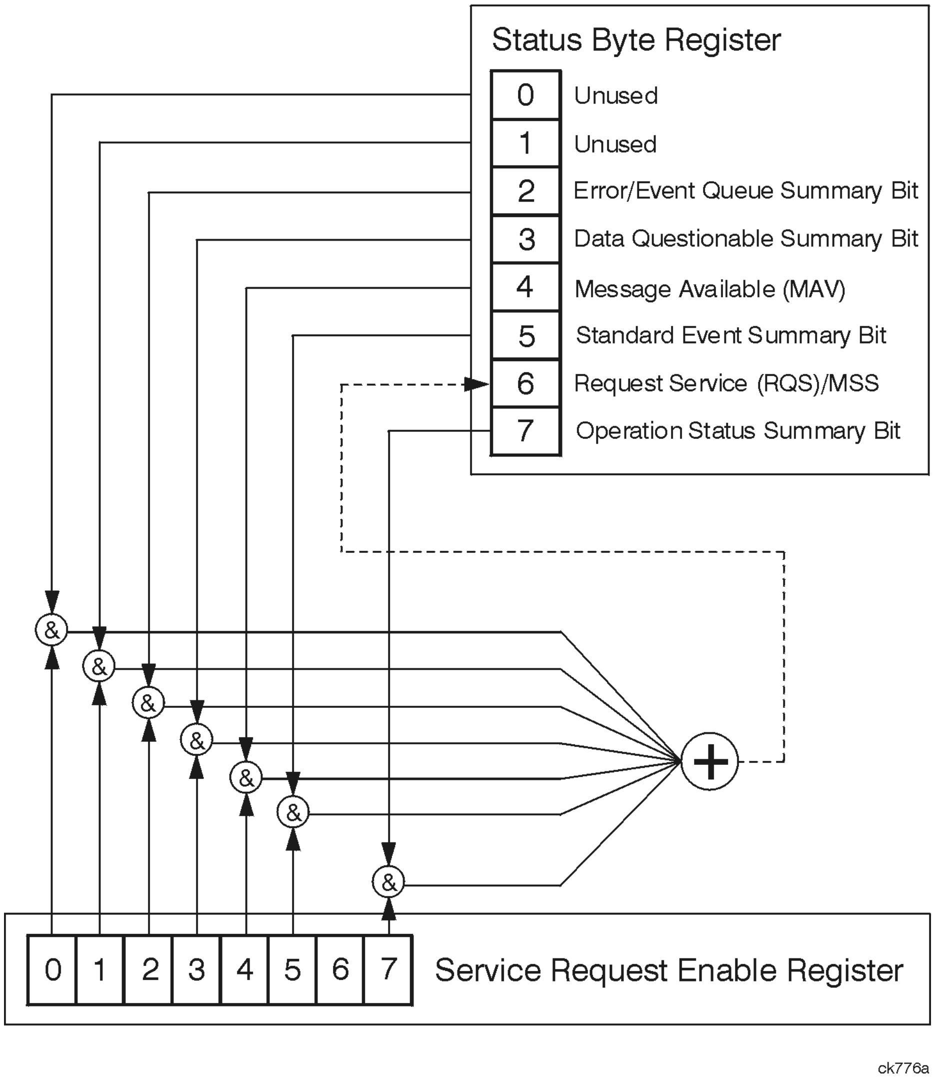
Restarting a measurement (INIT command) can cause the measuring bit to pulse low, which causes an SRQ if the status register is configured to SRQ on end-of-measurement. To avoid this:

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

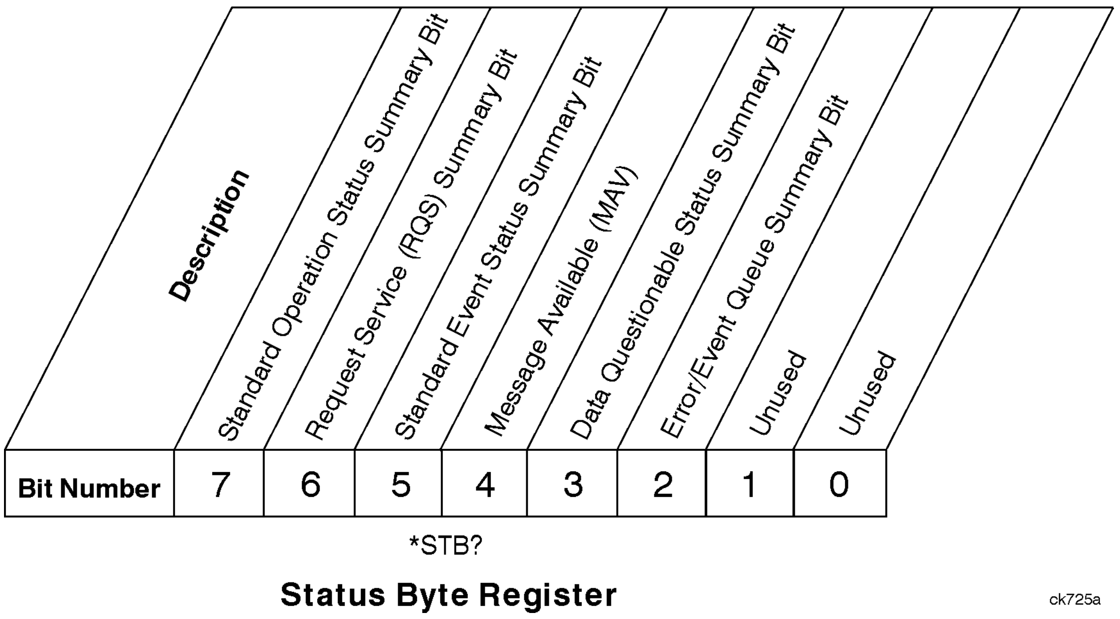
Overall Status Register System



Status Byte Register


Status Byte Register
0 Unused
1 Unused
2 Error/Event Queue Summary Bit
3 Data Questionable Summary Bit
4 Message Available (MAV)
5 Standard Event Summary Bit
6 Request Service (RQS)/MSS
7 Operation Status Summary Bit
&
+
0 1 2 3 4 5 6 7
Service Request Enable Register
ck776a

The RQS bit is read and reset by a serial poll. MSS (the same bit position) 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.


Description
Standard Operation Status Summary Bit
Request Service (RQS) Summary Bit
Standard Event Status Summary Bit
Message Available (MAV)
Data Questionable Status Summary Bit
Error/Event Queue Summary Bit
Unused
Unused
Bit Number
7
6
5
4
3
2
1
0
*STB?
Status Byte Register
ck725a

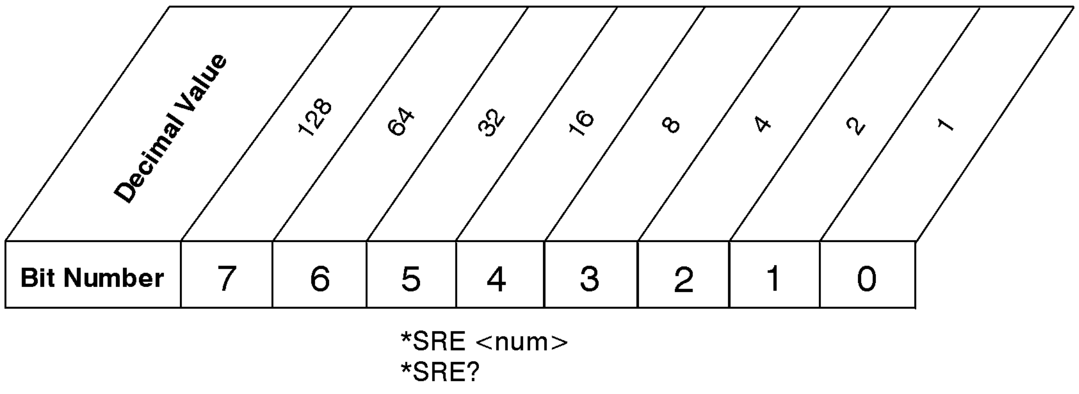
Bit	Description
0, 1	These bits are always set to 0.
2	A 1 in this bit position indicates that the SCPI error queue is not empty. The SCPI error queue contains at least one error message.
3	A 1 in this bit position indicates that the data questionable summary bit has been set. The data questionable event 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.

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.

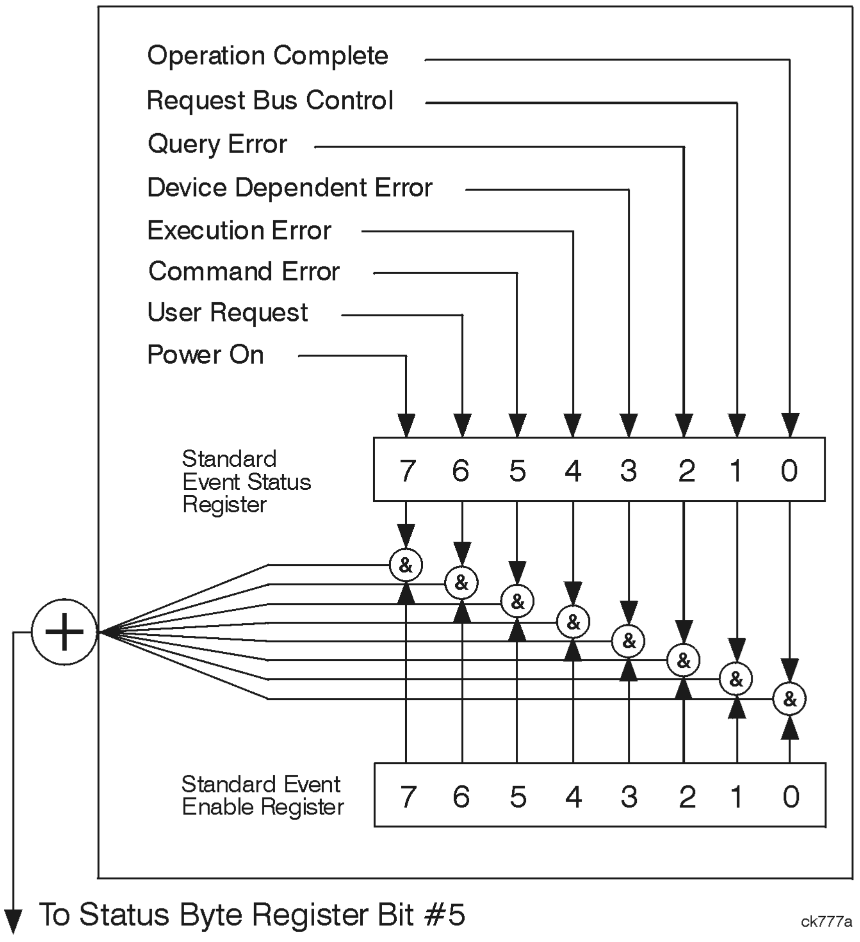
Send the `*SRE <number>` command where `<number>` is the sum of the decimal values of the bits you want to enable plus the decimal value of bit 6. For example, assume that you want to enable bit 7 so that whenever the standard operation status register summary bit is set to 1 it will trigger a service request. Send the command `*SRE 192` (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).

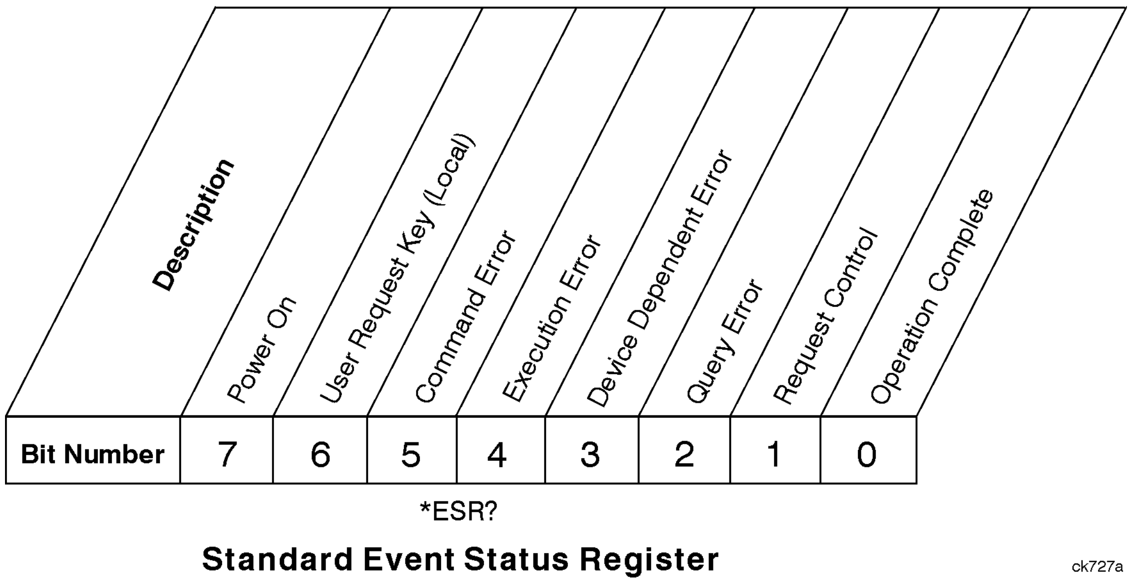


Service Request Enable Register

Standard Event Status Register



The standard event status register contains the following bits:

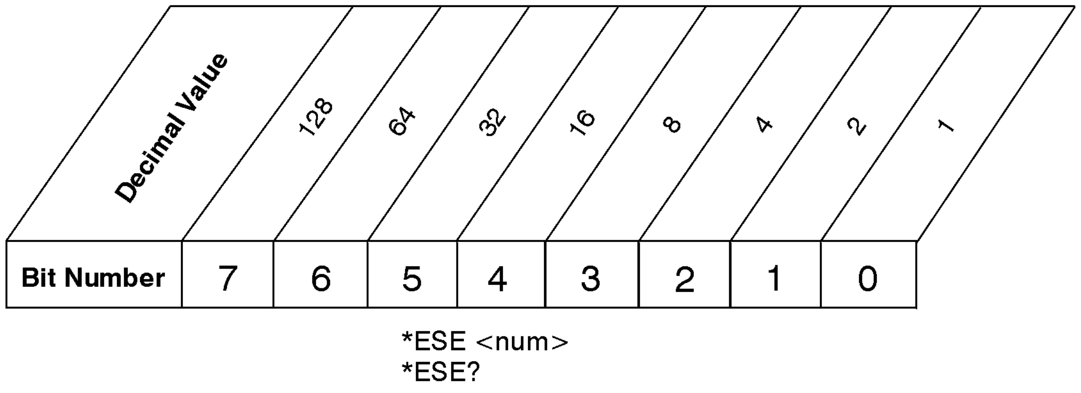


Bit	Description
0	A 1 in this bit position indicates that all pending operations were completed following execution of the *OPC command.
1	This bit is always set to 0. (The instrument does not request control.)
2	A 1 in this bit position indicates that a query error has occurred. Query errors have SCPI 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	Currently not used.
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 6 so that whenever either of those bits is set to 1, the standard event status summary bit of the status byte register will be set to 1, send the command `*ESE 192` (128 + 64). The command `*ESE?` returns the decimal value of the sum of the bits previously enabled with the `*ESE <number>` command.

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



Standard Event Status Enable Register

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.

Operation Status Register

The operation status register monitors the current instrument measurement state. It checks to see if the instrument is measuring, calibrating, sweeping, waiting for a trigger, or what?

Questionable Status Register

The questionable status register monitors the instrument to see if anything questionable has happened. 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.

C Programming Examples using VTL

The programming examples that are provided are written using the C programming language and the HP/Agilent VTL (VISA transition library). This section includes some basic information about programming in the C language. 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 si 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

GPIB Instrument Nomenclature

An instrument that is part of an 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, an 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).
- 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 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.

Using the LAN to Control the Analyzer

Using ftp for File Transfers

File transfers can be done using the instrument LAN connection. For example, you can use the ftp functionality to download instrument screen dumps to an external server.

A sample ftp session might be:

`ftp 15.88.163.118` (<instrument IP address>)

At the name prompt enter:
`vsa`

At the password prompt enter:
`service`

You are now in the instrument `/users` directory and can get files from the analyzer. Type in `help` at the prompt to see the ftp commands that are available on your system. Typing `quit` will end your ftp session.

NOTE Do *NOT* delete files from this directory. Most of the files are required for the instrument, and it's optional personality modes, to operate.

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.

Table lists the available user commands.

Table 2-1 `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.

Table 2-1 `ftp` Commands

Command	Description
quit	Closes the connection to the host and exits ftp.

Using Telnet to Send Commands

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

NOTE If you need to control the GPIB using “device clear” or SRQ’s, you can use SICL LAN. SICL LAN provides control of your analyzer via IEEE 488.2 GPIB over the LAN. See “Using SICL LAN to Control the Analyzer” on page 77. in this chapter.

On unix:

The syntax of the telnet command is:

```
telnet <vsa hostname> 5023
```

or

```
telnet <IP address>
```

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:

You would type at the dos prompt

```
telnet
```

The telnet gui has the host/port setting menu.

Unix Telnet Example:

To connect to the instrument with host name `my4406` and port number `5023`, enter the following command:

```
telnet my4406 5023
```

NOTE You must have changed your instrument host name from the default (for example, change E566DD69 to my4406) in order to specify your instrument by name. Press **System, Config I/O, Host Name**.

Alternately, you can enter the IP address directly in the `telnet` command, in place of the analyzer name:

```
telnet 15.4.45.255 5023
```

The computer responds with the following messages:

```
Trying...
Connected to 15.4.45.255.
Escape character is '^]'.
```

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. When you are done, break the telnet connection using the escape character (in this case `Ctrl ]`), and type `quit`.

The analyzer responds with the a welcome message and the SCPI prompt. You can immediately enter programming (SCPI) commands. Typical commands might be:

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

The small program above sets the analyzer to measure a signal in the frequency domain, places a marker on the maximum point, and then queries the analyzer 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 analyzer returns the amplitude level of the marker to your computer and displays it on the next line. For example, after typing `CALC:SPECTRUM:MARK:MAX?` and pressing Enter, the computer would 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.

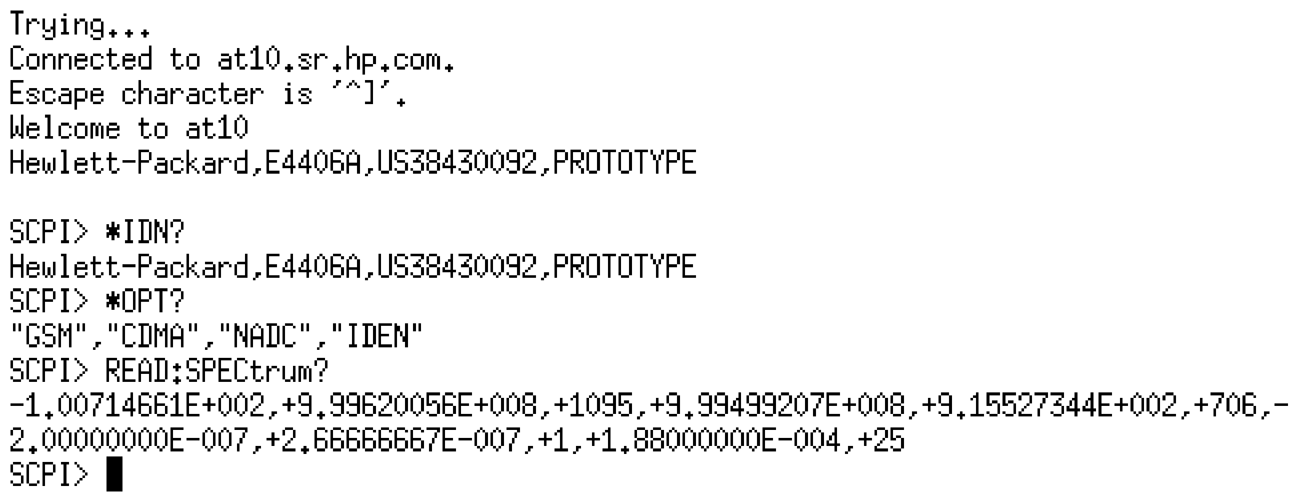
At the telnet prompt, type `quit` or `close`.

The telnet connection closes and you see your regular prompt.

```
Connection closed.
```

Figure 2-2 shows a terminal screen using the example commands above.

Figure 2-2 Example telnet Session



```
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 analyzer 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 analyzer 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 analyzer.

Setting Up Your Analyzer for Socket Programming

Before you can use socket programming, you must identify your analyzer’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 analyzer is 5025.

Using SICL LAN to Control the Analyzer

SICL LAN is a LAN protocol using the Standard Instrument Control Library (SICL). It provides control of your analyzer over the LAN, using a variety of computing platforms, I/O interfaces, and operating systems. With SICL LAN, you control your remote analyzer over the LAN with the same methods you use for a local analyzer 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 analyzer implements a SICL LAN *server*. To control the analyzer, you need a SICL LAN *client* application running on a computer or workstation that is connected to the analyzer 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.

At the time of the publication of this manual, National Instruments' VISA does not support the VXI-11 Instrument Protocol. However, future revisions of National Instruments VISA will support the VX-11 protocol. Contact National Instruments for their release date.

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

Collecting SICL LAN Set-up Information

Before you set up your analyzer as a SICL LAN server, you will need to collect some information about your VISA/SICL LAN client application. The “value” of the following parameters can be found from the front panel **System** keys. They can then be 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 analyzer. Your analyzer 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 analyzer is shipped with the logical unit number set to 8.
Emulated GPIB Address	The emulated GPIB address (bus address) is assigned to the device to be controlled using SICL LAN. The emulated GPIB address is automatically set to be the same as the current GPIB address. The instrument is shipped with the emulated GPIB address set to 18.

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 Analyzer 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 analyzer 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 the same as the GPIB address.

Configuring Your 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’s VISA can be found on Agilent’s website:

```
http://www.tm.agilent.com/tmo/software/English/HP_IO_Libraries.html
```

These descriptions assume a LAN connection between your computer and network analyzer. They are not written 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 hostname or IP address of your analyzer in the hostname field, such as my4406a.companyname.com
12. Enter a Remote SICL address, such as GPIB1.
13. Set the LAN interface to match the defined LAN client (lan1 in this example).
14. Select OK.
15. Close I/O Configuration by selecting OK.

Controlling Your Analyzer 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.

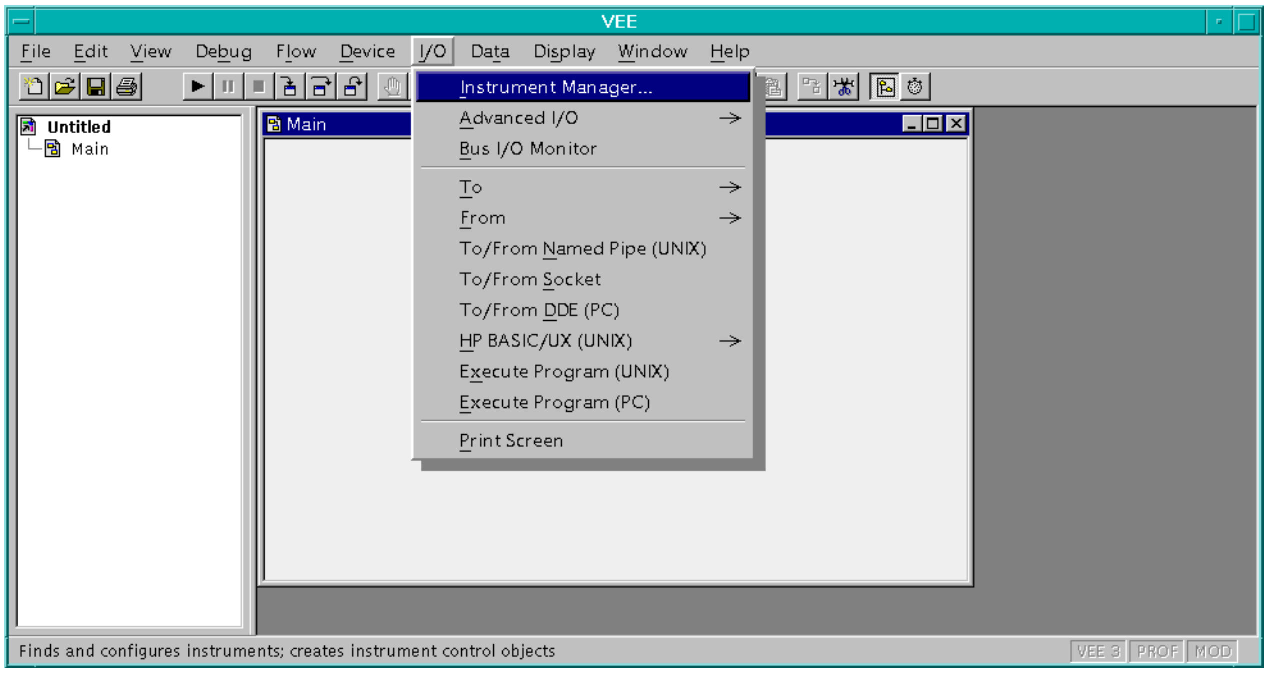
NOTE If you are using HP/Agilent VEE and SICL LAN, the logical unit number is limited to the range of 0-8.

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 analyzer. 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 analyzer:

1. On your computer or workstation, select `I/O|Instrument Manager`.

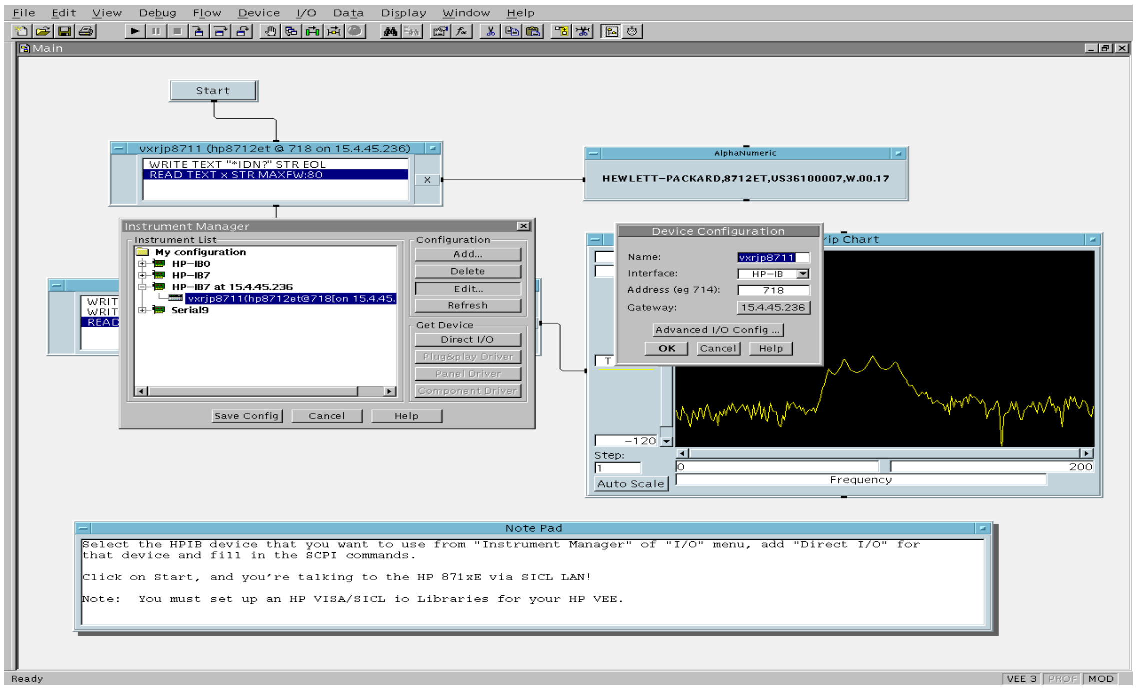
Figure 2-3 I/O | Instrument Manager Menu



Finds and configures instruments; creates instrument control objects

2. Add a new GPIB device with an address of 7XX, where XX is the GPIB device address from your analyzer.

Figure 2-4 Adding Your Analyzer as an VEE Device



```
WRITE TEXT "*IDN?" STR EOL
READ TEXT x STR MAXFW:80
```

```
HEWLETT-PACKARD,8712ET,US36100007,W.00.17
```

Name: vxrjp8711
Interface: HP–IB
Address (eg 714): 718
Gateway: 15.4.45.236

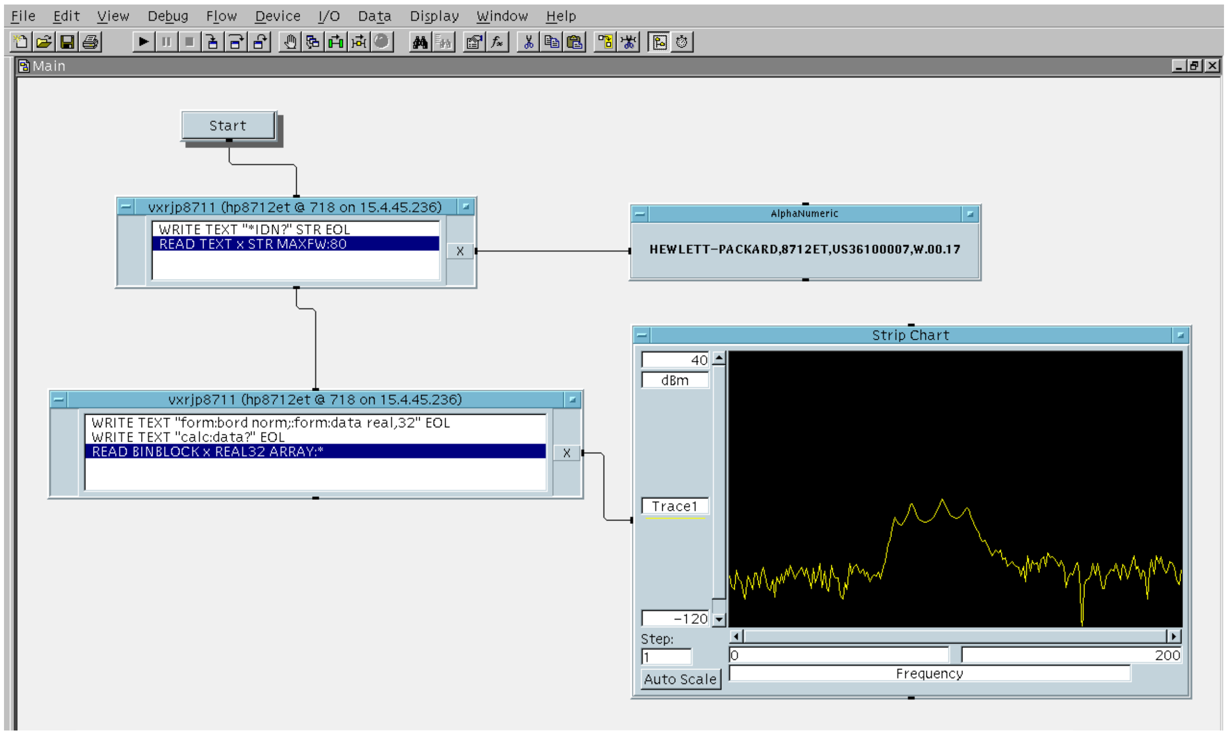
```
Select the HPIB device that you want to use from "Instrument Manager" of "I/O" menu, add "Direct I/O" for
that device and fill in the SCPI commands.

Click on Start, and you're talking to the HP 871xE via SICL LAN!

Note:  You must set up an HP VISA/SICL io Libraries for your HP VEE.
```

To send SCPI commands to the analyzer, 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 analyzer.

Figure 2-5 Sending SCPI Commands Directly to your Analyzer



```
WRITE TEXT "*IDN?" STR EOL
READ TEXT x STR MAXFW:80
```

```
HEWLETT-PACKARD,8712ET,US36100007,W.00.17
```

```
WRITE TEXT "form:bord norm;:form:data real,32" EOL
WRITE TEXT "calc:data?" EOL
READ BINBLOCK x REAL32 ARRAY:*
```

See the VEE example program for more details.

Controlling Your Analyzer with SICL LAN and HP/Agilent BASIC for Windows

Before you can use HP/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 analyzer, `GPIP name` with the GPIB name given to your analyzer, 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:

analyzer IP address **12.22.344.225**

analyzer GPIB name **inst0**

logical unit number **7**

timeout value (seconds) **30**

`LOAD` statement (all on a single line)

LOAD BIN “GPIBS;DEV lan[**12.22.344.225**]:**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 analyzer using commands such as the following:

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

where 18 is the device address for the analyzer.

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

Controlling Your Analyzer 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 analyzer, and `GPIB name` with the GPIB name given to your analyzer. 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 analyzer using commands such as the following:

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

where 18 is the device address for the analyzer.

Using HP/Agilent VEE Over Socket LAN

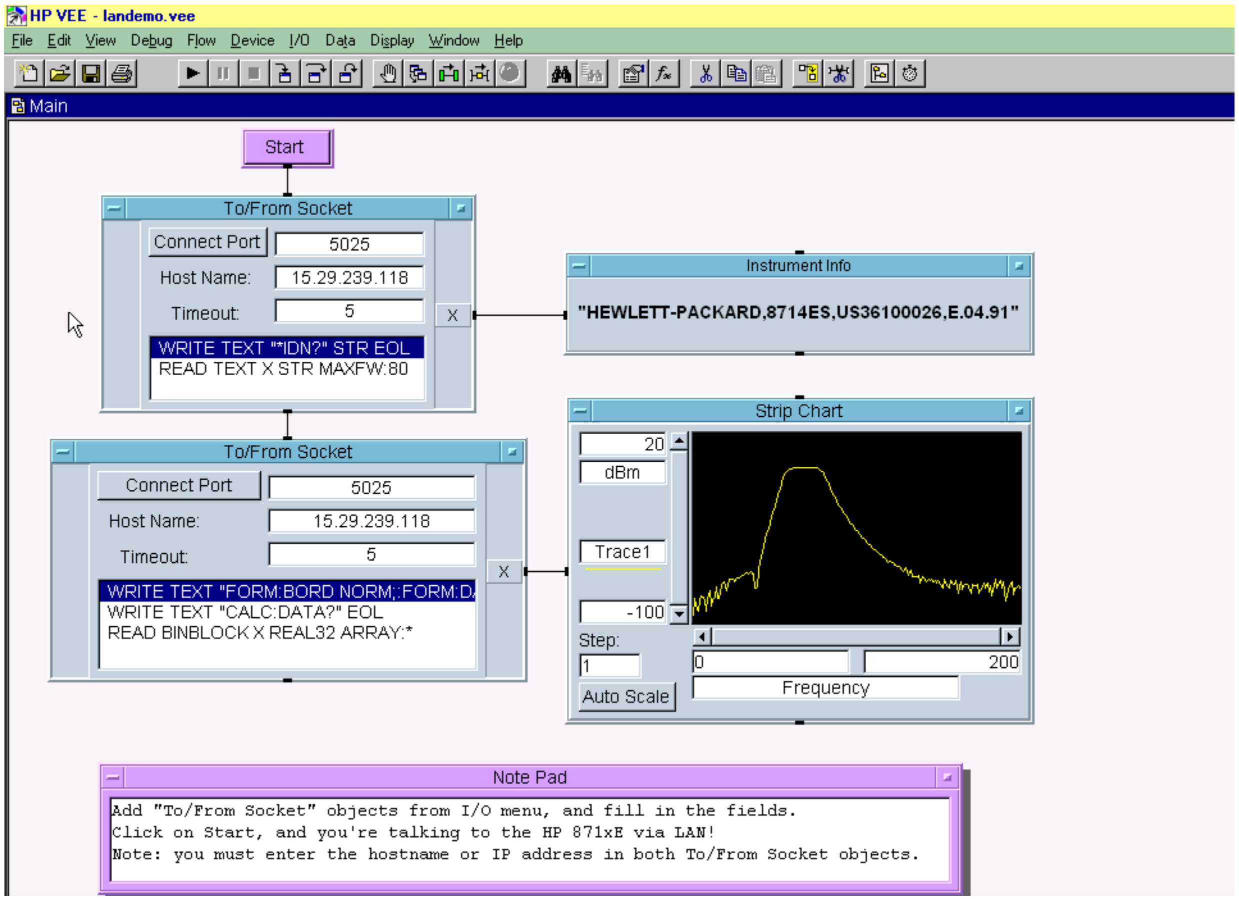
To control your analyzer via socket LAN using VEE, click on the VEE menu titled "I/O." Then select "To/From Socket" and position the I/O object box on the screen. Fill in the following fields:

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

For faster troubleshooting, you may want to set the timeout to a smaller number. If the hostname you enter doesn't work, try using the IP address of your analyzer (example: `15.4.43.5`). Using the IP address rather than the hostname may also be faster. See Figure 2-6 for an example of an VEE screen.

NOTE If you need to control the GPIB using “device clear” or SRQ’s, you can use SICL LAN. SICL LAN provides control of your analyzer via IEEE 488.2 GPIB. See See “Using SICL LAN to Control the Analyzer” on page 77. in this chapter.

Figure 2-6 Sample VEE Screen



To/From Socket
Connect Port: 5025
Host Name: 15.29.239.118
Timeout: 5
WRITE TEXT "*IDN?" STR EOL
READ TEXT X STR MAXFW:80

Instrument Info
"HEWLETT-PACKARD,8714ES,US36100026,E.04.91"

To/From Socket
Connect Port: 5025
Host Name: 15.29.239.118
Timeout: 5
WRITE TEXT "FORM:BORD NORM;:FORM:D
WRITE TEXT "CALC:DATA?" EOL
READ BINBLOCK X REAL32 ARRAY:*

Note Pad
```
Add "To/From Socket" objects from I/O menu, and fill in the fields.
Click on Start, and you're talking to the HP 871xE via LAN!
Note: you must enter the hostname or IP address in both To/From Socket objects.
```

Using a Java™ Applet Over Socket LAN

The example program "Using Java Programming Over Socket LAN" on page 126 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 on your documentation CD ROM that shipped with the product.

Using a C Program Over Socket LAN

The example programs "Using C Programming Over Socket LAN" on page 109 and "Using C Programming Over Socket LAN (Windows NT)" on page 123demonstrate simple socket programming. They are written in C, and compile in the HP-UX UNIX environment, or 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, since fread( ) and fwrite( ) may not work on sockets.

General LAN Troubleshooting

Troubleshooting the Initial Connection

Getting the analyzer 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 analyzer 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 analyzer'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 analyzer via ftp or telnet, go directly to "Pinging the Analyzer from Your Computer or Workstation" on page 89.

If you have previously been able to access the analyzer 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 hostname.

❑ Is the upper- and lower-case character usage in your hostname 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 analyzer, you might be experiencing timeout problems. See the previous section on "Timeout Errors."

Common Problems After You've Made the Connection

This section describes common problems you may encounter when using the analyzer on a LAN. It assumes you have been able to connect to the analyzer 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**.

You cannot connect to the analyzer

- If you suspect a bad LAN connection between your computer and analyzer, you can verify the network connection by using the ping command described earlier in this chapter or another similar echo request utility.
- If a bad connection is revealed, try the following solutions:

  — Make sure the analyzer is turned on.

  — Check the physical connection to the LAN.

— Make sure the internet (IP) Address of the analyzer is set up correctly in the LAN port setup menu. (Press **System, Config I/O, IP Address**.)

— If the analyzer and the computer are on different networks or subnets, make sure the gateway address and subnet mask values are set correctly. See "Troubleshooting Subnet Problems" earlier in this chapter.

You cannot access the file system via ftp

- If you get a "connection refused" message, try the following solutions:

  — If the power to the analyzer 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 analyzer. Refer to "If you cannot connect to the analyzer" earlier in this section.

You cannot telnet to the command parser port

- If you get a "connection refused" message

  — Check the telnet port number from the front panel keys.

- If you get a "connection timed out" or "no response from host" message

  — Verify the LAN connection between your computer and the analyzer. Refer to "If you cannot connect to the analyzer" earlier in this section.

- If you get a "connection refused" or "no response from host" message

  — If the analyzer was just turned on, make sure that you wait about 25 seconds before attempting the connection.

You get an "operation timed-out" message

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

You 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 analyzer'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 Analyzer from Your Computer or Workstation

Verify the communications link between the computer and the analyzer 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 analyzer 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 hostname. 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 hostname 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 analyzer, starting with the your workstation. Ping each gateway, then attempt a ping of the remote file server.

If the analyzer still does not respond to ping, then you should suspect a hardware problem with the analyzer. To check the analyzer performance, refer to "Verify the Analyzer 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 analyzer 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 analyzer. To check the analyzer, refer to "Verify the Analyzer 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 2-2 **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 2-3 **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	

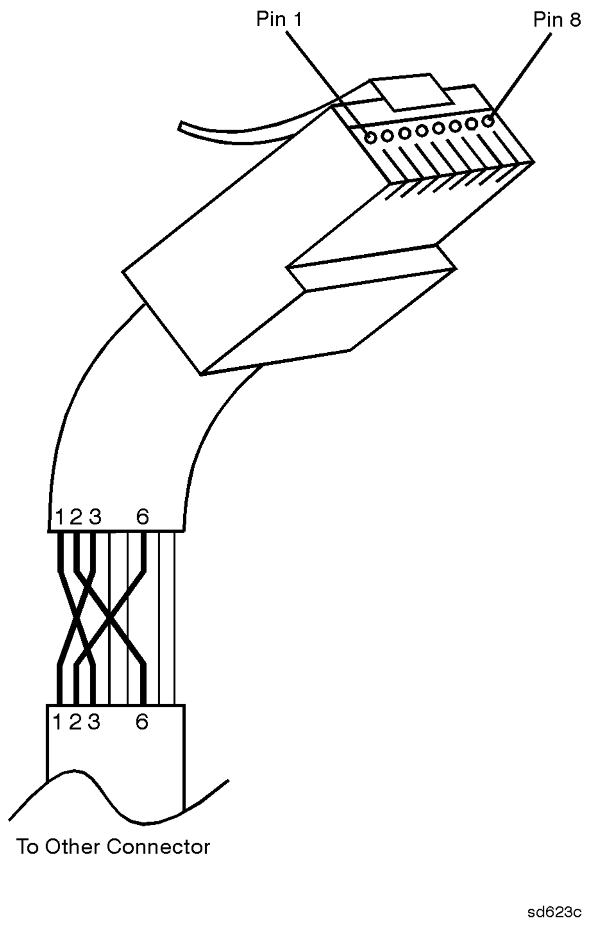
a. Either end of this cable can be used at the analyzer or LAN device. The connector names are a convention useful during cable construction only.

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 , above. Standard straight-through patch cables can then be used from the analyzer 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 analyzer. Please refer to Table , above, and verify all connections before using cables not made by Agilent Technologies.

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



3 Programming Examples

Types of Examples

This section includes examples of how to program the instrument using the instrument SCPI programming commands. The most of the examples are written for a PC, using GPIB. They are written in the C programming language and use the HP/Agilent VISA transition library. The VISA transition library must be installed and the GPIB card configured.

These examples are available on the Agilent Technologies E4406A documentation CD-ROM. They are also available at the URL **`http://www.agilent.com/find/vsa`**

The section "C Programming Examples using VTL" on page 57, 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 VXIplug&play instrument driver that is provided. The 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++, BASIC. The software driver can be found at the URL **`http://www.agilent.com/find/vsa`**.

The programming examples include:

- "Using Markers" on page 95
- "Saving Binary Trace Data in an ASCII File" on page 98
- "Saving ASCII Trace Data in an ASCII File" on page 101
- "Saving and Recalling Instrument State Data" on page 104
- "Performing Alignments and Getting Pass/Fail Results" on page 107
- "Using C Programming Over Socket LAN" on page 109
- "Using C Programming Over Socket LAN (Windows NT)" on page 123
- "Using Java Programming Over Socket LAN" on page 126

Using Markers

This C programming example (HPE4406Markers.c):

- uses the VISA library for input/output, opens a session to a GP-IB device at address 18 and presets the instrument.
- sets the input to the internal 50 MHz reference source and tunes the instrument to the signal.
- turns off the active trace and put the instrument in single measurement mode.
- initiates a spectrum measurement and waits for the operation to complete.
- puts marker 1 on the peak signal of the average trace and measures the amplitude.
- puts a noise type marker (marker 2) on the average trace and measures the amplitude.
- calculates the difference between the peak and the noise floor.
- puts the instrument back in continuous measurement mode and prints the results.

Example:

```
#include <stdio.h>
#include <stdlib.h>
#include <stdmath.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);
     }

     /*reset the instrument */
     viPrintf(viVSA, "*RST\n");

     /*set the input port to the internal 50Mhz reference
       source*/
     viPrintf(viVSA, "SENS:FEED AREF\n");

     /*tune the instrument to 50MHZ*/
     viPrintf(viVSA, "SENS:FREQ:CENT 50E6\n");

     /*put the instrument in single measurement 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 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*/
     viPrintf (viVSA,"%lf",&dNoiseMarker);

     /*put the instrument back to continuous mode*/
     viScanf (viVSA,"INIT:CONT 1\n");

     /*calculate the difference between the marker peak and
       the NOISE marker*/
     dResult = fabs(dNoiseMarker - dPeakPower);

     /*print result to the standard 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 C programming example (HPE4406Trace.c):

- uses the VISA library for input/output, opens a session to a GP-IB device at address 18 and presets the instrument.
- sets the input to the internal 50 MHz reference source and tunes the instrument to the signal.
- sets the instrument to single measurement mode.
- selects binary data output format (causing the trace header to be 6 bytes) and sets the binary byte order to SWAP.
- initiates a spectrum measurement and waits for the operation to complete.
- sets the instrument back to continuous measurement mode and queries the trace data.
- calculates the number of points in the trace by:
    1. getting the trace header data. (In this case we know it's the first 6 bytes.)
    2. extracting the information on the number of bytes in the block of data, from the trace header.
    3. calculating the number of trace points given the number of bytes in the trace. (REAL,64 binary format means each number is represented by 8 bytes.)
- Gets and saves the trace in a buffer
- Copies the trace buffer to the array of real numbers
- Saves the trace data to an ASCII file

Example:

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

void main ()
{

     /*program variable*/
     ViSession defaultRM, viVSA;
     ViStatus viStatus = 0;
     char sTraceBuffer[10240] = {0};
     char sBufferInfo[4] = {0};
     FILE *fTraceFile;
```

```
long lNumberPoints = 0;
long lNumberBytes = 0;
long lComplete = 0;
unsigned long lBytesRetrieved;
ViReal64 adTraceArray[10240];

/*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);
}

/* 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 instrument to 50MHz*/
viPrintf(viVSA, "SENS:FREQ:CENT 50E6\n");

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

/*set the instrument in single mode*/
viPrintf(viVSA, "INIT:CONT 0\n");

/*Set the ouput format to a binary format.
  This will cause the trace header to be 6 bytes*/
viPrintf(viVSA, "FORM REAL,64\n");

/*set the binary byte order to SWAP*/
viPrintf(viVSA, "FORM:BORD SWAP\n");

/*trigger a  spectrum measurement*/
viPrintf(viVSA, "MEAS:SPEC?;*OPC?\n");

/*poll the operation complete query*/
while (!lComplete)
     viScanf (viVSA,"%d",&lComplete);

/*query the spectrum trace data*/
viPrintf(viVSA, "FETCH:SPEC7?\n");

/*set the instrument back to continuous mode*/
viPrintf(viVSA, "INIT:CONT 1\n");
```

```
     /*get trace header data, in this case we know it's 6
       bytes*/
     viRead (viVSA,(ViBuf)sTraceBuffer,6,&lBytesRetrieved);

     /*Extract the number of bytes from the trace header*/
     memcpy(sBufferInfo,sTraceBuffer+2,4);
     lNumberBytes = atoi(sBufferInfo);

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

     /*get and save trace  in a buffer*/
     viRead (viVSA,(ViBuf)sTraceBuffer,lNumberBytes,
       &lBytesRetrieved);

     /*copy the trace buffer to the array of real*/
     memcpy(adTraceArray,sTraceBuffer,(size_t)lNumberBytes);

     /*save trace data to an ASCII file*/
     fTraceFile=fopen("C:\\HPE4406ATrace.txt","w");
     fprintf(fTraceFile,"HPE4406ATrace.exe Output\nAgilent
        1998\n\n");
     fprintf(fTraceFile,"List of %d points of the averaged
       spectrum trace:\n\n",lNumberPoints);
     for (long 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:\\HPE4406ATrace.txt file\n\n",lNumberPoints);

     /* close session */
     viClose (viVSA);
     viClose (defaultRM);
}
```

Saving ASCII Trace Data in an ASCII File

This C programming example (HPE4406TraceASCII.c):

- uses the VISA library for input/output, opens a session to a GP-IB device at address 18 and presets the instrument.
- sets the input to the internal 50 MHz reference source and tunes the instrument to the signal.
- sets the instrument to single measurement mode.
- initiates a spectrum measurement and waits for the operation to complete.
- gets and saves the trace in a buffer.
- queries the spectrum trace data and gets the trace header (which is 6 bytes for ASCII data format).
- sets the instrument back to continuous measurement mode.
- saves the trace data to an ASCII file.

Example:

```
#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);
```

```
}

/* 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 instrument to 50MHz*/
viPrintf(viVSA, "SENS:FREQ:CENT 50E6\n");

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

/*set the instrument in single mode*/
viPrintf(viVSA, "INIT:CONT 0\n");

/*trigger a  spectrum measurement*/
viPrintf(viVSA, "MEAS:SPEC1?;*OPC?\n");

/*poll the operation complete query*/
while (!lComplete)
     viScanf (viVSA,"%d",&lComplete);

/*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 instrument back to continuous mode*/
viPrintf(viVSA, "INIT:CONT 1\n");

/*save trace data to an ASCII file*/
fTraceFile=fopen("C:\\HPE4406ATraceASCII.txt",
  "w");
fprintf(fTraceFile,"HPE4406ATraceASCII.exe
  Output\nHewlett-Packard 1998\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:\\HPE4406ATraceASCII.txt file\n\n");

        /* close session */
        viClose (viVSA);
        viClose (defaultRM);
}
```

Saving and Recalling Instrument State Data

This C programming example (HPE4406State.c):

- uses the VISA library for input/output, opens a session to a GP-IB device at address 18 and presets the instrument
- sets the input to the internal 50 MHz reference source and tunes the instrument to the signal
- selects a resolution bandwidth, display scaling, and reference level
- initiates a spectrum measurement and waits for the operation to complete
- saves the current state to register 10, replacing anything that is currently in register 10
- displays a message and waits for a key press before resetting the instrument
- displays a message and waits for a key press before recalling the instrument stored in register 10
- zooms the spectrum display and again displays a message waiting for a key press before resetting the instrument

Example:

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

void main ()
{
     /*program variables*/
     ViSession defaultRM, viVSA;
     ViStatus viStatus = 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);
     }
```

```
/*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 instrument 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;*OPC?\n");

/*poll the operation complete query*/
while (!lComplete)
     viScanf (viVSA,"%d",&lComplete);

/*save this state in register 10.
  !!!Carefull this will overwrite register 10*/
viPrintf(viVSA, "*SAV 10\n");

/*display message*/
printf("E4406A 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 the again the input port to the internal 50Mhz
  reference source*/
viPrintf(viVSA, "INP:PORT AREF\n");

/*display message*/
printf("\n\nThe instrument was reset to the factory
  default setting\n");
```

```
     printf("Notice the absence 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\n");

     /* close session */
     viClose (viVSA);
     viClose (defaultRM);
}
```

Performing Alignments and Getting Pass/Fail Results

This C programming example (HPE4406Align.c):

- uses the VISA library for input/output, opens a session to a GP-IB device at address 18 and presets the instrument
- increases the instrument timeout to one minute (60 sec) to allow time for the auto-alignment to run
- runs the auto-alignment procedure
- queries the auto-alignment results and outputs the success/failure information

Example:

```
#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 60 sec*/
     viSetAttribute(viVSA,VI_ATTR_TMO_VALUE,60000);

     /*reset the instrument*/
     viPrintf(viVSA, "*RST\n");

     /*print message */
     printf("The auto-alignment is in progress...\nPlease
       wait...\n\n");

     /*auto-align the instrument*/
     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 instrument auto-alignement was
            successful!\n\n");
     else

          /*print failure message to standard output*/
          printf("The instrument auto-alignement was not
           successful!\n\n");

     /* reset timeout to 3 sec*/
     viSetAttribute(viVSA,VI_ATTR_TMO_VALUE,3000);

     /* Close session */
     viClose (viVSA);
     viClose (defaultRM);
}
```

Using C Programming Over Socket LAN

This is a C programming example (socketio.c) that 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 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, 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.

Example:

```
/***************************************************************************
*  $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 a 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.

Example:

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

 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 a 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.

Example:

```
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;
    }

}
```

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	
Commands: listing of all	SYSTem:HELP:HEADers	Lists only the commands in the current selected mode.
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

Function	SCPI Command Subsystems	Remarks
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, Service, GSM, and CDMA.
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>	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:	

Function	SCPI Command Subsystems	Remarks
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	

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

IEEE Common 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.

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. 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?**

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

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:

```
Hewlett-Packard, E4406A,US00000040, A.01.42
```

Remarks:	An @ in the firmware revision information indicates that it is proto firmware.
Front Panel Access:	**System, Show System**

Instrument State Query

***LRN?**

Returns current instrument state data in a block of defined length. The information is in a machine readable format only. Sending the query returns the following format:

`SYST:SET #NMMM<state_data>`

The following example is a response to ***LRN?** The actual sizes will vary depending on the instrument state data size.

Example: `#42031SYST:SET #42016<state data>`

4 (N in the preceding query response format) represents the number of bits to follow
2031 and 2016 (MMM in the preceding query response format) represents file size in bytes

The state can be changed by sending this block of data to the instrument after removing the size information:

`SYST:SET #NMMM<state_data>`

Operation Complete

***OPC**

***OPC?**

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

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

Query Instrument Options

***OPT?**

Returns a string of all the installed instrument options.It is a comma separated list such as: “BAC, 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`**.

The **`:SYSTem:PRESet`** command is equivalent to a front panel **Preset**. The front panel **Preset** sets instrument parameters to values for good local/human interaction. 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`

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.

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**

Wait-to-Continue

`*WAI`

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

ABORt Subsystem

Abort

: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 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.

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.

Query the Current Measurement Status

`:CALCulate:CLIMits:FAIL?`

Checks if the current measurement is outside its limits. It returns a 0 (zero) if it is passing or a 1 (one) if it is failing.

Front Panel Access:	None

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 the "MEASure Group of Commands" on page 187 for information on the data that can be returned for each measurement.

Calculate/Compress Trace Data Query

:CALCulate:DATA[n]:COMPress? MAXimum|MEAN|MINimum|RMS|SAMPle|SDEViation|CFIT {,<soffset>}{,<length>}{,<roffset>}

Returns the designated trace data for the currently selected measurement. The command can be used with sub-opcodes (*n*) for measurement results that are trace data. See the following table.

This command is used to compress/decimate a long trace to extract the desired data and only return to the computer the necessary data. A typical example would be to acquire N bursts of GSM data and return the mean power of each burst.

The command can also be used to identify the best curve fit for the data.

Curve Fit - applies curve fitting routines to the data. Where <soffset> and <length> are required, and <roffset> is an optional parameter for the desired order of the curve equation. The query will return the following values: the x-offset (in points) and the curve coefficients ((order + 1) values).

<Start offset> - is an optional integer. It specifies the amount of data, at the beginning of the trace, that will be ignored before the decimation process starts. It is an integer index (that starts counting at zero) for all the elements in the trace. The default value is zero.

<Length> - is an optional integer that defines how many trace elements will be compressed into one value. This parameter has a default value equal to the current trace length.

<Repeat offset> - is an optional real number. 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. Select a number such that repeated additions will round to the correct starting index.

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

1. Set the waveform measurement sweep time to acquire the required number of bursts.
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,62,1315,1442.3** (These parameter values correspond to GSM signals.)

Remarks: The optional parameters must be entered in the specified order. If you want to specify <length>, you must also specify <soffset> or it's default. (e.g. **CALC:DATA2:COMP? MEAN,62,1315**

This command uses the data setting specified by the

FORMat:DATA command and can return binary or ascii data.

History: Added in revision A.03.00 and later

Measurement	Available Traces	Markers Available?
ACP - adjacent channel power (Basic, cdmaOne, cdma2000, W-CDMA, 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 (W-CDMA mode)	CDPower (*n*=2)[a] EVM (*n*=4)[a] MERRor (*n*=5)[a] PERRor (*n*=6)[a] SPOWer (*n*=8)[a]	yes
CHPower - channel power (Basic, cdmaOne, cdma2000, W-CDMA mode)	SPECtrum (*n*=2)[a]	no markers
CSPur - spurs close (cdmaOne mode)	SPECtrum (*n*=2)[a] ULIMit (*n*=3)[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 modes)	EVM (*n*=2)[a] MERRor (*n*=3)[a] PERRor (*n*=4)[a]	yes
OBW - occupied bandwidth (iDEN, PDC modes)	no traces	no markers

Measurement	**Available Traces**	**Markers Available?**
ORFSpectrum - output RF spectrum (GSM mode)	RFEModulation (n=2)[a] RFESwitching (n=3)[a]	no markers
PFERror - phase and frequency error (GSM mode)	PERRor (n=2)[a] PFERror (n=3)[a] RFENvelope (n=4)[a]	yes
PSTatistic - power statistics CCDF (cdma2000, W-CDMA modes)	MEASured (n=2)[a] GAUSsian (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 mode)	EVM (n=2)[a] MERRor (n=3)[a] PERRor (n=4)[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.

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
- SPECtrum - markers available
- WAVeform - markers available

Service Mode - <measurement> key words

- PVTime - no markers
- 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. Using the MEASure or READ command, before the marker command, 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.

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)
Front Panel Access:	None

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, 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 (W-CDMA mode)	CDPower (*n*=2)[a] EVM (*n*=4)[a] MERRor (*n*=5)[a] PERRor (*n*=6)[a] SPOWer (*n*=8)[a]	yes
CHPower - channel power (Basic, cdmaOne, cdma2000, W-CDMA mode)	SPECtrum (*n*=2)[a]	no markers
CSPur - spurs close (cdmaOne mode)	SPECtrum (*n*=2)[a] ULIMit (*n*=3)[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 modes)	EVM (*n*=2)[a] MERRor (*n*=3)[a] PERRor (*n*=4)[a]	yes
OBW - occupied bandwidth (iDEN, PDC modes)	no traces	no markers
ORFSpectrum - output RF spectrum (GSM mode)	RFEModulation (*n*=2)[a] RFESwitching (*n*=3)[a]	no markers

Measurement	Available Traces	Markers Available?
PFERror - phase and frequency error (GSM mode)	PERRor (n=2)[a] PFERror (n=3)[a] RFENvelope (n=4)[a]	yes
PSTatistic - power statistics CCDF (cdma2000, W-CDMA 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 mode)	EVM (n=2)[a] MERRor (n=3)[a] PERRor (n=4)[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 -20 dB**
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)

CALibration Subsystem

These commands control the self-alignment and self-diagnostic processes.

Calibration Abort

:CALibration:ABORt

Abort any alignment in progress.

The query stops any other processing until the abort is complete.

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.

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.

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.

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.

Front Panel
Access: **System, Alignments, Align subsystem, Align ADC???**

Align Everything

: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.

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.

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 24 hours, or if the ambient temperature changes by 3 degrees. If the alignment is turned off, the instrument may drift out of specification. The alert mode allows you to turn off the automatic alignment, but have the instrument display a warning message if 24 hours has expired or the temperature has change by 3 degrees.

Factory Preset and *RST:	Off 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.

Remarks:	A valid service password needs to be entered prior to sending the command.
Front Panel Access:	**System, Alignments, Align Subsystem, RF**

Calibration Correction On/Off

:CALibration:CORRection OFF|ON|0|1

:CALibration:CORRection?

Turns on and off the corrections for RF flatness, IF flatness, attenuator error, system gain error, IF gain DAC curve (the usage of it). Also, if the corrections are turned off, the automatic IF flatness and current state gain alignment routines will not run.

Factory Preset and *RST:	On
Front Panel Access:	**System, Alignments, 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 they are off, so they do not have to update the display.

Off - displays no trace points

Low - displays every 10th trace

High - displays every trace trace

Factory Preset and *RST:	Low
Front Panel Access:	**System, Alignments, Visible Align**

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.

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.

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.

Front Panel Access:	**System, Alignments, Align Subsystem, IF**

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.

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.

Front Panel
Access: **System, Alignments, Align Subsystem, Align RF**

Align the Image Filter Circuitry

:CALibration:IMAGefilter

:CALibration:IMAGefilter?

Align the eight image filter tuning DACs.

Remarks: A valid service password needs to be entered prior to sending the command.

Front Panel
Access: **System, Diagnostics**

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 Wide LC Prefilter

:CALibration:PFILter:LCWide

:CALibration:PFILter:LCWide?

Align the wide LC prefilter. (1.2 MHz to 7.5 MHz)

Remarks:	A valid service password needs to be entered prior to sending the command.
Front Panel Access:	**System, Diagnostics**

Align the Narrow LC Prefilter

:CALibration:PFILter:LCNarrow

:CALibration:PFILter:LCNarrow?

Align the narrow LC prefilter. (200 kHz to 1.2 MHz)

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 Crystal Prefilter

:CALibration:PFILter:XTALWide

:CALibration:PFILter:XTALWide?

Align the wide crystal prefilter. (20 kHz to 200 kHz)

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 Narrow Crystal Prefilter

:CALibration:PFILter:XTALNarrow

:CALibration:PFILter:XTALNarrow?

Align the narrow crystal prefilter. (2.5 kHz to 20 kHz)

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**

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:ABSLevel?

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**

Align the RF Circuitry

:CALibration:RF

:CALibration:RF?

Performs an alignment of the RF assembly.

The query performs the alignment and returns a zero if the alignment is successful.

Remarks:	info
Front Panel Access:	**info**

Select the Source for Calibration

:CALibration:SOURce INTernal|EXTernal

:CALibration:SOURce?

Controls the source of the 50 MHz alignment signal. When it is set to INT, the 50MHz alignment signal is routed through internal circuitry to the analyzers RF input and the front panel RF INPUT connector is disconnected. When it is set to EXT, an external reference can be applied to the analyzers RF input.

Factory Preset and *RST:	Internal
Remarks:	info
Front Panel Access:	**info**

Select the Source State for Calibration

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

:CALibration:SOURce:STATe?

Controls the source of the 50 MHz alignment signal.

Factory Preset and *RST:	Off
Remarks:	info
Front Panel Access:	**info**

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.

Front Panel Access:	**System**, **Alignments, Align subsystem**, **Align 50 MHz Reference**

Align the Trigger Interpolator

`:CALibration:TRIGger:INTerp`

`:CALibration:TRIGger:INTerp?`

Align the partial sample trigger interpolator. This same alignment is run as part of the CAL:ALL routine.

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

:CONFigure:<measurement>

The CONFigure commands are used with several other commands and are documented in the section on the "MEASure Group of Commands" on page 187.

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.

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, Edit Title**

Display Annotation Title On/Off

`:DISPlay:ANNotation:TITLe[:STATe] OFF|ON|0|1`

`:DISPlay:ANNotation:TITLe[:STATe]?`

Turns the display of the title annotation on or off.

Front Panel Access:	**Display, Title, Title On Off**

Turn the Whole Display On/Off

`:DISPlay:ENABle OFF|ON|0|1`

Controls the updating of the display. If enable is set to off, the display will appear to “freeze” in its current state. Measurements will run faster if the instrument doesn’t have to keep updating the display.

Factory Preset and *RST:	On

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**

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**

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, 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:	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 187 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, 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 (W-CDMA mode)	CDPower (*n*=2)[a] EVM (*n*=4)[a] MERRor (*n*=5)[a] PERRor (*n*=6)[a] SPOWer (*n*=8)[a]	yes
CHPower - channel power (Basic, cdmaOne, cdma2000, W-CDMA mode)	SPECtrum (*n*=2)[a]	no markers
CSPur - spurs close (cdmaOne mode)	SPECtrum (*n*=2)[a] ULIMit (*n*=3)[a]	yes

Measurement	Available Traces	Markers Available?
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 modes)	EVM (*n*=2)[a] MERRor (*n*=3)[a] PERRor (*n*=4)[a]	yes
OBW - occupied bandwidth (iDEN, PDC modes)	no traces	no markers
ORFSpectrum - output RF spectrum (GSM mode)	RFEModulation (*n*=2)[a] RFESwitching (*n*=3)[a]	no markers
PFERror - phase and frequency error (GSM mode)	PERRor (*n*=2)[a] PFERror (*n*=3)[a] RFENvelope (*n*=4)[a]	yes
PSTatistic - power statistics CCDF (cdma2000, W-CDMA modes)	MEASured (*n*=2)[a] GAUSsian (*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 mode)	EVM (*n*=2)[a] MERRor (*n*=3)[a] PERRor (*n*=4)[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

Measurement	Available Traces	Markers Available?
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, RF envelope

n=2, I/Q waveform

m, selects the window within the view. The default is 1.

Factory Preset and *RST:	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

:FETCh:<measurement>[n]?

The FETCh? commands are used with several other commands and are documented in the section on the "MEASure Group of Commands" on page 187.

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 data output. It controls whether binary data is transferred in normal or swapped mode.

Factory Preset and *RST:	Normal

Numeric Data format

`:FORMat[:DATA] ASCii|REAL,32|REAL,64`

`:FORMat[:DATA]?`

This command changes the format of the data output, switching between different forms of ASCII and floating point binary. It specifies the format used for trace data during data transfer across any remote port. 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 a lot of time and storage space.

Real,32 (or 64) - Binary 32-bit, or 64-bit, real values in amplitude units. Transfers of real data are done in a binary block format. The block of data starts with a header that indicates how many additional data points are following in the block. Suppose the header is 512320.

— The first byte in the header (5) tells you how many additional 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 data format bytes/point, either 8 (for real 64), or 4 (for real 32). If you're 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 plotting and printing to an external device.

Print a Hard Copy

:HCOPy[:IMMediate]

The entire screen is output.

Front Panel Access:	**Print**

Screen Dump Query

:HCOPy:SDUMp:DATA? [GIF|XWD]

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 file is formatted as block data where the block of data starts with a header that indicates how many additional data points 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 header bytes (#D) first. The # tells you to read the next byte. That byte tells you how many additional bytes there are in the header. (In the above example D=3.)
- The read the next D bytes. NNN tells you the number of bytes of data there are following the header.
- Those data bytes can then be saved as a file with a .gif or .xwd suffix to use in other applications.

Factory Preset and *RST:	GIF
History:	Firmware revision A.03.28 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.

Continuous or Single Measurements

:INITiate:CONTinuous OFF|ON|0|1

:INITiate:CONTinuous?

Selects whether the trigger system is continuously initiated or not. This corresponds to continuous measurement or single measurement operation.

When set to ON, at the completion of each trigger cycle, the trigger system immediately initiates another trigger cycle.

When set to OFF, the trigger system remains in an “idle” state until CONTinuous is set to ON or an INITiate[:IMMediate] command is received. On receiving the INITiate{:IMMediate] command, it will go through a single trigger cycle, and then return to the “idle” state.

Factory Preset: ON

Front Panel
Access: **Meas Control, Measure Single Cont**

Take New Data Acquisitions

:INITiate[:IMMediate]

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.

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 *TRG command

Restart the Measurement

:INITiate:RESTart

Restarts the current measurement regardless of its current operating state. It is equivalent to:

INITiate[:IMMediate] (for single measurement mode)

ABort (for continuous measurement mode)

Front Panel Access:	**Restart**
	or
	Meas Control, Restart

INPut Subsystem

The INPut subsystem controls the characteristics of all the instrument input ports.

Input Impedance for IQ Input

`:INPut:IMPedance:IQ 50|600`

`:INPut:IMPedance:IQ?`

Select the impedance for the baseband I/Q input.

Factory Preset and *RST:	50 Ohm
Front Panel Access:	**Input, I/Q Input Z**

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 installed applications. If the optional **FULL** keyword is specified, each name is followed by its associated instrument number, also comma-separated. These instrument numbers are assigned internally and can be used with the **INST:NSELect** command.

Select Application by Number

:INSTrument:NSELect <integer>

:INSTrument:NSELect?

Select the measurement application by its instrument number. The actual available choices depends upon which applications are installed in the instrument. These instrument numbers can be identified with **INST:CATalog:FULL**.

Factory Preset and *RST:	Persistent state with factory default of 1
Range:	1 to x, where x depends upon which applications are installed.
Front Panel Access:	**Mode**

Select Application

`:INSTrument[:SELect] BASIC|SERVICE|CDMA|GSM`

`:INSTrument[:SELect]?`

Select the measurement application. The actual available choices depends upon which applications (modes) are installed in the instrument.

Once the instrument mode is selected, only the commands that are valid for that mode can be executed. SYSTem:HELP:HEADers? provides a list of the valid commands.

Basic mode - Makes basic receiver measurements

Service mode - Used only for servicing the instrument

CDMA mode - Makes cdmaOne (code division multiple access) standard measurements

GSM mode - Makes GSM (global system for mobile communications) standard measurements

Factory Preset and *RST:	Persistent state with factory default of the first installed application other than the service mode.
Front Panel Access:	**Mode**

MEASure Group of Commands

This group includes commands used to make measurements and return results. The different commands can be used to provide fine control of the overall measurement process. Most measurements should be done in single measurement mode, rather than doing the measurement continuously.

Each measurement sets the instrument state that is appropriate for that measurement. Other commands are available for each **Mode** to allow changing settings, view, limits, etc. Refer to:

SENSe:<measurement>, SENSe:CHANnel, SENSe:CORRection,
SENSe:FREQuency, SENSe:POWer, SENSe:RADio, SENSe:SNYC
CALCulate:<measurement>, CALCulate:CLIMits/DATA
DISPlay:<measurement>
TRIGger

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 Standard.

- Stops the current measurement 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.

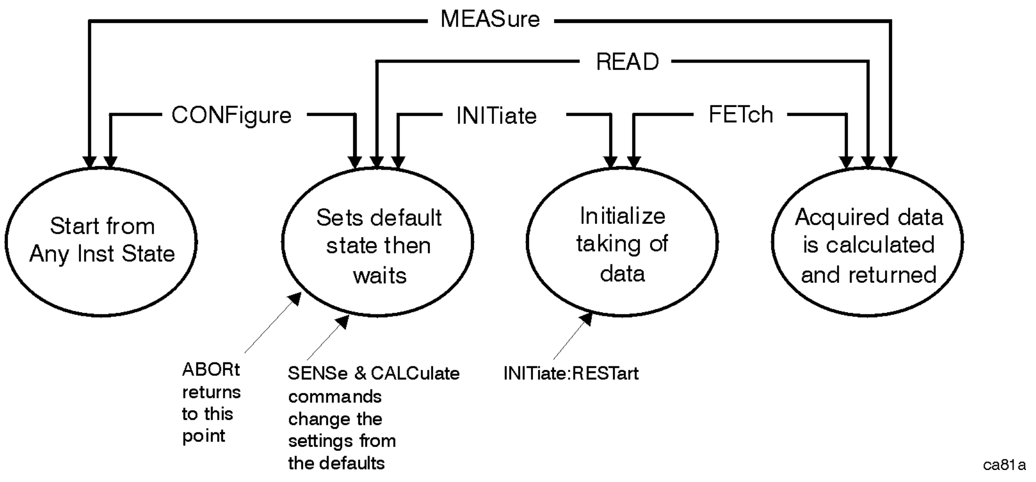
  If the optional [n] value is not included, or is set to 1, the scalar measurement results will be returned. If the [n] value is set to a value other than 1, the selected trace data results will be returned. See each command for details of what types of scalar results or trace data results are available.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

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, or the INITiate and FETCh? commands, to initiate the measurement and query the results. See Figure 5-1.

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 or INITiate and FETCh? commands, 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>?.

Figure 5-1 Measurement Group of Commands


MEASure
READ
CONFigure
INITiate
FETCh
Start from Any Inst State
Sets default state then waits
Initialize taking of data
Acquired data is calculated and returned
ABORt returns to this point
SENSe & CALCulate commands change the settings from the defaults
INITiate:RESTart
ca81a

Configure Commands

:CONFigure:<measurement>

This command sets up the instrument for the specified measurement using the factory default instrument settings and stops the current measurement. It does not initiate the taking of measurement data.

The CONFigure? query returns the current measurement name.

Fetch Commands

:FETCh:<measurement>[n]?

This command puts valid data into the output buffer, but does not initiate data acquisition. Use the INITiate[:IMMediate] command to acquire data before you use the FETCh command. You can only fetch results from the measurement that is currently selected.

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 for handling large blocks of data since they are smaller and faster then the ASCII format.

Read Commands

:READ:<measurement>[n]?

- Does not preset the measurement to the factory defaults. (The MEASure? command does preset.) It uses the settings from the last measurement.
- 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.
- 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.

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>?.

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, 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, 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.
	not specified or n=1 NADC and PDC mode	Returns 22 comma-separated scalar results, in the following order: Center freq – absolute power (dBm) Center freq – absolute power (W) Negative offset freq(1) – relative power (dB) Negative offset freq(1) – absolute power (dBm) Positive offset freq(1) – relative power (dB) Positive offset freq(1) – absolute power (dBm) . . . Positive offset freq(5) – relative power (dB) Positive offset freq(5) – absolute power (dBm)

Measurement Type	n	Results Returned
	not specified or n=1 iDEN mode	Returns 13 comma-separated scalar results, in the following order: Center freq – relative power (dB) Center freq – absolute power (dBm) Lower offset freq – relative power (dB) Lower offset freq– absolute power (dBm) Upper offset freq – relative power (dB) Upper offset freq – absolute power (dBm) Total power (dBm) Offset frequency (Hz) Reference BW (Hz) Offset BW (Hz) Carrier/center frequency (Hz) Frequency span (Hz) Average count
Total power reference	not specified or n=1 Basic, cdmaOne, cdma2000, or W-CDMA mode	Returns 24 comma-separated scalar results, in the following order: Center freq - relative power (dB) Center freq - absolute power (dBm) Center freq - relative power (dB) Center freq - absolute power (dBm) Negative offset freq(1) - relative power (dB), Negative offset freq(1) - absolute power (dBm) Positive offset freq(1) - relative power (dB) Positive offset freq(1) - absolute power (dBm) . . . Positive offset freq(5) - relative power (dB) Positive offset freq(5) - absolute power (dBm)
Power spectral density reference	not specified or n=1 Basic, cdmaOne, cdma2000, or W-CDMA mode	Returns 24 comma-separated scalar results, in the following order: Center freq - relative power (dB) Center freq - absolute power (dBm/Hz) Center freq - relative power (dB) Center freq - absolute power (dBm/Hz) Negative offset freq(1) - relative power (dB) Negative offset freq(1) - absolute power (dBm/Hz) Positive offset freq(1) - relative power (dB) Positive offset freq(1) - absolute power (dBm/Hz) . . . Positive offset freq(5) - relative power (dB) Positive offset freq(5) - absolute power (dBm/Hz)

Measurement Type	n	Results Returned
	2 NADC and PDC mode	Returns 10 comma-separated scalar values of the pass/fail (1=passed, or 0=failed) results determined by testing the absolute power of the offset frequencies: Negative offset frequency(1) absolute power Positive offset frequency(1) absolute power . . . Negative offset frequency(5) absolute power Positive offset frequency(5) absolute power
	2 iDEN mode	Returns 3 comma-separated scalar values of the histogram absolute power trace: Lower offset frequency – absolute power Reference frequency – absolute power Upper offset frequency – absolute power
Total power reference	2 Basic, cdmaOne, cdma2000, or W-CDMA mode	Returns 11 comma-separated scalar values (in dBm) corresponding to the total power histogram display. The values are returned in ascending frequency order: Negative offset frequency(5) Negative offset frequency(4) . . . Center frequency Positive Offset frequency(1) . . . Positive Offset frequency(5)
	3 NADC and PDC mode	Returns 10 comma-separated scalar values of the pass/fail (1=passed, or 0=failed) results determined by testing the relative power of the offset frequencies: Negative offset frequency(1) relative power Positive offset frequency(1) relative power . . . Negative offset frequency(5) relative power Positive offset frequency(5) relative power
	3 iDEN mode	Returns 3 comma-separated scalar values of the histogram relative power trace: Lower offset frequency – relative power Reference frequency – relative power Upper offset frequency – relative power

Measurement Type	n	Results Returned
Power spectral density reference	3 Basic, cdmaOne, cdma2000, or W-CDMA 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: Negative offset frequency(5) Negative offset frequency(4) . . . Center frequency Positive Offset frequency(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. Reference channel – absolute power Reference channel – absolute power (duplicate of above) Lower offset channel – absolute power Upper offset channel – absolute power
(For cdma2000 and W-CDMA the data is only available with spectrum display selected)	4 Basic, cdmaOne, cdma2000, or W-CDMA mode	Returns the frequency-domain spectrum trace data for the entire frequency range being measured. With the spectrum view selected (DISPlay:ACP:VIEW SPEC) and the spectrum trace on (SENSe:ACP:SPECtrum:ENABle): • In FFT mode (SENSe:ACPR:SWEep:TYPE FFT) the number of trace points returned are 343 (cdma2000 SR1), 1029 (cdma2000 SR3) or 1715 (W-CDMA). This is with the default span of 5 MHz (cdma2000 SR1), 15 MHz (cdma2000 SR3), or 25 MHz (W-CDMA). The number of points also varies if another offset frequency is set. • In sweep mode (SENSe:ACPR: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: Reference channel – relative power Reference channel – relative power (duplicate of above) Lower offset channel – relative power Upper offset channel – relative power

Measurement Type	**n**	**Results Returned**
Total power reference	5 Basic, cdmaOne, cdma2000, or W-CDMA mode	Returns 12 comma-separated scalar values (in dBm) of the absolute power of the center and the offset frequencies: Center frequency Center frequency Negative offset frequency(1) Positive offset frequency(1) . . . Negative Offset frequency(5) Positive Offset frequency(5)
Power spectral density reference	5 Basic, cdmaOne, cdma2000, or W-CDMA mode	Returns 12 comma-separated scalar values (in dBm/Hz) of the absolute power of the center and the offset frequencies: Center frequency Center frequency Negative offset frequency(1) Positive offset frequency(1) . . . Negative offset frequency(5) 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: Reference channel absolute power pass/fail Reference channel absolute power pass/fail (duplicate of above) Lower offset channel absolute power pass/fail Upper offset channel absolute power pass/fail
Total power reference	6 Basic, cdmaOne, cdma2000, or W-CDMA 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: Center frequency Center frequency Negative offset frequency(1) Positive offset frequency(1) . . . Negative offset frequency(5) Positive offset frequency(5)

Measurement Type	**n**	**Results Returned**
Power spectral density reference	6 Basic, cdmaOne, cdma2000, or W-CDMA 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: Center frequency Center frequency Negative offset frequency(1) Positive offset frequency(1) . . . Negative offset frequency(5) 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: Reference channel relative power pass/fail Reference channel relative power pass/fail (duplicate of above) Lower offset channel relative power pass/fail Upper offset channel relative power pass/fail
Total power reference	7 Basic, cdmaOne, cdma2000, or W-CDMA mode	Returns 12 comma-separated scalar values of the pass/fail (1=passed, or 0=failed) results determined by testing the absolute power limit of the center and offset frequencies (measured as total power in dB): Center frequency Center frequency Negative offset frequency(1) Positive offset frequency(1) . . . Negative offset frequency(5) Positive offset frequency(5)
Power spectral density reference	7 Basic, cdmaOne, cdma2000, or W-CDMA mode	Returns 12 comma-separated scalar values of the pass/fail (1=passed, or 0=failed) results determined by testing the absolute power limit of the center and offset frequencies (measured as power spectral density in dB): Center frequency Center frequency Negative offset frequency(1) Positive offset frequency(1) . . . Negative offset frequency(5) Positive Offset frequency(5)

Measurement Type	**n**	**Results Returned**
Total power reference	8 Basic, cdmaOne, cdma2000, or W-CDMA mode	Returns 12 comma-separated scalar values of the pass/fail (1=passed, or 0=failed) results determined by testing the power limit relative to the center frequency (measured as total power spectral in dB): Center frequency Center frequency Negative offset frequency(1) Positive offset frequency(1) . . . Negative offset frequency(5) Positive Offset frequency(5)
Power spectral density reference	8 Basic, cdmaOne, cdma2000, or W-CDMA mode	Returns 12 comma-separated scalar values of the pass/fail (1=passed, or 0=failed) results determined by testing the power limit relative to the center frequency (measured as power spectral density in dB): Center frequency Center frequency Negative offset frequency(1) Positive offset frequency(1) . . . Negative offset frequency(5) 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.t

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
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 3 scalar results: 1. RF input average amplitude 2. 50 MHz reference oscillator average amplitude 3. Average amplitude error
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 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 vs. Time Measurement [VSA-G,S ESA-G]

This measures the average power during the “useful part” of the burst comparing the power ramp to required timing mask. You must be in 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.

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 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** is the index of the data point at the start of the useful part of the burst 6. **Stop** is the index of the data point at the end of the useful part of the burst 7. **T**$_0$ is the index of the data point where t_0 occurred 8. **Burst width** is the width of the burst measured at –3dB 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. (e.g. for *n*=2, Start (for the IQ trace data) = Start + IQ_point_delta)
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).

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 select the appropriate mode using INSTrument:SELect, to use these commands.

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 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. **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. 7. **First time point** is the time of the first time domain point, where time zero is the trigger event. 8. **Time spacing** is the time spacing between the time domain points. 9. **Time domain** returns a 1, if time domain is complex (I/Q), or 0 if it is real. (raw ADC samples) 10. **Scan time** is the total scan time of the time domain trace used for the FFTThe 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.)
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.

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 select the appropriate mode using INSTrument:SELect, to use these commands.

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 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 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**.

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 Hewlett-Packard 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.

The forward slash / and the reverse slash \ are both acceptable delimiters for specifying a directory path.

Delete a File

`: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 useable instrument memory.

Front Panel Access:	**System, File System**

READ Subsystem

:READ:<measurement>[n]?

The READ? commands are used with several other commands and are documented in the section on the "MEASure Group of Commands" on page 187.

SENSe Subsystem

Sets the instrument state parameters so that you can measure the input signal.

Adjacent Channel Power Measurement

Commands for querying the adjacent channel power measurement results and for setting to the default values are found in the "MEASure Group of Commands" on page 187. The equivalent front panel keys for the parameters described in the following commands, are found under the **Meas Setup** key, after the **ACP** 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 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 mode
	20, for Basic, cdmaOne, iDEN mode
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 mode
	Exponential, for NADC, PDC, iDEN mode
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 mode to use this command. Use INSTrument:SELect to set the mode.
History:	Revision A.03.00 or later

Adjacent Channel Power—Channel Integration BW

Basic, iDEN mode

[:SENSe]:ACP:BANDwidth|BWIDth:INTegration <freq>

[:SENSe]:ACP:BANDwidth|BWIDth:INTegration?

cdmaOne, cdma2000, W-CMDA mode

[:SENSe]:ACP:BANDwidth|BWIDth[n]:INTegration[n] <freq>

[:SENSe]:ACP:BANDwidth|BWIDth[n]:INTegration[n]?

Set the Integration bandwidth that will be used for the main (carrier) channel.

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.
cdma2000 mode	n=1 is SR1, 2 is SR3 DS, and 3 is SR3 MC. The default is SR1 (1).
W-CDMA mode	n=1 is ARIB, 2 is 3GPP, and 3 is Trial. The default is ARIB (1).

Factory Preset
and *RST:

Mode	**Format (Modulation Standard)**		
Basic	1.23 MHz		
cdmaOne	1.23 MHz		
iDEN	18 kHz		
cdma2000	SR1 (n=1)	SR3 DC (n=2)	SR3 MC (n=3)
	1.23 MHz	3.69 MHz	3.69 MHz
W-CDMA	ARIB (n=1)	3GPP (n=2)	Trial (n=3)
	4.069 MHz	3.84 MHz	4.096 MHz

Range:	300 Hz to 20 MHz for Basic, cdmaOne, cdma2000, W-CDMA 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, iDEN 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 mode to use this command. Use INSTrument:SELect to set the mode.
History:	Revision A.03.00 or later

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 mode to use this command. Use INSTrument:SELect to set the mode.
History:	Revision A.03.00 or later

Adjacent Channel Power—Absolute Amplitude Limits

iDEN mode

[:SENSe]:ACP:OFFSet:ABSolute <power>

[:SENSe]:ACP:OFFSet:ABSolute?

Basic mode

[:SENSe]:ACP:OFFSet:LIST:ABSolute <power>,<power>,<power>,<power>,<power>

[:SENSe]:ACP:OFFSet:LIST:ABSolute?

cdmaOne, cdma2000, W-CDMA mode

[:SENSe]:ACP:OFFSet[n]:LIST[n]:ABSolute <power>,<power>,<power>,<power>,<power>

[:SENSe]:ACP:OFFSet[n]:LIST[n]: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. ACP:OFFS[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 SENS:ACP:OFFSet: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[n]	

cdmaOne mode n=1 is cellular bands and 2 is pcs bands. The default is cellular.

cdma2000 mode n=1 is SR1, 2 is SR3 DS, and 3 is SR3 MC. The default is SR1 (1).

W-CDMA mode n=1 is ARIB, 2 is 3GPP, and 3 is Trial. The default is ARIB (1).

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
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		50 dBm	50 dBm	50 dBm	50 dBm	50 dBm
iDEN		0 dBm	n/a	n/a	n/a	n/a

Range:	–200 dBm to 50 dBm
Default Unit:	dBm
Remarks:	You must be in Basic, cdmaOne, cdma2000, W-CDMA, 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 LOG|MAXimum|MINimum|RMS|SCALar

[: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	RMS	RMS	RMS	RMS	RMS

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 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?

cdmaOne, cdma2000, W-CDMA 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 ACP:OFFSet[n]:LIST[n][:FREQ. You can turn off (not use) specific offsets with the SENS: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.

cdma2000 mode n=1 is SR1, 2 is SR3 DS, and 3 is SR3 MC. The default is SR1 (1).

W-CDMA mode n=1 is ARIB, 2 is 3GPP, and 3 is Trial. The default is ARIB (1).

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
W-CDMA	Trial and ARIB	4.096 MHz	4.096 MHz	4.096 MHz	4.096 MHz	4.096 MHz
	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, W-CDMA mode

1 kHz to 5 MHz for iDEN mode

Default Unit: Hz

Remarks: You must be in Basic, cdmaOne, cdma2000, W-CDMA, 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	1	1	1	1	1

Range:	1 to 12
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—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	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

`[:SENSe]:ACP:OFFSet:LIST[:FREQuency] <f_offset>,<f_offset>,<f_offset>,<f_offset>,<f_offset>`

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

cdmaOne, cdma2000, W-CDMA 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 resolution 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 SENS: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.
cdma2000 mode	n=1 is SR1, 2 is SR3 DS, and 3 is SR3 MC. The default is SR1 (1).
W-CDMA mode	n=1 is ARIB, 2 is 3GPP, and 3 is Trial. The default is ARIB (1).
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 SR1	750 kHz	1.98 MHz	0 Hz	0 Hz	0 Hz
	BTS SR3 DS	2.655 MHz	3.75 MHz	0 Hz	0 Hz	0 Hz
	BTS SR3 MC	2.135 kHz	2.5 MHz	0 Hz	0 Hz	0 Hz
	MS SR1	885 kHz	1.98 MHz	0 Hz	0 Hz	0 Hz
	MS SR3 DS	2.655 MHz	3.75 MHz	0 Hz	0 Hz	0 Hz
	MS SR3 MC	2.655 kHz	3.75 MHz	0 Hz	0 Hz	0 Hz
W-CDMA		5 MHz	10 MHz	15 MHz	20 MHz	25 MHz

Range:	0 Hz to 20 MHz for iDEN, Basic mode
	0 Hz to 45 MHz for cdmaOne mode
	0 Hz to 100 MHz for cdma2000, W-CDMA mode
Default Unit:	Hz
Remarks:	You must be in Basic, cdmaOne, cdma2000, W-CDMA, 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	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 mode to use this command. Use INSTrument:SELect to set the mode.
History:	Revision A.03.00 or later

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	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—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	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 mode to use this command. Use INSTrument:SELect to set the mode.

History:	Revision A.03.00 or later

Adjacent Channel Power—Amplitude Limits Relative to the Carrier

iDEN mode

`[:SENSe]:ACP:OFFSet:RCARrier <rel_power>`

`[:SENSe]:ACP:OFFSet:RCARrier?`

Basic mode

`[:SENSe]:ACP:OFFSet:LIST:RCARrier`
`<rel_powr>,<rel_powr>,<rel_powr>,<rel_powr>,<rel_powr>`

`[:SENSe]:ACP:OFFSet:LIST:RCARrier?`

cdmaOne, cdma2000, W-CDMA mode

`[:SENSe]:ACP:OFFSet[n]:LIST[n]:RCARrier`
`<rel_powr>,<rel_powr>,<rel_powr>,<rel_powr>,<rel_powr>`

`[: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. ACP:OFFS[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 SENS:ACP:OFFSet:LIST: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.
cdma2000 mode	n=1 is SR1, 2 is SR3 DS, and 3 is SR3 MC. The default is SR1 (1).
W-CDMA mode	n=1 is ARIB, 2 is 3GPP, and 3 is Trial. The default is ARIB (1).
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
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
cdmaOne		0 dBc	0 dBc	0 dBc	0 dBc	0 dBc
W-CDMA		0 dBc	0 dBc	0 dBc	0 dBc	0 dBc

Range:	–150 dB to 50 dB for cdmaOne, Basic mode
	–200 dB to 50 dB for cdma2000, W-CDMA, iDEN mode
Default Unit:	dB
Remarks:	You must be in Basic, cdmaOne, cdma2000, W-CDMA, 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

`[:SENSe]:ACP:OFFSet:LIST:RPSDensity <rel_powr>,<rel_powr>,<rel_powr>,<rel_powr>,<rel_powr>`

`[:SENSe]:ACP:OFFSet:LIST:RPSDensity?`

cdmaOne, cdma2000, W-CDMA mode

`[:SENSe]:ACP:OFFSet[n]:LIST[n]:RPSDensity <rel_powr>,<rel_powr>,<rel_powr>,<rel_powr>,<rel_powr>`

`[: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. ACP:OFFS[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 SENS:ACP:OFFSet: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.
cdma2000 mode	n=1 is SR1, 2 is SR3 DS, and 3 is SR3 MC. The default is SR1 (1).
W-CDMA mode	n=1 is ARIB, 2 is 3GPP, and 3 is Trial. The default is ARIB (1).
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		0 dB	0 dB	0 dB	0 dB	0 dB

Range:	–150 dB to 50 dB for cdmaOne, Basic, cdma2000, W-CDMA mode –200 dB to 50 dB for iDEN mode
Default Unit:	dB
Remarks:	You must be in Basic, cdmaOne, cdma2000, W-CDMA, 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	Both	Both	Both	Both	Both

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—Control Offset Frequency List

Basic mode

[: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?

cdmaOne, cdma2000, W-CDMA 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 ACP:OFFS[n]:LIST[n]:ABS, or the relative values defined with ACP:OFFS[n]:LIST[n]:RPSD and ACP:OFFS[n]:LIST[n]:RCAR.

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.

cdma2000 mode n=1 is SR1, 2 is SR3 DS, and 3 is SR3 MC. The default is SR1 (1).

W-CDMA mode n=1 is ARIB, 2 is 3GPP, and 3 is Trial. The default is ARIB (1).

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		On	On	Off	Off	Off

Remarks: You must be in Basic, cdmaOne, cdma2000, W-CDMA, 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	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 mode to use this command. Use INSTrument:SELect to set the mode.
History:	Revision A.03.00 or later

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	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 Type of Offset Frequency List

iDEN mode

[:SENSe]:ACP:OFFSet:TEST ABSolute|AND|RELative|OR

[:SENSe]:ACP:OFFSet:TEST?

Basic mode

[:SENSe]:ACP:OFFSet:LIST:TEST ABSolute|AND|RELative|OR, ABSolute|AND|RELative|OR, ABSolute|AND|RELative|OR, ABSolute|AND|RELative|OR, ABSolute|AND|RELative|OR

[:SENSe]:ACP:OFFSet:LIST:TEST?

cdmaOne, cdma2000, W-CDMA mode

[:SENSe]:ACP:OFFSet[n]:LIST[n]:TEST ABSolute|AND|RELative|OR, ABSolute|AND|RELative|OR, ABSolute|AND|RELative|OR, ABSolute|AND|RELative|OR, ABSolute|AND|RELative|OR

[: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 ACP:OFFS[n]:LIST[n]:ABS, or the relative values defined with ACP:OFFS[n]:LIST[n]:RPSD and ACP:OFFS[n]:LIST[n]:RCAR.

You can turn off (not use) specific offsets with the SENS: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.
cdma2000 mode	n=1 is SR1, 2 is SR3 DS, and 3 is SR3 MC. The default is SR1 (1).

W-CDMA mode n=1 is ARIB, 2 is 3GPP, and 3 is Trial. The default is ARIB (1).

The types of testing that can be done for each offset include:

- 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.
- Absolute - Test the absolute power measurement. If it fails, 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.

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		REL	REL	REL	REL	REL

Remarks: You must be in Basic, cdmaOne, cdma2000, W-CDMA, 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
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 mode to use this command. Use INSTrument:SELect to set the mode.
History:	Revision A.03.00 or later

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 mode to use this command. Use INSTrument:SELect to set the mode.
History:	Revision A.03.00 or later

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, cdma2000, W-CDMA, iDEN mode to use this command. Use INSTrument:SELect to set the mode.

History:	Revision A.03.27 or later

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.

Factory Preset and *RST:	11.20 ms
Range:	1 µs to 50 ms
Default Unit:	seconds
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—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 mode to use this command. Use INSTrument:SELect to set the mode.
History:	Revision A.03.00 or later

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).

RF Burst – internal wideband RF burst envelope trigger that has automatic level control for periodic burst signals.

Factory Preset and *RST:	IMMediate for BS
	RFBurst for MS
Range:	EXT1 | EXT2 | IMM | RFB for Basic mode
Remarks:	You must be in Basic, 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 TPRef|PSDRef

[: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.

Total Power Reference - the total power is used as the power reference

Power Spectral Density Reference - the power spectral density is used as the power reference

Factory Preset and *RST:	Total power reference
Remarks:	You must be in the Basic, cdmaOne, cdma2000, W-CDMA, NADC, PDC mode to use this command. Use INSTrument:SELect to set the mode.

Channel Power Measurement

Commands for querying the channel power measurement results and for setting to the default values are found in the "MEASure Group of Commands" on page 187. 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
Range:	1 to 10,000
Remarks:	You must be in the cdmaOne, cdma2000, W-CDMA, 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, 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, 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, SR1 of cdma2000
	3.69 MHz for SR3 of cdma2000
	5 MHz for W-CDMA
Default Unit:	Hz
Remarks:	You must be in the cdmaOne, cdma2000, W-CDMA, 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 MHz for Basic, cdma2000, SR1 of cdma2000
	5 MHz for SR3 of cdma2000
	6 MHz for W-CDMA
Range:	1 kHz to 10 MHz
Default Unit:	Hz
Remarks:	You must be in the cdmaOne, cdma2000, W-CDMA, 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, 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.

Auto - couples the Data Points to the Integration BW.

Manual - the Data Points is uncoupled from the Integration BW.

Factory Preset and *RST:	Auto
Remarks:	You must be in the cdmaOne, cdma2000, W-CDMA, 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:	625 μs (1 slot)
Range:	500 μs to 10 ms
Default Unit:	seconds
Remarks:	You must be in Basic, cdmaOne 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 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]|EXTernal 2|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 (Free Run)
Remarks:	You must be in the cdmaOne, cdma2000, W-CDMA, or Basic mode to use this command. Use INSTrument:SELect to set the mode.

Correction for Base Station RF Port External Attenuation

[:SENSe]:CORRection:BS[:RF]:LOSS <rel_power>

[:SENSe]:CORRection:BS[:RF]:LOSS?

Set the correction equal to the external attenuation used when measuring base stations.

Factory Preset and *RST:	0 dB
Range:	0 to 100 dB for cdmaOne
	–50 to 50 dB for Basic, iDEN, NADC or PDC
Default Unit:	dB
Remarks:	You must be in the Basic, iDEN, cdmaOne, NADC or PDC mode to use this command. Use INSTrument:SELect to set the mode.
	Value is global to the current mode.

Select the Input Port

[:SENSe]:FEED IQ|RF|IFALign|AREFerence

Select the input port.

IQ is the IQ Input port

RF in the RF INPUT port

IF Align is the IF alignment signal source (internal, 321.4 MHz)

Amplitude Reference is the internal amplitude reference source (50 MHz)

Factory Preset and *RST:	RF
Remarks:	info

Center Frequency

[:SENSe]:FREQuency:CENTer <freq>

[:SENSe]:FREQuency:CENTer?

Set the center frequency.

Factory Preset

and *RST:	1.00 GHz
	942.6 MHz for GSM
	806 MHz for iDEN
Range:	1 kHz to 4.321 GHz
Default Unit:	Hz
Front Panel Access:	**FREQUENCY/Channel, Center Freq**

Center Frequency Step Size Automatic

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

`[:SENSe]:FREQuency:CENTer:STEP:AUTO?`

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

Factory Preset and *RST:	On
History:	Version A.03.00 or later
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 MHz
	1.25 MHz for SR1 of cdma2000
Range:	1 kHz to 1 GHz, in 10 kHz steps
Default Unit:	Hz
History:	Version A.03.00 or later
Front Panel Access:	**FREQUENCY/Channel, Center Freq**

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
	12.0 dB for iDEN
Range:	0 to 40 dB
Default Unit:	dB
Front Panel Access:	**Input, Input Atten**

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 to 80 dBm for GSM
	–100 to 27.7 dBm for cdmaOne, iDEN
	–200 to 50 dBm for NADC, PDC
	–200 to 100 dBm for cdma2000, W-CDMA
Default Unit:	dBm
Remarks:	Global to the current mode. This is coupled to the RF input attenuation
	You must be in the Service, cdmaOne, GSM, NADC, PDC, cdma2000, W-CDMA mode to use this command. Use INSTrument:SELect to set the mode.
Front Panel Access:	**Input, Max Total Pwr (at UUT)**

Power vs. Time (Burst Power) Measurement

Commands for querying the power versus time measurement results and for setting to the default values are found in the "MEASure Group of Commands" on page 187. 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?`

#define help_SENS_PVT_AVER_COUN \

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:	10
Range:	1 to 10,000
Remarks:	You must be in the 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]?`

#define help_SENS_PVT_AVER_STAT \

Turn averaging on or off.

Factory Preset and *RST:	Off
Remarks:	You must be in the 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 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|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.

RMS - The power is averaged, providing the rms of the voltage.

Factory Preset and *RST:	RMS
Remarks:	You must be in GSM or Service mode to use this command. Use INSTrument:SELect to set the mode. We are using this command somewhat differently than defined in SCPI Manual. We use this command to specify what data type is being averaged (LPOW or POW). The SCPI manual defines a SCALar average; and would require a separate command to select the type of data to which the scalar averaging is applied.

Power vs. Time—Resolution BW [VSA-G,S, ESA-G]

[: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 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 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 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]|EXTernal 2|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).

RF Burst - internal wideband RF burst envelope trigger that has automatic level control for periodic burst signals.

Factory Preset and *RST:	RF burst
Remarks:	You must be in the GSM or Service mode to use this command. Use INSTrument:SELect to set the mode.

Reference Oscillator External Frequency

`[:SENSe]:ROSCillator:EXTernal:FREQuency <frequency>`

`[:SENSe]:ROSCillator:EXTernal:FREQuency?`

Set to the frequency of the external reference oscillator being supplied to the instrument. Switch to the external reference with ROSC:SOUR.

Option oscillator commands, if applicable, are found as SENSe:OPTion:ROSCillator. (ESA?)

Preset and *RST:	Persistent state with factory default of 10 MHz
Range:	1 MHz to 40 MHz, in steps of >100 Hz. There is a ‘special case’ frequency of 19.6608 MHz
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.

Option oscillator commands, if applicable, are found as SENSe:OPTion:ROSCillator. (ESA?)

Preset and *RST:	Persistent State with factory default of On
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.

Option oscillator commands, if applicable, are found as SENSe:OPTion:ROSCillator. (ESA?)

Internal - uses internal 50 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 the "MEASure Group of Commands" on page 187. 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.

Factory Preset and *RST:	Auto
Remarks:	To use this command, the appropriate mode should be selected with INSTrument:SELect.

Spectrum—ADC Range

[:SENSe]:SPECtrum:ADC:RANGe AUTO|APEak|APLock|M6|P0|P6|P12|P18|P24|

[: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 - 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 - 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.

- M6 - 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:	Auto peak
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—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 Auto

[:SENSe]:SPECtrum:BANDwidth|BWIDth:PFFT:AUTO OFF|ON|0|1

[:SENSe]:SPECtrum:BANDwidth|BWIDth:PFFT:AUTO?

Select auto or manual control of the pre-FFT BW. This is an advanced control that normally does not need to be changed.

Auto - couples the pre-FFT BW to the frequency span.

Manual - the pre-FFT BW is uncoupled from the frequency span.

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 kHz, for iDEN mode
Range:	1 Hz to 10 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- 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 top
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 kHz
	250 Hz, for iDEN mode
Range:	0.10 Hz to 3 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]?

Set the amount of data decimation done by the hardware and/or the software. 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 1000, where 0 sets the function to automatic
Remarks:	To use this command, the appropriate mode should be selected with INSTrument:SELect.
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:	4096
	32768, for iDEN mode
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 Length Auto

`[:SENSe]:SPECtrum:FFT:LENGth:AUTO OFF|ON|0|1`

`[:SENSe]:SPECtum: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.

Auto - the window lengths are coupled to resolution bandwidth, window type (FFT), pre-FFT bandwidth (sample rate) and SENSe:SPECtrum:FFT:RBWPoints.

Manual - lets you set `SENSe:SPECtrum:FFT:LENGth` and `SENSe:SPECtrum:FFT:WINDow:LENGth`.

Factory Preset and *RST:	Auto
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—Window Delay

`[: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 +10.0s
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
	5648, for iDEN mode
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

Flat Top - 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:	Flat top
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 MHz
	100 kHz for iDEN mode
Range:	10 Hz to 10 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 <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
	15.059 ms, for iDEN mode
Range:	100 ns to 10 s
Default Unit:	seconds
Remarks:	NOTE: 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]|EXTernal 2|FRAMe|IF|LINE|IMMediate|RFBurst`

`[:SENSe]:SPECtrum: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

Line - internal line trigger

Immediate - the next data acquisition is immediately taken (also called free run)

RF Burst - internal wideband RF burst envelope trigger that has automatic level control for periodic burst signals

Factory Preset and *RST:	Immediate (free run)
	RF burst, 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 the “MEASure Group of Commands” on page 187. 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]?

This is an Advanced control that normally does not need to be changed.

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

[:SENSe]:WAVeform:ADC:RANGe AUTO|APEak|APLock|GROund|M6|P0|P6|P12|P18|P24|

[: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 - automatically peak the range

Auto Peak Lock - automatically peak lock the range

Ground - ground

M6 - subtracts 6 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—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 kHz, for NADC, PDC, cdma2000, W-CDMA, basic, service mode 500 kHz, for GSM mode 2 MHz. for cdmaOne
Range:	1 kHz to 5 MHz
Remarks:	To use this command, the appropriate mode should be selected with INSTrument:SELect.

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.

Flat top - 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.

Decimation of Waveform Display

[:SENSe]:WAVeform:DECimate[:FACTor] <integer>

[:SENSe]:WAVeform:DECimate[:FACTor]?

Set the amount of data decimation done by the hardware and/or the firmware. 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.

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

RF Burst - internal 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
	RF burst, for GSM, iDEN mode
Remarks:	To use this command, the appropriate mode should be selected with INSTrument:SELect.

STATus Subsystem

The STATus subsystem controls the SCPI-defined status-reporting structures.

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.

Operation Enable

:STATus:OPERation:ENABle <number>

:STATus:OPERation:ENABle?

This command determines what bits in the Operation Condition Register will set bits in the Operation Event register, which also sets 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, 1 or more bits need to be set to 1. There is little reason to have any bits enabled for typical manufacturing tests. Enabling bits in this register would be of more value during test development.

Factory Preset and *RST:	0, factory default. (The user setting is persistant.)
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 equivalent 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 data is cleared.

Operation Negative Transition

:STATus:OPERation:NTRansition <number>

:STATus:OPERation:NTRansition?

This command determines what bits in the Operation Condition register will set the corresponding bit in the Operation Event register when that 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.

Factory Preset and *RST:	0, factory default. (The user setting is persistant.)
Range:	0 to 32767

Operation Positive Transition

:STATus:OPERation:PTRansition <number>

:STATus:OPERation:PTRansition?

This command determines what bits in the Operation Condition register will set the corresponding bit in the Operation Event register when that 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.

Factory Preset and *RST:	32767 (all 1's), factory default. (The user setting is persistant.)
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.

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.

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.

Factory Preset and *RST:	32767 (all 1's), factory default. (The user setting is persistant.)
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 equivalent 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 data is cleared.

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 that 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.

Factory Preset and *RST:	0, factory default. (The user setting is persistant.)
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 that 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.

Factory Preset and *RST:	32767 (all 1's), factory default. (The user setting is persistant.)
Range:	0 to 32767

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.

Questionable Enable

:STATus:QUEStionable:ENABle <number>

:STATus:QUEStionable:ENABle?

This command determines what bits in the Questionable Condition Register will set bits in the Questionable Event register, which also sets 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 to have all bits in this enable register set to 0. To have any Questionable Events reported to the Status Byte Register, 1 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, there was some kind of condition during the test, that might make the test results invalid. If it is equal to 0, this indicates that no hardware problem, or measurement problem was detected by the analyzer.

Factory Preset and *RST:	0, factory default. (The user setting is persistant.)
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 equivalent 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 data is cleared.

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.

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.

Factory Preset and *RST:	32767 (all 1's), factory default. (The user setting is persistant.)
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 equivalent 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 data is cleared.

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 that 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.

Factory Preset and *RST:	0, factory default. (The user setting is persistant.)
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 that 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.

Factory Preset and *RST:	32767 (all 1's), factory default. (The user setting is persistant.)
Range:	0 to 32767

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.

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.

Factory Preset and *RST:	32767 (all 1's), factory default. (The user setting is persistant.)
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 equivalent 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 data is cleared.

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 that 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.

Factory Preset and *RST:	0, factory default. (The user setting is persistant.)
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 that 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.

Factory Preset and *RST:	32767 (all 1's), factory default. (The user setting is persistant.)
Range:	0 to 32767

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.

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.

Factory Preset and *RST:	32767 (all 1's), factory default. (The user setting is persistant.)
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 equivalent 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 data is cleared.

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 that 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.

Factory Preset and *RST:	0, factory default. (The user setting is persistant.)
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 that 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.

Factory Preset and *RST:	32767 (all 1's), factory default. (The user setting is persistant.)
Range:	0 to 32767

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 that 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.

Factory Preset and *RST:	0, factory default. (The user setting is persistant.)
Range:	0 to 32767

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.

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.

Factory Preset and *RST:	32767 (all 1's), factory default. (The user setting is persistant.)
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 equivalent 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 data is cleared.

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 that 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.

Factory Preset and *RST:	0, factory default. (The user setting is persistant.)
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 that 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.

Factory Preset and *RST:	32767 (all 1's), factory default. (The user setting is persistant.)
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 that 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.

Factory Preset and *RST:	32767 (all 1's), factory default. (The user setting is persistant.)
Range:	0 to 32767

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.

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.

Factory Preset and *RST:	32767 (all 1's), factory default. (The user setting is persistant.)
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 equivalent 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 data is cleared

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 that 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.

Factory Preset and *RST:	0, factory default. (The user setting is persistant.)
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 that 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.

Factory Preset and *RST:	32767 (all 1's), factory default. (The user setting is persistant.)
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.

Factory Preset and *RST:	Persistent State with factory default of 18
Range:	Integer, 0 to 30
Front Panel Access:	**System, Config I/O, GPIB Addr**

LAN IP Address & Names

`: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> <domain name>, and <node name> with a form as shown in the following example:

```
15.4.402.222 at35.sr.hp.com at25 betabox
```

Front Panel Access:	**System, Config I/O, Config LAN**

Hardware Configuration Query

:SYSTem:CONFigure:DEFault

Resets all instrument functions to the factory defaults, including the persistent functions. Persistent functions are system settings, such as I/O bus addresses and preset preferences, that stay at their current settings even through instrument power-on.

Table 5-1 Factory Defaults for persistent Functions

Function	Default
???	

Front Panel
Access: **System, Restore Sys Defaults**

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 1 to 12

Day - is an integer 1 to 31 (depending on the month)

Front Panel
Access: **System, Time/Date, Set Date**

Error Information Query

:SYSTem:ERRor[:NEXT]?

This command queries the earliest entry to the error queue and then deletes that entry. *CLS clears the entire error queue.

Front Panel
Access: **System, Show Errors**

Exit Main Firmware for Upgrade

:SYSTem:EXIT

Exit the main firmware to allow the firmware to be upgraded.

Front Panel
Access: **System, More, More, Install, Exit Main Firmware**

SCPI Command Help Headers Query

:SYSTem:HELP:HEADers?

Outputs a list of valid SCPI commands from the instrument.

NOTE The commands that are listed are only for the currently selected mode (e.g. Service mode, GSM mode) Use INSTrument:SELect to change the mode. The core set of system-type commands are common to all modes.

Host Identification Query

:SYSTem:HID?

Return a string that contains the host identification. This ID is required in order to obtain the license key that allows installation of a new application (mode).

Front Panel
Access: **System, More, Show System**

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). The query returns a string that contains the license key for a specified application 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.

Application – is a string that is the same as one of the enumerated items used in the INSTrument[:SELect] command. The application name must be enclosed in quotes.

License key – is a12 character alphanumeric string given to you with your application. The license key is unique to the specific application installed in the instrument with the specified serial number.

Example: **SYST:LKEY? “BAC”,“123A456B789C”**

Front Panel Access: **System, More, More, Install, License Key**

Delete a License Key

:SYSTem:LKEY:DELete <“application”>,<“license key”>

Allows you to delete the license key, for the selected application, from instrument memory.

NOTE Do not delete the license key number. 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 a12 character alphanumeric string given to you with your application

Front Panel Access: **None**

Service Password

`:SYSTem:PASSword[:CENable]<string>`

Enables access to the service functions by means of the password.

Front Panel
Access: **System, More, More, Service**

Preset

`:SYSTem:PRESet`

Returns the instrument to a set of defined conditions. This command does not change any persistent parameters.

Front Panel
Access: **Preset**

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 0 to 23.

Minute - must be an integer 0 to 59.

Second - must be an integer 0 to 59.

Front Panel
Access: **System, Time/Date, Set Time**

Adjust Time

:SYSTem:TIME:ADJust <seconds>

Adjust the time by the positive or negative value entered, within the allowed range.

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:	Added 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 Trigger Source selection is local to each measurement, so there is a separate Trigger Source command for each measurement, which is in the SENSe subsystem. The equivalent front panel keys for the parameters described in the following commands, can be found under the **Mode Setup, Trigger** key. These functions can also be found under the **Mode Setup** key, after you have selected the **Spectrum** or **Waveform** measurement from the **Measure** menu.

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.

Factory Preset and *RST	Off
	On for cdma2000, W-CDMA, NADC, PDC
Front Panel Key Access	**Mode Setup, Trigger, Auto Trigger**

Automatic Trigger Time

`:TRIGger[:SEQuence]:AUTO[:TIME] <number>`

`:TRIGger[:SEQuence]:AUTO[:TIME]?`

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.

Factory Preset and *RST:	100 ms
Range:	1 ms to 1000 s
	0 to 1000 s for cdma2000, W-CDMA
Default Unit:	Seconds

Front Panel External Trigger Delay Value

`:TRIGger[:SEQuence]:EXTernal[1]:DELay <time>`

`:TRIGger[:SEQuence]:EXTernal[1]:DELay?`

Set the amount of trigger delay when using the front panel external trigger input. Set the trigger value to zero (0) seconds to turn off trigger delay.

Factory Preset and *RST:	0 s
Range:	–500 ms to 500 ms
	–100 ms to 500 ms for cdma2000, W-CDMA
Default Unit:	seconds
Front Panel Access:	**Mode Setup, Trigger, Ext Front, Delay**

Front Panel External Trigger Level

:TRIGger[:SEQuence]:EXTernal[1]:LEVel <level>

:TRIGger[:SEQuence]:EXTernal[1]:LEVel?

Set the trigger level when using the front panel external 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 Front, Level**

Front Panel External Trigger Slope

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

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

Sets the triggering to occur on a positive-going edge or a negative-going edge of the trigger when using the front panel external trigger input.

Factory Preset and *RST:	Positive
Front Panel Access:	**Mode Setup, Trigger, Ext Front, Slope**

Rear Panel External Trigger Delay

:TRIGger[:SEQuence]:EXTernal2:DELay <time>

:TRIGger[:SEQuence]:EXTernal2:DELay?

Set the trigger delay when using the rear panel external trigger.

Factory Preset and *RST:	0 s
Range:	–500 ms to 500 ms
	–100 ms to 500 ms for cdma2000, W-CDMA
Default Unit:	seconds
Front Panel Access:	**Mode Setup, Trigger, Ext Rear, Delay**

Rear Panel External Trigger Level

:TRIGger[:SEQuence]:EXTernal2:LEVel <level>

:TRIGger[:SEQuence]:EXTernal2:LEVel?

Set the trigger level when using the rear panel external 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**

Rear Panel External Trigger Slope

:TRIGger[:SEQuence]:EXTernal2:SLOPe POSitive|NEGative

:TRIGger[:SEQuence]:EXTernal2:SLOPe?

Sets the trigger slope when using the rear panel external trigger input.

Factory Preset and *RST:	Positive
Front Panel Access:	**Mode Setup, Trigger, Ext Rear, Slope**

Frame Trigger Adjust

:TRIGger[:SEQuence]:FRAMe:ADJust <time>

:TRIGger[:SEQuence]:FRAMe:ADJust?

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 s
Range:	0 to 10 s
Default Unit:	seconds
Front Panel Access:	**Mode Setup, Trigger, Frame, Offset**

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 µs for Basic, cdmaOne
	4.615383 ms, for GSM
	26.666667 ms for cdma2000
	10.0 ms (1 radio frame) for W-CDMA
	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 ms to 559 ms for Basic, cdmaOne, GSM, cdma 2000, W-CDMA
	1 ms to 559 ms for iDEN, NADC, PDC
Default Unit:	Seconds
Front Panel Access:	**Mode Setup, Trigger, Frame Timer, Period**

Frame Trigger Sync Mode

:TRIGger[:SEQuence]:FRAMe:SYNCmode EXTFront|EXTRear|OFF

:TRIGger[:SEQuence]:FRAMe:SYNCmode?

Selects the input port location for the external frame trigger that you are using.

Factory Preset and *RST:	Off
Front Panel Access:	**Mode Setup, Trigger, Frame, Sync Source**

Frame Trigger Syncronization Offset

:TRIGger[:SEQuence]:FRAMe:SYNCmode:ADJust <time>

Lets you adjust the frame triggering with respect to the external trigger input that you are using.

Factory Preset and *RST:	0 s
Range:	0 to 10 s
	33 ns to 559 ms for cdma2000, W-CDMA
Default Unit:	seconds
Remarks:	cdma2000 and W-CDMA do not allow the query form of this command.
History:	Revision A.03.27 or later
Front Panel Access:	**Mode Setup, Trigger, Frame, Offset**

Trigger Hold Off

: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 s
	20 ms for iDEN
	10 ms for NADC or PDC
Range:	0 to 500 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 s
Range:	–500 ms to 500 ms
	–100 ms to 500 ms for cdma2000, W-CDMA
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 modes
	–20.0 dBm for iDEN
	–30.0 dBm for NADC, PDC
Range:	–200 to 50 dBm
Default Unit:	dBm
Front Panel Access:	**Mode Setup, Trigger, Video (IF Envlp), Level**

Video (IF) Trigger Slope

`:TRIGger[:SEQuence]:IF:SLOPe POSitive|NEGative`

`: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 sec
Range:	–500 ms to 500 ms
	–100 ms to 500 ms for cdma2000, W-CDMA
Default Unit:	seconds
Front Panel Access:	**Mode Setup, Trigger, RF Burst, Delay**

RF Burst Trigger Level

:TRIGger[:SEQuence]:RFBurst:LEVel <percent>

: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 to 0 dB
	–200 to 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 POSitive|NEGative

: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, W-CDMA mode to use this command. Use :INSTrument:SELect to set the mode.
Front Panel Access:	**Mode Setup, Trigger, RF Burst, Slope**

6 Error Messages

Error Queues

If an error condition occurs in the instrument, it may be reported to both the history error queue (front panel display) and the SCPI error queue (remote interface). These two queues are viewed and managed separately. Some programming errors are not applicable to front panel operation, so they are only reported through the SCPI remote interface error queue.

Error messages will appear as they occur in the Status/Info bar that appears at the bottom of the display. To view error messages fully you will use keys in the **System**, **Show Errors** menu.

NOTE If there are any messages in the history error queue, the `Err` annunciator will be activated on the instrument display.

Front Panel Error Messages

Annunciators

The display annunciators show the status of some of the transmitter tester functions and indicate error conditions of the instrument. Error annunciators are shown in red text on the instrument display. Where applicable, some states will appear in green, indicating that the feature is active and performing correctly. The state will change to red if the feature fails. The following annunciators are available:

`Unlock` - This annunciator indicates that one or more of the internal phase-locked loops are unable to maintain a phase-locked state.

`Corr Off` (corrections off) - This annunciator appears when the **Corrections** softkey is set to off.

`Err` (error) - This annunciator appears when an error message is placed in the history error queue. It will persist until you use the **Clear Error Queue(s)** key to clear the history error queue.

`Ext Ref` (external reference) - The green `Ext Ref` annunciator indicates that the external reference has been selected and the instrument is locked to it. The red `Ext Ref` annunciator indicates that the external reference has been selected, but the instrument is not locked to that reference. Note that the external reference on this instrument can be set at any frequency between 1 and 30 MHz; if the entered value does not correspond to the external reference that is in use, a red `Ext Ref` annunciator will appear. Also, be aware that the value entered for the external reference frequency will persist, even after the instrument has been powered off. The user must manually enter a new value for the external reference if a different value is required, even if it corresponds with the default value. An `Ext Ref` annunciator will appear only if the external reference has been activated by the user.

ESec (even second clock) - The green ESec annunciator indicates that the external even second clock has been selected as the sync type and a sync signal is present at the even second input (rear panel Trigger In), and the measurement is using it as the demodulation sync type. The red ESec annunciator indicates that an external even second clock has been selected as the sync type but a sync signal is not present at the even second input (rear panel Trigger In). In this case, the error message `Even Second Clock Missing` will appear in the Status/Info bar at the bottom of the display. The even second clock detection is updated every 2 seconds.

The History Error Queue

This queue is designed in a circular (rotating) fashion. It can hold up to 250 error messages. If the queue is full, and additional error messages arrive, the oldest errors are lost. The previously read messages are not cleared from the queue; they remain in the queue until they are overwritten by a new error message.

The history error queue information can be accessed by pressing **System**, **Show Errors**. From this menu you can choose **Top Page**, **Last Page**, **Next Page**, or **Prev Page**, to switch between pages (if there are more than 17 error messages). To empty the queue, press **Clear Error Queue(s).**

You can exit the error queue display by pressing either **ESC** or **Return**. Selecting a measurement under the **Measure** key will also exit the error queue display.

Error Message Format

Error messages will appear (in the format described below) in the Status/Info bar that appears at the bottom of the display. Generally the most recent message will appear, however there are occasions when an error message that has a higher priority will appear instead of the most recent one.

Error messages appear in the following format:

```
<error number><error message><context-specific information><occurrences>
```

`<error number>` - unique numeric identifier
(refer to the Error Message Descriptions section)

`<error message>` - generic description

`<context-specific information>` - (optional) additional information about this particular occurrence of the error

`<occurrences>` - Many repetitive type errors are counted rather than being individually logged. Occurrences (enclosed in parentheses) show the number of times the error has occurred since the queue was last cleared.

SCPI Remote Interface Error Messages

Remote Error Queue

This queue is constructed in a linear first-in/first-out fashion. It can hold up to 30 error messages. As errors and events are detected, they are placed in the queue. Unlike the history error queue, errors in this queue are not overwritten by the latest incoming error messages. If the queue overflows, the last error in the queue is replaced with the error:

```
(-350) Queue overflow
```

When the queue overflows, the early errors remain in the queue, and the most recent error is discarded. Reading an error from the beginning of the queue removes that error from the queue, and opens a position at the end of the queue for a new error, if one is subsequently detected.

The queue overflow message remains in the queue until it is read. If errors continue to occur as the queue is read, the `Queue Overflow` message will be followed by as many of the new messages as will fit in the remaining queue space. If the queue fills again and another error occurs, another `Queue Overflow` message will be placed in the queue.

Querying the Error Queue

The `SYSTem:ERRor[:NEXT]?` query is a request for the next entry from the instrument's error queue. The instrument responds to the query with the next error number in the queue and its description in the format:

```
<error number><error message><context-specific information>
```

The `<error number>` is a unique error identifier in the range from –32768 to 32767. A negative error value indicates a general SCPI programming error, while a positive error is more instrument specific. An error value of zero indicates that no error or event has occurred. Short descriptions of the standard error numbers are described in this section. The `<context-specific information>` section of the error message may contain information which allows you to determine the exact error and context. For example:

```
Invalid suffix; FREQuency:CENT 2.0E+5 dBmV
```

The maximum string length of the `<error message>` including the `<context-specific information>` is 255 characters. The `<error message>` will be sent exactly as indicated in this document, including case. In this example, the context-specific information was the `FREQ:CENT` command.

If there has been more than one error, the instrument will respond with the first one in the queue. Subsequent responses to `SYSTem:ERRor?` will return errors until the queue is empty.

Clearing the Error Queue

The error queue will only be cleared upon:

- power up
- receipt of a `*CLS` command
- reading the last error from the queue

No Error

When all the errors have been read from the queue, further error queries will return:

```
(0) No error
```

This message indicates that the error queue contains no errors.

(Number)	**Description**
`(0)`	`No error` The queue is empty. Every error in the queue has been read or the queue was purposely cleared by power-on or `*CLS`.

Error Message Descriptions

Messages with No Numbers

Unnumbered messages are for operator information only and do not appear in any error queue.

Description

`Acquiring Data...`

A warning used when the data acquisition time is long enough to be noticeable.

`AFUN not implemented`

`Awaiting Trigger, no AUTO Trig`

Auto Trig is off and a trigger has not been detected for more than 4 seconds.

`Break freq > FFT filter edge - clipping to %f kHz`

`Correction off`

`Data Acquisition FIFO_OVERFLOW, use AUTO DataPacking..."`

Data acquisition malfunction; need to use auto data packing to resolve.

`GSM Hopping enabled, waiting for valid burst`

When GSM hopping is enabled, this indicates that a valid GSM burst has not yet been found.

`IF synthesizer unlocked`

`LAN external loopback test failed`

This message will appear during boot-up if the instrument is not connected to a LAN cable. You can ignore the message if you are not using the LAN.

`Please wait - Printing`

Waiting for the print job to complete.

`Settling Hardware...`

A warning used when the hardware settling time is long enough to be noticeable.

`Sync is RF ampl (not Training Seq). Bits not accurate.`

Query Error Messages [–499 to –400]

An error number in the range [–499 to –400] indicates the instrument has found a problem when trying to respond to a SCPI query. The occurrence of any error in this class will cause the error query bit (bit 2) to be set in the event status register. If a query error occurs one of the following is true:

- An attempt is being made to read data from the output queue when no output is either present or pending.
- Data in the output queue has been lost.

Query Error Message Descriptions

(Number)	Description
`(-440)`	`Query UNTERMINATED after indefinite response` Indicates that a query was received in the same program message after a query requesting an indefinite response was executed (see IEEE 488.2, 6.3.7.5).
`(-430)`	`Query DEADLOCKED` Indicates that a condition causing a DEADLOCKED query error occurred (see IEEE 488.2, 6.3.1.7) (for example, both the input buffer and the output buffer are full and the device cannot continue).
`(-420)`	`Query UNTERMINATED` Indicates that a condition causing an UNTERMINATED query error occurred (see IEEE 488.2, 6.3.2.2) (for example, the device was addressed to talk and an incomplete program message was received).
`(-410)`	`Query INTERRUPTED` Indicates that a condition causing an INTERRUPTED query error occurred (see IEEE 488.2, 6.3.2.7) (for example, a query was followed by DAB or GET before a response was completely sent).
`(-400)`	`Query Error` This is a generic query error for devices that cannot detect more specific errors. The code indicates only that a query error as defined in IEEE 488.2, 11.5.1.1.7 and

6.3 has occurred.

Device-Specific Error Messages [–399 to –300]

An error number in the range [–399 to –300] indicates that the instrument has detected an error where some device operations did not properly complete, possibly due to an abnormal hardware or firmware condition. This is not a error in response to a SCPI query or command, or command execution. These errors are also used for self-test response errors. The occurrence of any error in this class will cause the device-specific error bit (bit 3) in the event status register to be set.

Device-Specific Error Message Descriptions

(Number)	Description
(-362)	`Framing error in program message` Indicates that a stop bit was not detected when data was received (for example, a baud rate mismatch).
(-361)	`Parity error in program message` Indicates that the parity bit was not correct when data was received (for example, an incorrect parity bit on a serial port).
(-360)	`Communication error` This is the generic communication error for devices that cannot detect more specific errors.
(-350)	`Queue overflow` This is a specific code entered into the queue in lieu of the code that caused the error. This message indicates that there is no more room in the queue and an error occurred but was not recorded.
(-340)	`Calibration failed` Indicates that the device has detected a failure during its calibration procedure.
(-330)	`Self-test failed` Indicates that the device has detected a failure during its self-test procedure.

(-321) `Out of memory`

Indicates that an internal operation needed more memory than was available.

If this occurs during a memory catalog display, it means the system did not have enough free RAM to prepare the catalog.

(-320) `Storage fault`

Indicates that the firmware detected a fault when using data storage. This error is not an indication of physical damage or failure of any mass storage element.

(-315) `Configuration memory lost`

Indicates that non-volatile configuration data saved by the device has been lost. The meaning of this error is device-dependent.

(-314) `Save/recall memory loss`

Indicates that the non-volatile data saved by the *SAV? command has been lost.

(-313) `Calibration memory lost`

Indicates that non-volatile calibration data has been lost.

(-312) `PUD memory lost`

Indicates that the protected user data saved by the *PUD command has been lost.

(-311) `Memory error`

Indicates that an error was detected in the device’s memory.

(-310) `System error`

Indicates that an error, termed “system error” by the device, has occurred.

(-300) `Device-specific error`

This is a generic device-dependent error for devices that cannot detect more specific errors. The code indicates only that a device-dependent error as defined in IEEE 488.2, 11.5.1.1.6 has occurred.

Execution Error Messages [–299 to –200]

An error number in the range [–299 to –200] indicates that an error has been detected during instrument execution. The occurrence of any error in this class will cause the execution error bit (bit 4) in the event status register to be set. If this bit is set, one of the following events has occurred:

- A <program data> element following a header was evaluated by the device as outside of its legal input range or as otherwise inconsistent with the device capabilities.
- A valid program command could not be properly executed due to some device condition.

Execution errors will be reported by the device after rounding and expression evaluation operations have been completed. Rounding a numeric data element, for example, will not be reported as an execution error.

Execution Error Message Descriptions

(Number)	Description
(-294)	`Incompatible type` Indicates that the type or structure of a memory item is inadequate.
(-293)	`Referenced name already exists` A downloaded program attempted to define an element (a variable, constant, filename, etc.) that had already been defined.
(-292)	`Referenced name does not exist` A downloaded program attempted to access an undefined element (a variable, constant, filename, etc.)
(-291)	`Out of memory` A downloaded program required more memory than was available in the instrument.
(-286)	`Program runtime error` Indicates that a runtime error was detected in a downloaded program.
(-285)	`Program syntax error` Indicates that a syntax error appears within a downloaded program. The syntax used when parsing a downloaded program is device-specific.

`(-284)`	`Program currently running` Indicates that certain operation related to programs may be illegal while the program is running (for example, deleting a running program may be illegal).
`(-283)`	`Illegal variable name` Indicates that an attempt was made to reference a nonexistent variable.
`(-282)`	`Illegal program name` Indicates that the name used to reference a program was invalid (for example, redefining an existing program, deleting a nonexistent program, or in general, referencing a nonexistent program).
`(-281)`	`Cannot create program` Indicates that an attempt to create a program was unsuccessful. This may be due to insufficient memory.
`(-280)`	`Program error` Indicates that a downloaded program-related execution error occurred. This error message is used when the device cannot detect more specific errors. The syntax used in a program and the mechanism for downloading a program is device-specific.
`(-278)`	`Macro header not found` Indicates that a syntactically legal macro label in the `*GMC?` query could not be executed because the header was not previously defined.
`(-277)`	`Macro redefinition not allowed` Indicates that the macro label defined in the `*DMC` command could not be executed because the macro label was already defined (see IEEE 488.2, 10.7.6.4).
`(-276)`	`Macro recursion error` Indicates that a syntactically legal macro program data sequence could not be executed because the device found it to be recursive (see IEEE 488.2, 10.7.6.4).
`(-275)`	`Macro definition too long` Indicates that a syntactically legal macro program data sequence could not be executed because the string or block contents were too long for the device too handle (see IEEE 488.2, 10.7.6.1).

(-274)	`Macro parameter error`
	Indicates that the macro definition improperly used a macro parameter place holder (see IEEE 488.2, 10.7.3).
(-273)	`Illegal macro label`
	Indicates that the macro label defined in the `*DMC` command was a legal string syntax, but could not be accepted by the device (see IEEE 488.2, 10.7.3 and 10.7.6.2) (for example, the label was too long, the same as a common command header, or contained invalid header syntax).
(-272)	`Macro execution error`
	Indicates that a syntactically legal macro program data sequence could not be executed due to an error within the macro definition (see IEEE 488.2, 10.7.6.3).
(-261)	`Math error in expression`
	Indicates that a syntactically legal expression program data element could not be executed due to a math error (for example, a divide-by-zero was attempted). The definition of a math error is device-specific.
(-260)	`Expression error`
	Indicates that an expression data element related error occurred. This error message is used when the device cannot detect more specific errors.
(-258)	`Media protected`
	Indicates that the device or user has attempted to write to a read-only memory subsystem (msus). The definition of a protected media is device-specific.
(-257)	`File name error`
	Indicates that a legal program command or query could not be executed because a file name on the device media was in error (for example, an attempt was made to copy to a duplicate filename). The definition of what constitutes a file name error is device-specific.
(-256)	`File name not found`
	Indicates that a legal program command or query could not be executed because the file name on the device media could not be found (for example, an attempt was made to read or copy a nonexistent file). The definition of what constitutes a file not being found is device-specific.

(-255)	`Directory full`
	Indicates that a legal program command or query could not be executed because the media directory was full. The definition of what constitutes a full media directory is device-specific.
(-254)	`Media full`
	Indicates that a legal program command or query could not be executed because the media was full (for example, there was no space left on the disk). The definition of what constitutes full media is device-specific.
(-253)	`Corrupt media`
	Indicates that a legal program command or query could not be executed because of corrupt media, for instance a bad disk or incorrect disk format. The definition of what constitutes corrupt media is device-specific.
(-252)	`Missing media`
	Indicates that a legal program command or query could not be executed because of missing media, for instance no disk in the disk drive. The definition of what constitutes missing media is device-specific. If this occurs during a memory catalog display, it means the default memory system could not be located. The instrument is likely not functioning properly. Report this error to the nearest Agilent Technologies Sales and Service office. Refer to the Sales and Service Office table in the user’s guide for your instrument.
(-250)	`Mass storage error`
	Indicates that a mass storage error has occurred. This message is used when a device cannot detect more specific errors.
(-241)	`Hardware missing`
	Indicates that a legal program command or query could not be executed because of missing device hardware (for example, an option was not installed).
(-240)	`Hardware error`
	Indicates that a legal program command or query could not be executed because of a hardware problem in the device. The definition of what constitutes a hardware problem is completely device-specific. This error is used when the device cannot detect more specific errors.

(-233)	`Invalid version`
	Indicates that a legal program data element was parsed but could not be executed because the version of the data is incorrect to the device. This particular error is used when file or block data elements are recognized by the instrument, but cannot be executed for reasons of version incompatibility (for example, a non-supported file version or a non-supported instrument version).
(-232)	`Invalid format`
	Indicates that a legal program data element was parsed but could not be executed because the data format or structure is inappropriate (for example, when loading memory tables or when sending a `SYSTem:SET` parameter for an unknown instrument).
(-231)	`Data questionable`
	Indicates that the measurement accuracy is questionable.
(-230)	`Data corrupt or stale`
	Possibly invalid data. A new reading was started but not completed since last access.
(-226)	`Lists not same length`
	Attempted to use LIST structure having individual LISTs of unequal length.
(-225)	`Out of memory`
	The device has insufficient memory to perform the requested operation.
(-224)	`Illegal parameter value`
	Used where exact value, from a list of possibilities, was expected.
(-223)	`Too much data`
	Indicates that a legal program data element of block, expression or string type was received that contained more data than the device could handle due to memory or related device-specific requirements.
(-222)	`Data out of range`
	Indicates that a legal program data element was parsed but could not be executed because the interpreted value was outside the legal range defined by the device (see IEEE 488.2 11.5.1.1.5).

`(-221)` `Settings conflict`

Indicates that a legal program data element was parsed but could not be executed due to the current device state (see IEEE 488.2 11.5.1.1.5).

`(-220)` `Parameter error`

Indicates that a program data element related error has occurred. This particular error message is used if the device cannot detect more specific errors.

`(-215)` `Arm deadlock`

Indicates that the arm source for the initiation of a measurement is set to GET and a subsequent measurement query is received. The measurement cannot begin until a GET is received, but the GET would cause an INTERRUPTED error.

`(-214)` `Trigger deadlock`

Indicates that a trigger source for the initiation of a measurement is set to GET and a subsequent measurement query is received. The measurement cannot begin until a GET is received, but the GET would cause an INTERRUPTED error.

`(-213)` `Init ignored`

Indicates that a request for a measurement initiation was ignored as another measurement was already in progress.

`(-212)` `Arm ignored`

Indicates that an arming signal was received and recognized by the device, but was ignored.

`(-211)` `Trigger ignored`

Indicates that a GET, *TRG, or triggering signal was received and recognized by the device, but was ignored because of device timing considerations (for example, the device was not ready to respond).

`(-210)` `Trigger error`

Indicates that a `GET, *TRG,` or a triggering signal could not be executed due to an error.

`(-202)` `Settings lost due to rtl`

Indicates that a setting associated with a hard local control (see IEEE 488.2, 5.6.15) was lost when the device changed to LOCS from REMS or to LWLS from RWLS.

`(-201)`	`Invalid while in local`
	Indicates that a command is not executable while the device is in local mode due to a hard local control (see IEEE 488.2, 5.6.1.5) (for example, a device with a rotary switch receives a message which would change the switch's state, but the device is in local, so the message cannot be executed).
`(-200)`	`Execution Error`
	This is a generic syntax error for devices that cannot detect more specific errors. The code indicates only that an execution error as defined in IEEE 488.2, 11.5.1.1.5 has occurred.

Command Error Messages [–199 to –100]

An error number in the range [–199 to –100] indicates that an IEEE 488.2 syntax error has been detected by the instrument's parser. The occurrence of any error in this class will cause the command error bit (bit 5) in the event status register to be set. If this bit is set, one of the following events has occurred:

- An IEEE 488.2 syntax error has been detected by the parser. That is, a control-to-device message was received which is in violation of the IEEE 488.2 standard. Possible violations include a data element which violates device listening formats or whose type is unacceptable to the device.
- An unrecognized header was received. Unrecognized headers include incorrect device-specific headers and incorrect or unimplemented IEEE 488.2 common commands.

Command Error Message Descriptions

(Number)	Description
(-184)	`Macro parameter error` Indicates that a command inside the macro definition had the wrong number or type of parameters.
(-183)	`Invalid inside macro definition` Indicates that the program message unit sequence, sent with a *DDT or a *DMC command, is syntactically invalid (see IEEE 488.2, 10.7.6.3).
(-181)	`Invalid outside macro definition` Indicates that a macro parameter place holder ($<number>) was encountered outside of a macro definition.
(-180)	`Macro error` This error is generated when using a macro or executing a macro. This error message is used if the device cannot detect a more specific error.
(-178)	`Expression data not allowed` A legal expression data was encountered, but was not allowed by the device at this point in parsing.
(-171)	`Invalid expression` The expression data element was invalid (see IEEE 488.2, 7.7.7.2) (for example, unmatched parentheses or an illegal character).

	This error also occurs if a command is executed that is not valid for the current selected instrument mode. Use INSTrument:SELect to change the mode.
(-170)	`Expression data error`
	This error is generated when parsing an expression data element. This particular error message is used if the device cannot detect a more specific error.
(-168)	`Block data not allowed`
	A legal block data element was encountered, but not allowed by the device at this point in the parsing.
(-161)	`Invalid block data`
	A block data element was expected, but was invalid (see IEEE 488.2, 7.7.6.2) (for example, an END message was received before the end length was satisfied).
(-160)	`Block data error`
	This error is generated when parsing a block data element. This particular error message is used if the device cannot detect a more specific error.
(-158)	`String data not allowed`
	A string data element was encountered, but not allowed by the device at this point in the parsing.
(-151)	`Invalid string data`
	A string data element was expected, but was invalid (see IEEE 488.2, 7.7.5.2) (for example, an END message was received before the terminal quote character).
(-150)	`String data error`
	This error is generated when parsing a string data element. This particular error message is used if the device cannot detect a more specific error.
(-148)	`Character data not allowed`
	A legal character data element was encountered where prohibited by the device.
(-144)	`Character data too long`
	The character data element contains more that twelve characters (see IEEE 488.2, 7.7.1.4).

(-141)	`Invalid character data`
	Either the character data element contains an invalid character or the particular element received is not valid for the header.
(-140)	`Character data error`
	This error is generated when parsing a character data element. This particular error message is used if the device cannot detect a more specific error.
(-138)	`Suffix not allowed`
	A suffix was encountered after a numeric element which does not allow suffixes.
(-134)	`Suffix too long`
	The suffix contained more than twelve characters (see IEEE 488.2, 7.7.3.4).
(-131)	`Invalid suffix`
	The suffix does not follow the syntax described in IEEE 488.2, 7.7.3.2, or the suffix is inappropriate for this device.
(-130)	`Suffix error`
	This error is generated when parsing a suffix. This particular error message is used if the device cannot detect a more specific error.
(-128)	`Numeric data not allowed`
	A legal numeric data element was received, but the device does not accept one in this position for the header.
(-124)	`Too many digits`
	The mantissa of a decimal-numeric data element contained more than 255 digits excluding leading zeros (see IEEE 488.2, 7.7.2.4.1).
(-123)	`Exponent too large`
	The magnitude of an exponent was greater than 32000 (see IEEE 488.2, 7.7.2.4.1).
(-121)	`Invalid character in number`
	An invalid character for the data type being parsed was encountered (for example, an alpha in a decimal numeric or a “9” in octal data).

(-120)	`Numeric data error`
	This error is generated when parsing a data element which appears to be numeric, including non-decimal numeric types. This particular error message is used if the device cannot detect a more specific error.
(-114)	`Header suffix out of range`
	The value of a header suffix attached to a program mnemonic makes the header invalid.
(-113)	`Undefined header`
	The header is syntactically correct, but it is undefined for this specific device (for example, *XYZ is not defined for any device).
	The command (header) may not be valid for the current instrument mode. Use INST:SELect to change the mode.
	The command may not be valid for the current (specified) measurement. (e.g. CALC:WAV:MARK:MAX is not valid because the waveform measurement does not use the marker maximum command.)
(-112)	`Program mnemonic too long`
	The header contains more than twelve characters (see IEEE 488.2, 7.6.1.4.1).
(-111)	`Header separator error`
	A character which is not a legal header separator was encountered while parsing the header.
(-110)	`Command header error`
	An error was detected in the header. This message is used when the device cannot detect more specific errors.
(-109)	`Missing parameter`
	Fewer parameters were received than required for the header (for example, the *ESE common command requires one parameter, so receiving *ESE is not allowed).
(-108)	`Parameter not allowed`
	More parameters were received than expected for the header (for example, the *ESE common command only accepts one parameter, so receiving `*ESE 0,1` is not allowed).

(-105)	`GET not allowed`
	A Group Execute Trigger was received within a program message (see IEEE 488.2, 7.7). Correct the GPIB controller program so that the GET does not occur within a line of GPIB program code.
(-104)	`Data type error`
	The parser recognized a data element that is not allowed (for example, numeric or string data was expected, but block data was encountered).
(-103)	`Invalid separator`
	The parser was expecting a separator and encountered an illegal character (for example, the semicolon was omitted after a program message unit).
(-102)	`Syntax error`
	An unrecognized command or data type was encountered (for example, a string was received when the device does not accept strings).
(-101)	`Invalid character`
	A syntactic command contains a character which is invalid for that type (for example, a header containing an ampersand, SETUP&).
(-100)	`Command error`
	This is a generic syntax error for devices that cannot detect more specific errors. The code indicates only that a command error as defined in IEEE 488.2, 11.5.1.1.4 has occurred.

Instrument-Specific Error Messages [positive numbers]

Some instrument-specific error messages use the existing negative or "generic" SCPI error numbers with the addition of device-dependent or instrument-specific information following the semicolon in the error message. A positive error number indicates that the instrument has detected an error within the GPIB system, within the instrument firmware or hardware, during the transfer of block data, or during calibration.

An error number in the positive range indicates that the instrument has detected an error relating to the core operation [1 to 99], or to a personality loaded into the instrument, GSM [100 to 199] or CDMA [200 to 299].

Core-Specific Error Messages [1 to 99]

An error number in the range [1 to 99] indicates the instrument has detected an error relating to the core functionality of the instrument.

Core-Specific Error Message Descriptions

(Number)	Description
(1)	`Synthesizer unlocked` The A19 synthesizer assembly has lost phase lock. Suspect a problem with the A19 hardware or absence of 10 MHz from the A18 reference assembly.
(2)	`Frequency reference unlocked` The 100 MHz VCXO on the A18 reference assembly is no longer phase locked. Possible causes are; a faulty internal OCXO, the external reference bad or missing, or faulty phase lock circuitry on the A18 reference assembly.
(3)	`Third LO unlocked` The third LO on the A12 analog IF assembly has lost phase lock. Possible causes are; a faulty A12 analog IF assembly, or a missing 10 MHz from the A18 reference assembly.
(4)	`Cal oscillator unlocked` The 42.8 MHz calibrator oscillator on the A12 analog IF assembly is unlocked.

(5)	`Analog IF sample rate osc unlocked` The 30 MHz sample rate oscillator on the A12 analog IF assembly is unlocked.
(6)	`Even second clock failing` The even second clock is unlocked.
(7)	`No application file`
(8)	`Catalog incomplete`
(9)	`Application not licensed` License key "word" is not entered into instrument memory.
(10)	`Application not installed` Measurement application could not be found.
(11)	`Invalid application file` Caused by an invalid personality file.
(12)	`Application load failed` Measurement application could not load.
(13)	`Invalid trace number` Caused by an invalid trace number. Some measurements only have 1 or 2 valid traces, rather then the indicated 4.
(14)	`Trace data not ready` This may be caused by sending a command that asks for trace data from a measurement that has not finished calculating, or from a measurement that is not currently active (running).
(15)	`Measurement data not available` This may be caused by sending a command that asks for data from a measurement that is not currently active (running).

(16)	`Input Overload Decrease max total power in input.`
	Excessive input power has been detected which will cause the ADC to clip the signal. Reduce the signal level, change the attenuator/max total power setting (under **Input** menu), or press **Restart** if the **RF Input Range** is **Auto**.
(17)	`Data Acquisition forcing SHORT packing`
	The data acquisition rate is too high to use longer word packing.
(18)	`Command not implemented`
	The requested command is not implemented.
(19)	`Function not implemented`
	The requested function is not implemented.
(20)	`Signal exceeds maximum allowable power - Reduce input power`
	Excessive input power has been detected which will cause the ADC to clip the signal.
(21)	`Memory Allocation FAILURE`
(22)	`Memory limit caused Data Acquisition to be truncated`
	Caused by a Memory Allocation failure. The measurement limited the acquisition time in order to complete the measurement.
(23)	`Setup INVALID`
	The parameters chosen create a measurement request that is impossible to complete, often due to memory limitations.
(24)	`External reference missing`
	The external frequency reference signal either is missing, has too low an amplitude, or does not match the frequency value previously entered into the instrument memory by the operator.
(25)	`Even Second Clock missing`
	The even second clock signal supplied from the base station is missing. Check the external trigger input connection where the even second clock signal is fed into the instrument.

(26)	`Oven temp low`
	The oven-controlled crystal oscillator is not at the desired operating temperature.
(27)	`Alignment Needed`
	The Auto Align routine needs to be run. At least 24 hours has passed since the last full alignment, or the temperature has changed 6° C.
(28)	`Printer failure`
	Check for proper printer operation.
(29)	`Printer not available`
	The requested printer is not available. Check for proper printer hookup.
(30)	`Printer out of paper`
	Put paper in the printer.
(31)	`Data Acquisition TIMEOUT, repairs underway...”`
	Hardware malfunction, data acquisition subsystem.
(32)	`ADC Alignment Failure`
	One or more built-in alignment tests have failed.
(33)	`IF Alignment Failure`
	One or more built-in alignment tests have failed.
(34)	`RF Alignment Failure`
	One or more built-in alignment tests have failed.
(35)	`System Alignment Failure`
	One or more built-in alignment tests have failed.

GSM-Specific Error Messages [100 to 199]

An error number in the range [100 to 199] indicates the instrument has detected an error relating to the GSM personality.

GSM-Specific Error Message Descriptions

(Number)	Description
(100)	`Not enough data to fit into GSM mask` An attempt to position a GSM trace into the mask, when not enough data was present. Try using the **Restart** key to clear the problem. This can be caused by a bad GSM burst, or the RF Sync Delay set too far.
(101)	`GSM burst out of limits` The GSM signal did not fit into the mask in the Power vs. Time measurement.
(102)	`Insufficient pre-Trig for demod - decrease Trig Delay`
(103)	`Incorrect RBW for demod - change RBW`
(104)	`Invalid GSM burst timing` A GSM-like burst was acquired, but it's timing is not valid. Ensure the correct **Burst Type** has been selected.
(105)	`Valid GSM burst not found` In a GSM measurement, data was acquired but a GSM burst was not found.
(106)	`Cannot synchronize frame trigger` Cannot synchronize the frame trigger to the even second clock.
(107)	`Dynamic range not optimum - set AUTO RF input`
(108)	`Cannot synchronize to RF amplitude (burst error)`
(109)	`GSM RF sync delay is out of range` Change RF Sync Delay.

(110) `Sync word not found`

In a GSM measurement using demodulation, the training sequence code (sync word) could not be found.

(111) `Signal too noisy`

In a GSM measurement, indicates that a burst could not be found in a signal that appears noisy.

(112) `Incorrect trigger holdoff - set to 0 sec`

(113) `SCPI marker query not available in GSM Rise&Fall`

(114) `GSM Pwr Meas requires trig delay < –50us. Delay set to –50us`

CDMA-Specific Error Messages [200 to 299]

An error number in the range [200 to 299] indicates the instrument has detected an error relating to the CDMA personality.

CDMA-Specific Error Message Descriptions

(Number)	Description
(200)	`Signal near noise floor - Power accuracy degraded`
(201)	`Signal exceeds maximum allowable power - Reduce input power`
(202)	`Input overload` Excessive input power has been detected which will cause the ADC to clip the signal. Reduce the signal level, change the attenuator/max total power setting (under **Input** menu), or press **Restart** if the **RF Input Range** is **Auto**.
(203)	`Channel center frequency outside device’s transmit band`
(205)	`No power at carrier frequency` No power was detected as a CW or a modulated signal.
(206)	`Cannot correlate to input signal` A correlation failure with the pilot CDMA channel occurred during synchronous demodulation.

NADC-Specific Error Messages [300 to 399]

An error number in the range [300 to 399] indicates the instrument has detected an error relating to the NADC personality.

NADC-Specific Error Message Descriptions

(Number)	Description
(300)	`Sync word not found` In an EVM measurement, the sync word is not found and the synchronization cannot be established when **Sync Word** is selected in the **Burst Sync** menu.
(301)	`Valid NADC burst not found` A valid NADC burst is not found when the **Device** is MS.
(302)	`Signal too noisy` The valid EVM measurement cannot be performed, because the input signal is too noisy.
(303)	`Burst Delay exceeds 2 ms limit for EVM` In an EVM measurement, the **Burst Delay** value must be less than 2 ms.

PDC-Specific Error Messages [400 to 499]

An error number in the range [400 to 499] indicates the instrument has detected an error relating to the PDC personality.

PDC-Specific Error Message Descriptions

(Number)	Description
(400)	`Sync word not found` In an EVM measurement, the sync word is not found and the synchronization cannot be established when **Sync Word** is selected in the **Burst Sync** menu.
(401)	`Valid PDC burst not found` A valid PDC burst is not found when the **Device** is MS.
(402)	`Signal too noisy` The valid EVM measurement cannot be performed, because the input signal is too noisy.
(412)	`Burst Delay exceeds 2 ms limit for EVM` In an EVM measurement, the **Burst Delay** value must be less than 2 ms.

Index

Index

Index

Index

Index