Programmer's Guide

Agilent Technologies ESA Spectrum Analyzers

This guide documents firmware revision 04.00 and prior versions.

This guide provides documentation for the following instruments:

ESA-E Series

E4401B (9 kHz–1.5 GHz)
E4402B (9 kHz–3.0 GHz)
E4404B (9 kHz–6.7 GHz)
E4405B (9 kHz–13.2 GHz)
E4407B (9 kHz–26.5 GHz)

and

ESA-L Series

E4411B (9 kHz–1.5 GHz)
E4403B (9kHz–3.0 GHz)
E4408B (9 kHz–24.5 GHz)

Agilent Technologies

Manufacturing Part Number: E4401-90179
Supersedes E4401-90109

Printed in USA

February 2000

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

WARNING

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

CAUTION

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

NOTE

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

⚠	The instruction documentation symbol. The product is marked with this symbol when it is necessary for the user to refer to the instructions in the documentation.
|	This symbol is used to mark the on position of the power line switch.
⏻	This symbol is used to mark the standby position of the power line switch.
~	This symbol indicates that the input power required is AC.

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 **If this product is not used as specified, the protection provided by the equipment could be impaired. This product must be used in a normal condition (in which all means for protection are intact) only.**

Warranty

This Agilent Technologies instrument product is warranted against defects in material and workmanship for a period of three years from date of shipment. During the warranty period, Agilent Technologies 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.

Where to Find the Latest Information

Documentation is updated periodically. For the latest information about Agilent ESA Spectrum Analyzers, including firmware upgrades and application information, please visit the following Internet URL:

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

Contents

1 Programming Fundamentals

The purpose of this chapter is to serve as a reminder of SCPI (Standard Commands for Programmable Instruments) fundamentals to those who have previous experience in programming SCPI. This chapter is not intended to teach you everything about the SCPI programming language.

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

Valid ESA Spectrum Analyzer SCPI commands are used for examples in this chapter. Topics included in this chapter are:

- “Creating Valid Commands”
- “Command Notation Syntax”
- “Special Characters in Commands”
- “Putting Multiple Commands on the Same Line”
- “Overview of GPIB”
- “Overview of RS-232”
- “Printer Setup and Operation”

Creating Valid Commands

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

Command Syntax	Sample Valid Commands
`[:SENSe]:BANDwidth[:RESolution] <freq>`	The following sample commands are all identical. They will all cause the same result. • `:Sense:Band:Res 1700` • `:BANDWIDTH:RESOLUTION 1.7e3` • `:sens:band 1.7KHZ` • `:SENS:band 1.7E3Hz` • `:band 1.7kHz` • `:bandwidth:RES 1.7e3Hz`
`:MEASure:HARMonics:AMPLitude [n?]`	The last command below returns different results than the commands above it. The number 3 in the command causes this. See the command description for more information. • `:MEAS:HARM:AMPL?` • `:Meas:Harm:Ampl?` • `:MEAS:HARM:AMPL3?`
[:SENSe]:DETector[:FUNCtion] NEGative|POSitive|SAMPle	• `DET:FUNC NEG` • `:Sense:Detector:Function Sample`
:INITiate:CONTinuous OFF|ON|0|1	The sample commands below are identical. • `:INIT:CONT ON` • `:init:continuous 1`

Command Notation 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:SEQuence:VIDeo:LEVel 2.5V`

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

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

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

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:**[:SENSe]:DETector[:FUNCtion] NEGative|POSitive|SAMPle** The choices are neg, pos, and samp.. **:SENSe:DETector:FUNCtion SAMPle** 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]:ACPower:BANDwidth|BWIDth:ACHannel`** Two identical commands are: **`:SENSe:ACPower:BANDwidth:ACHannel`** **`:SENSe:ACPower:BWIDth:ACHannel`**
[ ]	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]:ACPower:AVERage[:STATe]OFF|ON|0|1`** The following commands are all valid and have identical effects: **`:SENSe:ACPower:AVERage:STATe OFF`** **`:ACPower:AVERage:STATe OFF`** **`ACPower:AVERage OFF`**
< >	Angle brackets around a word, or words, indicates they are not to be used literally in the command. They represent the needed item.	Command: **:SENSe:ACPower:CSPacing <freq>** In this command example the word **`<freq>`** should be replaced by an actual frequency: **:SENSe:ACPower:CSPacing 9.7MHz**

Special Character	Meaning	Example
{ }	Parameters in braces can optionally be used in the command either not at all, once, or several times.	Command: **[SENSe:]CORRection:CSET[1]|2|3|4:DATA:MERGe <freq>,<rel_ampl>{,<freq>,<rel_ampl>}** A valid form of this command is: **[SENSe:]CORRection:CSET1:DATA:MERGe 740000,.94 1250000,.31 3320000,1.7**

Parameters in Commands

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

Boolean

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

Key Word

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

Units

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

Variable

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

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

MINimum - sets the parameter to the smallest possible value.

MAXimum - sets the parameter to the largest possible value.

Include the key word MINimum or MAXimum after the question mark in a query in order to return the numeric value of the key word.

Example query: **`[:SENSE]:FREQuency:CENTer? MAXimum`**

Variable Parameters

<freq>

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

<time>

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

<ampl>, <rel_ampl>

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

<angle>

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 characters.

Block Program Data

Definite length arbitrary block response data is defined in section 8.7.9.2 of IEEE Standard 488.2-1992, *IEEE Standard Codes, Formats, Protocols and Common Commands for Use with ANSI/IEEE Std 488.1-1987*. New York, NY, 1992.

<definite_length_block>

It allows data to be transmitted over the system interface as a series of 8 bit data bytes. This element is particularly useful for sending large quantities of data, 8 bit extended ASCII codes, or other data that are not able to be directly displayed.

Putting Multiple Commands on the Same Line

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

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

SCPI Termination and Separator Syntax

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

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

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

The following are some examples of good and bad commands. The examples are created from an ESA spectrum analyzer with the simple set of commands indicated below:

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

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

[:SENSe]
```

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

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

Overview of GPIB

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 RS-232

Serial interface programming techniques are similar to most general I/O applications.

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 or 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.

Printer Setup and Operation

Equipment

- ESA Spectrum Analyzer equipped with Options A4H and Parallel Interface) or 1AX (RS-232 and Parallel Interface).
- IEEE 1284 compliant printer cable (such as C2950A).
- Supported printer equipped with a parallel interface. (A supported printer is one that accepts Printer Control Language Level 3 or 5).
  - — PCL3 printers include most HP DeskJet printers.
  - — PCL5 printers include most HP LaserJet printers and the 1600C DeskJet printer.

Interconnection and Setup

1. Turn off the printer and the analyzer.
2. Connect the printer to the analyzer parallel I/O interface connector using an IEEE 1284 compliant parallel printer cable.
3. If appropriate, configure your printer using configuration menus or switches. Refer to your printer's documentation for more specific information on configuring your printer.
4. Turn on the analyzer and printer.
5. Press **Print Setup** on the front panel and then press the **Printer Type** menu key. **Printer Type** accesses the following keys:

None	**None** disables the analyzer from attempting to print to a printer. This is the appropriate setting if no printer is connected to the analyzer.
Custom	**Custom** allows you to access the **Define Custom** menu keys. The **Define Custom** menu keys allow you to specify printer characteristics such as PCL Level and printer color capability.
Auto	**Auto** enables the analyzer to automatically attempt to identify the connected printer when the **Print** key is pressed or when **Printer Type** is set to **Auto**.

6. Press **Printer Type** to access the **Printer Type** menu keys. Press **Auto** to make the analyzer attempt to identify the connected printer. When you press **Auto**, the analyzer will respond in one of the three following ways:

   - The **Print Setup** menu will be displayed with the **Auto** key selected and no new message will be displayed in the display status line. This indicates that the analyzer has successfully identified the connected printer and no further setup is required. As long as **Auto** remains selected in the **Printer Type** menu, the analyzer will attempt to identify the printer when the front panel **Print** key is pressed.

   - The **Print Setup** menu will be displayed with the **Custom** key selected and one of the following diagnostic messages will be displayed in the display status line:

     `Unknown printer, Define Custom to set up printer`

     `No printer response, Define Custom to set up printer`

     `Invalid printer response, Define Custom to set up printer`

     This indicates that the analyzer was unable to automatically identify the connected printer, and **Custom** has been selected in the **Printer Type** menu. Press **Print Setup**, **Define Custom** to select specific printer characteristics such as the printer language (PCL3 or PCL5) and color printing capability. Once you have set these characteristics to match those of your connected printer, the printer setup process is complete. As long as **Custom** remains selected in the **Printer Type** menu, the analyzer will not attempt to automatically identify the connected printer when the front panel **Print** key is pressed.

   - The **Print Setup** menu will be displayed with the **None** key selected and the following message will appear in the display status line:

     `Unsupported printer, Printer Type set to None`

     This indicates that the analyzer has successfully identified the connected printer, but the printer is not supported by the analyzer. As long as **None** is selected in the **Printer Type** menu, the analyzer will respond to any print command by displaying the message **`Printer Type is None`** in the display status line.

Testing Printer Operation

When you have completed the printer setup for the analyzer, press **Print Setup**, **Print (Screen)** and then press **Print** on the front panel. If the printer is ready and the printer setup was successful, a printout of the analyzer display will be printed. If the printer is not ready, the message **`Printer Timeout`** will appear on the analyzer display. **`Printer Timeout`** will remain on the display until the printer is ready or until you press **ESC** to cancel the printout request.

2 Status Registers

This chapter contains a comprehensive description of status registers explaining what status registers are and how to use them. Information pertaining to all bits of the registers in Agilent ESA analyzers is also provided.

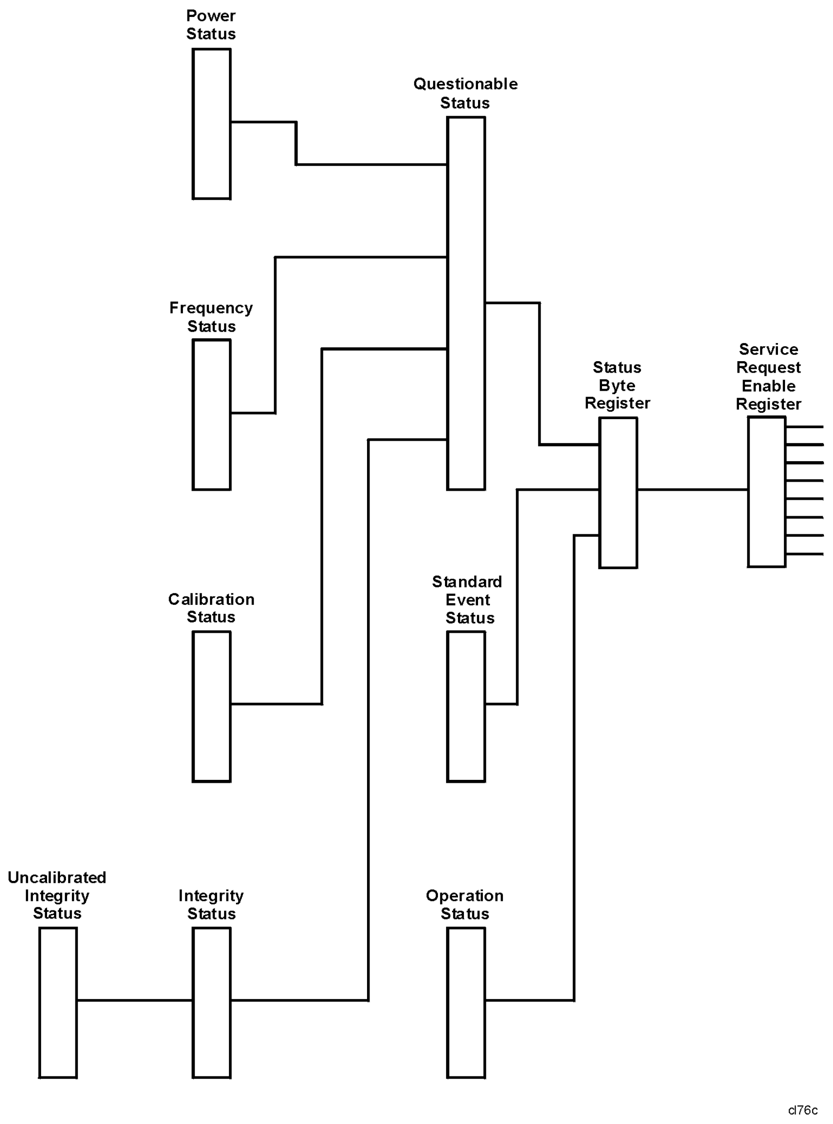
Use Status Registers to Determine the State of Analyzer Events and Conditions

Programs often need to detect and manage error conditions or changes in analyzer status. Agilent ESA products allow this function to be performed using status registers. You can determine the state of certain analyzer hardware and firmware events and conditions by programming the status register system.

Refer to Figure 2-1. The status system is comprised of multiple registers arranged in a hierarchical order. The service request enable register is at the top of the hierarchy and contains the general status information for the analyzer events and conditions. The lower-priority status registers propagate their data to the higher-priority registers in the data structures by means of summary bits. These registers are used to determine the states of specific events or conditions.

The two methods used to programmatically access the information in status registers are the polling method and the service request method. An explanation of these methods is given in the next section "What are the Status Registers?"

Figure 2-1 Status Register System Simplified Block Diagram



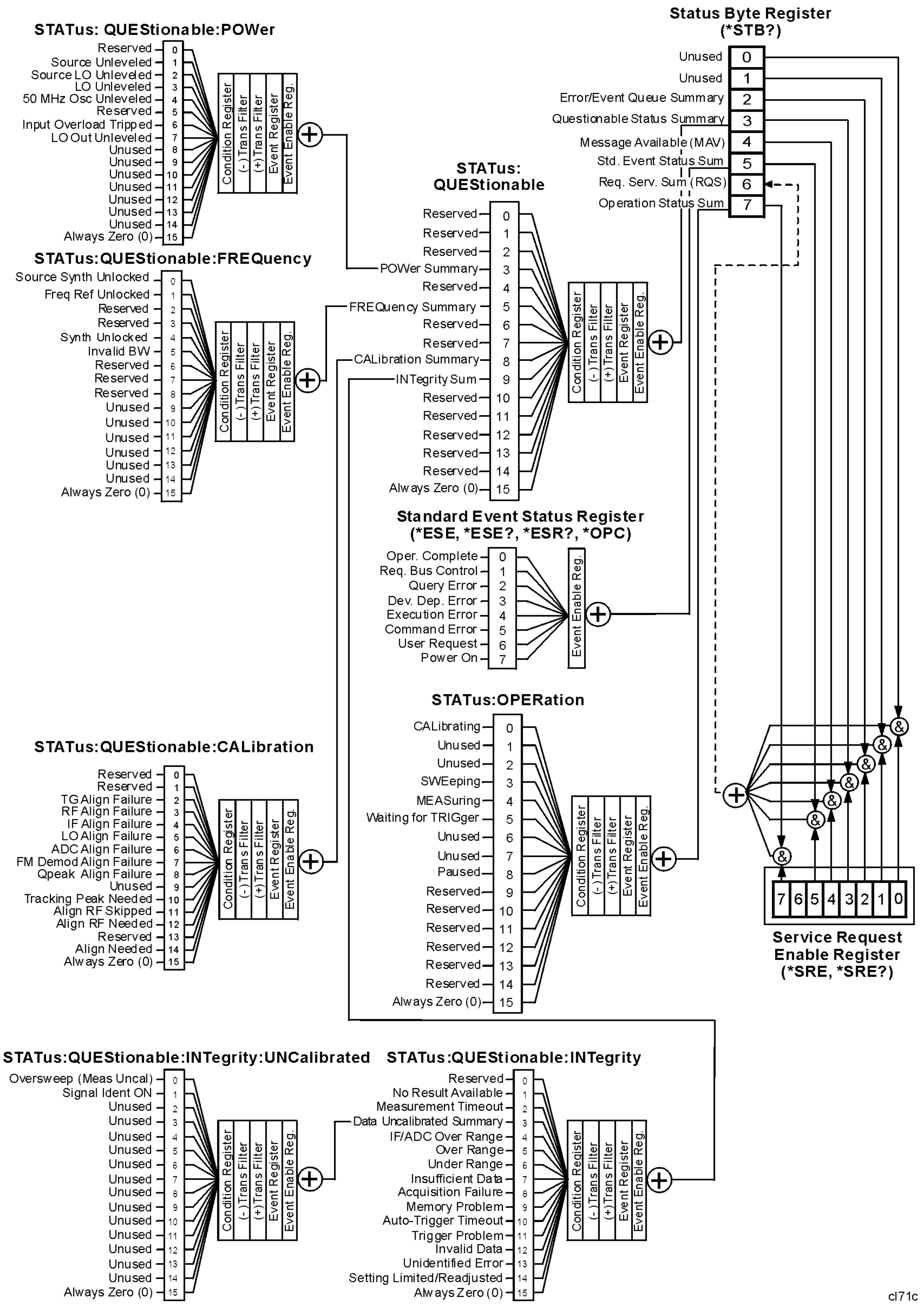
What are the Status Registers?

Refer to Figure 2-2, which shows the overall status register system in detail. Most status registers are composed of the five individual registers described below. One such status register in the figure is entitled “STATus: QUEStionable,” which is both the name of the register, and the SCPI command form used to access the register. From now on, the SCPI command form will be used when referring to the various registers. There are IEEE common SCPI commands noted under some register names in parenthesis. These commands are associated with those registers, and their effects are described under “How Do You Access the Status Registers?” in this chapter, and in the beginning of Chapter 5, “Language Reference” in this guide.

Refer to the right-hand part of the STATus: QUEStionable register while reading the following register descriptions.

Condition Register	A condition register continuously monitors the hardware and firmware status of the analyzer. There is no latching or buffering for a condition register.
Negative Transition Filter	A negative transition filter 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 Filter	A positive transition filter 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 Register	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 Register	An event enable register specifies the bits in the event register that can generate a summary bit. Summary bits are, in turn, used by the status byte register.

Figure 2-2 Overall Status Register System



Status registers (except for the status byte register and the standard event status register) consist of the registers whose contents can be used to produce status summary bits.

These summary bits are then manipulated as follows: The condition register passes summary bits to the negative and positive transition filters, after which they are stored in the event register. The contents of the event register are logically ANDed with the contents of the event enable register and the result is logically ORed to produce a status summary bit. The status summary bit is then passed to the status byte register either directly, or through the STATus: QUEStionable register. Next, the summary bits are logically ANDed with the contents of the service request enable register and the result is logically ORed to produce the request service (*RQS) bit in the status byte register.

How Do You Access the Status Registers?

There are two different methods to access the status registers:

- Common Commands Accesses and Controls
- Status Subsystem Commands

Common Command Access and Control

Most monitoring of the analyzer conditions is done at the highest level using the following IEEE common commands:

***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 standard event status register.

***OPC** (operation complete) sets bit 0 in the standard event status register when all operations are complete.

***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.

Complete command descriptions are given in Chapter 5, “Language Reference” under the subsection entitled “IEEE Common Commands” on page 5-5.

NOTE If you are using the status bits and the analyzer mode is changed, the status bits should be read, and any error conditions resolved, prior to switching modes. Error conditions that exist prior to switching modes cannot be detected using the condition registers after the mode change. This is true unless they recur after the mode change, although transitions of these conditions can be detected using the event registers.

Changing modes resets all SCPI status registers and mask registers to their power-on defaults. Hence any event or condition register masks must be re-established after a mode change. Also note that the power up status bit is set by any mode change, since that is the default state after power up.

Status Subsystem Commands

Individual status registers can be set and queried using the commands in the STATus subsystem in Chapter 5, “Language Reference” in this guide. There are two methods used to programmatically detect and manage error conditions or changes in analyzer status. Either method allows you to monitor one or more conditions. The two methods are:

- The Polling Method
- The Service Request (SRQ) Method

The Polling Method

In the polling method, the analyzer has a passive role. It only tells the controller that conditions have changed when the controller asks the right question. The polling method works well if you do not need to know about changes the moment they occur. This method is very efficient.

Use the polling method when either:

— 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

The Service Request (SRQ) Method

The SRQ method allows timely communication of information without requiring continuous controller involvement. Using this method, the analyzer takes a more active role. It tells the controller when there has been a condition change without the controller asking. The SRQ method should be used if you must know immediately when a condition changes. This is in contrast to the polling method, which requires the program to repeatedly read the registers to detect a change.

Use the SRQ method when either:

— 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 the analyzer is making a measurement
— you can't afford the performance penalty inherent to polling

Using the Service Request (SRQ) Method

Your language, bus, and programming environment must be able to support SRQ interrupts (for example, using C and C++ with the GPIB). When you monitor a condition with the SRQ method, you must establish the following parameters:

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 bits that monitor 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 analyzer sets the 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 also determines how the controller responds to the SRQ.

Generating a Service Request

Before using the SRQ method of generating a service request, first become familiar with how service requests are generated. Bit 6 of the status byte register is the request service summary (RQS) bit. The RQS bit is set whenever there is a change in the register bit that it has been configured to monitor. The RQS bit will remain set until the condition that caused it is cleared. It can be queried without erasing the contents using the ***STB?** command. Configure the RQS function using the ***SRE** command.

When a register set causes a summary bit in the status byte to change from 0 to 1, the analyzer 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 analyzer does not have a service request pending. (A service request is considered to be pending between the time the analyzer 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 request service (RQS) bit to 1. Both actions are necessary to inform the controller that the analyzer 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 device 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 analyzer 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 with the **`:INITiate`** command can cause the measuring bit to pulse low. A low pulse causes an SRQ if the status register is configured to SRQ upon end-of-measurement. To avoid this, perform the following steps:

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

Example of Monitoring Conditions Using the :STATus Command

Use the following steps to monitor a *specific* 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.
4. Act upon the cause of the condition and the SRQ to re-enable the method.

The examples below show how to use the **`:STATus`** command to perform the following tasks:

- Check the analyzer hardware and firmware status.

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

- Monitor a particular bit (condition), or bits.

Once you have enabled a bit using the event enable register, the analyzer 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 a change in the condition of a particular bit, or bits.

  Once you have enabled a bit, the analyzer will monitor it for a change in its condition. The transition registers are preset to respond to the condition of going from 0 to 1 (positive transitions). This can be changed so that the selected bit is detected if it goes from 1 to 0 (negative transition), or if either transition occurs. Query the event register to determine whether or not a change has been made to how the transition registers respond. The event register can only be cleared by querying it or sending the ***CLS** command, which clears all event registers.

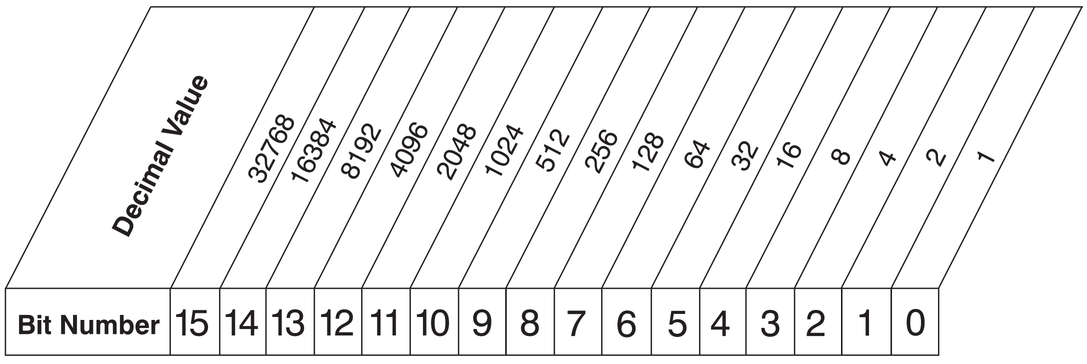
Setting and Querying the Status Register

See Figure 2-3. Each bit in a register is represented by a numerical value based on its location. This number is sent with the command to enable a particular bit. To enable more than one bit, send the sum of all of the bits involved.

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

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

Figure 2-3 Status Register Bit Values

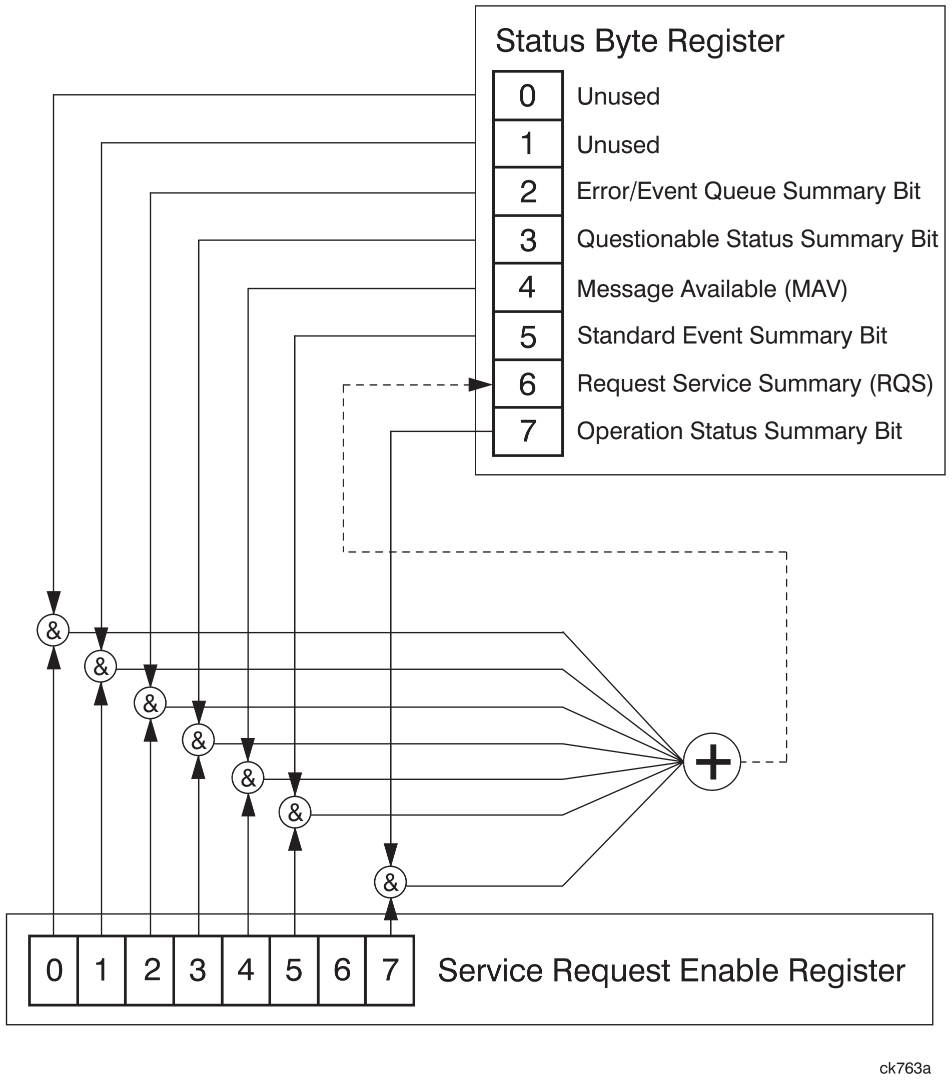


Details of Bits in All Registers

Refer to Figure 2-2. The rest of this chapter lists the bits in each register shown in the figure, along with descriptions of their purpose.

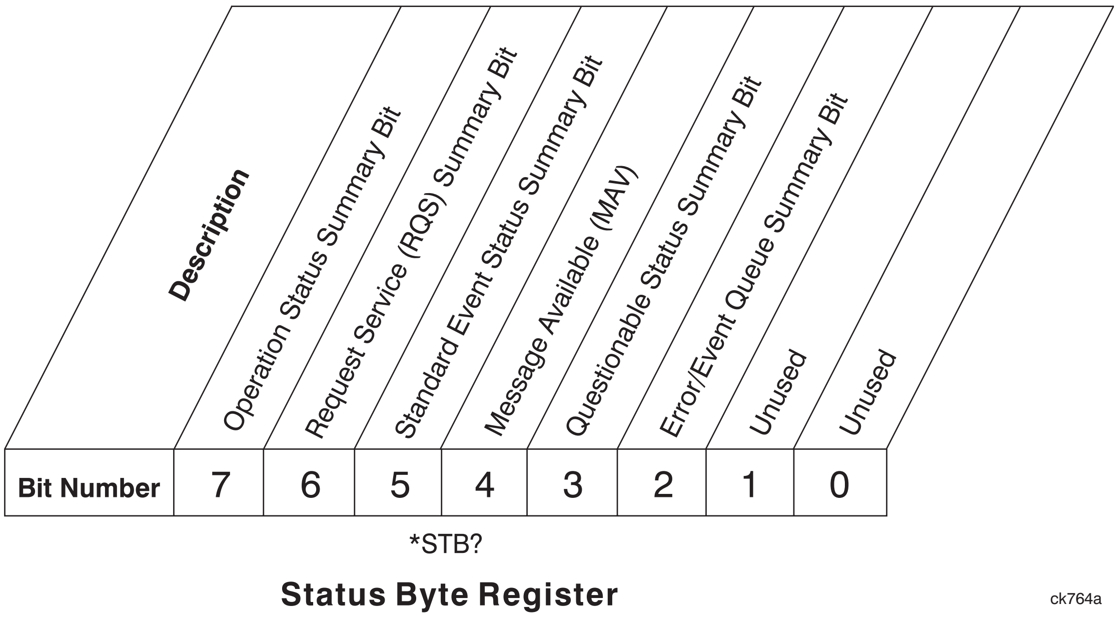
Status Byte Register

Figure 2-4 Status Byte Register


Status Byte Register
0 Unused
1 Unused
2 Error/Event Queue Summary Bit
3 Questionable Status Summary Bit
4 Message Available (MAV)
5 Standard Event Summary Bit
6 Request Service Summary (RQS)
7 Operation Status Summary Bit
&
+
0 1 2 3 4 5 6 7
Service Request Enable Register
ck763a

The status byte register contains the following bits:

Figure 2-5


Description
Operation Status Summary Bit
Request Service (RQS) Summary Bit
Standard Event Status Summary Bit
Message Available (MAV)
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
ck764a

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 questionable status summary bit has been set. The questionable status event register can then be read to determine the specific condition that caused this bit to be set.
4	A 1 in this bit position indicates that the analyzer has data ready in the output queue. There are no lower status groups that provide input to this bit.
5	A 1 in this bit position indicates that the standard event status summary bit has been set. The standard event status register can then be read to determine the specific event that caused this bit to be set.
6	A 1 in this bit position indicates that the analyzer has at least one reason to report a status change. This bit is also called the master summary status bit (MSS).
7	A 1 in this bit position indicates that the operation status summary bit has been set. The operation status event register can then be read to determine the specific event that caused this bit to be set.

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

Service Request Enable Register

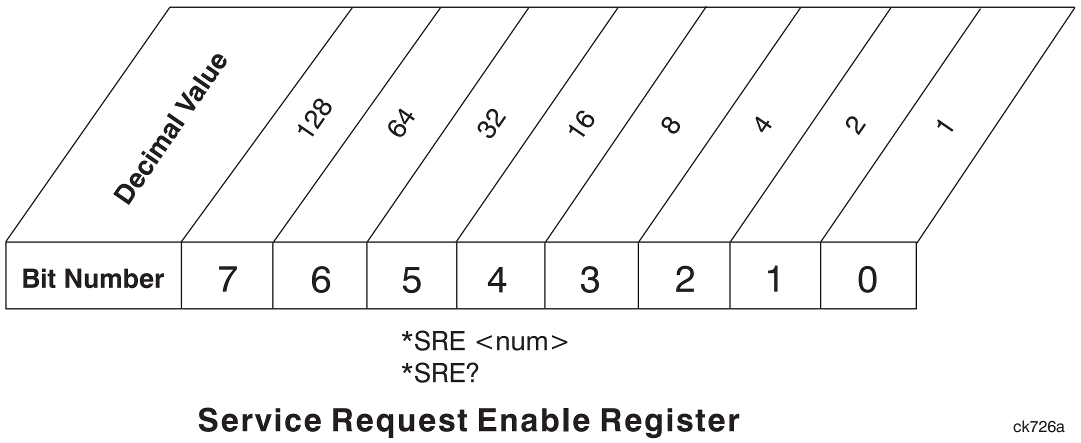
In addition to the status byte register, the status byte group also contains the service request enable register. The status byte service request enable register lets you choose which bits in the Status Byte Register will trigger a service request.

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 operation status summary bit is set to 1, it will trigger a service request. Send the **`*SRE 192`** (128 + 64) command. The **`*SRE?`** command returns the decimal value of the sum of the bits enabled previously with the **`*SRE <number>`** command.

NOTE 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 service request enable register contains the following bits:

Figure 2-6



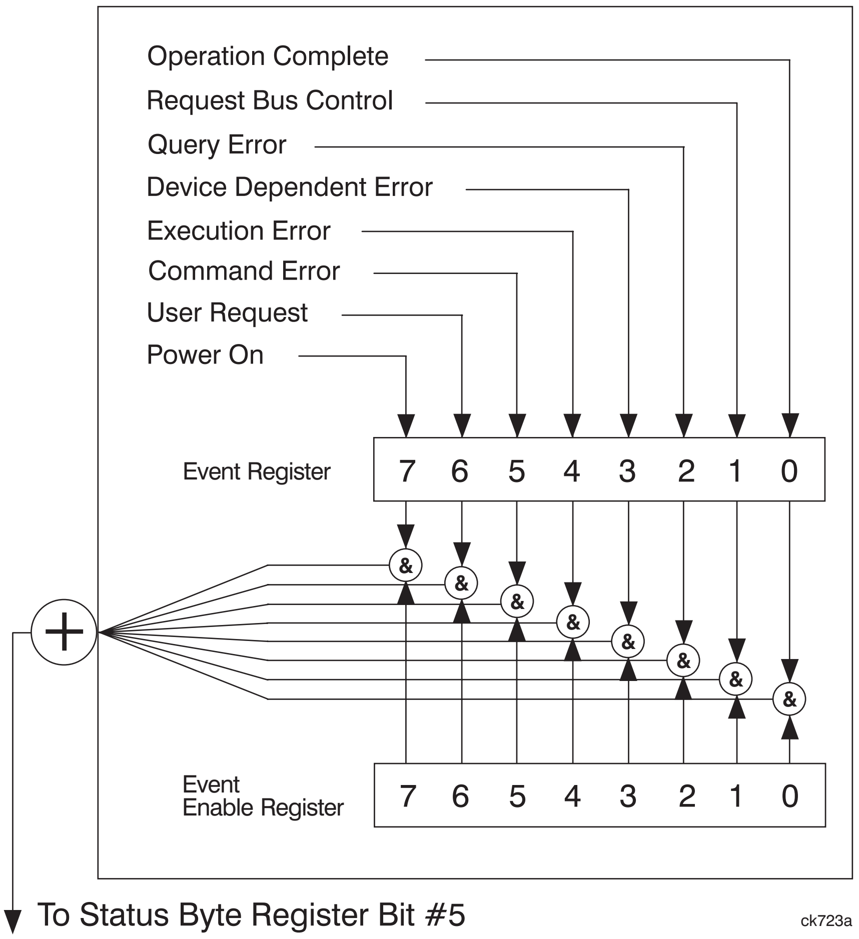
NOTE The service request enable register presets to zeros (0).

Standard Event Status Register

The standard event status register is used to determine the specific event that sets bit 5 in the status byte register. The standard event status register does *not* have negative and positive transition registers, nor a condition register. Use the IEEE common commands at the beginning of Chapter 5, “Language Reference” in this guide to access the register.

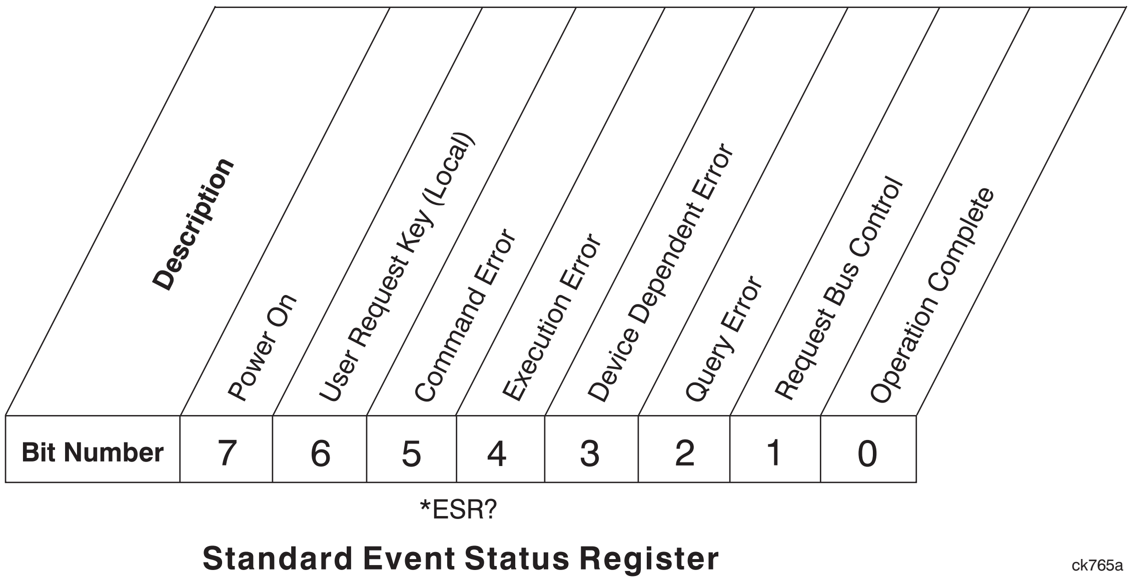
To query the standard event status register, send the ***ESR** command. The response will be the *decimal* sum of the bits which are set to 1. For example, if bit number 7 and bit number 3 are set to 1, the decimal sum of the 2 bits is 128 plus 8. So the decimal value 136 is returned.

Figure 2-7 Standard Event Status Register



The standard event status register contains the following bits:

Figure 2-8



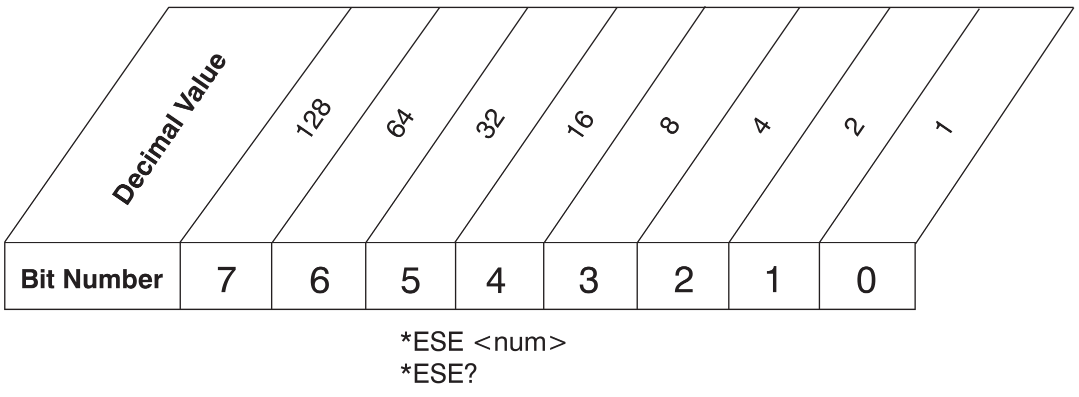
Bit	Description
0	A 1 in this bit position indicates that all operations were completed following execution of the ***OPC** command.
1	This bit is always set to 0. (The analyzer does not request control.)
2	A 1 in this bit position indicates that a query error has occurred. Query errors have SCPI error 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	A 1 in this bit position indicates that the **LOCAL** key has been pressed. This is true even if the analyzer is in local lockout mode.
7	A 1 in this bit position indicates that the analyzer has been turned off and then on.

Standard Event Status Event Enable Register

The event enable register (contained in the standard event status register) lets you choose which bits 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 also be set to 1, send the **`*ESE 192`** (128 + 64) command. The **`*ESE?`** command returns the decimal value of the sum of the bits previously enabled with the **`*ESE <number>`** command.

Figure 2-9



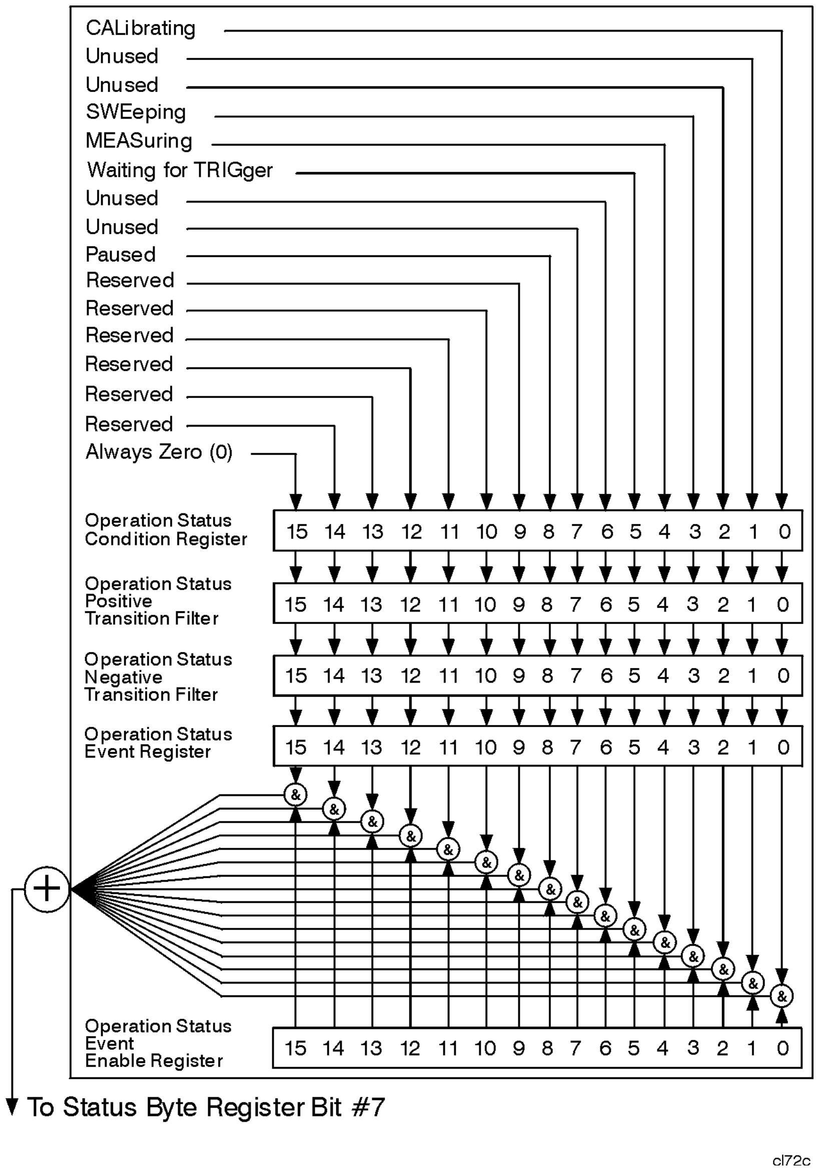
Standard Event Status Enable Register

STATus:OPERation Register

The STATus:OPERation register is used to determine the specific event that sets bit 7 in the status byte register. This register also monitors the current measurement state and checks to see if the analyzer is performing any of these functions:

- measuring
- calibrating
- sweeping
- waiting for a trigger

Figure 2-10 STATus:OPERation Register


CALibrating
Unused
Unused
SWEeping
MEASuring
Waiting for TRIGger
Unused
Unused
Paused
Reserved
Reserved
Reserved
Reserved
Reserved
Reserved
Always Zero (0)
Operation Status Condition Register
Operation Status Positive Transition Filter
Operation Status Negative Transition Filter
Operation Status Event Register
Operation Status Event Enable Register
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
&
To Status Byte Register Bit #7
cl72c

STATus:OPERation Condition Register

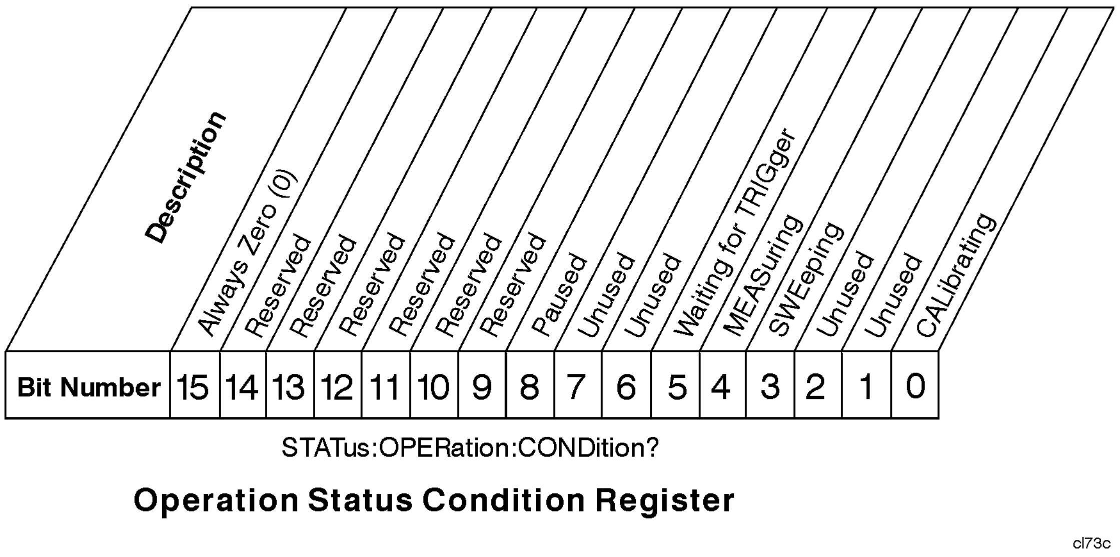
The STATus:OPERation condition register continuously monitors the hardware and firmware status of the analyzer, and is read-only. To query the register, send the **:STATus:OPERation:CONDition?** command. The response will be the *decimal* sum of the bits that are set to 1. For example, if bit number 9 and bit number 3 are set to 1, the decimal sum of the 2 bits is 512 plus 8. So the decimal value 520 is returned.

The transition filter specifies which types of bit state changes in the condition register will set corresponding bits in the event register. The changes may be positive (from 0 to 1) or negative (from 1 to 0). Send the **:STATus:OPERation:NTRansition <num>** (negative transition) command or the **:STATus:OPERation:PTRansition <num>** (positive transition) command (where **<num>** is the sum of the decimal values of the bits you want to enable).

The STATus:OPERation event register latches transition events from the condition register as specified by the transition filters. Event registers are destructive read-only data. Reading data from an event register will clear the content of that register. To query the event register, send the **:STATus:OPERation:[:EVENt]?** command.

The STATus:OPERation condition register contains the following bits:

Figure 2-11



Bit	Description
0	A 1 in this bit position indicates that the analyzer is performing a self-calibration.
1, 2	Unused. These bits are always set to 0.
3	A 1 in this bit position indicates that a sweep is in progress.
4[a]	A 1 in this bit position indicates that a measurement is in progress.
5	A 1 in this bit position indicates that a measurement is in a “wait for trigger” state.
6, 7	Unused. These bits are always set to 0.
8[b]	A 1 in this bit position indicates that the instrument is in the paused state of the measurement.
9–14	Reserved. These bits are not used by the analyzer, but are for future use with other Agilent products.
15	Always Zero (0).

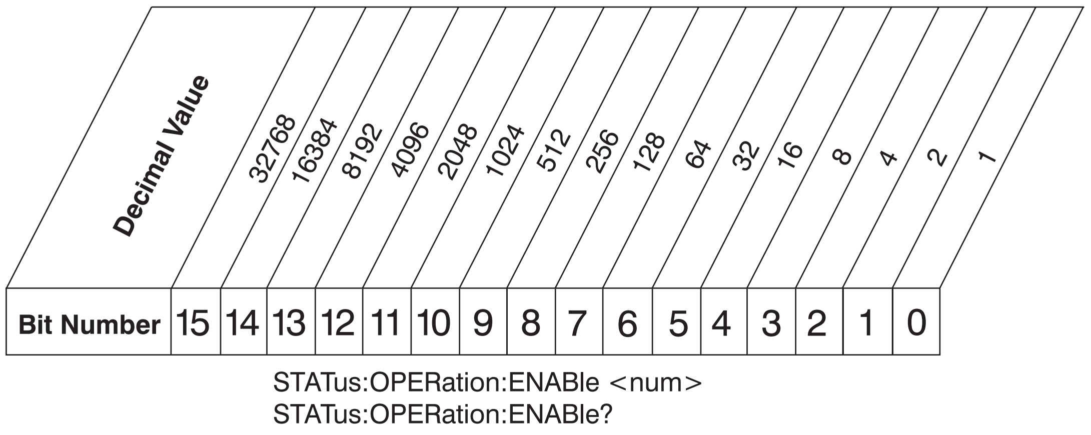
a. The description of this bit refers to any measurement under the **MEASURE** key.
b. This bit applies to ESA optional measurement personalities only, and may or may not be implemented in all such personalities.

STATus:OPERation Event Enable Register

The STATus:OPERation event enable register lets you choose the bits that will set the operation status summary bit (bit 7) of the status byte register to 1. Send the **`:STATus:OPERation:ENABle <num>`** command where **`<num>`** is the sum of the decimal values of the bits you want to enable.

For example, to enable bit 9 and bit 3 (so that whenever either bit 9 or 3 is set to 1, the operation status summary bit of the status byte register will be set to 1), send the **`:STATus:OPERation:ENABle 520`** (512 + 8) command. The **`:STATus:OPERation:ENABle?`** command returns the decimal value of the sum of the bits previously enabled with the **`:STATus:OPERation:ENABle <num>`** command.

Figure 2-12



Operation Status Event Enable Register

STATus:QUEStionable Registers

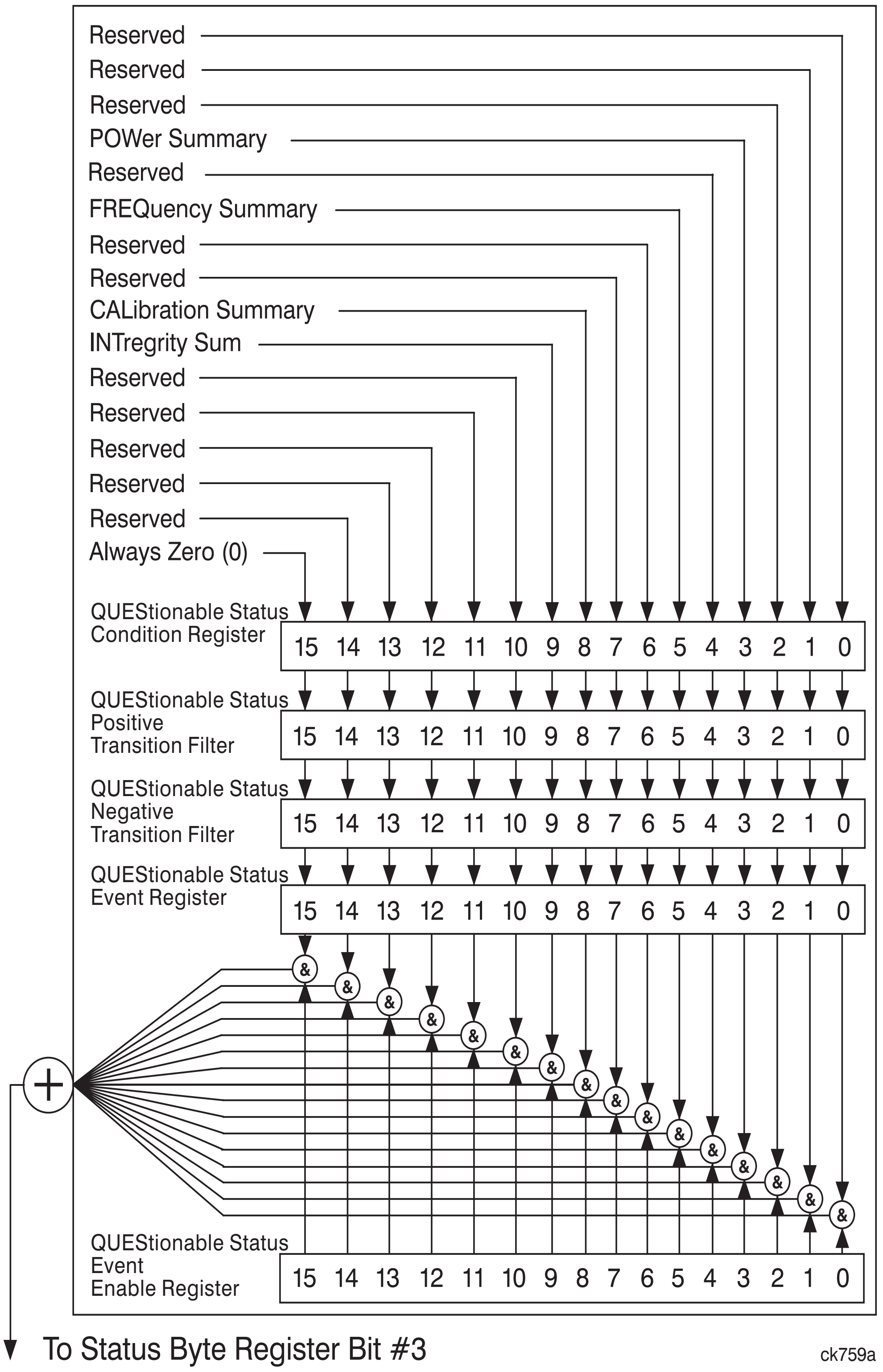
STATus:QUEStionable registers monitor the overall analyzer condition. They are accessed with the **`:STATus:OPERation`** and **`:STATus:QUEStionable`** commands in the **`:STATus`** command subsystem.

The STATus:QUEStionable registers also monitor the analyzer to see if there are any questionable events that occurred. These registers look for anything that may cause an error or that may induce a faulty measurement. Signs of a faulty measurement include the following:

- hardware problems
- out of calibration situations
- unusual signals

NOTE All bits are summary bits from lower-level event registers. (For a general diagram of the STATus:QUEStionable register, see Figure 2-13 on page 2-21.)

Figure 2-13 STATus:QUEStionable Register



STATus:QUEStionable:POWer Condition Register Bits

Bit	Decimal Value	Description
0	0	**Reserved** This bit is not used by the analyzer, but is for future use with other Agilent products.
1	2	**Source Unleveled** A 1 in this bit position indicates that the source (tracking generator) output is unlevelled.
2	4	**Source LO Unleveled** A 1 in this bit position indicates that the local oscillator (LO) in the source (tracking generator) is unlevelled.
3	8	**LO Unleveled** A 1 in this bit position indicates that the analyzer local oscillator (LO) is unlevelled.
4	16	**50 MHz Osc Unleveled** A 1 in this bit position indicates that the 50 MHz amplitude reference signal is unlevelled.
5	32	**Reserved** This bit is not used by the analyzer, but is for future use with other Agilent products.
6	64	**Input Overload Tripped** A 1 in this bit position indicates that the input overload protection is tripped (Agilent ESA models E4401B and E4411B only).
7	128	**LO Out Unleveled** A 1 in this bit position indicates that the first local oscillator (LO) output is unlevelled. (Agilent ESA model E4407B option AYZ, External Mixing, only).
8–14	–––	**Unused** These bits are always set to 0.
15	32768	**Always Zero (0)** This bit is always set to 0.

STATus:QUEStionable:FREQuency Condition Register Bits

Bit	Decimal Value	Description
0	0	**Source Synth Unlocked** A 1 in this bit position indicates that the synthesizer in the source (tracking generator) is unlocked.
1	2	**Freq Ref Unlocked** A 1 in this bit position indicates that the analyzer frequency reference is unlocked.
2, 3	4, 8	**Reserved** This bit is not used by the analyzer, but is for future use with other Agilent products.

Bit	Decimal Value	Description
4	16	**`Synth Unlocked`** A 1 in this bit position indicates that the analyzer synthesizer is unlocked.
5	32	**`Invalid BW`** A 1 in this bit position indicates that an invalid bandwidth setting has been requested.
6–8	–––	**`Reserved`** These bits are not used by the analyzer, but are for future use with other Agilent products.
9–14	–––	**`Unused`** These bits are always set to 0.
15	32768	**`Always Zero (0)`** This bit is always set to 0.

STATus:QUEStionable:CALibration Condition Register Bits

Bit	Decimal Value	Description
0, 1	0, 2	**`Reserved`** These bits are not used by the analyzer, but are for future use with other Agilent products.
2	4	**`TG Align Failure`** A 1 in this bit position indicates that a failure has occurred while trying to align the tracking generator (TG).
3	8	**`RF Align Failure`** A 1 in this bit position indicates that a failure has occurred while trying to align the RF section.
4	16	**`IF Align Failure`** A 1 in this bit position indicates that a failure has occurred while trying to align the IF section.
5	32	**`LO Align Failure`** A 1 in this bit position indicates that a failure has occurred while trying to align the local oscillator (LO).
6	64	**`ADC Align Failure`** A 1 in this bit position indicates that a failure has occurred while trying to align the analog-to-digital converter (ADC).
7	128	**`FM Demod Align Failure`** A 1 in this bit position indicates that a failure has occurred while trying to align the FM demodulation circuitry. (Agilent ESA models E4401B, E4402B, E4404B, E4405B, and E4407B Option BAA, FM Demodulation, only).

Bit	Decimal Value	Description
8	256	**Qpeak Align Failure** A 1 in this bit position indicates that a failure has occurred while trying to align the quasi-peak detector. (Agilent ESA- E700A Series models only).
9	512	**Unused** This bit is always set to 0.
10	1024	**Tracking Peak Needed** A 1 in this bit position indicates that a tracking peak needs to be performed (the tracking generator is in operation). (Agilent ESA models E4402B, E4403B, E4405B, E4407B, and E4408B, Option 1DN, Tracking Generator, only).
11	2048	**Align RF Skipped** A 1 in this bit position indicates that the alignment of the RF section was skipped, perhaps due to an external 50 MHz signal having been detected.
12	4096	**Align RF Needed** A 1 in this bit position indicates that the RF section needs to be aligned.
13	8192	**Reserved** This bit is not used by the analyzer, but is for future use with other Agilent products.
14	16384	**Align Needed** A 1 in this bit position indicates that a full alignment is needed, perhaps due to a large temperature change having been detected with auto align off, or due to default data being used.
15	32768	**Always Zero (0)** This bit is always set to 0.

STATus:QUEStionable:INTegrity Condition Register Bits

Bit	Decimal Value	Description
0	0	**Reserved** This bit is not used by the analyzer, but is for future use with other Agilent products.
1[a]	2	**No Result Available** A 1 in this bit position indicates that a measurement terminated with no measurement results.
2[a]	4	**Measurement Timeout** A 1 in this bit position indicates that a measurement terminated due to a timeout.
3	8	**Data Uncalibrated Summary** This is the summary bit for the Questionable Status Integrity Uncalibrated Register.

Bit	Decimal Value	Description
4	16	**`IF/ADC Over Range`** The signal input level is too high, causing the analyzer analog-to-digital converter (ADC) range to be exceeded. This may occur with resolution bandwidths less than or equal to 300 Hz in zero span. (Agilent ESA models E4401B, E4402B, E4404B, E4405B, and E4407B only).
5[a]	32	**`Over Range`** A 1 in this bit position indicates that the signal is too large at the analog-to-digital converter (ADC).
6[a]	64	**`Under Range`** A 1 in this bit position indicates that the signal is too small at the analog-to-digital converter (ADC).
7[a]	128	**`Insufficient Data`** A 1 in this bit position indicates that there is not enough information to perform the measurement or function.
8[a]	256	**`Acquisition Failure`** A 1 in this bit position indicates that the demod algorithm cannot correlate to the signal.
9[a]	512	**`Memory Problem`** A 1 in this bit position indicates a failure of the file system memory or digital signal processor (DSP) memory.
10[a]	1024	**`Auto-Trigger Timeout`** A 1 in this bit position indicates that the measurement timed out due to no trigger.
11[a]	2048	**`Trigger Problem`** A 1 in this bit position indicates that the measurement timed out due to no trigger.
12	4096	**`Invalid Data`** A 1 in this bit position indicates that the present trace data does not reflect the existing analyzer state. Trigger a new sweep and/or measurement.
13[a]	8192	**`Unidentified Error`** A 1 in this bit position indicates that a measurement has terminated for a reason other than that given in any of the other bits.
14[a]	16384	**`Setting Limited/Readjusted`** A 1 in this bit position indicates that the user settings could not be achieved with the existing hardware; values were set to limits.
15	32768	**`Always Zero (0)`** This bit is always set to 0.

a. This bit applies to ESA optional measurement personalities only, and may or may not be implemented in all such personalities.

STATus:QUEStionable:INTegrity:UNCalibrated Condition Register Bits

Bit	Decimal Value	Description
0	0	**Oversweep (Meas Uncal)** A 1 in this position indicates that the anlyzer is in a state that could lead to uncalibrated measurements. This is typically caused by sweeping too fast for the current combination of span, resolution bandwidth, and video bandwidth. Auto coupling may resolve this problem.
1	2	**Signal Ident ON** A 1 in this bit position indicates that amplitude measurements may be in error due to signal identification routines being active. Amplitude accuracy is degraded when signal identification is active. (Agilent ESA model E4407B Option AYZ, External Mixing, only).
2–14	–––	**Reserved** These bits are not used by the analyzer, but are for future use with other Agilent products.
15	32768	**Always Zero (0)** This bit is always set to 0.

3 Programming Examples

This chapter includes examples of how to program the analyzer using the analyzer SCPI programming commands. Twelve examples are written for an analyzer with an HP-IB interface (Option A4H). Three examples are written for an analyzer with an RS-232 interface (Option 1AX).

List of Programming Examples

The programming examples included in this chapter are:

Programming Examples System Requirements

These examples were written for use on an IBM compatible PC configured as follows:

- ❏ Pentium processor
- ❏ Windows 95 or Windows NT 4.0 operating system
- ❏ C programming language
- ❏ HP/Agilent 82341C HP-IB interface card (for ESAs with Option A4H)
- ❏ HP VISA Transition Libraries (VTL)
- ❏ COM1 serial port configured as follows (for ESAs with Option 1AX)
    - 9600 baud
    - 8 data bits
    - 1 stop bit
    - no parity bits
    - hardware flow control

A National Instruments GPIB card may be substituted for the HP/Agilent 82341C, and the National Instruments VISA libraries may be substituted for the HP VISA Transition Libraries. If substitutions are made, the subdirectories for the include and library files will be different than those listed in the following paragraphs. Refer to the documentation for your interface card and the VISA libraries for details.

C Programming Examples using VTL

The programming examples that are provided in this guide are written using the C programming language and the 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 "HP VISA Transition Library", HP part number E2090-90026.) If you are using the National Instruments VISA library, most of this information will still apply, but the include and library files will be in different subdirectories. Also, this information assumes a computer running a Windows 95 operating system with an HP/Agilent 82341C HP-IB interface card is being used. 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 SCPI commands to the analyzer. The `viPrintf` call can also be used to query the analyzer. The `viScanf` call is then used to read the results.
`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 deallocated. 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 (assumes Windows 95 operating system):

`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 for information on how to use the VTL run-time libraries.

Compiling and Linking a VTL Program

32-bit Applications (assumes Windows 95 operating system)

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 if something other than the ESA default value of 18 is required.

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

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

void main ()
{

     /*Open session to HP-IB 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 the sessions */
     viClose(vi);
     viClose(defaultRM);
}
```

Including the VISA Declarations File

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

```
#include “visa.h”
```

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

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

Opening a Session

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

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

NOTE

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

Device Sessions

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

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

viOpenDefaultRM (*sesn*);

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

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

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

The following summarizes the parameters in the previous function calls:

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

The following is an example of opening sessions with an HP-IB multimeter and an HP-IB/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 HP-IB 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 HP-IB/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]
HP-IB/VXI	GPIB-VXI [*board*]::*VXI logical address*[::INSTR]
HP-IB	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.
VXI logical address	This is the logical address of the VXI instrument.
primary address	This is the primary address of the HP-IB device.
secondary address	This optional parameter is the secondary address of the HP-IB 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:

VXI0::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 an HP-IB/VXI command module.
GPIB0::7::0	An HP-IB device at primary address 7 and secondary address 0 on the HP-IB interface.

The following is an example of opening a device session with the HP-IB device at primary address 23.

```
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.

Using Marker Peak Search and Peak Excursion

Example:

```
/************************************************************/
/* Using Marker Peak Search and Peak Excursion              */
/*                                                          */
/* This C programming example does the following.           */
/* The SCPI instrument commands used are given as           */
/* reference.                                               */
/*                                                          */
/* - Opens an HP-IB session at address 18                   */
/* - Clears the Analyzer                                    */
/*      *CLS                                                */
/* - Resets the Analyzer                                    */
/*      *RST                                                */
/* - Sets the analyzer center frequency and span            */
/*      SENS:FREQ:CENT freq                                 */
/*      SENS:FREQ:SPAN freq                                 */
/* - Set the input port to the 50 MHz amplitude reference   */
/*      CAL:SOUR:STAT ON                                    */
/* - Set the analyzer to single sweep mode                  */
/*      INIT:CONT 0                                         */
/* - Prompt the user for peak excursion and set them        */
/*      CALC:MARK:PEAK:EXC dB                               */
/* - Set the peak threshold to -90 dBm                      */
/*      TRAC:MATH:PEAK:THR:STAT ON                          */
/*      TRAC:MATH:PEAK:THR -90                              */
/* - Trigger a sweep                                        */
/*      INIT:IMM                                            */
/* - Check for operation complete                           */
/*      *OPC?                                               */
/* - Set the marker to the maximum peak                     */
/*      CALC:MARK:MAX                                       */
/* - Query and read the marker frequency and amplitude      */
/*      CALC:MARK:X?                                        */
/*      CALC:MARK:Y?                                        */
/* - Close the session                                      */
/************************************************************/

#include <stdio.h>
```

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

#define hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"

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

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

  viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
  iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)));
  if( iResult == 0 )
  {
       /*Set the input port to the 50MHz amplitude reference for the models*/
       /*E4411B and E4401B*/
       viPrintf(viESA,"CAL:SOUR:STAT ON \n");
  }
  else
  {
       /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
       /* to connect the amplitude reference output to the input*/
       printf ("Connect AMPTD REF OUT to the INPUT \n");
       printf ("......Press Return to continue \n");
       scanf( "%c",&cEnter);

       /*Externally route the 50MHz Signal*/
       viPrintf(viESA,"CAL:SOUR:STAT ON \n");
  }
}

void main()
{
  /*Program Variables*/
  ViStatus viStatus  = 0;
```

```
double dMarkerFreq = 0;
double dMarkerAmpl = 0;
float fPeakExcursion =0;
long lOpc = 0L;

/*Open an HP-IB session at address 18.*/
viStatus=viOpenDefaultRM(&defaultRM);
viStatus=viOpen(defaultRM,"GPIB0::18",VI_NULL,VI_NULL,&viESA);
if(viStatus)
{
     printf("Could not open a session to HP-IB device at address 18!\n");
     exit(0);
}
/*Clear the instrument*/
viClear(viESA);

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

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

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

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

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

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

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

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

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

/*Trigger a sweep*/
viPrintf(viESA,"INIT:IMM\n");
```

```
  /*Make sure the previous command has been completed*/
  viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
  if (!lOpc)
  {
       printf("Program Abort! error ocurred: last command was not completed!\n");
       exit(0);
  }
  /*Set the marker to the maximum peak*/
  viPrintf(viESA,"CALC:MARK:MAX \n");

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

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

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

Using Marker Delta Mode and Marker Minimum Search

```
/*************************************************************/
/* Using Marker Delta Mode and Marker Minimum Search        */
/*                                                           */
/* This C programming example does the following.           */
/* The SCPI instrument commands used are given as           */
/* reference.                                                */
/*                                                           */
/* - Opens an HP-IB session at address 18                   */
/* - Clears the Analyzer                                     */
/* - Resets the Analyzer                                     */
/*      *RST                                                 */
/* - Set the input port to the 50 MHz amplitude reference   */
/*      CAL:SOUR:STAT ON                                     */
/* - Set the analyzer to single sweep mode                  */
/*      INIT:CONT 0                                          */
/* - Prompts the user for the start and stop frequencies    */
/* - Sets the start and stop frequencies                    */
/*      SENS:FREQ:START freq                                 */
/*      SENS:FREQ:STOP freq                                  */
/* - Trigger a sweep                                         */
/*      INIT:IMM                                             */
/* - Check for operation complete                           */
/*      *OPC?                                                */
/* - Set the marker to the maximum peak                     */
/*      CALC:MARK:MAX                                        */
/* - Set the analyzer to activate the delta marker          */
/*      CALC:MARK:MODE DELT                                  */
/* - Trigger a sweep                                         */
/*      INIT:IMM                                             */
/* - Check for operation complete                           */
/*      *OPC?                                                */
/* - Set the marker to the minimum amplitude mode           */
/*      CALC:MARK:MIN                                        */
/* - Query and read the marker amplitude                    */
/*      CALC:MARK:Y?                                         */
/* - Close the session                                       */
/*************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
```

```
#include <conio.h>
#include <ctype.h>
#include <string.h>
#include "visa.h"

#define hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"

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

/*Set the input port to the 50MHz amplitude reference*/
void Route50MHzSignal()
{
  viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
  iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)));
  if( iResult == 0 )
  {
       /*Set the input port to the 50MHz amplitude reference for the models*/
       /*E4411B and E4401B*/
       viPrintf(viESA,"CAL:SOUR:STAT ON \n");
  }
  else
  {
       /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
       /* to connect the amplitude reference output to the input*/
       printf ("Connect AMPTD REF OUT to the INPUT \n");
       printf ("......Press Return to continue \n");
       scanf( "%c",&cEnter);

       /*Externally route the 50MHz Signal*/
       viPrintf(viESA,"CAL:SOUR:STAT ON \n");
  }
}

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

```
 long   lOpc =0L;

 /* Open an HP-IB session at address 18*/
 viStatus=viOpenDefaultRM(&defaultRM);
 viStatus=viOpen(defaultRM,"GPIB0::18",VI_NULL,VI_NULL,&viESA);
 if(viStatus)
 {
      printf("Could not open a session to HP-IB device at address 18!\n");
      exit(0);
 }
 /*Clear the instrument*/
 viClear(viESA);

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

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

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

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

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

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

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

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

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

 /*Trigger a sweep*/
 viPrintf(viESA,"INIT:IMM\n");

 /*Check for operation complete*/
 viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
```

```
  if (!lOpc)
  {
       printf("Program Abort! error ocurred: last command was not completed!\n");
       exit(0);
  }
  /*Set the marker to the maximum peak*/
  viPrintf(viESA,"CALC:MARK:MAX\n");

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

  /*Trigger a sweep*/
  viPrintf(viESA,"INIT:IMM\n");

  /*Check for operation complete*/
  viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
  if (!lOpc)
  {
       printf("Program Abort! error ocurred: last command was not completed!\n");
       exit(0);
  }
  /*Set the marker to minimum amplitude*/
  viPrintf(viESA,"CALC:MARK:MIN\n");

  /*Query and read the marker amplitude*/
  viPrintf(viESA,"CALC:MARK:Y?\n");
  viScanf(viESA,"%lf",&dMarkerAmplitude);

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

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

Performing Internal Self-alignment

```
/*************************************************************/
/* Performing Internal Self-alignment                        */
/*                                                           */
/* This example shows two ways of executing an internal      */
/* self-alignment. The first demonstrates using the *OPC?    */
/* query to determine when the alignment has completed. The */
/* second demonstrates using the query form of the CAL:ALL  */
/* command to not only determine when the alignment has      */
/* been completed, but the pass/fail status of the align-    */
/* ment process.                                             */
/*                                                           */
/* This C programming example does the following.            */
/* The SCPI instrument commands used  are given as           */
/* reference.                                                */
/*                                                           */
/* - Opens an HP-IB session at address 18                    */
/* - Clears the Analyzer                                     */
/*      *CLS                                                 */
/* - Resets the Analyzer                                     */
/*      *RST                                                 */
/* - VISA function sets the time out to infinite             */
/* - Initiate self-alignment                                 */
/*      CAL:ALL                                              */
/* - Query for operation complete                            */
/*      *OPC?                                                */
/* - Query for results of self-alignment                     */
/*      CAL:ALL?                                             */
/* - Report the results of the self-alignment                */
/* - Close the session                                       */
/*************************************************************/

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

#define hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"

ViSession defaultRM, viESA;
```

```
ViStatus  errStatus;
ViChar    cIdBuff[256]= {0};
char      cEnter = 0;
int       iResult = 0;

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

  viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
  iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)));
  if( iResult == 0 )
  {
       /*Set the input port to the 50MHz amplitude reference for the models*/
       /*E4411B and E4401B*/
       viPrintf(viESA,"CAL:SOUR:STAT ON \n");
  }
  else
  {
       /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
       /* to connect the amplitude reference output to the input*/
       printf ("Connect AMPTD REF OUT to the INPUT \n");
       printf ("......Press Return to continue \n");
       scanf( "%c",&cEnter);

       /*Externally route the 50MHz Signal*/
       viPrintf(viESA,"CAL:SOUR:STAT ON \n");
  }
}

void main()
{
  /*Program Variables*/
  ViStatus viStatus  = 0;
  long lOpc =0L;
  long lResult =0L;

  /* Open a HP-IB session at address 18*/
  viStatus=viOpenDefaultRM(&defaultRM);
  viStatus=viOpen(defaultRM,"GPIB0::18",VI_NULL,VI_NULL,&viESA);
  if(viStatus)
  {
       printf("Could not open a session to HP-IB device at address 18!\n");
       exit(0);
  }
```

```
/*Clear the instrument*/
viClear(viESA);

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

/*Display the program heading */
printf("\n\t\t Internal Self-Alignment Program \n\n" );

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

/*VISA function sets the time out to infinite for this specified session*/
viSetAttribute(viESA, VI_ATTR_TMO_VALUE, VI_TMO_INFINITE);
printf("\t Performing first self alignment ..... " );

/*Initiate a self-alignment */
viPrintf(viESA,"CAL:ALL\n");

/*Query for operation complete*/
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
printf ("\n\n\t First Self Alignment is Done \n\n");
if (!lOpc)
{
     printf("Program Abort! error ocurred: last command was not completed!\n");
     exit(0);
}
printf ("\n\n\t Press Return to continue with next alignment \n\n");
scanf( "%c",&cEnter);
printf("\t Performing next self alignment ..... " );

/* Query for self-alignment results*/
viPrintf(viESA,"CAL:ALL?\n");
viScanf(viESA,"%d",&lResult);
if (lResult)
     printf ("\n\n\t Self-alignment Failed \n");
else
     printf ("\n\n\t Self-alignment Passed \n");

/* Query for operation complete*/
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
{
     printf("Program Abort! error ocurred: last command was not completed!\n");
     exit(0);
}
```

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

Reading Trace Data using ASCII Format (HP-IB)

```
/************************************************************/
/* Reading Trace Data using ASCII Format (HP-IB)            */
/*                                                          */
/* This C programming example does the following.           */
/* The required SCPI instrument commands are given as       */
/* reference.                                               */
/*                                                          */
/* - Opens an HP-IB session at address 18                   */
/* - Clears the Analyzer                                    */
/* - Resets the Analyzer                                    */
/*      *RST                                                */
/* - Set the input port to the 50 MHz amplitude reference   */
/*   E4411B or E4401B                                       */
/*      CAL:SOUR:STAT ON                                    */
/*   E4402, E4403B, E4404BE, 4405B, E4407B or E4408B        */
/*      Prompt to connect AMPTD REF OUT to INPUT            */
/*      CAL:SOUR STAT ON                                    */
/* - Query for the number of sweep points (only applies to  */
/*   firmware revisions A.04.00 and later); default is 401  */
/*      SENS:SWE:POIN?                                      */
/* - Sets the analyzer center frequency to 50 MHz           */
/*      SENS:FREQ:CENT 50 MHZ                               */
/* - Sets the analyzer span to 50 MHz                       */
/*      SENS:FREQ:SPAN 50 MHZ                               */
/* - Set the analyzer to single sweep mode                  */
/*      INIT:CONT 0                                         */
/* - Trigger a sweep                                        */
/*      INIT:IMM                                            */
/* - Check for operation complete                           */
/*      *OPC?                                               */
/* - Specify units in dBm                                   */
/*      UNIT:POW DBM                                        */
/* - Set the analyzer trace data to ASCII                   */
/*      FORM:DATA: ASC                                      */
/* - Trigger a sweep                                        */
/*      INIT:IMM                                            */
/* - Check for operation complete                           */
/*      *OPC?                                               */
/* - Query the trace data                                   */
/*      TRAC:DATA? TRACE1                                   */
/* - Remove the "," from the ACSII data                     */
/* - Save the trace data to an ASCII file                  */
/* - Close the session                                     */
/**********************************************************/

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

#define hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"

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

/*Set the input port to 50MHz amplitude reference*/
void Route50MHzSignal()
{
  viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
  iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)));
  if( iResult == 0 )
  {
      /*Set the input port to the 50MHz amplitude reference for the models*/
      /*E4411B and E4401B*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
  }
  else
  {
      /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
      /* to connect the amplitude reference output to the input*/
      printf ("Connect AMPTD REF OUT to the INPUT \n");
      printf ("......Press Return to continue \n");
      scanf( "%c",&cEnter);

      /*Externally route the 50MHz Signal*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
  }
}
```

```
void main()
{
  /*Program Variable*/
  ViStatus viStatus = 0;
  /*Dimension cResult to 13 bytes per sweep point, 8192 sweep points maximum*/
    ViChar _VI_FAR cResult[106496] = {0};
  FILE *fTraceFile;
    static ViChar *cToken ;
  int iNum  =0;
  int iSwpPnts = 401;
  long lCount=0L;
  long lOpc=0;

  /*iNum set to 13 times number of sweep points, 8192 sweep points maximum*/
  iNum =106496;
    lCount =0;

  /* Open an HP-IB session at address 18*/
  viStatus=viOpenDefaultRM(&defaultRM);
  viStatus=viOpen(defaultRM,"GPIB0::18",VI_NULL,VI_NULL,&viESA);
  if(viStatus)
  {
      printf("Could not open a session to HP-IB device at address 18!\n");
      exit(0);
  }
  /* Clear the instrument */
  viClear(viESA);

  /*Reset the instrument. This will set number of sweep points to default of 401*/
  viPrintf(viESA,"*RST\n");

  /*Display the program heading */
  printf("\n\t\t Read in Trace Data using ASCII Format (HPIB) Program \n\n" );

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

  /*Query number of sweep points per trace (firmware revision A.04.00 and later)*/
  /*For firmware revisions prior to A.04.00, the number of sweep points is 401*/
  iSwpPnts = 401;
    viQueryf(viESA,"SENSE:SWEEP:POINTS?\n","%d",&iSwpPnts);

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

  /*Set the analyzer to 50MHz Span*/
```

```
viPrintf(viESA,"SENS:FREQ:SPAN 50 MHz\n");

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

/*Trigger a spectrum measurement*/
viPrintf(viESA,"INIT:IMM\n");

/*Read the operation complete query */
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
{
     printf("Program Abort! error ocurred: last command was not completed!\n");
     exit(0);
}
/* Specify units in dBm*/
viPrintf(viESA,"UNIT:POW DBM \n");

/*Set analyzer trace data format to ASCII Format*/
viPrintf(viESA,"FORM:DATA ASC \n");

/*Trigger a spectrum measurement*/
viPrintf(viESA,"INIT:IMM \n");

/*Read the operation complete query */
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
{
     printf("Program Abort! error ocurred: last command was not completed!\n");
     exit(0);
}
/*Query the Trace Data using ASCII Format */
viQueryf(viESA,"%s\n", "%#t","TRAC:DATA? TRACE1" , &iNum , cResult);

/*Remove the "," from the ASCII trace data for analyzing data*/
  cToken  = strtok(cResult,",");

/*Save trace data to an ASCII to a file, by removing the "," token*/
fTraceFile=fopen("C:\\temp\\ReadAscHpib.txt","w");
fprintf(fTraceFile,"ReadAscHpib.exe Output\nHewlett-Packard 1999\n\n");
fprintf(fTraceFile,"\tAmplitude of point[%d] = %s dBm\n",lCount+1,cToken);
  while (cToken != NULL)
     {
     lCount++;
     cToken =strtok(NULL,",");
            if (lCount != iSwpPnts)
```

```
             fprintf(fTraceFile,"\tAmplitude of point[%d] = %s
dBm\n",lCount+1,cToken);
      }
  fprintf(fTraceFile,"\nThe Total trace data points of the spectrum are :[%d]
\n\n",lCount);
  fclose(fTraceFile);

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

Reading Trace Data Using 32-bit Real Format (HP-IB)

```
/*************************************************************/
/* Reading Trace Data using 32-bit Real Format (HP-IB)      */
/*                                                           */
/* This C programming example does the following.            */
/* The SCPI instrument commands used are given as            */
/* reference.                                                */
/*                                                           */
/* - Opens an HP-IB session at address 18                    */
/* - Clears the Analyzer                                     */
/* - Resets the Analyzer                                     */
/*      *RST                                                 */
/* - Set the input port to the 50 MHz amplitude reference    */
/*      CAL:SOUR:STAT ON                                     */
/* - Query for the number of sweep points (for firmware      */
/*   revisions A.04.00 and later). Default is 401.           */
/*       SENS:SWE:POIN?                                      */
/* - Calculate the number of bytes in the header             */
/* - Set the analyzer to single sweep mode                   */
/*      INIT:CONT 0                                          */
/* - Sets the analyzer center frequency and span to 50 MHz   */
/*      SENS:FREQ:CENT 50 MHZ                                */
/*      SENS:FREQ:SPAN 50 MHZ                                */
/* - Specify 10 dB per division for the amplitude scale in   */
/*   and dBm Units                                           */
/*      DISP:WIND:TRAC:Y:SCAL:PDIV 10 dB                     */
/*      UNIT:POW DBM                                         */
/* - Set the analyzer trace data to 32-bit Real              */
/*      FORM:DATA: REAL,32                                   */
/* - Set the binary order to swap                            */
/*      FORM:BORD SWAP                                       */
/* - Trigger a sweep                                         */
/*      INIT:IMM                                             */
/* - Check for operation complete                            */
/*      *OPC?                                                */
/* - Calculate the number of bytes in the trace record       */
/* - Query the trace data                                    */
/*      TRAC:DATA? TRACE1                                    */
/* - Remove the "," from the ACSII data                      */
/* - Save the trace data to an ASCII file                    */
/* - Close the session                                       */
/*************************************************************/
```

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

#define  hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define  hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"

ViSession defaultRM, viESA;
ViChar    cIdBuff[256];
char      cEnter =0;
int       iResult =0;

void Route50MHzSignal()
{
viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)));
if( iResult == 0 )
{
     /*Set the input port to the 50MHz internal reference source for the models*/
     /*E4411B and E4401B*/
     viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
else
{
     /* For the analyzers having frequency limits >= 3GHz, prompt the user to*/
     /* connect the amplitude reference output to the input*/
     printf ("Connect AMPTD REF OUT to the INPUT \n");
     printf ("......Press Return to continue \n");
     scanf( "%c",&cEnter);

     /*Externally route the 50MHz Signal*/
     viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
}

void main()
{
/*Program Variables*/
ViStatus viStatus= 0;
    ViChar _VI_FAR cResult[5000] = {0};
```

```
ViReal32 dTraceArray[401] = {0};
char cBufferInfo[6]= {0};
long lNumberBytes =0L;
long lOpc =0L;
unsigned long lRetCount = 0L;
int iSize = 0;
/*BytesPerPoint is 4 for Real32 or Int32 formats, 8 for Real64, and 2 for Uint16*/
int iBytesPerPnt = 4;
int iSwpPnts = 401;
int iDataBytes=1604;
int iHeaderBytes=6;
FILE *fTraceFile;

/* Open a HP-IB session at address 18*/
viStatus=viOpenDefaultRM(&defaultRM);
viStatus=viOpen(defaultRM,"GPIB0::18",VI_NULL,VI_NULL,&viESA);
if(viStatus)
{
     printf("Could not open a session to HP-IB device at address 18!\n");
     exit(0);
}
/*Clear the instrument */
viClear(viESA);

/*Reset the instrument. This will set number of sweep points to default of 401*/
viPrintf(viESA,"*RST\n");

/*Display the program heading */
printf("\n\t\t Read in Trace Data using 32-bit Real Format (using HP-IB) \n\n" );

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

/*Query number of sweep points per trace (firmware revision A.04.00 and later)*/
/*For firmware revisions prior to A.04.00, the number of sweep points is 401*/
iSwpPnts=401;
viQueryf(viESA,"SENSE:SWEEP:POINTS?\n","%d",&iSwpPnts);

/*Calculate number of bytes in the header. The header consists of the "#" sign*/
/*followed by a digit representing the number of digits to follow. The digits */
/*which follow represent the number of sweep points multiplied by the number  */
/*of bytes per point. */
iHeaderBytes = 3;              /*iDataBytes >3, plus increment for "#" and "n"*/
iDataBytes = (iSwpPnts*iBytesPerPnt);
lNumberBytes = iDataBytes;
while ((iDataBytes = (iDataBytes / 10 )) > 0 )
```

```
{
     iHeaderBytes++;
}

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

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

/*Set the analyzer to 50MHz Span */
viPrintf(viESA,"SENS:FREQ:SPAN 50 MHZ\n");

/* Specify dB per division of each vertical division and  Units */
viPrintf(viESA,"DISP:WIND:TRAC:Y:SCAL:PDIV 10dB\n");
viPrintf(viESA,"UNIT:POW DBM\n");

/*Set analyzer trace data format to 32-bit Real */
viPrintf(viESA,"FORM:DATA REAL,32 \n");

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

/*Trigger a sweep */
viPrintf(viESA,"INIT:IMM \n");

/*Read the operation complete query */
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
{
     printf("Program Abort! error ocurred: last command was not completed!\n");
     exit(0);
}
/*Calculate size of trace record. This will be sum of HeaderBytes, NumberBytes*/
/*(the actual data bytes) and the "/n" terminator*/
iSize = lNumberBytes +iHeaderBytes+1;

/*Get trace header data and trace data */
viPrintf(viESA,"TRAC:DATA? TRACE1\n");
viRead (viESA,(ViBuf)cResult,iSize,&lRetCount);

/*Extract the trace data*/
 memcpy(dTraceArray,cResult+iHeaderBytes,(size_t)lNumberBytes);

/*Save trace data to an ASCII file*/
fTraceFile=fopen("C:\\temp\\ReadTrace32Hpib.txt","w");
```

```
fprintf(fTraceFile,"ReadTrace32Hpib.exe Output\nHewlett-Packard 1999\n\n");
fprintf(fTraceFile,"The %d trace data points of the
spectrum:\n\n",(lNumberBytes/4));
for ( long i=0;i<lNumberBytes/4;i++)
     fprintf(fTraceFile,"\tAmplitude of point[%d] = %.2lf
dBm\n",i+1,dTraceArray[i]);
fclose(fTraceFile);

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

Reading Trace Data Using ASCII Format (RS-232)

```
/************************************************************/
/* Reading Trace Data using ASCII Format (RS-232)          */
/*                                                          */
/* This C programming example does the following.          */
/* The SCPI instrument commands used are given as          */
/* reference.                                               */
/*                                                          */
/* - Opens an RS-232 session at COM1/COM2                   */
/* - Clears the Analyzer                                    */
/* - Resets the Analyzer                                    */
/*      *RST                                                */
/* - Set the input port to the 50 MHz amplitude reference   */
/*      CAL:SOUR:STAT ON                                    */
/* - Query for the number of sweep points (for firmware     */
/*   revisions A.04.00 and later). Default is 401.          */
/*      SENS:SWE:POIN?                                      */
/* - Set the analyzer to single sweep mode                  */
/*      INIT:CONT 0                                         */
/* - Sets the analyzer center frequency and span to 50 MHz  */
/*      SENS:FREQ:CENT 50 MHZ                               */
/*      SENS:FREQ:SPAN 50 MHZ                               */
/* - Trigger a sweep                                        */
/*      INIT:IMM                                            */
/* - Check for operation complete                           */
/*      *OPC?                                               */
/* - Specify dBm Unit                                       */
/*      UNIT:POW DBM                                        */
/* - Set the analyzer trace data ASCII                      */
/*      FORM:DATA: ASC                                      */
/* - Trigger a sweep                                        */
/*      INIT:IMM                                            */
/* - Check for operation complete                           */
/*      *OPC?                                               */
/* - Query the trace data                                   */
/*      TRAC:DATA? TRACE1                                   */
/* - Remove the "," from the ACSII data                     */
/* - Save the trace data to an ASCII file                   */
/* - Close the session                                      */
/************************************************************/

#include <stdio.h>
```

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

#define  hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define  hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"

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

/*Set the input port to 50MHz amplitude reference*/
void Route50MHzSignal()
{
viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)));
if( iResult == 0 )
{
      /*Set the input port to the 50MHz amplitude reference for the models*/
      /*E4411B and E4401B*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
else
{
      /* For the analyzers having frequency limits >= 3GHz, prompt the user to/*
      /* connect the amplitude reference output to the input*/
      printf ("Connect AMPTD REF OUT to the INPUT \n");
      printf ("......Press Return to continue \n");
      scanf( "%c",&cEnter);

      /*Externally route the 50MHz Signal*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
}

void main()
{
/*Program Variable*/
ViStatus viStatus = 0;
/*Dimension cResult to 13 bytes per sweep point, 8192 sweep points maximum*/
```

```
ViChar _VI_FAR cResult[106496] = {0};
FILE *fTraceFile;
    static ViChar *cToken;
int  iNum  =0;
int  iSwpPnts = 401;
long lCount=0L;
long lOpc=0L;

/*iNum set to 13 times number of sweep points, 8192 sweep points maximum*/
iNum =106496;
    lCount =0;

/* Open a serial session at COM1 */
viStatus=viOpenDefaultRM(&defaultRM);
if (viStatus =viOpen(defaultRM,"ASRL1::INSTR",VI_NULL,VI_NULL,&viESA) !=
VI_SUCCESS)
{
      printf("Could not open a session to ASRL device at COM1!\n");
      exit(0);
}
/* Clear the instrument */
viClear(viESA);

/*Reset the instrument. This will set number of sweep points to default of 401*/
  viPrintf(viESA,"*RST\n");

/*Display the program heading */
printf("\n\t\tRead in Trace Data using ASCII Format (RS232) Program \n\n" );

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

/*Query number of sweep points per trace (firmware revision A.04.00 and later)*/
/*For firmware revisions prior to A.04.00, the number of sweep points is 401  */
iSwpPnts = 401;
viQueryf(viESA, "SENSE:SWEEP:POINTS?\n","%d",&iSwpPnts);

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

/*Set the analyzer to 50MHz Span*/
viPrintf(viESA,"SENS:FREQ:SPAN 50 MHz\n");

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

```
/*Trigger a spectrum measurement*/
viPrintf(viESA,"INIT:IMM \n");

/*Read the operation complete query*/
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
{
      printf("Program Abort! error ocurred: last command was not completed!\n");
      exit(0);
}
/*Specify units in dBm*/
viPrintf(viESA,"UNIT:POW DBM \n");

/*Set analyzer trace data format to ASCII Format*/
viPrintf(viESA,"FORM:DATA ASC \n");

/*Trigger a spectrum measurement*/
viPrintf(viESA,"INIT:IMM \n");

/*Read the operation complete query */
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
{
      printf("Program Abort! error ocurred: last command was not completed!\n");
      exit(0);
}
/*Query the Trace Data using ASCII Format */
viQueryf(viESA,"%s\n", "%#t","TRAC:DATA? TRACE1" , &iNum , cResult);

/*Remove the "," from the ASCII trace data for analyzing data*/
    cToken  = strtok(cResult,",");

/*Save trace data to an ASCII to a file, by removing the "," token*/
fTraceFile=fopen("C:\\temp\\ReadAscRS232.txt","w");
fprintf(fTraceFile,"ReadAscRS232.exe Output\nHewlett-Packard 1999\n\n");
fprintf(fTraceFile,"\tAmplitude of point[%d] = %s dBm\n",lCount+1,cToken);
    while (cToken != NULL)
      {
      lCount++;
      cToken =strtok(NULL,",");
             if (lCount != iSwpPnts)
             fprintf(fTraceFile,"\tAmplitude of point[%d] = %s
dBm\n",lCount+1,cToken);
      }
fprintf(fTraceFile,"\nThe Total trace data points of the spectrum are :[%d]
\n\n",lCount);
```

```
fclose(fTraceFile);

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

Reading Trace Data Using 32-bit Real Format (RS-232)

```
/***********************************************************/
/* Reading Trace Data using 32-bit Real Format (RS-232)     */
/*                                                          */
/* This C programming example does the following.           */
/* The SCPI instrument commands used are given as           */
/* reference.                                               */
/*                                                          */
/* - Opens an RS-232 session at COM1/COM2                   */
/* - Clears the Analyzer                                    */
/* - Resets the Analyzer                                    */
/*      *RST                                                */
/* - Set the input port to the 50 MHz amplitude reference   */
/*      CAL:SOUR:STAT ON                                    */
/* - Query for the number of sweep points (for firmware     */
/*   revision A.04.00 and later). Default is 401.           */
/*      SENS:SWE:POIN?                                      */
/* - Calculate the number of bytes in the header            */
/* - Set the analyzer to single sweep mode                  */
/*      INIT:CONT 0                                         */
/* - Sets the analyzer center frequency and span to 50 MHz  */
/*      SENS:FREQ:CENT 50 MHZ                               */
/*      SENS:FREQ:SPAN 50 MHZ                               */
/* - Specify 10 dB per division for the amplitude scale in  */
/*   and dBm Units                                          */
/*      DISP:WIND:TRAC:Y:SCAL:PDIV 10 dB                    */
/*      UNIT:POW DBM                                        */
/* - Set the analyzer trace data to 32-bit Real             */
/*      FORM:DATA: REAL,32                                  */
/* - Set the binary order to swap                           */
/*      FORM:BORD SWAP                                      */
/* - Trigger a sweep                                        */
/*      INIT:IMM                                            */
/* - Check for operation complete                           */
/*      *OPC?                                               */
/* - Calculate the number of bytes in the trace record      */
/* - Set VISA timeout to 60 seconds, to allow for slower    */
/*   transfer times caused by higher number of sweep points */
/*   at low baud rates.                                     */
/* - Set VISA to terminate read after buffer is empty       */
/* - Query the trace data                                   */
/*      TRAC:DATA? TRACE1                                   */
/* - Reset VISA timeout to 3 seconds                          */
/* - Remove the "," from the ACSII data                       */
/* - Save the trace data to an ASCII file                     */
/* - Close the session                                        */
/**************************************************************/

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

#define hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"

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

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

viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)));
if( iResult == 0 )
{
     /*Set the input port to the 50MHz amplitude reference for the models*/
     /*E4411B and E4401B*/
     viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
else
{
     /* For the analyzers having frequency limits >= 3GHz, prompt the user to*/
     /* connect the amplitude reference output to the input*/
     printf ("Connect AMPTD REF OUT to the INPUT \n");
     printf ("......Press Return to continue \n");
     scanf( "%c",&cEnter);

     /*Externally route the 50MHz Signal*/
     viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
}

void main()
{
/*Program Variables*/
ViStatus viStatus= 0;
    ViChar _VI_FAR cResult[1024000] = {0};
ViReal32 dTraceArray[1024] = {0};
char cBufferInfo[7]= {0};
long lNumberBytes =0L;
long lOpc =0L;
unsigned long lRetCount = 0L;
int iSize = 0;
/*BytesPerPnt is 4 for Real32 or Int32 formats, 8 for Real64, and 2 for Uint16*/
int iBytesPerPnt = 4;
int iSwpPnts = 401;   /*Number of points per sweep*/
int iDataBytes = 1604;/*Number of data points, assuming 4 bytes per point*/
int iHeaderBytes = 6; /*Number of bytes in the header, assuming 1604 data bytes*/
FILE *fTraceFile;

/* Open a serial session at COM1 */
viStatus=viOpenDefaultRM(&defaultRM);
if (viStatus =viOpen(defaultRM,"ASRL1::INSTR",VI_NULL,VI_NULL,&viESA) !=
VI_SUCCESS)
{
     printf("Could not open a session to ASRL device at COM1!!\n");
     exit(0);
}
/*Clear the instrument */
viClear(viESA);

/*Reset the instrument. This will set number of sweep points to default of 401*/
viPrintf(viESA,"*RST\n");

/*Display the program heading */
printf("\n\t\t Read in Trace Data using ASCII Format (using RS-232) Program \n\n"
);

/* Set the input port to the internal 50MHz reference source */
Route50MHzSignal();

/*Query number of sweep points per trace (firmware revision A.04.00 or later)*/
/*For firmware revisions prior to A.04.00, the number of sweep points is 401 */
iSwpPnts = 401;
viQueryf(viESA,"SENSE:SWEEP:POINTS?\n","%d",&iSwpPnts);
```

```
/*Calculate number of bytes in the header. The header consists of the "#" sign*/
/*followed by a digit representing the number of digits to follow. The digits */
/*which follow represent the number of sweep points multiplied by the number  */
/*of bytes per point. */
iHeaderBytes = 3;             /*iDataBytes >0, plus increment for "#" and "n" */
iDataBytes = (iSwpPnts*iBytesPerPnt);
    lNumberBytes = iDataBytes;
while ((iDataBytes = (iDataBytes / 10 )) > 0 )
{
     iHeaderBytes++;
}

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

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

/*Set the analyzer to 50MHz Span */
viPrintf(viESA,"SENS:FREQ:SPAN 50 MHZ\n");

/* Specify dB per division of each vertical division & Units */
viPrintf(viESA,"DISP:WIND:TRAC:Y:SCAL:PDIV 10dB\n");
viPrintf(viESA,"UNIT:POW DBM\n");

/*Set analyzer trace data format to 32-bit Real */
viPrintf(viESA,"FORM:DATA REAL,32\n");

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

/*Trigger a sweep */
viPrintf(viESA,"INIT:IMM\n");

/*Read the operation complete query */
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
{
     printf("Program Abort! error ocurred: last command was not completed!\n");
     exit(0);
}
/*Calculate size of trace record. This will be the sum of HeaderBytes, Number*/
/*Bytes (the actual data bytes) and the "\n" terminator*/
iSize = lNumberBytes + iHeaderBytes + 1;
```

```
/*Increase timeout to 60 sec*/
viSetAttribute(viESA,VI_ATTR_TMO_VALUE,60000);

/*Set RS-232 interface to terminate when the buffer is empty*/
viSetAttribute(viESA,VI_ATTR_ASRL_END_IN,VI_ASRL_END_NONE);

/*Get trace header data and trace data*/
viPrintf(viESA,"TRAC:DATA? TRACE1\n");
viRead (viESA,(ViBuf)cResult,iSize,&lRetCount);

/*Reset timeout to 3 sec*/
viSetAttribute(viESA,VI_ATTR_TMO_VALUE,3000);

/*Extract the trace data*/
memcpy(dTraceArray,cResult+iHeaderBytes,(size_t)lNumberBytes);

/*Save trace data to an ASCII file*/
fTraceFile=fopen("C:\\temp\\ReadTrace32Rs232.txt","w");
fprintf(fTraceFile,"ReadTrace32Rs232.exe Output\nHewlett-Packard 1999\n\n");
fprintf(fTraceFile,"The %d trace data points of the
spectrum:\n\n",(lNumberBytes/4));
for ( long i=0;i<lNumberBytes/iBytesPerPnt;i++)
     fprintf(fTraceFile,"\tAmplitude of point[%d] = %.2lf
dBm\n",i+1,dTraceArray[i]);
fclose(fTraceFile);

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

Using Limit Lines

```
/************************************************************/
/* Using Limit Lines                                        */
/*                                                          */
/* This C programming example does the following.           */
/* The SCPI instrument commands used are given as           */
/* reference.                                               */
/*                                                          */
/*                                                          */
/* - Open an HP-IB session at address 18.                   */
/* - Clear the analyzer.                                    */
/*       *CLR                                               */
/* - Reset the analyzer.                                    */
/*       *RST                                               */
/* - Define the upper limit line to have frequency/         */
/*   amplitude pairs.                                       */
/*       CALC:LLINE1:CONT:DOM FREQ                          */
/*       CALC:LLINE1:TYPE UPP                               */
/*       CALC:LLINE1:DISP ON                                */
/*       CALC:LLINE1:DATA freq1,amp1,1,freq2,amp2,1...      */
/* - Define the lower limit line to have frequency/amplitude*/
/*   pairs.                                                 */
/*       CALC:LLINE2:CONT:DOM FREQ                          */
/*       CALC:LLINE2:TYPE LOW                               */
/*       CALC:LLINE2:DISP ON                                */
/*       CALC:LLINE2:DATA freq1,amp1,1,freq2,amp2,1...      */
/* - Turn the limit line test function on.                  */
/*       CALC:LLINE2:STAT ON                                */
/* - Set the analyzer to a center frequency of 50 MHz, span */
/*   to 20 MHz, and resolution bandwidth to 1 MHz.          */
/*       SENS:FREQ:SPAN 20 MHZ                              */
/*       SENS:FREQ:CENT 50 MHZ                              */
/*       SENS:BWID:RES 1 MHZ                                */
/* - Turn the limit line test function on.                  */
/* - Set the analyzer reference level to 0 dBm.             */
/*       DISP:WIND:TRAC:Y:SCAL:RLEV 0                       */
/* - Set the input port to the 50 MHz amplitude reference.  */
/*       CAL:SOUR:STAT ON                                   */
/* - Check to see if limit line passes or fails.  It should */
/*   pass.                                                  */
/*       CALC:LLINE:FAIL?                                   */
/* - Pause for 5 seconds.                                   */
/* - Deactivate the 50 MHz alignment signal.                */
/*       CAL:SOUR:STAT OFF                                */
/* - The limit line test should fail.                     */
/* - Close the session.                                   */
/*********************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>
#include <ctype.h>
#include <string.h>
#include <windows.h>
#include "visa.h"
#define  YIELD Sleep(5000)

#define  hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define  hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"

ViSession defaultRM, viESA;
ViStatus  errStatus;
ViChar    cIdBuff[256]= {0};
char      cEnter = 0;
int       iResult = 0;
long      lLimitTest =0L;

/*Set the input port to 50MHz amplitude reference*/
void Route50MHzSignal()
{
  viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
  iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)));
  if( iResult == 0 )
  {
       /*Set the input port to the 50MHz amplitude reference for the models*/
       /*E4411B and E4401B*/
       viPrintf(viESA,"CAL:SOUR:STAT ON \n");
  }
  else
  {
       /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
       /* to connect the amplitude reference output to the input*/
       printf ("Connect AMPTD REF OUT to the INPUT \n");
       printf ("......Press Return to continue \n");
       scanf( "%c",&cEnter);

       /*Externally route the 50MHz Signal*/
```

```
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
  }
}

void printResult()
{
  viQueryf(viESA, "CALC:CLIM:FAIL?\n", "%ld", &lLimitTest);
    if (lLimitTest!=0)
  {
      printf ("\n\t..Limit Line Failed.....\n");
      viQueryf(viESA, "CALC:LLINE1:FAIL?\n", "%ld", &lLimitTest);
      if (lLimitTest==0)
            printf ("\n\t......Limit Line1 Passed \n");
      else  printf ("\n\t......Limit Line1 Failed \n");

      viQueryf(viESA, "CALC:LLINE2:FAIL?\n", "%ld", &lLimitTest);
      if (lLimitTest==0)
            printf ("\n\t......Limit Line2 Passed \n");

      else printf ("\n\t......Limit Line2 Failed \n");
  }
  else
  printf ("\n\t..Limit Test Pass\n");
}

void main()
{
  /*Program Variable*/
  ViStatus viStatus  = 0;
  long   lOpc =0L;

  /* Open an HP-IB session at address 18*/
  viStatus=viOpenDefaultRM(&defaultRM);
  viStatus=viOpen(defaultRM,"GPIB0::18",VI_NULL,VI_NULL,&viESA);
  if(viStatus)
  {
      printf("Could not open a session to HP-IB device at address 18!\n");
      exit(0);
  }
  /*Clear the instrument*/
  viClear(viESA);

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

  /* Check for the instrument model number and route the 50MHz signal accordingly*/
```

```
/*Route50MHzSignal();

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

  /*Set to Frequency Domain Mode*/
    viPrintf(viESA,"CALC:LLINE1:CONT:DOM FREQ\n");

  /*Delete any current limit line and define the upper limit line
    to have the following frequency/amplitude pairs*/
    viPrintf(viESA,"CALC:LLINE1:TYPE UPP\n");

  /* Turn on display*/
    viPrintf(viESA,"CALC:LLINE1:DISP ON\n");

  /*Send the upper limit line data*/
    viPrintf(viESA,"CALC:LLINE1:DATA 40E06,-50,1, 45E06,-20,1, 50E06,-15,1,
55E06,-20,1, 60E06,-50,1\n");

  /* Turn on display*/
    viPrintf(viESA,"CALC:LLINE1:DISP ON\n");

  /*Delete any current limit line and define the lower limit line
    to have the following frequency/amplitude pairs*/
    viPrintf(viESA,"CALC:LLINE2:TYPE LOW\n");

  /*Send the lower limit line data*/
    viPrintf(viESA,"CALC:LLINE2:DATA
40E06,-100,1,49.99E06,-100,1,50E06,-30,1,50.01E06,-100,1,60E06,-100,1\n");

  /* Turn on display*/
    viPrintf(viESA,"CALC:LLINE2:DISP ON\n");

  /*Turn the limit line test function on.*/
  viPrintf(viESA,"CALC:LLINE2:STAT ON\n");

  /*Set the analyzer to a center frequency of 50 MHz, span to 20 MHz,
    and resolution bandwidth to 1 MHz.*/
    viPrintf(viESA,"SENS:FREQ:CENT 50e6\n");
    viPrintf(viESA,"SENS:FREQ:SPAN 20e6\n");
    viPrintf(viESA,"SENS:BWID:RES 1e6\n");

  /*Set the analyzer reference level to 0 dBm*/
    viPrintf(viESA,"DISP:WIND:TRAC:Y:SCAL:RLEV 0 \n");

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

  /*Trigger a spectrum measurement*/
  viPrintf(viESA,"INIT:IMM \n");
  /*Check for operation complete */

  viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
  if (!lOpc)
  {
      printf("Program Abort! error ocurred: last command was not completed!\n");
      exit(0);
  }
  /*Check to see if limit line passes or fails. It should pass.*/
  printf ("\n\t Limit Line status after activating the 50MHz signal \n");

  /*Print the limits line result*/
  printResult();

  /*Pause for 5 seconds*/
  YIELD;

  /*Deactivate the 50 MHz alignment signal.*/
  viPrintf(viESA,"CAL:SOUR:STAT OFF\n");

  /*Trigger a spectrum measurement*/
  viPrintf(viESA,"INIT:IMM \n");

  /*Check for operation complete */
  viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
  if (!lOpc)
  {
      printf("Program Abort! error ocurred: last command was not completed!\n");
      exit(0);
  }
  /* The limit line test should fail.*/
  printf ("\n\t Limit Line status after de-activating the 50MHz signal \n");

  /*Print the limits line result*/
  printResult();

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

Measuring Noise

```
/************************************************************/
/* Measuring Noise                                          */
/*                                                          */
/* This C programming example does the following.           */
/* The SCPI instrument commands used are given as           */
/* reference.                                               */
/*                                                          */
/* - Opens an HP-IB session at address 18                   */
/* - Clears the Analyzer                                    */
/* - Resets the Analyzer                                    */
/*       *RST                                               */
/* - Sets the center frequency and span                     */
/*       SENS:FREQ:CENT 50 MHZ                              */
/*       SENS:FREQ:SPAN 10 MHZ                              */
/* - Set the input port to the 50 MHz amplitude reference   */
/*       CAL:SOUR:STAT ON                                   */
/* - Set the analyzer to single sweep mode                  */
/*       INIT:CONT 0                                        */
/* - Trigger a sweep                                        */
/*       INIT:IMM                                           */
/* - Check for operation complete                           */
/*       *OPC?                                              */
/* - Set the marker to the maximum peak                     */
/*       CALC:MARK:MAX                                      */
/* - Set the analyzer to active delta marker                */
/*       CALC:MARK:MODE DELT                                */
/* - Set the delta marker to 2 MHZ                          */
/*       CALC:MARK:X 2E+6                                   */
/* - Activate the noise marker function                     */
/*       CALC:MARK:FUNC NOIS                                */
/* - Trigger a sweep                                        */
/*       INIT:IMM                                           */
/* - Check for operation complete                           */
/*       *OPC?                                              */
/* - Query the marker delta amplitude from the analyzer     */
/*       CALC:MARK:Y?                                       */
/* - Report the marker delta amplitude as the carrier to    */
/*   noise ratio in dBc/Hz                                  */
/* - Close the session                                      */
/************************************************************/

#include <stdio.h>
```

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

#define hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"

ViSession defaultRM, viESA;
ViStatus  errStatus;
ViChar    cIdBuff[256]= {0};
char      cEnter = 0;
int       iResult = 0;
long      lOpc =0L;

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

viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)));
if( iResult == 0 )
{
      /*Set the input port to the 50MHz amplitude reference for the models*/
      /*E4411B amd E4401B*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
else
{
      /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
      /* to connect the amplitude reference output to the input*/
      printf ("Connect AMPTD REF OUT to the INPUT \n");
      printf ("......Press Return to continue \n");
      scanf( "%c",&cEnter);

      /*Externally route the 50MHz Signal*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
}

void main()
{
/*Program Variables*/
```

```
ViStatus viStatus  = 0;
double dMarkAmp =0.0;
long lOpc=0L;

/*Open an HP-IB session at address 18*/
viStatus=viOpenDefaultRM(&defaultRM);
viStatus=viOpen(defaultRM,"GPIB0::18",VI_NULL,VI_NULL,&viESA);
if(viStatus)
{
      printf("Could not open a session to HP-IB device at address 18!\n");
      exit(0);
}
/*Clear the Instrument*/
viClear(viESA);

/*Reset the Instrument*/
viPrintf(viESA,"*RST\n");

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

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

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

/*Set the analyzer span to 10MHz*/
viPrintf(viESA,"SENS:FREQ:SPAN 10e6\n");

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

/*Trigger a spectrum measurement*/
viPrintf(viESA,"INIT:IMM \n");

/*Check for operation complete*/
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
{
      printf("Program Abort! error ocurred: last command was not completed!\n");
      exit(0);
}

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

```
/*Check for operation complete*/
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
{
      printf("Program Abort! error ocurred: last command was not completed!\n");
      exit(0);
}

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

/*Trigger a spectrum measurement*/
viPrintf(viESA,"INIT:IMM \n");

/*Set the analyzer in active delta marker mode*/
    viPrintf(viESA,"CALC:MARK:MODE DELT \n");

/*Set the marker delta frequency to 2 MHz. This places the
  active marker two divisions to the right of the input signal.*/
    viPrintf(viESA,"CALC:MARK:X 2E+6 \n");

/*Activate the noise marker function.*/
    viPrintf(viESA,"CALC:MARK:FUNC NOIS \n");

/*Trigger a spectrum measurement*/
viPrintf(viESA,"INIT:IMM \n");

/*Check for operation complete*/
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
{
      printf("Program Abort! error ocurred: last command was not completed!\n");
      exit(0);
}
/*Query and read the marker delta amplitude from the analyzer*/
viPrintf(viESA,"CALC:MARK:Y? \n");
viScanf(viESA,"%lf",&dMarkAmp);

/*Report the marker delta amplitude as the carrier-to-noise ratio in dBc/Hz*/
printf("\t Marker Amplitude = %lf dBc/Hz\n",dMarkAmp);

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

Entering Amplitude Correction Data

```
/*************************************************************/
/* Entering Amplitude Correction Data                        */
/*                                                           */
/* This C programming example does the following.            */
/* The SCPI instrument commands used are given as            */
/* reference.                                                */
/*                                                           */
/* - Opens an HP-IB session at address 18                    */
/* - Clears the Analyzer                                     */
/* - Resets the Analyzer                                     */
/*       *RST                                                */
/* - Sets the stop frequency to 1.5 GHz                      */
/*       SENS:FREQ:STOP 1.5 GHZ                              */
/* - Set the input port to the 50 MHz amplitude reference    */
/*       CAL:SOUR:STAT ON                                    */
/* - Enter amplitude correction frequency/amplitude pairs:   */
/*    0 Hz/ 0 dB, 100 MHz/5 dB, 1 GHz/-5 dB, 1.5 GHz/ 10 dB  */
/*       SENS:CORR:CSET1:DATA 0,0,100E6,5.0,1.0E9,-5.0,...   */
/* - Activate amplitude correction                           */
/*       SENS:CORR:CSET1:DATA                                */
/*       SENS:CORR:CSET1:ALL:STAT ON                         */
/* - Query the analyzer for the amplitude corection factors */
/*       SENS:CORR:CSET1:DATA?                               */
/* - Store them in an array                                  */
/* - Display the array                                       */
/* - Close the session                                       */
/*************************************************************/

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

#define hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"

ViSession defaultRM, viESA;
ViStatus  errStatus;
ViChar    cIdBuff[256]= {0};
```

```
char        cEnter = 0;
int         iResult = 0;

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

viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)));
if( iResult == 0 )
{
      /*Set the input port to the 50MHz amplitude reference for the models*/
      /*E4411B and E4401B*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
else
{
      /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
      /* to connect the amplitude reference output to the input*/
      printf ("Connect AMPTD REF OUT to the INPUT \n");
      printf ("......Press Return to continue \n");
      scanf( "%c",&cEnter);

      /*Externally route the 50MHz Signal*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
}

void main()
{
/*Program Variables*/
    ViChar _VI_FAR cResult[1024] ={0};
ViReal64 _VI_FAR aRealArray[2][100] ={0};
ViStatus viStatus  = 0;
int  iNum =0;
int  iNoOfPoints =0;
long lCount = 0;
long lFreq=0L;
long lAmpltd=1;
    static ViChar *cToken;

/*No of amplitude corrections points */
iNoOfPoints = 4;

/* Open an HP-IB session at address 18*/
```

```
viStatus=viOpenDefaultRM(&defaultRM);
viStatus=viOpen(defaultRM,"GPIB0::18",VI_NULL,VI_NULL,&viESA);
if(viStatus)
{
      printf("Could not open a session to HP-IB device at address 18!\n");
      exit(0);
}
/*Clear the instrument*/
viClear(viESA);

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

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

/*Set the stop frequency to 1.5 GHz */
viPrintf(viESA,"SENS:FREQ:STOP 1.5 GHz\n");

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

/*  Purge any currently-loaded amplitude correction factors*/
    viPrintf(viESA,"SENS:CORR:CSET1:DEL \n");

/* Enter amp cor frequency/amplitude pairs:
   0 Hz, 0 dB, 100 MHz, 5 dB, 1 GHz, -5 dB, 1.5GHz,10*/
viPrintf(viESA,"SENS:CORR:CSET1:DATA ");
viPrintf(viESA,"0, 0.0,");
viPrintf(viESA,"100.E6, 5.0,");
viPrintf(viESA,"1.E9, -5.0,");
viPrintf(viESA,"1.5E9, 10  \n");

/* Activate amplitude correction. Notice that the noise floor slopes
up from 0 Hz to 100 MHz, then downward by 10 dB to 1 GHz, then upwards
again by 15 dB to 1.5 GHz.*/
    viPrintf(viESA,"SENS:CORR:CSET1:STATE ON \n");
    viPrintf(viESA,"SENS:CORR:CSET1:ALL:STAT ON \n");

/*Query the analyzer for its amplitude correction factors */
   viQueryf(viESA,"%s\n", "%#t","SENS:CORR:CSET1:DATA?" , &iNum , cResult);

/*Remove the "," from the amplitude correction for analyzing data*/
    cToken  = strtok(cResult,",");

/*Store the array (frequency) value into a two-dimensional real array*/
```

```
aRealArray[lFreq=0][lCount=0] = atof( cToken);

/*Remove the "," from the amplitude correction for analyzing data*/
cToken =strtok(NULL,",");

/*Store the array(amplitude) value into a two-dimensional real array*/
aRealArray[lAmpltd=1][lCount] = atof(cToken);
while (cToken != NULL)
{
      lCount++;
      if (lCount == iNoOfPoints)
      {
            lCount --;
            break;
      }
      /*Remove the "," from the amplitude correction for analyzing data*/
      cToken =strtok(NULL,",");

      /*Store the array (frequency) value into a two-dimensional real array*/
      aRealArray[lFreq][lCount] = atof(cToken);
      cToken =strtok(NULL,",");

      /*Store the array (amplitude) value into a two-dimensional real array*/
      aRealArray[lAmpltd][lCount] = atof(cToken);
}
/*Display the contents of the array.*/
for (long i=0;i<=lCount;i++)
{
      printf("\tFrequency of point[%d] = %f MHz\n",i,aRealArray[lFreq][i]/1e6);
      printf("\tAmplitude of point[%d] = %f dB\n",i,aRealArray[lAmpltd][i]);
}
/*Close the session*/
viClose(viESA);
viClose(defaultRM);
}
```

Status Register–Determine When a Measurement is Done

```
/*************************************************************/
/* Status Register - Determine when a measurement is done   */
/*                                                           */
/* This C programming example does the following.            */
/* The SCPI instrument commands used are given as            */
/* reference.                                                */
/*                                                           */
/* - Opens an HP-IB session at address 18                    */
/* - Resets the Analyzer                                     */
/*      *RST                                                 */
/* - Clears the analyzer status byte                         */
/*      *CLS                                                 */
/* - Sets the analyzer to single sweep mode                  */
/*      INIT:CONT 0                                          */
/* - Route the amplitude reference to the analyzer input     */
/*      CAL:SOUR:STAT ON                                     */
/* - Set the analyzer center frequency, span and Res BW      */
/*      SENS:FREQ:CENT 50 MHz                                */
/*      SENS:FREQ:SPAN 10 MHz                                */
/*      SENS:BAND:RES 300 kHz                                */
/* - Trigger a sweep and wait for completion of sweep        */
/*      INIT:IMM                                             */
/*      *OPC?                                                */
/* - Sets the service request mask to assert SRQ when        */
/*   either a measurement is uncalibrated or an error        */
/*   message has occurred.                                   */
/*      *SRE 32                                              */
/*      *ESE 35                                              */
/* - Set the computer to response to an interrupt            */
/* - Send an undefined command to the ESA                    */
/*       IDN (illegal command)                               */
/* - Wait for the SRQ                                        */
/* - When an interrupt occurs, poll all instruments          */
/* - Report the nature of the interrupt on the ESA analyzer */
/* - Pause 5 seconds to observe the analyzer                 */
/* - Set the ESA to perform 80 video averages                */
/*       SENS:AVER:TYPE LPOW                                 */
/*       SENS:AVER:COUN 80                                   */
/*       SENS:AVER:STAT ON                                   */
/* - Trigger a measurement, and set OPC bit when done        */
/*       INIT:IMM                                            */
/*       *OPC                                            */
/* - Wait for the SRQ                                    */
/* - When an interrupt occurs, poll all instruments      */
/* - Report the nature of the interrupt on the ESA analyzer */
/* - Clear the status register enable                    */
/*      *SRE 0                                           */
/* - Clear the status byte of the ESA                    */
/*      *CLS                                             */
/* - Close the session                                   */
/*************************************************************/

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

#define  hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define  hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#define  YIELD Sleep(10)

ViSession defaultRM, viESA;
ViStatus  errStatus;
ViChar    cIdBuff[256] = {0};
char      cEnter =0;
int       iResult =0;
int       iSrqOccurred=0;
char      cBuf[3]={0};


/*Wait until SRQ is generated and for the handler to be called. Print
  something while waiting. When interrupt occurs it will be handled by
  interrupt handler*/
void WaitForSRQ()
{
long   lCount = 0L;
iSrqOccurred    =0;

printf ("\t\nWaiting for an SRQ to be generated ...");
for (lCount =0;(lCount<10) && (iSrqOccurred    ==0); lCount++)
{
      long lCount2 =0;
      printf(".");
```

```
        while ((lCount2++ < 100) && (iSrqOccurred   ==0))
        {
                YIELD;
        }
}
printf("\n");

}

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

viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)));
if( iResult == 0 )
{
        /*Set the input port to the 50MHz amplitude reference for the models*/
        /*E4411B and E4401B*/
        viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
else
{
        /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
        /* to connect the amplitude reference output to the input*/
        printf ("Connect AMPTD REF OUT to the INPUT \n");
        printf (".......Press Return to continue \n");
        scanf( "%c",&cEnter);

        /*Externally route the 50MHz Signal*/
        viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
}

/*Interrupt handler,trigger event handler */
ViStatus _VI_FUNCH  mySrqHdlr(ViSession viESA, ViEventType eventType, ViEvent
ctx,ViAddr userHdlr)
{
ViUInt16 iStatusByte;

/* Make sure it is an SRQ event, ignore if stray event*/
if (eventType!=VI_EVENT_SERVICE_REQ)
{
        printf ("\n Stray event type0x%1x\n",eventType);
        /*Return successfully*/
        return VI_SUCCESS;
}
/* When an interrupt occurs,determine which device generated the interrupt
(if an instrument other than the ESA generates the interrupt, simply report
"Instrument at HP-IB Address xxx Has Generated an Interrupt").*/
printf ("\n\n SRQ event occurred!\n");
printf ("\n ... Original Device Session = %t\n",viESA);

/*Get the HP-IB address of the insrument, which has interrupted*/
viQueryf(viESA,"SYST:COMM:GPIB:SELF:ADDR?\n","%t", cBuf);
printf ("\n Instrument at HP-IB address %s has generated an interrupt!\n",cBuf);

/*Get the status byte*/
/* If the ESA generated the interrupt, determine the nature of the interrupt;
  did is the measurement complete or an error message occur?*/
viQueryf(viESA, "*ESR?\n", "%d", &iStatusByte);
if ( (0x01 & iStatusByte))
        printf("\n SRQ message:\t Measurement complete\n");
else if ( (0x02 | 0x10 | 0x20 & iStatusByte ))
        printf ("\n SRQ message:\t Error Message Occurred\n");

/*Return successfully*/
iSrqOccurred   =1;
return VI_SUCCESS;
}
/* Main Program*/
void main()
{
/*Program Variables*/
ViStatus viStatus  = 0;
long   lOpc=0;

/* Open a HP-IB session at address 18*/
viStatus=viOpenDefaultRM(&defaultRM);
int address =18;
viStatus=viOpen(defaultRM,"GPIB0::18",VI_NULL,VI_NULL,&viESA);
if(viStatus)
{
        printf("Could not open a session to HP-IB device at address 18!\n");
        exit(0);
}
/*Clear the instrument*/
viClear(viESA);

/*Reset the instrument*/
```

```
viPrintf(viESA,"*RST\n");

/*Clear the status byte of the instrument*/
viPrintf(viESA,"*CLS\n");

/*Display the program heading */
printf("\n\t Status Register - Determine When a Measuremenet is Done \n\n" );

/*Put the analyzer in a single sweep*/
viPrintf(viESA,"INIT:CONT 0 \n");

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

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

/*Set the analyzer resolution bandwidth to 300 Khz*/
viPrintf(viESA,"SENS:BAND:RES 300 KHz\n");

/*Set the analyzer to 10MHz span*/
viPrintf(viESA,"SENS:FREQ:SPAN 10MHz\n");

/*Trigger a sweep*/
viPrintf(viESA,"INIT:IMM\n");

/*Make sure the previous command has been completed*/
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
{
      printf("Program Abort! error ocurred: last command was not completed!\n");
      exit(0);
}
/* Set the service request mask to assert SRQ when either a measurement
   is completed or an error message has occurred.*/
viPrintf(viESA,"*SRE 32\n");
viPrintf(viESA,"*ESE 35\n");

/* Configure the computer to respond to an interrupt*/
/*install the handler and enable it */
viInstallHandler(viESA, VI_EVENT_SERVICE_REQ, mySrqHdlr,ViAddr(10));
viEnableEvent(viESA, VI_EVENT_SERVICE_REQ,VI_HNDLR,VI_NULL);

/*Send an undefined command to the device*/
viPrintf(viESA,"IDN\n");
```

```
/*Wait for SRQ */
WaitForSRQ();

/* Pause 5 seconds to observe error message displayed on ESA*/
Sleep(5000);

/*Averaging the successive measurements, Set video averaging to 80 sweeps,
/*Turn the avarage On*/
viPrintf(viESA,":SENS:AVER:TYPE LPOW;:SENS:AVER:COUN 80;:SENS:AVER:STAT ON\n");

/*Trigger the sweeps and set the *OPC bit after the sweeps are completed*/
viPrintf(viESA,":INIT:IMM;*OPC\n");

/*Wait for SRQ */
WaitForSRQ();

/*Disable and uninstall the interrupt handler*/
viDisableEvent  (viESA, VI_EVENT_SERVICE_REQ,VI_HNDLR);
viUninstallHandler(viESA, VI_EVENT_SERVICE_REQ, mySrqHdlr,ViAddr(10));

/*Clear the instrument status register*/
viPrintf(viESA,"*SRE 0 \n");

/*Clear the status byte of the instrument*/
viPrintf(viESA,"*CLS\n");

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

Determine if an Error has Occurred

```
/************************************************************/
/* Determine if an error has occurred                       */
/*                                                          */
/* This C programming example does the following.           */
/* The SCPI instrument commands used are given as           */
/* reference.                                               */
/*                                                          */
/* - Opens an HP-IB session at address 18                   */
/* - Clears the Analyzer                                    */
/*      *CLS                                                */
/* - Resets the Analyzer                                    */
/*      *RST                                                */
/* - Sets the service request mask to assert SRQ when       */
/*   either a measurement is uncalibrated or an error       */
/*   message has occurred.                                  */
/*      STAT:QUES:ENAB 512                                  */
/*      STAT:QUES:INT:ENAB 8                                */
/*      *ESE 35                                             */
/*      *SRE 255                                            */
/* - Set the center frequency to 500MHz and span to 100MHz  */
/*      SENS:FREQ:CENT 500 MHZ                              */
/*      SENS:FREQ:SPAN 100 MHZ                              */
/* - Set the analyzer to an uncalibrated state              */
/* - When an interrupt occurs, poll all instruments         */
/* - Report the nature of the interrupt on the ESA analyzer */
/* - Pause 5 seconds to observe the analyzer                */
/* - Sets the service request mask to assert SRQ when       */
/*   either a measurement is uncalibrated or an error       */
/*   message has occurred.                                  */
/*      *ESE 35                                             */
/*      *SRE 255                                            */
/* - Send an illegal command to the ESA                     */
/*       IDN  (illegal command)                             */
/* - When an interrupt occurs, poll all instruments         */
/* - Report the nature of the interrupt on the ESA analyzer */
/* - Clear the analyzer status registers                    */
/*      *SRE 0                                              */
/*      *ESE 0                                              */
/*      STAT:QUES:ENAB 0                                    */
/*      STAT:QUES:INT:ENAB 0                                */
/*      *CLS                                                */
/* - Continue monitoring for an interrupt                   */
/* - Close the session                                           */
/**************************************************************/

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

#define   hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define   hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#define   YIELD Sleep(10)

ViSession defaultRM, viESA;
ViStatus  errStatus;
ViChar    cIdBuff[256] = {0};
char      cEnter =0;
int       iResult =0;
int       iSrqOccurred = 0;
char      cBuf[3]={0};


/*Wait until SRQ is generated and for the handler to be called. Print
 something while waiting. When interrupt occurs it will be handled by
 interrupt handler*/
void WaitForSRQ()
{
long   lCount = 0L;
iSrqOccurred    =0;

printf ("\t\nWaiting for an SRQ to be generated ...");
for (lCount =0;(lCount<10) && (iSrqOccurred    ==0); lCount++)
{
      long lCount2 =0;
      printf(".");
      while ((lCount2++ < 100) && (iSrqOccurred   ==0))
      {
            YIELD;
      }
}
printf("\n");

}
```

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

viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)));
if( iResult == 0 )
{
      /*Set the input port to the 50MHz amplitude reference for the models*/
      /*E4411B and E4401B*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
else
{
      /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
      /* to connect the amplitude reference output to the input*/
      printf ("Connect AMPTD REF OUT to the INPUT \n");
      printf ("......Press Return to continue \n");
      scanf( "%c",&cEnter);

      /*Externally route the 50MHz Signal*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
}

/*Interrupt handler,trigger event handler */
ViStatus _VI_FUNCH  sSrqHdlr(ViSession viESA, ViEventType eventType, ViEvent
ctx,ViAddr userHdlr)
{
ViUInt16 iStatusByte=0;
long lCond = 0L;

/* Make sure it is an SRQ event, ignore if stray event*/
if (eventType!=VI_EVENT_SERVICE_REQ)
{
      printf ("\n Stray event type0x%1x\n",eventType);
      /*Return successfully*/
      return VI_SUCCESS;
}
/* When an interrupt occurs, determine which device generated the interrupt
   (if an instrument other than the ESA generates the interrupt, simply report
   "Instrument at HP-IB Address xxx Has Generated an Interrupt").*/
printf ("\n SRQ Event Occurred!\n");
printf ("\n ... Original Device Session = %1d\n",viESA);
```

```
/*Get the HP-IB address of the insrument, which has interrupted*/
viQueryf(viESA,"SYST:COMM:GPIB:SELF:ADDR?\n","%t", cBuf);
printf ("\n Instrument at HP-IB Address %s Has Generated an Interrupt!\n",cBuf);

/*Get the status byte*/
/* If the ESA generated the interrupt, determine the nature of the interrupt
   either a measurement is uncalibrated or an error message has occurred?*/
viQueryf(viESA, "STAT:QUES:INT:EVEN?\n", "%d", &iStatusByte);
if ( (0x08 & iStatusByte))
      printf("\n SRQ message:\t Measurement uncalibrated\n");

/* If the ESA generated the interrupt, determine the nature of the interrupt;
  did is the measurement complete or an error message occur?*/
viQueryf(viESA, "*ESR?\n", "%d", &iStatusByte);
if ( (iStatusByte !=0) && (0x01 & iStatusByte))
      printf("\n SRQ message:\t Measurement complete\n");
else if ( (iStatusByte !=0) && (0x02 | 0x10 | 0x20 & iStatusByte ))
      printf ("\n SRQ message:\t Error Message Occurred\n");

/*Return successfully*/
iSrqOccurred   =1;
return VI_SUCCESS;
}

void main()
{
/*Program Variables*/
ViStatus viStatus  = 0;
long     lOpc= 0L;
int      iIntNum= 0;
long    lCount= 0L;

/* Open a HP-IB session at address 18*/
viStatus=viOpenDefaultRM(&defaultRM);
viStatus=viOpen(defaultRM,"GPIB0::18",VI_NULL,VI_NULL,&viESA);
if(viStatus)
{
      printf("Could not open a session to HP-IB device at address 18!\n");
      exit(0);
}
/*Clear the instrument*/
viClear(viESA);

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

```
/*Clear the status byte of the instrument*/
viPrintf(viESA,"*CLS\n");

/*Display the program heading */
printf("\n\t\t Status register - Determine if an Error has Occurred\n\n" );

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

/*Put the analyzer in single sweep*/
viPrintf(viESA,"INIT:CONT 0 \n");

/*Set the service request mask to assert SRQ when either a measurement
is uncalibrated (i.e. "Meas Uncal" displayed on screen) or an error
message has occurred.*/
viPrintf(viESA,"STAT:QUES:ENAB 512\n");
 viPrintf(viESA,"STAT:QUES:INT:ENAB 8\n");
viPrintf(viESA,"*ESE 35\n");
viPrintf(viESA,"*SRE 255\n");

/*Configure the computer to respond to an interrupt,install the handler
  and enable it */
viInstallHandler(viESA, VI_EVENT_SERVICE_REQ, sSrqHdlr,ViAddr(10));
viEnableEvent(viESA, VI_EVENT_SERVICE_REQ,VI_HNDLR,VI_NULL);
iSrqOccurred    =0;

/*Set the analyzer to a 500 MHz center frequency*/
viPrintf(viESA,"SENS:FREQ:CENT 500 MHZ \n");

/*Set the analyzer to a 100 MHz span*/
viPrintf(viESA,"SENS:FREQ:SPAN 100 MHZ\n");

/*Set the analyzer to a  auto resolution BW*/
viPrintf(viESA,"SENS:BAND:RES:AUTO 1\n");

/*Set the analyzer to a  Auto Sweep Time*/
viPrintf(viESA,"SENS:SWE:TIME:AUTO 1\n");

/*Allow analyzer to sweep several times.*/
viPrintf(viESA,"INIT:CONT 1 \n");

/*Manually couple sweeptime to 5ms. reduce resolution BW to 30 KHz.
"Meas Uncal" should be displayed on the screen, and an interrupt should
 be generated.*/
viPrintf(viESA,"SENS:SWE:TIME 5 ms \n");
viPrintf(viESA,"SENS:BAND:RES 30 KHZ  \n");

/*Wait for SRQ*/
WaitForSRQ();

/*Pause for 5 seconds to observe "Meas Uncal" message on ESA display*/
Sleep(5000);

/* Set the service request mask to assert SRQ when either a measurement
   is completed or an error message has occurred.*/
viPrintf(viESA,"*SRE 32\n");
viPrintf(viESA,"*ESE 35\n");

/*Send an undefined command to the device*/
viPrintf(viESA,"IDN\n");

/*Wait for SRQ*/
WaitForSRQ();

/*Disable and uninstall the interrupt handler*/
viDisableEvent  (viESA, VI_EVENT_SERVICE_REQ,VI_HNDLR);
viUninstallHandler(viESA, VI_EVENT_SERVICE_REQ, sSrqHdlr,ViAddr(10));

/*Clear the instrument status register*/
viPrintf(viESA,"*SRE 0 \n");
viPrintf(viESA,"*ESE 0 \n");
viPrintf(viESA,"STAT:QUES:ENAB 0\n");
 viPrintf(viESA,"STAT:QUES:INT:ENAB 0\n");

/*Clear the status byte of the instrument*/
viPrintf(viESA,"*CLS\n");

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

Measuring Harmonic Distortion (HP-IB)

```
/*************************************************************/
/* Measuring Harmonic Distortion (HP-IB)                     */
/*                                                           */
/* This C programming example does the following.            */
/* The SCPI instrument commands used are given as            */
/* reference.                                                */
/*                                                           */
/* - Opens an HP-IB session at address 18                    */
/* - Clears the Analyzer                                     */
/*      *CLS                                                 */
/* - Resets the Analyzer                                     */
/*      *RST                                                 */
/* - Set the input port to the 50 MHz reference              */
/*      CAL:SOUR:STAT ON                                     */
/* - Set the analyzer center frequency to the fundamental    */
/*      SENS:FREQ:CENT freq                                  */
/* - Set the analyzer to 10 MHz span                         */
/*      SENS:FREQ:SPAN 10 MHZ                                */
/* - Set the analyzer to single sweep mode                   */
/*      INIT:CONT 0                                          */
/* - Trigger a sweep                                         */
/*      INIT:IMM                                             */
/* - Check for operation complete                            */
/*      *OPC?                                                */
/* - Perform the peak search                                 */
/*      CALC:MARK:MAX                                        */
/* - Set the marker to reference level                       */
/*      CALC:MARK:SET:RLEV                                   */
/* - Trigger a sweep                                         */
/*      INIT:IMM                                             */
/* - Check for operation complete                            */
/*      *OPC?                                                */
/* - Perform the peak search                                 */
/*      CALC:MARK:MAX                                        */
/* - Change VISA timeout to 60 seconds                       */
/* - Activate signal track                                   */
/*      CALC:MARK:TRCK:STAT ON                               */
/* - Perform narrow span and wait                            */
/*      SENS:FREQ:SPAN 10e4                                  */
/* - Check for operation complete                            */
/*      *OPC?                                                */
/* - De-activate signal track                                */
/*      CALC:MARK:TRCK:STAT OFF                              */
/* - Reset VISA timeout to 3 seconds                         */
/* - Set units to dBm                                        */
/*      UNIT:POW DBM                                         */
/* - Take a sweep                                            */
/*      INIT:IMM                                             */
/* - Check for operation complete                            */
/*      *OPC?                                                */
/* - Perform the peak search                                 */
/*      CALC:MARK:MAX                                        */
/* - Read the marker amplitude,this is the fundamental Level*/
/*      CALC:MARK:Y?                                         */
/* - Change the amplitude units to volts                     */
/*      UNIT:POW V                                           */
/* - Take a sweep                                            */
/*      INIT:IMM                                             */
/* - Check for operation complete                            */
/*      *OPC?                                                */
/* - Read the marker amplitude in volts, this is the         */
/*   fundamental amplitude in volts.                         */
/*      CALC:MARK:Y?                                         */
/* - Read the marker frequency                               */
/*      CALC:MARK:X?                                         */
/* - Measure each harmonic amplitude as follows:             */
/*     Set the span to 20 MHz                                */
/*        SENS:FREQ:SPAN 20 MHZ                              */
/*     Set the center frequency to the desired harmonic      */
/*        SENS:FREQ:CENT <freq>                              */
/*     Take a sweep and wait for operation complete          */
/*        INIT:IMM                                           */
/*        *OPC?                                              */
/*     Perform peak search                                   */
/*        CALC:MARK:MAX                                      */
/*     Set VISA timeout to 60 seconds                        */
/*     Activate signal track                                 */
/*        CALC:MARK:TRCK:STAT ON                             */
/*     Zoom down to a 100 kHz span                           */
/*        SENS:FREQ:SPAN 10E4                                */
/*     Take a sweep and wait for operation complete          */
/*        INIT:IMM                                           */
/*        *OPC?                                              */
/*     Signal track off                                      */
/*        CALC:MARK:TRCK:STAT OFF                            */
/*     Reset VISA timeout to 3 seconds                       */
/*     Perform Peak Search                                   */
/*        CALC:MARK:MAX                                      */
/*     Set marker amplitude in volts                          */
/*        UNIT:POW V                                          */
/*     Query, read the marker amplitude in volts              */
/*        CALC:MARK:Y?                                        */
/*     Change the amplitude units to dBm and read the         */
/*     marker amplitude.                                      */
/*        UNIT:POW DBM                                        */
/* - Calculate the relative amplitude of each harmonic        */
/*   reletive to the fundamental                              */
/* - Calculate the total harmonic distortion                  */
/* - Display the fundamental amplitude in dBm, fundamental  */
/*   frequency in MHz, relative amplitude of each harmonic  */
/*   in dBc and total harmonic distortion in percent          */
/* - Close the session                                        */
/************************************************************/

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

#define hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"

ViSession defaultRM, viESA;
ViStatus  errStatus;
ViChar    cIdBuff[256]= {0};
char      cEnter = 0;
int       iResult = 0;
long      lOpc =0L;

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

viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)));
if( iResult == 0 )
{
     /*Set the input port to the 50MHz amplitude reference for the models*/
     /*E4411B and E4401B*/
     viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
else
{
     /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
     /* to connect the amplitude reference output to the input*/
     printf ("Connect AMPTD REF OUT to the INPUT \n");
     printf ("......Press Return to continue \n");
     scanf( "%c",&cEnter);
     /*Externally route the 50MHz Signal*/
     viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
}

void TakeSweep()
{
/*Take a sweep and wait for the sweep completion*/
viPrintf(viESA,"INIT:IMM\n");
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
{
     printf("Program Abort! Error occurred: last command was not completed! \n");
     exit(0);
}
}

void main()
{
/*Program Variables*/
ViStatus viStatus  = 0;
double dFundamental = 0.0;
double dHarmFreq =0.0;
float fHarmV[10] ={0.0};
float fHarmDbm[10]={0.0};
float fRelAmptd[10]={0.0};
float fFundaAmptdDbm=0.0;
double dFundaAmptdV=0.0;
double dMarkerFreq = 0.0;
double dPrcntDistort =0.0;
double dSumSquare =0.0;
long   lMaxHarmonic =0L;
long   lNum=0L;

/*Setting default values*/
lMaxHarmonic =5;
dFundamental =50.0;
```

```
/* Open a HP-IB session at address 18*/
viStatus=viOpenDefaultRM(&defaultRM);
viStatus=viOpen(defaultRM,"GPIB0::18",VI_NULL,VI_NULL,&viESA);
if(viStatus)
{
     printf("Could not open a session to HP-IB device at address 18!\n");
     exit(0);
}
/*Clear the instrument*/
viClear(viESA);

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

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

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

/*Prompt user for fundamental frequency*/
printf("\t Enter the input signal fundamental frequency in MHz ");

/*The user enters fundamental frequency*/
scanf("%lf",&dFundamental);

/*Set the analyzer center frequency to the fundamental frequency. */
viPrintf(viESA,"SENS:FREQ:CENT %lf MHZ \n;",dFundamental);

/*Set the analyzer to 10MHz Span */
viPrintf(viESA,"SENS:FREQ:SPAN 10 MHZ\n");

/*Put the analyzer in a single sweep  */
viPrintf(viESA,"INIT:CONT 0 \n");

/*Trigger a sweep*/
viPrintf(viESA,"INIT:IMM\n");

/*Check for operation complete*/
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
{
     printf("Program Abort! Error occurred: last command was not completed !\n");
     exit(0);
}
/*Perform a peak search */
```

```
viPrintf(viESA,"CALC:MARK:MAX \n");

/* Place the signal at the reference level using the
   marker-to-reference level command and take sweep */
viPrintf(viESA,"CALC:MARK:SET:RLEV \n");

/*Trigger a sweep*/
viPrintf(viESA,"INIT:IMM\n");

/*Check for operation complete */
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
{
     printf("Program Abort! Error occurred: last command was not completed! \n");
     exit(0);
}
/*Perform a peak search */
viPrintf(viESA,"CALC:MARK:MAX \n");

/*increase timeout to 60 sec*/
viSetAttribute(viESA,VI_ATTR_TMO_VALUE,60000);

/*Perform activate signal track */
viPrintf(viESA,"CALC:MARK:TRCK:STAT ON \n");

/*Take a sweep and wait for the sweep completion*/
TakeSweep();

/*Perform  narrow span and wait */
viPrintf(viESA,"SENS:FREQ:SPAN 10e4 \n");

/*Take a sweep and wait for the sweep completion*/
TakeSweep();

/*De activate the signal track */
viPrintf(viESA,"CALC:MARK:TRCK:STAT OFF \n");

/*Reset timeout to 3 sec*/
viSetAttribute(viESA,VI_ATTR_TMO_VALUE,3000);

/*Set units to DBM*/
viPrintf(viESA,"UNIT:POW DBM \n");

/*Perform a peak search */
viPrintf(viESA,"CALC:MARK:MAX \n");
```

```
/*Read the marker amplitude, this is the fundamental amplitude
  in dBm */
viQueryf(viESA,"CALC:MARK:Y?\n","%1f", &fFundaAmptdDbm);

/*Change the amplitude units to Volts */
viPrintf(viESA,"UNIT:POW V \n");

/*Read the marker amplitude in volts, This is the fundamental amplitude
  in Volts (necessary for the THD calculation).*/
viPrintf(viESA,"CALC:MARK:Y?\n");
viScanf(viESA,"%lf",&dFundaAmptdV);

/*Read the marker frequency. */
viPrintf(viESA,"CALC:MARK:X? \n");
viScanf(viESA,"%lf",&dMarkerFreq);
dFundamental = dMarkerFreq;

/*Change the center frequency step size equal to the marker frequency.*/
/*viPrintf(viESA,"CALC:MARK:SET:STEP \n"); */

/*Measure each harmonic amplitude as follows: */
for ( lNum=2;lNum<=lMaxHarmonic;lNum++)
{
     /*Measuring the Harmonic No#[%d] message */
     printf("\n\t Measuring the Harmonic No [%d] \n",lNum );

     /*Set the span to 20 MHz*/
     viPrintf(viESA,"SENS:FREQ:SPAN 20 MHZ \n");

     /*Set the center frequency to the nominal harmonic frequency*/
     dHarmFreq = lNum*dFundamental;
     viPrintf(viESA,"SENS:FREQ:CENT %lf HZ \n;",dHarmFreq);

     /*Take a sweep and wait for the sweep completion*/
     TakeSweep();

     /*Perform a peak search and wait for completion */
     viPrintf(viESA,"CALC:MARK:MAX\n");

     /*increase timeout to 60 sec*/
     viSetAttribute(viESA,VI_ATTR_TMO_VALUE,60000);

     /*Activate signal track */
     viPrintf(viESA,"CALC:MARK:TRCK:STAT ON\n");

     /*Zoom down to a 100 kHz span */
     viPrintf(viESA,"SENS:FREQ:SPAN 10e4\n");

     /*Take a sweep and  wait for the sweep completion*/
     TakeSweep();

     /* Signal track off */
     viPrintf(viESA,"CALC:MARK:TRCK:STAT OFF\n");

     /*Reset timeout to 3 sec*/
     viSetAttribute(viESA,VI_ATTR_TMO_VALUE,3000);

     /*Set Marker Amplitude in Volts*/
     viPrintf(viESA,"UNIT:POW V\n");

     /*Perform a peak search and  wait for completion*/
     viPrintf(viESA,"CALC:MARK:MAX\n");

     /*Query and read the Marker Amplitude in Volts*/
     /*Store the result in the array.*/
     viQueryf(viESA,"CALC:MARK:Y?\n","%1f", &fHarmV[lNum]);

     /*Change the amplitude units to DBM  */
     viPrintf(viESA,"UNIT:POW DBM\n");

     /* Read the marker amplitude */
     viQueryf(viESA,"CALC:MARK:Y?\n","%1f", &fHarmDbm[lNum]);
     }

/*Sum the square of each element in the fHarmV array. Then
  calculate the relative amplitude of each harmonic relative
  to the fundamental */
for (lNum=2;lNum<=lMaxHarmonic;lNum++)
{
     dSumSquare= dSumSquare + (pow (double(fHarmV[lNum]) ,2.0));
     /* Relative Amplitude  */
     fRelAmptd[lNum] = fHarmDbm[lNum] - fFundaAmptdDbm ;
}
/*Calculate the total harmonic distortion by dividing the square root of
the sum of the squares (dSumSquare) by the fundamental amplitude in Volts
(dFundaAmptdV).Multiply this value by 100 to obtain a result in percent*/
dPrcntDistort = ((sqrt(double (dSumSquare)))  /dFundaAmptdV) *100 ;

/*Fundamental amplitude in dBm */
printf("\n\t Fundamental Amplitude: %lf dB \n\n",fFundaAmptdDbm);

/*Fundamental Frequency in MHz*/
```

```
printf("\t Fundamental Frequency is: %lf MHz \n\n",dFundamental/10e5);

/*Relative amplitude of each harmonic in dBc*/
for (lNum=2;lNum<=lMaxHarmonic;lNum++)
     printf("\t Relative amplitude of Harmonic[%d]: %lf dBc
\n\n",lNum,fRelAmptd[lNum]);

/*Total harmonic distortion in percent*/
printf("\t Total Harmonic Distortion: %lf percent \n\n",dPrcntDistort);

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

Measuring Harmonic Distortion (RS-232)

```
/************************************************************/
/* Measuring Harmonic Distortion (RS-232)                   */
/*                                                          */
/* This C programming example does the following.           */
/* The SCPI instrument commands used are given as           */
/* reference.                                               */
/*                                                          */
/* - Opens an RS-232 session to the COM1 serial port        */
/* - Clears the Analyzer                                    */
/*      *CLS                                                */
/* - Resets the Analyzer                                    */
/*      *RST                                                */
/* - Set the input port to the 50 MHz reference             */
/*      CAL:SOUR:STAT ON                                    */
/* - Set the analyzer center frequency to the fundamental   */
/*      SENS:FREQ:CENT freq                                 */
/* - Set the analyzer to 10 MHz span                        */
/*      SENS:FREQ:SPAN 10 MHZ                               */
/* - Set the analyzer to single sweep mode                  */
/*      INIT:CONT 0                                         */
/* - Trigger a sweep                                        */
/*      INIT:IMM                                            */
/* - Check for operation complete                           */
/*      *OPC?                                               */
/* - Perform the peak search                                */
/*      CALC:MARK:MAX                                       */
/* - Set the marker to reference level                      */
/*      CALC:MARK:SET:RLEV                                  */
/* - Trigger a sweep                                        */
/*      INIT:IMM                                            */
/* - Check for operation complete                           */
/*      *OPC?                                               */
/* - Perform the peak search                                */
/*      CALC:MARK:MAX                                       */
/* - Change VISA timeout to 60 seconds                      */
/* - Activate signal track                                  */
/*      CALC:MARK:TRCK:STAT ON                              */
/* - Perform narrow span and wait                           */
/*      SENS:FREQ:SPAN 10e4                                 */
/* - Check for operation complete                           */
/*      *OPC?                                               */
/* - De-activate signal track                               */
/*      CALC:MARK:TRCK:STAT OFF                                    */
/* - Reset VISA timeout to 3 seconds                               */
/* - Set units to dBm                                              */
/*      UNIT:POW DBM                                               */
/* - Take a sweep                                                  */
/*      INIT:IMM                                                   */
/* - Check for operation complete                                  */
/*      *OPC?                                                      */
/* - Perform the peak search                                       */
/*      CALC:MARK:MAX                                              */
/* - Read the marker amplitude,this is the fundamental Level*/
/*      CALC:MARK:Y?                                               */
/* - Change the amplitude units to volts                           */
/*      UNIT:POW V                                                 */
/* - Take a sweep                                                  */
/*      INIT:IMM                                                   */
/* - Check for operation complete                                  */
/*      *OPC?                                                      */
/* - Read the marker amplitude in volts, this is the               */
/*   fundamental amplitude in volts.                               */
/*      CALC:MARK:Y?                                               */
/* - Read the marker frequency                                     */
/*      CALC:MARK:X?                                               */
/* - Measure each harmonic amplitude as follows:                   */
/*     Set the span to 20 MHz                                      */
/*        SENS:FREQ:SPAN 20 MHZ                                    */
/*     Set the center frequency to the desired harmonic           */
/*        SENS:FREQ:CENT <freq>                                    */
/*     Take a sweep and wait for operation complete               */
/*        INIT:IMM                                                 */
/*        *OPC?                                                    */
/*     Perform peak search                                         */
/*        CALC:MARK:MAX                                            */
/*     Set VISA timeout to 60 seconds                              */
/*     Activate signal track                                       */
/*        CALC:MARK:TRCK:STAT ON                                   */
/*     Zoom down to a 100 kHz span                                 */
/*        SENS:FREQ:SPAN 10E4                                      */
/*     Take a sweep and wait for operation complete               */
/*        INIT:IMM                                                 */
/*        *OPC?                                                    */
/*     Signal track off                                            */
/*        CALC:MARK:TRCK:STAT OFF                                  */
/*     Reset VISA timeout to 3 seconds                             */
/*     Perform Peak Search                                         */
/*        CALC:MARK:MAX                                            */
/*     Set marker amplitude in volts                        */
/*        UNIT:POW V                                        */
/*     Query, read the marker amplitude in volts            */
/*        CALC:MARK:Y?                                      */
/*     Change the amplitude units to dBm and read the       */
/*     marker amplitude.                                    */
/*        UNIT:POW DBM                                      */
/* - Calculate the relative amplitude of each harmonic      */
/*   reletive to the fundamental                            */
/* - Calculate the total harmonic distortion                */
/* - Display the fundamental amplitude in dBm, fundamental  */
/*   frequency in MHz, relative amplitude of each harmonic  */
/*   in dBc and total harmonic distortion in percent        */
/* - Close the session                                      */
/***********************************************************/

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

#define hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"

ViSession defaultRM, viESA;
ViStatus  errStatus;
ViChar    cIdBuff[256]= {0};
char      cEnter = 0;
int       iResult = 0;
long      lOpc =0L ;

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

viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)));
if( iResult == 0 )
{
      /*Set the input port to the 50MHz amplitude reference for the models*/
      /*E4411B, E4401B*/
      viPrintf(viESA,"CAL:SOUR:STAT ON\n");
}
else
{
      /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
      /* to connect the amplitude reference output to the input*/
      printf ("Connect AMPTD REF OUT to the INPUT \n");
      printf ("......Press Return to continue \n");
      scanf( "%c",&cEnter);

      /*Externally route the 50MHz Signal*/
      viPrintf(viESA,"CAL:SOUR:STAT ON\n");
}
}

void TakeSweep()
{
/*Take a sweep and wait for the sweep completion*/
viPrintf(viESA,"INIT:IMM\n");
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
{
      printf("Program Abort! Error occurred: last command was not completed! \n");
      exit(0);
}
}

void main()
{
/*Program Variables*/
ViStatus viStatus  = 0;
double dFundamental = 0.0;
double dHarmFreq = 0.0;
float fHarmV[10] ={0.0};
float fHarmDbm[10]={0.0};
float fRelAmptd[10]={0.0};
float fFundaAmptdDbm=0.0;
double dFundaAmptdV=0.0;
double dMarkerFreq = 0.0;
double dPrcntDistort =0.0;
double dSumSquare =0.0;
long   lMaxHarmonic =0L;
long   lNum=0L;


/*Setting default values*/
lMaxHarmonic =5;
```

```
dFundamental =50.0;


/* Open a serial session at COM1 */
viStatus=viOpenDefaultRM(&defaultRM);
if (viStatus =viOpen(defaultRM,"ASRL1::INSTR",VI_NULL,VI_NULL,&viESA) !=
VI_SUCCESS)
{
      printf("Could not open a session to ASRL device at COM1!\n");
      exit(0);
}
/*Clear the instrument*/
viClear(viESA);

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

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

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

/*Prompt user for fundamental frequency*/
printf("\t Enter the input signal fundamental frequency in MHz ");

/*The user enters fundamental frequency*/
scanf("%lf",&dFundamental);

/*Set the analyzer center frequency to the fundamental frequency. */
viPrintf(viESA,"SENS:FREQ:CENT %lf MHZ\n",dFundamental);

/*Set the analyzer to 10MHz Span */
viPrintf(viESA,"SENS:FREQ:SPAN 10 MHZ\n");

/*Put the analyzer in a single sweep mode */
viPrintf(viESA,"INIT:CONT 0\n");

/*Trigger a sweep*/
viPrintf(viESA,"INIT:IMM\n");

/*Check for operation complete*/
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
{
      printf("Program Abort! Error occurred: last command was not completed !\n");
```

```
        exit(0);
}
/*Perform a peak search */
viPrintf(viESA,"CALC:MARK:MAX\n");

/* Place the signal at the reference level using the
   marker-to-reference level command and take sweep */
viPrintf(viESA,"CALC:MARK:SET:RLEV\n");

/*Trigger a sweep */
viPrintf(viESA,"INIT:IMM\n");

/*Check for operation complete */
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
{
        printf("Program Abort! Error occurred: last command was not completed! \n");
        exit(0);
}
/*Perform a peak search */
viPrintf(viESA,"CALC:MARK:MAX\n");

/*Increase timeout to 60 sec*/
viSetAttribute(viESA,VI_ATTR_TMO_VALUE,60000);

/*Perform activate signal track */
viPrintf(viESA,"CALC:MARK:TRCK:STAT ON\n");

/*Perform  narrow span and  wait */
viPrintf(viESA,"SENS:FREQ:SPAN 10e4\n");

/*Take a sweep and wait for the sweep completion*/
TakeSweep();

/*De activate the signal track */
viPrintf(viESA,"CALC:MARK:TRCK:STAT OFF\n");

/*Reset timeout to 3 sec*/
viSetAttribute(viESA,VI_ATTR_TMO_VALUE,3000);

/*Set units to dBm*/
viPrintf(viESA,"UNIT:POW DBM\n");

/*Perform a peak search */
viPrintf(viESA,"CALC:MARK:MAX\n");
```

```
/*Read the marker amplitude, this is the fundamental amplitude
  in dBm */
viQueryf(viESA,"CALC:MARK:Y?\n","%1f", &fFundaAmptdDbm);

/*Change the amplitude units to Volts */
viPrintf(viESA,"UNIT:POW V\n");

/*Read the marker amplitude in volts, This is the fundamental amplitude
  in Volts (necessary for the THD calculation).*/
viPrintf(viESA,"CALC:MARK:Y?\n");
viScanf(viESA,"%lf",&dFundaAmptdV);

/*Read the marker frequency. */
viPrintf(viESA,"CALC:MARK:X?\n");
viScanf(viESA,"%lf",&dMarkerFreq);
dFundamental = dMarkerFreq;

/*Measure each harmonic amplitude as follows: */
for ( lNum=2;lNum<=lMaxHarmonic;lNum++)
{
      /*Measuring the Harmonic No#[%d] message */
      printf("\n\t Measuring the Harmonic No [%d] \n",lNum );

      /*Set the span to 20 MHz*/
      viPrintf(viESA,"SENS:FREQ:SPAN 20 MHZ\n");

      /*Set the center frequency to the nominal harmonic frquency*/
      dHarmFreq = lNum * dFundamental;
      viPrintf(viESA,"SENS:FREQ:CENT %lf HZ\n",dHarmFreq);

      /*Take a sweep and wait for the sweep completion*/
      TakeSweep();

      /*Perform a peak search and  wait for completion */
      viPrintf(viESA,"CALC:MARK:MAX\n");

      /*Increase timeout to 60 sec*/
      viSetAttribute(viESA,VI_ATTR_TMO_VALUE,60000);

      /*Activate signal track */
      viPrintf(viESA,"CALC:MARK:TRCK:STAT ON\n");

      /*Zoom down to a 100 KHz span */
      viPrintf(viESA,"SENS:FREQ:SPAN 10e4\n");

      /*Take a sweep and wait for the sweep completion*/
```

```
        TakeSweep();

        /* Signal track off */
        viPrintf(viESA,"CALC:MARK:TRCK:STAT OFF\n");

        /*Reset timeout to 3 sec*/
        viSetAttribute(viESA,VI_ATTR_TMO_VALUE,3000);

        /*Perform a peak search and wait for completion*/
        viPrintf(viESA,"CALC:MARK:MAX\n");

        /*Set marker amplitude in Volts*/
        viPrintf(viESA,"UNIT:POW V\n");

        /*Query and read the marker amplitude in Volts*/
        /*Store the result in the fHarmV array.*/
        viQueryf(viESA,"CALC:MARK:Y?\n","%1f", &fHarmV[lNum]);

        /*Change the amplitude units to dBm */
        viPrintf(viESA,"UNIT:POW DBM\n");

        /* Read the marker amplitude */
        viQueryf(viESA,"CALC:MARK:Y?\n","%1f", &fHarmDbm[lNum]);
        }

/*Sum the square of each element in the fHarmV array and calculate
  the relative amplitude of each harmonic relative to the fundamental*/
for (lNum=2;lNum<=lMaxHarmonic;lNum++)
{
        dSumSquare= dSumSquare + (pow (double(fHarmV[lNum]) ,2.0));

        /* Relative Amplitude  */
        fRelAmptd[lNum] = fHarmDbm[lNum] - fFundaAmptdDbm ;
}
/*Calculate the total harmonic distortion by dividing the square root of
the sum of the squares (dSumSquare) by the fundamental amplitude in Volts
(dFundaAmptdV).Multiply this value by 100 to obtain a result in percent*/
dPrcntDistort = ((sqrt(double (dSumSquare)))  /dFundaAmptdV) *100 ;

/*Fundamental amplitude in dBm */
printf("\nFundamental Amplitude: %lf dB \n",fFundaAmptdDbm);

/*Fundamental frequency in MHz*/
printf("Fundamental Frequency is: %lf MHz \n",dFundamental/10e5);

/*Relative amplitude of each harmonic in dBc*/
```

```
for (lNum=2;lNum<=lMaxHarmonic;lNum++)
      printf("Relative amplitude of Harmonic[%d]: %lf dBc
\n",lNum,fRelAmptd[lNum]);

/*Total harmonic distortion in percent*/
printf("Total Harmonic Distortion: %lf percent\n",dPrcntDistort);

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

Making Faster Measurements (multiple measurements)

```
/************************************************************/
/* Making Faster Measurements (multiple measurements)       */
/*                                                          */
/* This C programming example does the following:           */
/* Performs Power Averaging of Multiple ESA Measurements    */
/* and Writes the Result back to a Trace for display        */
/*                                                          */
/* This program demonstrates a quick way to obtain          */
/* trace data from the analyzer. The ESA display            */
/* and the IF, Video, and Sweep ports are disabled to       */
/* improve trace data throughput. It also performs          */
/* computations on the trace data (a power average in this  */
/* case) and returns the result to the analyzer for display.*/
/*                                                          */
/* The example below uses the VISA library functions. The   */
/* speed will in most cases improve if written to use the   */
/* National Instruments IEEE-488.2 library functions        */
/* instead of the VISA library functions.                   */
/*                                                          */
/* This program uses the following commands found in ESA    */
/* firmware revisions A.04.00 and later:                    */
/*    :SYST:PRES:TYPE FACT (sets ESA to factory preset)     */
/*    :SYST:PORT:IFVS:ENAB OFF (improves speed)             */
/*    :SWE:POINTS (sets number of sweep points, fewer       */
/*        points improve speed)                             */
/* The above commands may be omitted in ESAs with firmware  */
/* revisions prior to A.04.00, but the speed may be         */
/* degraded slightly.                                       */
/*                                                          */
/* The associated SCPI instrument commands are given as     */
/* reference.                                               */
/*                                                          */
/* - Opens a GPIB device at address 18                      */
/* - Clears and Resets the Analyzer to a known state        */
/*      SYST:PRES:TYPE FACT                                 */
/*      *RST                                                */
/* - Identify the Instrument model                          */
/*      *IDN?                                               */
/* - Sets the analyzer center frequency and span            */
/*      SENS:FREQ:CENT freq                                 */
/*      SENS:FREQ:SPAN freq                                 */
/* - Sets the analyzer resolution bandwidth                  */
/*      SENS:BAND rbw                                        */
/* - Selects sampled as the detector mode                    */
/*      SENS:DET SAMP                                        */
/* - Disable optional Output functions                       */
/*      :SYST:PORT:IFVS:ENAB OFF                             */
/* - Turn off auto-alignment                                 */
/*      CAL:AUTO OFF                                         */
/* - Select the desired number of sweep points               */
/*      SWE:POINTS points                                    */
/* - Connect the input port to the 50 MHz amptd reference    */
/*   E4402B, E4403B, E4404B, E4405B, E4407B or E4408B        */
/* - Activate the 50 MHz amplitude reference signal          */
/*      CAL:SOUR STAT ON                                     */
/* - Select the appropriate display reference level          */
/*      E4402, E4403B, E4404B, E4405B, E4407B or E4408B      */
/*   DISP:WIND:TRAC:Y:RLEV -20 DBM                           */
/*      E4411B or E4401B                                     */
/*   DISP:WIND:TRAC:Y:RLEV -25 DBM                           */
/* - Select single sweep mode                                */
/*      INIT:CONT OFF                                        */
/* - Disable local display                                   */
/*      DISP:ENAB OFF                                        */
/* - Select internal machine binary data format (milli-dBm) */
/*      FORM:DAT INT,32                                      */
/* - Select appropriate byte order (Intel)                   */
/*      FORM:BORD SWAP                                       */
/* - Repeat the following the requested number of times:     */
/*   - Trigger a measurement and wait for completion         */
/*      INIT:*OPC?                                           */
/*   - Read the resulting measurement trace                  */
/*      TRAC:DATA? TRACE1                                    */
/* - Compute running averaged power at all trace points      */
/* - Display measurement statistics                          */
/* - Write averaged data to second trace display             */
/*      TRAC:DATA TRACE2 <definite length block of data>     */
/* - Enable viewing of second trace                          */
/*      TRACE2:MODE VIEW                                     */
/* - Enable local display for viewing                        */
/*      DISP:ENAB ON                                         */
/* - Select continuous sweep mode                            */
/*      INIT:CONT ON                                         */
/* - Close session and Return instrument to local control    */
/***********************************************************/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <sys\timeb.h>
#include "visa.h"

#define NUM_TRACES 100      /* number of traces to average                  */
#define NUM_POINTS 401      /* requested number of points/trace             */
#define CENTER 50           /* center frequency in MHz, an integer          */
#define SPAN 20             /* span frequency in MHz, an integer            */
#define RBW 300             /* resolution BW in kHz, an integer             */

define DISPLAY  0           /* ESA display enable, disable for speed        */

#define DATA_LENGTH     4   /* number of data bytes in one trace point      */
#define MAX_POINTS 8192    /* maximum number of points/trace in ESA        */

int iNumTraces = NUM_TRACES,/* number of traces to average                  */
    iRbw = RBW,             /* resolution bandwidth                         */
    iNumPoints = NUM_POINTS,/* actual number of trace points per sweep      */
    iSpan = SPAN,           /* Analyzer Frequency Span in MHz               */
    iCenter = CENTER;       /* Analyzer Center frequency in MHz             */

unsigned long lRetCount;  /* number of bytes transferred in one trace record  */

double dDelta, dTimePer, dPower;

struct timeb start_time, stop_time, elapsed_time;

char cCommand[100];
char cBuffer[100];
double dPwrAvgArray[MAX_POINTS];

ViUInt32 iHeaderLength,   /* header is "#nyyy..." n is number of chars in yyy,*/
                          /* yyy is the total data length in bytes            */
         iArrayLength,    /* iArrayLength is number of bytes of data          */
         iTermLength = 1, /* the response message includes a LF character    */
         iBlockSize,   /* number of bytes expected in one trace definite block*/
         iTotalRetCount;  /* total number of bytes actually transferred       */

ViSession defaultRM, viESA;

                      /* reserve space for the header, data and terminator    */
```

```
ViChar cInBuffer[sizeof("#nyyyyl") + (MAX_POINTS * DATA_LENGTH) ];
ViChar cOutBuffer[sizeof("TRAC:DATA TRACE2,#nyyyyl") + (MAX_POINTS * DATA_LENGTH
) ];

/******************   Calculate length byte in block header   *****************/
int HeaderLength(int iArrayLength)      {
        int iHeaderLength;

        iHeaderLength = 3; /* iArrayLength >0 plus increment for "#" and "n"  */
        while ( (iArrayLength = (iArrayLength / 10)) > 0 )      {
                iHeaderLength++;
        }

        return(iHeaderLength);
}

/*******************      prepare ESA for measurement       *******************/
void setup() {

        int iModelNumber;
        char * cpModel;

        /* Identify the instrument and get the model number                   */
        viQueryf(viESA, "*IDN?\n", "%t", &cBuffer);

        iModelNumber = 0;
        if( !(strstr(cBuffer,"E44") == NULL))  {
                cpModel = strstr(cBuffer,"44");
                cpModel[4] = 0;
                iModelNumber = atoi(cpModel);
        }
        else    {
                printf("\nNo E44xx instrument found, program is exiting\n");
                exit(1);
        }

        viPrintf(viESA, ":SENS:FREQ:CENT %i MHz\n", iCenter);
        viPrintf(viESA, ":SENS:FREQ:SPAN %i MHZ\n", iSpan);
        viPrintf(viESA, ":SENS:BAND %i KHZ\n", iRbw);

        /* use the sampling detector for power-average calculations           */
        viPrintf(viESA, ":DET SAMP\n");

       /* Turn off analog output of option board to maximize measurement rate*/
        viPrintf(viESA, ":SYST:PORT:IFVS:ENAB OFF\n");
        /* Turn auto align off to maximize measurement rate                  */
        viPrintf(viESA, ":CAL:AUTO OFF\n");

        /* set requested number of points                                    */
        viPrintf(viESA, ":SWE:POINTS %i\n", NUM_POINTS);

        printf("This program will measure and calculate\n");
        printf ("the power average of %i %i-point
traces.\n",iNumTraces,iNumPoints);

        /* Turn on 50 MHz amplitude reference signal                         */
        viPrintf(viESA, ":CAL:SOUR:STAT ON\n");

        /* route 50 MHz amptd ref to the input port of the spectrum analyzer  */
        /* & set reference level appropriate to the amplitude reference level */
        switch (iModelNumber)  {
                case 4401: case 4411:   {
                    viPrintf(viESA, ":DISP:WIND:TRAC:Y:RLEV -25 DBM\n");
                    break;
                }

               case 4402: case 4403: case 4404: case 4405: case 4407: case 4408:  {
                    viPrintf(viESA, ":DISP:WIND:TRAC:Y:RLEV -20 DBM\n");
                    printf ("\nConnect the calibrator output to the RF input.\n");
                    printf (".......Press <Enter> to continue \n");
                    scanf( "%c", &cBuffer);
                    break;
                }

                default:        {
                    printf("\nNo E-series ESA  found. The program is exiting\n");
                    exit(1);
                }
        }

        /* Single sweep mode                                                  */
        viPrintf(viESA, ":INIT:CONT OFF\n");

        /* Turn off the local display to maximize measurement rate            */
        if(!DISPLAY)    {
            viPrintf(viESA, ":DISP:ENAB OFF\n");
        }

        /* transfer data in definite length,32 bit integer blocks. Select     */
        /* machine units (milli-dBm) to maximize measurement rate             */
        viPrintf(viESA, ":FORM:DATA INT,32\n" );
        /* select the byte order; low-byte first for Intel platforms          */
        /* To further increase measurement rate,:FORM:BORD NORM could         */
        /* be used instead. The byte ordering would then need to be           */
        /* done within this program.                                          */
        viPrintf(viESA, ":FORM:BORD SWAP\n");

        /* pre-calculate amount of data to be transferred per measurement     */
        iTermLength = 1;
        iArrayLength = iNumPoints * DATA_LENGTH;
        iHeaderLength = HeaderLength(iArrayLength);
        iBlockSize = iHeaderLength + iArrayLength + iTermLength;

}

/****************        Write binary trace data to ESA         ***************/
void write_binary_trace(char *cScpiCommand, int *ipTraceData)   {

    /*  trace data must point to an integer array of size NUM_POINTS          */
        memcpy(&cOutBuffer[strlen(cScpiCommand)], ipTraceData, iArrayLength);
        memcpy(&cOutBuffer, cScpiCommand, strlen(cScpiCommand));

    /* Add a <newline> to the end of the data, This isn't necessary           */
    /* if the GPIB card has been configured to assert EOI when the last       */
    /* character is sent, but it ensures a valid terminator is provided.      */
    cOutBuffer[iArrayLength + strlen(cScpiCommand)] = 0x0A;
    iBlockSize = (strlen(cScpiCommand) + iArrayLength + 1);
    viWrite(viESA,(ViBuf) cOutBuffer, iBlockSize, &lRetCount );
}

/*****   Measure and calculate power-average of multiple measurements   *******/
void average()  {
    int i=0, iLoop=0;
    int iArray[NUM_POINTS];

    long lOpc =0L;
    double dLogTen = log(10.0);

    setup();

    iTotalRetCount = lRetCount = 0;

    /* start the timer                                                         */
    ftime( &start_time );

    /* Now run through the event loop iNumTraces times                         */
    for(i=0; i<iNumTraces; i++)     {

/* trigger a new measurement and wait for complete                 */
        viQueryf(viESA, ":INIT;*OPC?\n", "%d", &lOpc);
        if (!lOpc)  {
            printf("Program Abort! error occurred: last command was not
completed!\n");
            exit(0);
        }

        /* Read the trace data into a buffer                                 */
        viPrintf(viESA, ":TRAC:DATA? TRACE1\n");
        viRead(viESA,(ViBuf) cInBuffer, (ViUInt32) iBlockSize, &lRetCount );
        iTotalRetCount += lRetCount;

        /* copy trace data to an array,                                      */
        /* byte order swapping  could be done here rather than in ESA        */
        memcpy(iArray, &cInBuffer[iHeaderLength], iArrayLength);

        /* calculate a running power-average                                 */
        for(iLoop = 0; iLoop < NUM_POINTS; iLoop++) {
            /* running average of power, in milliwatts                       */
            dPower = exp( dLogTen * (iArray[iLoop]/10000.0));
            if(i > 0)   {
                dPwrAvgArray[iLoop] += ((dPower - dPwrAvgArray[iLoop])/(i+1));
            }
            else    {
                dPwrAvgArray[iLoop] = dPower;
            }
        }
    }   /* end of event loop                                                 */

        /* stop the timer                                                    */
        ftime( &stop_time );

        /* Calculate elapsed time                                            */
        if (start_time.millitm > stop_time.millitm) {
            stop_time.millitm += 1000;
            stop_time.time--;
        }
        elapsed_time.millitm = stop_time.millitm - start_time.millitm;
        elapsed_time.time = stop_time.time - start_time.time;

        /* This is measurement time in milliseconds                          */
        dDelta = (1000.0 * elapsed_time.time) + (elapsed_time.millitm);
        /* show measurement statistics                                    */
        dTimePer=dDelta/((float)iNumTraces);
        printf("\tPower average of %i %i-point traces performed in %3.1f
seconds\n",
iNumTraces,iNumPoints,dDelta/1000);
        printf("\t%6.1f milliseconds per averaged measurement\n",dTimePer);
        printf("\t%6.1f averaged measurements per second\n",1000.0/dTimePer);
        printf("\t%i bytes transferred per trace, %i bytes total\n\n",lRetCount,
iTotalRetCount);
        return;
}

/*********************************        Main       ***************************/
void  main(void)  {
    int iLoop;
    int iAvgArray[NUM_POINTS];
    ViStatus viStatus;

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

/*Clear the Instrument                                                 */
    viClear(viESA);

    /* go to known instrument state with cleared status byte             */
    viPrintf(viESA, ":SYST:PRES:TYPE FACT;*RST\n");

    /* measure, transfer and calculate power average of multiple traces      */
    average();

    /* convert average power array back to integer array                 */
    for (iLoop = 0; iLoop < iNumPoints; iLoop++)    {
        dPower = 10.0 * log10( dPwrAvgArray[iLoop]);
        iAvgArray[iLoop] = (int) (1000.0 * dPower);
    }

    /* build 'TRAC:DATA TRACE2,#nyyy' header and write the result to Trace 2  */
    sprintf(cCommand,":TRAC:DATA TRACE2,#%i%i", HeaderLength(iArrayLength)-2,
iArrayLength);
    write_binary_trace(cCommand, iAvgArray);
    /* enable the trace, local display and return to continuous sweep         */
    viPrintf(viESA,":TRACE2:MODE VIEW;:DISP:ENAB ON;:INIT:CONT ON\n");

    /* Close session                                                          */
    viClose(viESA);
    viClose(defaultRM);

}  /***************************    End of Main   ******************************/
```

4 Programming Command Cross References

Functional Index to SCPI Subsection

The following table lists the SCPI subsections or subsystems associated with the instrument function category you wish to perform. The commands listed that begin with an asterisk (*) are IEEE common commands. These commands, and the SCPI commands in the subsection or subsystem, are documented in Chapter 5, “Language Reference” in this guide.

Function Category	SCPI Subsection or Subsystem
ALIGNMENT	***CAL?** ***TST?** **:CALibration** **:STATus:QUEStionable**
ATTENUATOR	see function category: Internal Attenuation and Source
BANDWIDTH	**:CALCulate** **:FETCh** **:INITiate** **:MEASure** **:READ** **[:SENSe]:BANDwidth** **[:SENSe]:EBWidth** **[:SENSe]:OBWidth**
CONFIGURATION and STATUS	***RCL <register>** ***SRE <integer>** ***STB?** **:SYSTem**
CONTROL	**ABORt**
CORRECTED MEASUREMENTS	**[:SENSe]:CORRection**
COUPLING	**:COUPle** **[:SENSe]:BANDwidth**
DELETE, LOAD, OR SAVE	***SAV <register>** **:MMEMory**
DEMODULATION	**[:SENSe]:DEMod**
DISPLAY	**:UNIT**
FREQUENCY	**[:SENSe]:FREQuency** **:STATus:QUEStionable**

Function Category	**SCPI Subsection or Subsystem**
FREQUENCY SPAN	**[:SENSe]:FREQuency** **[:SENSe]:OBWidth**
INPUT and OUTPUT	**:INPut** **:OUTPut** **[:SENSe]:ACPower** **[:SENSe]:AVERage** **[:SENSe]:BANDwidth** **[:SENSe]:CHPower** **[:SENSe]:CORRection** **[:SENSe]:DEMod** **[:SENSe]:DETector** **[:SENSe]:EBWidth** **[:SENSe]:POWer** **[:SENSe]:SWEep** **:STATus:QUEStionable** **:UNIT**
INTERNAL ATTENUATION and SOURCE	**:OUTPut** **[:SENSe]:POWer** **:SOURce**
LIMIT LINES	**:CALCulate:LLINe** **:MMEMory** **:TRACe**
MARKER	**:CALCulate:MARKer**
MEASURE	**:CONFigure** **:FETCh** **:INITiate** **:MEASure** **:READ** **[:SENSe]:ACPower** **[:SENSe]:AVERage** **[:SENSe]:CHPower** **[:SENSe]:EBWidth** **[:SENSe]:HARMonics** **[:SENSe]:OBWidth** **[:SENSe]:POWer** **[:SENSe]:SWEep**
PRESET	***RST** **:STATus** **:SYSTem**
PRINTING	**:HCOPy**
SOURCE	see function category: Internal Attenuation and Source

Function Category	**SCPI Subsection or Subsystem**
SPAN	**[:SENSe]:EBWidth** see also functional category: FREQUENCY SPAN
SPEAKER	**:SYSTem**
SWEEP	**[:SENSe]:HARMonics** **[:SENSe]:SWEep** **:SOURce**
SYNCHRONIZATION	***OPC?** ***WAI** **:SYSTem**
SYSTEM INFORMATION	***CLS** ***ESE <number>** ***IDN?** ***ESR?** ***LRN?** **:STATus** **:STATus:QUEStionable** **:SYSTem**
TRACE	**:DISPlay** **:FORMat** **[:SENSe]:EBWidth** **:TRACe**
TRACE MATH	**:CALCulate:NTData** **:DISPlay** **:TRACe**
TRIGGER	***TRG** **:ABORt** **:INITiate** **:TRIGger**

5 Language Reference

This chapter contains SCPI (Standard Commands for Programmable Instruments) programming commands for the Agilent ESA spectrum analyzers.

The first few pages of this chapter contain common commands specified in IEEE Standard 488.2-1992, *IEEE Standard Codes, Formats, Protocols and Common Commands for Use with ANSI/IEEE Std 488.1-1987*. New York, NY, 1992. Following these commands, the Agilent ESA spectrum analyzers SCPI commands are listed.

NOTE Refer to Chapter 2 , "Status Registers" and the status messages in Chapter 7, "Error Messages" which supplements the information presented in this chapter. In addition, refer to Chapter 6, "Front-Panel Key Reference," in the Agilent ESA Spectrum Analyzers User's Guide for additional information about the operation of each analyzer function. Use the analyzer **HELP** key to obtain similar information about analyzer key functions.

Refer to Chapter 6, "Agilent 8590/ESA Spectrum Analyzers Programming Conversion Guide" for specific backwards compatibility information between commands for HP/Agilent 8590-Series spectrum analyzers and Agilent ESA analyzers.

SCPI Sections and Subsections

SCPI commands related to major functional areas (such as calculate or sense) are grouped into blocks, or subsystems. Some of these subsystems are further divided into subsections (such as calculate/marker, or sense/harmonics). An instrument model is then created to represent the way in which instument functionality is viewed and categorized by SCPI. Refer to *IEEE SCPI-1997 Volume 2: Command Reference, Standard Commands for Programmable Instruments*, Version 1997.0, May, 1997 for a more complete description of the SCPI instrument model.

The SCPI subsystems in this chapter are listed in alphabetical order. Likewise, the SCPI commands are in alphabetical order within the subsystem in which they belong. Refer to the following table to locate SCPI command subsystems and subsections by page number.

SCPI Subsystem/Subsection	**Page**
IEEE Common Commands	page 5-5
:ABORt	page 5-10
:CALCulate	page 5-11
:CALCulate:LLINe	page 5-12
:CALCulate:MARKer	page 5-19
:CALCulate:NTData	page 5-31
:CALibration	page 5-32
:CONFigure	page 5-37
:COUPle	page 5-38
:DISPlay	page 5-39
:FETCh	page 5-46
:FORMat	page 5-51
:HCOPy	page 5-54
IEEE Common Commands	page 5-5
:INITiate	page 5-57
:INPut	page 5-60
:INSTrument	page 5-62

SCPI Subsystem/Subsection	Page
:MEASure	page 5-63
:MMEMory	page 5-68
:OUTPut	page 5-73
:READ	page 5-74
[:SENSe]:	page 5-79
[:SENSe]:ACPower	page 5-80
[:SENSe]:AVERage	page 5-81
[:SENSe]:BANDwidth	page 5-82
[:SENSe]:CHPower	page 5-84
[:SENSe]:CORRection	page 5-85
[:SENSe]:DEMod	page 5-88
[:SENSe]:DETector	page 5-90
[:SENSe]:EBWidth	page 5-90
[:SENSe]:FREQuency	page 5-92
[:SENSe]:HARMonics	page 5-95
[:SENSe]:MIXer	page 5-97
[:SENSe]:OBWidth	page 5-100
[:SENSe]:POWer	page 5-101
[:SENSe]:SIDentify	page 5-103
[:SENSe]:SWEep	page 5-105
:SOURce	page 5-109
:STATus	page 5-114
:STATus:QUEStionable	page 5-116
:SWEEp	page 5-125
:SYSTem	page 5-126
:TRACe	page 5-136
:TRIGger	page 5-143
:UNIT	page 5-151

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 SCPI equivalent for this command is the same as **`:CALibrate[:ALL]?`**

NOTE Before executing this command, connect a cable between front panel connector **AMPTD REF OUT** and the **INPUT** connector for all Agilent ESA spectrum analyzers except Agilent models E4401B and E4411B. Except for the models indicated that do not require the cable, the alignment will fail using commands **CAL:RF** or **CAL:TG** if the cable is not connected.

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 094

***IDN?**

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

Manufacturer

Model

Serial number

Firmware version

For example:

Hewlett-Packard, E4401B, US00000013, A.00.00

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:

#PQQQSYST: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>`**

The number 4 (P in the preceding query response format) means there are 4 numbers that make up the size of the data that follows. In this example, 2031 bytes will follow the number 4 (42031).

2031 and 2016 (QQQ and MMM in the preceding query response format) represent data 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.

*OPC and *OPC? are currently effective only when immediately preceded by either the **`:INITiate:IMMediate`** or a **`:CALibration`** command.

Query Instrument Options

This function is provided in the ESA SCPI language reference in the SYSTem subsystem under **`:SYSTem:OPTions?`**.

Recall

`*RCL <register>`

This command recalls the instrument state from the specified instrument memory register.

Range:	Registers are an integer, 0 to 127
Remarks:	See also commands **`:MMEMory:LOAD:STATe`** and **`:MMEMory:STORe:STATe`**
	If the state being loaded has a newer firmware revision than the revision of the instrument, no state is recalled and an error is reported.
	If the state being loaded has an equal firmware revision than the revision of the instrument, the state will be loaded.
	If the state being loaded has an older firmware revision than the revision of the instrument, the instrument will only load the parts of the state that apply to the older revision.

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`**. This command always performs a factory preset.

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

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 127
Remarks:	See also commands **`:MMEMory:LOAD:STATe`** and **`:MMEMory:STORe:STATe`**
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 255

Read Status Byte Query

`*STB?`

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

Remarks:	See **`*CLS`**

Trigger

*TRG

This command triggers the instrument. Use the **`:TRIGger:SEQuence:SOURce`** command to select the trigger source.

Remarks: See also the **`:INITiate:IMMediate`** command

Self Test Query

*TST?

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

For Agilent ESA analyzers, ***TST?** always returns 0; no tests are performed.

Front Panel Access: **System, Alignments, Align All Now**

Wait-to-Continue

*WAI

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

ABORt Subsystem

Abort

`:ABORt`

Restarts 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:	**Restart** for continuous measurement mode

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.

NdBpoints

`:CALCulate:BWIDth|BANDwidth:NDB <rel_ampl>`

`:CALCulate:BWIDth|BANDwidth:NDB?`

Selects the power level, below the peak of the signal, at which the signal bandwidth will be measured by the markers. **`:CALCulate:BWIDth|BANDwidth[:STATe]`** must be ON.

NOTE To query the result of NdBpoints, use the command **`:CALCulate:BWIDth|BANDwidth:RESult?`**

Factory Preset and *RST:	–3 dB
Range:	–80 dB to –1 dB
Default Unit:	dB
Remarks:	Refer to **`:CALCulate:BWIDth|BANDwidth[:STATe]`** for an explanation of this marker function.
Front Panel Access:	**Peak Search (or Search), N dB Points**

NdBresults

`:CALCulate:BWIDth|BANDwidth:RESult?`

Returns the measured bandwidth at the power level defined by **`:CALCulate:BWIDth:NDB`**. –100 is returned if **`:CALCulate:BWIDth|BANDwidth[:STATe]`** is off, or when a result is not available.

Range:	Real value less than the current frequency span
Default Unit:	Hz
Remarks:	Refer to **`CALCulate:BWIDth|BANDwidth[:STATe]`** for an explanation of this marker function.
Front Panel Access:	**Peak Search (or Search), N dB Points**

NdBstate

:CALCulate:BWIDth|BANDwidth[:STATe] OFF|ON|0|1

:CALCulate:BWIDth|BANDwidth[:STATe]?

Controls the bandwidth measurement function. The function measures the bandwidth, at the number of dB down specified in **:CALCulate:BWIDth:NDB**, of the maximum signal on the display.

Factory Preset and *RST:	Off
Remarks:	When this command is turned on, the bandwidth measurement function (N dB Points) is associated with the active marker, and a peak search is performed. If no marker is active at the time this command is turned on, marker 1 becomes the active marker. No restrictions exist for moving the bandwidth measurement function markers to any other signal on the display. However, when this function is turned on, all other concurrent marker functions are suspended.
Front Panel Access:	**Peak Search (or Search), N dB Points On Off**

Test Current Trace Data Against all Limit Lines

:CALCulate:CLIMits:FAIL?

Queries the status of the limit line testing. Returns a 0 if the trace data passes when compared with all the current limit lines. Returns a 1 if the trace data fails any limit line test.

CALCulate:LLINe Subsection

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

NOTE Refer also to **:MMEMory** and **:TRACe** subsystems for more trace and limit line commands.

Control Limit Line Amplitude Interpolation

:CALCulate:LLINe[1]|2:AMPLitude:INTerpolate:TYPE LOGarithmic|LINear

:CALCulate:LLINe[1]|2:AMPLitude:INTerpolate:TYPE?

Selects the type of interpolation done for the amplitude values of the designated limit line when comparing to measured data.

Factory Preset and *RST:	Not affected by preset
Remarks:	Once this function is defined, the selected type is persistent. Persistent means that it retains the setting previously selected, even through a power cycle.
Front Panel Access:	**Display, Limits, Modify, Amptd Interp Log Lin**

Set Fixed or Relative Limit Lines

`:CALCulate:LLINe:CMODe FIXed|RELative`

`:CALCulate:LLINe:CMODe?`

Specifies whether the current limit lines are fixed or relative.

NOTE If you need to change the domain with **`:CALCulate:LLINe:CONTrol:DOMain`**, do it before this command. Changing the domain deletes all the existing limit line values.

Factory Preset and *RST:	Not affected by preset
Remarks:	For Amplitude Parameters:

Regardless of whether the limit line is based on frequency or sweep time, amplitude parameters in a limit line table represent absolute values or relative values. In fixed, the limit line amplitude values are specified in absolute amplitude and do not depend on the reference level. In relative, the limit line amplitude values are relative to the current reference level.

For Fixed Frequency Parameters:

The frequency values in a limit line table are fixed values, and the limit line is positioned accordingly. Fixed limit lines are specified in absolute frequency and do not depend upon the center frequency values.

For Relative Frequency Parameters:

The frequency values in a limit line table are relative values and positions the limit line relative to the center frequency settings. Relative limit lines are specified in relative frequency and are positioned with respect to the current center frequency. When the current center frequency value is changed, the segment frequencies are converted according to the current center frequency value.

For Time Parameters:

Limit lines that are based on sweep time are always relative to the start time. The horizontal position of the limit line is not affected by this command.

Front Panel Access: **Display, Limits, Properties, Limits Fixed Rel**

Set Limit Line X-axis Units

:CALCulate:LLINe:CONTrol:DOMain FREQuency|TIME

:CALCulate:LLINe:CONTrol:DOMain?

Selects how the limit line segments are defined: according to frequency, or according to the sweep time setting of the spectrum analyzer.

NOTE Changing this setting deletes *all* existing limit data from the analyzer. In other words, if a limit line has already been defined, changing the type clears the existing limit line.

Factory Preset and *RST: not affected by Preset

Remarks: For TIME, the limit line segments are placed on the spectrum analyzer display with respect to the sweep time setting of the analyzer, with 0 at the left edge of the display.

For FREQuency, segments are placed according to the frequency that is specified for each segment.

Front Panel Access: **Display, Limits, Properties, X Axis Units Freq Time**

Control Limit Line Frequency Interpolation

:CALCulate:LLINe[1]|2:CONTrol:INTerpolate:TYPE LOGarithmic|LINear

:CALCulate:LLINe[1]|2:CONTrol:INTerpolate:TYPE?

Selects the type of interpolation done for the frequency values of the designated limit line when comparing to measured data. This only applies in the frequency domain. This function does not work in zero span (when the analyzer is in the time domain).

Remarks: Once this function is defined, the selected type is persistent. Persistent means that it retains the setting previously selected, even through a power cycle.

Front Panel Access: **Display, Limits, Modify, Freq Interp Log Lin**

Define Limit Line Values

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

:CALCulate:LLINe[1]|2:DATA?

Defines limit line values, and destroys all existing data. Up to 200 points may be defined for each limit. No units are allowed.

- <x-axis> – can be frequency or time values as specified by **:CALCulate:LLINe:CONTrol:DOMain**. Frequencies are always in Hz. Time is always in seconds. No unit is allowed in this parameter.
- <ampl> – amplitude values are always in units of dBm. Up to two amplitude values can be provided for each x-axis value, by repeating <x-axis> in the data list. No unit is allowed in this parameter.
- <connected> – connected values are either 0 or 1. A 1 means this point should be connected to the previously defined point to define the limit line. A 0 means that it is a point of discontinuity and is not connected to the preceding point. The “connected” value is ignored for the first point.

Example: **CALC:LLIN1:DATA 1000000000,-20,0,200000000,-30,1**

Range: <**x-axis**> –30 Gs to +30 Gs for time limits

<**x-axis**> –30 GHz to +350 GHz for frequency limits

<ampl> –120 dBm to +100 dBm

<connected> 0 or 1

Remarks: If two amplitude values are entered for the same frequency, a single vertical line is the result. In this case, if an upper line is chosen, the amplitude of lesser frequency (amplitude 1) is tested. If a lower line is chosen, the amplitude of greater frequency (amplitude 2) is tested.

For linear amplitude interpolation and linear frequency interpolation, the interpolation is computed as:

$$y = \frac{y_{i+1} - y_i}{f_{i+1} - f_i}(f - f_i) + y_i$$

For linear amplitude interpolation and log frequency interpolation, the interpolation is computed as:

$$y = \frac{y_{i+1} - y_i}{\log f_{i+1} - \log f_i}(\log f - \log f_i) + y_i$$

For log amplitude interpolation and linear frequency interpolation, the interpolation is computed as:

$$\log y = \frac{\log y_{i+1} - \log y_i}{f_{i+1} - f_i}(f - f_i) + \log y_i$$

For log amplitude interpolation and log frequency interpolation, the interpolation is computed as:

$$\log y = \frac{\log y_{i+1} - \log y_i}{\log f_{i+1} - \log f_i}(\log f - \log f_i) + \log y_i$$

Front Panel
Access: **Display, Limits, Properties, X Axis Units Freq Time**

Display, Limits, Modify, Edit

Display, Limits, Modify, Edit, Point

Display, Limits, Modify, Edit, Frequency

Display, Limits, Modify, Edit, Amplitude

Display, Limits, Modify, Edit, Connected

Display, Limits, Modify, Edit, Delete Point

Merge Additional Values into the Existing Limit Line

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

Adds the points with the specified values to the current limit line, allowing you to merge limit line data. Up to two amplitude values are allowed for each x value. If too much data is merged, as many points as possible are merged into the existing limit and then an error is reported. Up to 200 points total may be defined for each limit.

- <x-axis> can be frequency or time values as specified by **`:CALCulate:LLINe:CONTrol:DOMain`**. Frequencies are always in Hz. Time is always in seconds. No unit is allowed in this parameter.
- <ampl> amplitude values are always in units of dBm. No unit is allowed in this parameter.
- <connected> connected values are either 0 or 1. A 1 means this point should be connected to the previously defined point to define the limit line. A 0 means that it is a point of discontinuity and is not connected to the preceding point. The “connected” value is ignored for the first point.

Range: <**`x-axis`**> –30 Gs to +30 Gs for time limits

<**`x-axis`**> –30 GHz to +350 GHz for frequency limits

	`<ampl>` –120 dBm to +100 dBm
	`<connected>` 0 or 1
Front Panel Access:	**Display, Limits, Properties, X Axis Units Freq Time**

Delete Limit Line

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

Deletes the selected limit line.

Display the Limit Line

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

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

Controls the display of the current limit line.

Factory Preset and *RST:	Off
Front Panel Access:	**Display, Limits, Modify, Limit On Off**

Test the Data Against the Limit Line

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

Queries the status of the limit line testing. Returns a 0 if the data passes, and returns a 1 if there is a failure. This query only makes sense if margin or limit test is On. Use the command **`:CALCulate:LLINe[1]|2:STATe OFF|ON|0|1`** to activate limit line testing.

Set the Margin Size

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

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

Allows you to define the amount of measurement margin that is added to the designated limit line.

Factory Preset and *RST:	not affected
Default Units:	dB
Remarks:	The margin must be negative for upper limit lines, and positive for lower limits.
Front Panel Access:	**Display, Limits, Modify, Margin On Off**

Display the Limit Margin

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

:CALCulate:LLINe[1]|2:MARGin:STATe?

Allows you to display a measurement margin that is added to the designated limit line to do secondary testing of the data.

Factory Preset and *RST:	Off
Front Panel Access:	**Display, Limits, Modify, Margin On Off**

Control Limit Line Testing

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

:CALCulate:LLINe[1]|2:STATe?

Turns limit line testing on/off. The limit and margin will only be tested if they are displayed. Use **:CALCulate:LLINe[1]|2:DISPlay** to turn on the display of limit lines, and **:CALCulate:LLINe[1]|2:MARGin:STATe** to turn on the display of margins. If margin and limit display are both turned off, limit test is automatically turned off. Use **:CALCulate:LLINe[1]|2:FAIL?** to return the state of pass or fail after limit line state has been turned on.

Factory Preset and *RST:	Off
Front Panel Access:	**Display, Limits, Modify, Limit On Off**

Select the Type of Limit Line

:CALCulate:LLINe[1]|2:TYPE UPPer|LOWer

:CALCulate:LLINe[1]|2:TYPE?

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

Factory Preset and *RST:	Upper; not affected by preset
Remarks:	If a margin has already been set for this limit line, and this command is used to change the limit type, then the margin value is reset to 0 dB.
Front Panel Access:	**Display, Limits, Modify, Type Upper Lower**

CALCulate:MARKer Subsection

Markers All Off on All Traces

:CALCulate:MARKer:AOFF

Turns off all markers on all the traces.

Front Panel Access:	**Marker, Marker All Off**

Continuous Peaking Marker Function

:CALCulate:MARKer[1]|2|3|4:CPEak[:STATe] OFF|ON|0|1

:CALCulate:MARKer[1]|2|3|4:CPEak[:STATe]?

Turns on or off continuous peaking. It continuously puts the selected marker on the highest displayed signal peak.

Factory Preset and *RST:	Off
Front Panel Access:	**Peak Search (or Search), N dB Points Continuous Pk On Off**

Frequency Counter Marker Resolution

:CALCulate:MARKer:FCOunt:RESolution <real>

:CALCulate:MARKer:FCOunt:RESolution?

Sets the resolution of the marker frequency counter. AUTO ON couples the marker counter resolution to the frequency span.

Factory Preset and *RST:	1 kHz
Range:	1 Hz to 100 kHz
Default Unit:	Hz
Front Panel Access:	**Freq Count, Resolution Auto Man**

Frequency Counter Marker Resolution Automatic

:CALCulate:MARKer:FCOunt:RESolution:AUTO OFF|ON|0|1

:CALCulate:MARKer:FCOunt:RESolution:AUTO?

Sets the resolution of the marker frequency counter so it is automatically coupled to the frequency span, generating the fastest accurate count.

Factory Preset and *RST:	On

Front Panel
Access: **Freq Count, Resolution Auto Man**

Frequency Counter Marker

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

`:CALCulate:MARKer[1]|2|3|4:FCOunt[:STATe]?`

`:CALCulate:MARKer[1]:FCOunt:X?`

Turns on or off the marker frequency counter. To query the frequency counter, use `:CALCulate:MARKer[1]:FCOunt:X?`

Factory Preset and *RST:	Off
Remarks:	If query with frequency count off, 9e15 is returned.
Front Panel Access:	**Freq Count, Marker Count On Off**

Marker Function

`:CALCulate:MARKer[1]|2|3|4:FUNCtion BPOWer|NOISe|OFF`

`:CALCulate:MARKer[1]|2|3|4:FUNCtion?`

Selects the marker function for the specified marker. To query the value returned by the function, use `:CALCulate:MARKer[1]|2|3|4:Y?`

BPOWer is the power integrated within the bandwidth

NOISe is a noise measurement

OFF turns off all functions

Remarks:	When a measurement under the front panel **MEASURE** key is started, this command is turned off. If this command is turned on when any of the **MEASURE** key measurements are in progress, that measurement will be stopped.
Front Panel Access:	**Marker, Function**

Marker Peak (Maximum) Search

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

Performs a peak search based on the search mode settings of `:CALCulate:MARKer:PEAK:SEARch:MODE`.

NOTE See command `:CALCulate:MARKer:PEAK:SEARch:MODE`

Front Panel
Access: **Peak Search (or Search), Meas Tools, Peak Search**

Marker Peak (Maximum) Left Search

:CALCulate:MARKer[1]|2|3|4:MAXimum:LEFT

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

Remarks:	The marker will be placed at the next highest peak that rises and falls by at least the peak excursion above the peak threshold. If no peak meets the excursion and threshold criteria, a No Peak Found error (202) is given.
Front Panel Access:	**Peak Search (or Search), Next Pk Left**

Marker Next Peak (Maximum) Search

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

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

Remarks:	The marker will be placed at the highest peak that rises and falls by at least the peak excursion above the peak threshold. If no peak meets the excursion and threshold criteria, a No Peak Found error (202) is given.
Front Panel Access:	**Peak Search (or Search), Next Peak**

Marker Peak (Maximum) Right Search

:CALCulate:MARKer[1]|2|3|4:MAXimum:RIGHt

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

Remarks:	The marker will be placed at the highest peak that rises and falls by at least the peak excursion above the peak threshold. If no peak meets the excursion and threshold criteria, a No Peak Found error (202) is given.
Front Panel Access:	**Peak Search (or Search), Next Pk Right**

Marker Peak (Minimum) Search

:CALCulate: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.

Front Panel Access:	**Peak Search (or Search), Min Search**

Marker Mode

:CALCulate:MARKer[1]|2|3|4:MODE POSition|DELTa|BAND|SPAN

:CALCulate:MARKer[1]|2|3|4:MODE?

Selects the type of markers that you want to activate. Refer to the "*Agilent ESA Analyzers User's Guide*" for a more complete explanation of this function.

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

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

Band activates a pair of band markers, where each marker can be independently positioned on the trace. The marker readout shows the difference between the two markers.

Span activates a pair of span markers, where the marker positioning is controlled by changing the span and/or center frequency between the two markers. The marker readout shows the difference between the two markers.

Front Panel Access:	**Marker, Normal**
	Marker, Delta
	Marker, Band Pair Start Stop
	Marker, Span Pair Span Center

Define Peak Excursion

:CALCulate:MARKer:PEAK:EXCursion <rel_ampl>

:CALCulate:MARKer:PEAK:EXCursion?

Specifies the minimum signal excursion above the threshold for the internal peak identification routine to recognize a signal as a peak. This applies to all traces and all windows. (The excursion is the delta power from the noise level to the signal peak.)

NOTE See command **:CALCulate:MARKer:PEAK:THReshold**

Factory Preset and *RST:	6 dB
Range:	0 to 100 dB
Default Unit:	dB
Front Panel	

Access:	**Peak Search (or Search), Search Param, Peak Excursn**

Define Peak Search

:CALCulate:MARKer:PEAK:SEARch:MODE PARameter|MAXimum

:CALCulate:MARKer:PEAK:SEARch:MODE?

Sets the peak search mode.

NOTE See command **:CALCulate:MARKer[1]|2|3|4:MAXimum**

Factory Preset and *RST:	MAXimum
Remarks:	If mode is set to MAXimum, peak search will place the marker at the maximum amplitude in the trace. If mode is set to PARameter, peak search will place the marker at the highest peak that rises and falls by at least the peak excursion above the peak threshold. If no peak meets the excursion and threshold criteria, a No Peak Found error (error 202) is issued. Next peak, next peak right, next peak left, and peak table are not affected by this command. They will always use peak excursion and peak threshold for search criteria.
Front Panel Access:	**Peak Search (or Search), Search Param, Peak Search Max**

Define Peak Threshold

:CALCulate:MARKer:PEAK:THReshold <ampl>

:CALCulate:MARKer:PEAK:THReshold?

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

NOTE See command **:CALCulate:MARKer:PEAK:EXCursion**

Range:	Reference level to the bottom of the display
Default Unit:	amplitude units
Front Panel Access:	**Peak Search (or Search), Search Param, Peak Threshold**

Peak to Peak Delta Markers

:CALCulate:MARKer[1]|2|3|4:PTPeak

Positions delta markers on the highest and lowest points on the trace.

Factory Preset and *RST:	Off
Front Panel Access:	**Peak Search (or Search), Pk-Pk Search**

Set Center Frequency to the Marker Value

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

Sets the center frequency equal to the specified marker frequency, which moves the marker to the center of the screen. In delta marker mode, the center frequency is set to the marker delta value. This command is not available in zero span.

Front Panel Access:	**Marker –>, Mkr –> CF**

Set Reference Level to the Marker Value

`:CALCulate:MARKer[1]|2|3|4[:SET]:RLEVel`

Sets the reference level to the specified marker amplitude. In delta marker mode, the reference level is set to the amplitude difference between the markers.

Front Panel Access:	**Marker –>, Mkr –> Ref Lvl**
	Peak Search (or Search), Meas Tools, Mkr –> Ref Lvl

Set Span to the Marker Value

`:CALCulate:MARKer[1]|2|3|4[:SET]:SPAN`

Sets the span to the value of the specified marker frequency. The specified marker must be in delta mode. Select the delta marker mode with CALCulate:MARKer[1]|2|3|4:MODE DELTa. This command is not available in zero span.

Front Panel Access:	**Marker, Delta, Marker –>, Mkr Δ –> Span**

Set Start Frequency to the Marker Value

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

Sets the start frequency to the value of the specified marker frequency. In delta marker mode, the start frequency is set to the marker delta value. This command is not available in zero span.

Front Panel Access:	**Marker –>, Mkr –> Start**

Set Center Frequency Step Size to the Marker Value

:CALCulate:MARKer[1]|2|3|4[:SET]:STEP

Sets the center frequency step size to match the marker frequency. In delta marker mode, the center frequency step size will be set to the frequency difference between the markers. Select the delta marker mode with **:CALCulate:MARKer[1]|2|3|4:MODE DELTa**. This command is not available if the delta marker is off, or in zero span.

Front Panel Access:	**Marker –>, Mkr –> CF Step**
	Peak Search (or Search), Meas Tools, Mkr –> CF

Set Stop Frequency to the Marker Value

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

Sets the stop frequency to the value of the active marker frequency. In delta marker mode, the stop frequency is set to the marker delta value. This command is not available in zero span.

Front Panel Access:	**Marker –>, Mkr –> Stop**

Marker On/Off

:CALCulate:MARKer[1]|2|3|4:STATe OFF|ON|0|1

:CALCulate:MARKer[1]|2|3|4:STATe?

Turns the selected marker on or off.

Front Panel Access:	**Marker, Off**

Marker Table On/Off

:CALCulate:MARKer:TABLe:STATe OFF|ON|0|1

:CALCulate:MARKer:TABLe:STATe?

Turns the marker table on or off

Front Panel Access:	**Marker, Marker Table On Off**

Marker to Trace

:CALCulate:MARKer[1]|2|3|4:TRACe <integer>

:CALCulate:MARKer[1]|2|3|4:TRACe?

Assigns the specified marker to the designated trace 1, 2, or 3.

Factory Preset and *RST:	1

Range:	1 to 3
Front Panel Access:	**Marker, Marker Trace Auto 1 2 3**

Marker to Trace Auto

:CALCulate:MARKer[1]|2|3|4:TRACe:AUTO OFF|ON|0|1

:CALCulate:MARKer[1]|2|3|4:TRACe:AUTO?

Turns on or off the automatic marker to trace function.

Factory Preset and *RST:	**AUTO ON**
Front Panel Access:	**Marker, Marker Trace Auto 1 2 3**

Continuous Signal Tracking Function

:CALCulate:MARKer[1]|2|3|4:TRCKing[:STATe] OFF|ON|0|1

:CALCulate:MARKer[1]|2|3|4:TRCKing[:STATe]?

Turns on or off marker signal tracking. It continuously puts the selected marker on the highest displayed signal peak and moves it to the center frequency. This allows you to keep a signal that is drifting in frequency, on the display.

Factory Preset and *RST:	Off
Remarks:	When a measurement under the front panel **MEASURE** key is started, this command is turned off. If this command is turned on when any of the **MEASURE** key measurements are in progress, that measurement will be stopped.
Front Panel Access:	**FREQUENCY/Channel, Signal Track On Off**

Marker X Value

:CALCulate:MARKer[1]|2|3|4:X <param>

:CALCulate:MARKer[1]|2|3|4:X?

Position the designated marker on its assigned trace at the specified trace X value. The value is in the X-axis units (which is often frequency or time).

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

Default Unit:	Matches the units of the trace on which the marker is positioned

Front Panel
Access: **Marker**

Span Markers Center Frequency X Value

`:CALCulate:MARKer[1]|2|3|4:X:CENTer <param>`

`:CALCulate:MARKer[1]|2|3|4:X:CENTer?`

Position the center frequency, of the designated span-type marker pair, at the specified trace X value. The value is in the X-axis units (which is often frequency or time) Use **`:CALCulate:MARKer:MODE SPAN`** to select span markers.

The query returns the current X-value center frequency of the designated markers.

Range: Matches the units of the trace on which the markers are positioned

Front Panel
Access: **Marker, <active marker>, Center**

Marker X Position

`:CALCulate:MARKer[1]|2|3|4:X:POSition <integer>`

`:CALCulate: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 401 points (X positions.)

The query returns the current X position for the designated marker.

Range: 0 to 400

Front Panel
Access: **Marker**

Span Markers Center Frequency X Position

`:CALCulate:MARKer[1]|2|3|4:X:POSition:CENTer <param>`

`:CALCulate:MARKer[1]|2|3|4:X:POSition:CENTer?`

Position the center frequency, of the designated span-type marker pair, at the specified trace X position. A trace is composed of 401 points (X positions.) Use **`:CALCulate:MARKer:MODE SPAN`** to select span markers.

The query returns the current X-position center frequency of the designated markers.

Range: 0 to 400

Front Panel
Access: **Marker, <active marker>, Center**

Span Markers Span X Position

:CALCulate:MARKer[1]|2|3|4:X:POSition:SPAN <param>

:CALCulate:MARKer[1]|2|3|4:X:POSition:SPAN?

Change the frequency span, of the designated span-type marker pair, to position the markers at the desired trace X positions. A trace is composed of 401 points (X positions.) Use CALCulate:MARKer:MODE SPAN to select span markers.

The query returns the current X-position frequency span of the designated markers.

Range:	0 to 400
Default Unit:	X-axis units (Hz or seconds)
Front Panel Access:	**Marker, <active marker>, Span**

Band Markers Start Frequency X Position

:CALCulate:MARKer[1]|2|3|4:X:POSition:STARt <param>

:CALCulate:MARKer[1]|2|3|4:X:POSition:STARt?

Position the left-most marker, the start (reference) frequency of the designated band-type marker pair, at the specified trace X position. A trace is composed of 401 points (X positions.) Use **:CALCulate:MARKer:MODE BAND** to select band markers.

The query returns the current X-position start/reference frequency of the designated marker.

Range:	0 to 400
Default Unit:	X-axis units (Hz or seconds)
Front Panel Access:	**Marker, <active marker>, Start**

Band Markers Stop Frequency X Position

:CALCulate:MARKer[1]|2|3|4:X:POSition:STOP <param>

:CALCulate:MARKer[1]|2|3|4:X:POSition:STOP?

Position the right-most marker, the stop frequency of the designated band-type marker pair, at the specified trace X position. A trace is composed of 401 points (X positions.) Use **:CALCulate:MARKer:MODE BAND** to select band markers.

The query returns the current X-position stop frequency of the designated marker.

Range:	0 to 400

Default Unit:	X-axis units (Hz or seconds)
Front Panel Access:	**Marker, <active marker>, Stop**

Marker X-Axis Readout

:CALCulate:MARKer[1]|2|3|4:X:READout FREQuency|TIME|ITIMe|PERiod

:CALCulate:MARKer[1]|2|3|4:X:READout?

Selects the units for the x-axis readout of the marker. Available units are:

Frequency
Time
Inverse of time
Period

Factory Preset and *RST:	Frequency
Front Panel Access:	**Marker, Readout, Frequency**
	Marker, Readout, Time
	Marker, Readout, Inverse Time
	Marker, Readout, Period

Span-Markers Span X Value

:CALCulate:MARKer[1]|2|3|4:X:SPAN <param>

:CALCulate:MARKer[1]|2|3|4:X:SPAN?

Change the frequency span of the designated span-type marker pair to position the markers at the desired trace X values. The value is in the X-axis units (which is usually frequency or time). Use **:CALCulate:MARKer:MODE SPAN** to select span markers.

The query returns the current X-value frequency span of the designated markers.

Default Unit:	Matches the units of the trace on which the markers are positioned.
Front Panel Access:	**Marker, <active marker>, Span**

Band-Markers Start Frequency X Value

:CALCulate:MARKer[1]|2|3|4:X:STARt <param>

:CALCulate:MARKer[1]|2|3|4:X:STARt?

Position the start (reference) frequency of the designated band-type marker pair, at the specified trace X value. The value is in the X-axis units (which is often frequency or time). Use **:CALCulate:MARKer:MODE BAND** to select band markers.

The query returns the current X-value start/reference frequency of the designated marker.

Default Unit:	Matches the units of the trace on which the markers are positioned
Front Panel Access:	**Marker, <active marker>, Start**

Band-Markers Stop Frequency X Value

:CALCulate:MARKer[1]|2|3|4:X:STOP <param>

:CALCulate:MARKer[1]|2|3|4:X:STOP?

Position the stop frequency of the designated band-type marker pair, at the specified trace X value. The value is in the X-axis units (which is often frequency or time). Use **:CALCulate:MARKer:MODE BAND** to select band markers.

The query returns the current X-value stop frequency of the designated marker.

Default Unit:	Matches the units of the trace on which the markers are positioned
Front Panel Access:	**Marker, <active marker>, Stop**

Marker Read Y Value

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

Read the current Y value for the designated marker or delta on its assigned trace. The value is in the Y-axis units for the current trace (which is often dBm).

Default Unit:	Matches the units of the trace on which the marker is positioned
Remarks:	This command can be used to read the results of marker functions such as band power and noise that are displayed in the marker value field on the analyzer.

CALCulate:NTData Subsection

Normalize the Trace Data

`:CALCulate:NTData[:STATe] OFF|ON|0|1`

`:CALCulate:NTData[:STATe]?`

One sweep of trace data is copied to trace NRML, which is used as the reference trace. Then for all subsequent trace sweeps, display trace 1 = data collected into trace 1 – data in trace NRML.

Front Panel Access:	**View/Trace, Normalize, Normalize On Off**

CALibration Subsystem

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

Align All Instrument Assemblies

:CALibration[:ALL]

:CALibration[:ALL]?

Performs an alignment of all the assemblies within the instrument, except for the tracking generator (Option 1DN or 1DQ), if installed.

Before executing this command, connect a cable between front panel connector **AMPTD REF OUT** and the **INPUT** connector for all Agilent ESA spectrum analyzers except Agilent models E4401B and E4411B. If the cable is not connected, **CAL:ALL** will perform a subset of the RF alignment and a subsequent **CAL:RF** will be required for the analyzer to meet its specified performance.

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, even if only a subset of the RF alignment is performed..

Front Panel Access:	**System, Alignments, Align Now, All**

Set Auto Align Mode All or Not RF

:CALibration:AUTO:MODE ALL|NRF

:CALibration:AUTO:MODE?

This command determines whether or not to include RF alignment as part of the automatic alignment routines. Eliminating automatic alignment of the RF prevents changes in the input impedance between sweeps, which could cause input device instability.

Factory Preset and *RST:	All at power-up
Front Panel Access:	**System, Alignments, Auto Align, All**
	System, Alignments, Auto Align, All but RF

Automatic Alignment

:CALibration:AUTO OFF|ON|0|1

:CALibration:AUTO?

Turns the automatic alignment on and off. This is run continuously, at the end of each sweep.

Factory Preset and *RST:	On at power-up
Front Panel Access:	**System, Alignments, Auto Align, All**
	System, Alignments, Auto Align, All but RF
	System, Alignments, Auto Align, Off

Return to the Default Alignment Data

:CALibration:DATA:DEFault

Initializes the alignment data to the factory defaults.

Front Panel Access:	**System, Alignments, Load Defaults**

Align FM Demodulation

:CALibration:FMDemod

:CALibration:FMDemod?

Performs an alignment of the FM Demodulation board if Option BAA (FM Demodulation) is installed. The query form of this command performs the alignment and returns zero if the alignment is successful.

NOTE Both this command and front panel access are available only when Option BAA (FM Demodulation) is installed.

Front Panel Access:	**System, Alignments, Align Now, FM Demod**

Query the Internal or External Frequency Reference

:CALibration:FREQuency:REFerence?

This is a query only.

Front Panel Access:	**none**

Coarse Adjust the Frequency Reference

:CALibration:FREQuency:REFerence:COARse <setting>

:CALibration:FREQuency:REFerence:COARse?

Allows coarse adjustment of the internal 10 MHz reference oscillator timebase of the analyzer.

NOTE **:CALibration:ALL** is required after **COARse** is set.

Range:	Integer, 0 to 255
Front Panel Access:	**System, Alignments, Time Base, Coarse**

Fine Adjust the Frequency Reference

:CALibration:FREQuency:REFerence:FINE <setting>

:CALibration:FREQuency:REFerence:FINE?

Allows fine adjustment of the analyzer internal 10 MHz reference oscillator timebase.

NOTE **:CALibration:ALL** is required after **FINE** is set.

Range:	Integer, 0 to 255
Front Panel Access:	**System, Alignments, Time Base, Fine**

Select the Frequency Corrections

:CALibration:FREQuency[:STATe] OFF|ON|0|1

:CALibration:FREQuency[:STATe]?

Turns on or off the frequency corrections.

Factory Preset and *RST:	On
Front Panel Access:	**System, Alignments, Freq Correct On Off**

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.

NOTE Before executing this command, connect a cable between front panel connector **AMPTD REF OUT** and the **INPUT** connector for all Agilent ESA spectrum analyzers except Agilent models E4401B and E4411B. Except for the models indicated that do not require the cable, the alignment will fail using command **CAL:RF** if the cable is not connected.

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

Select the Source State for Calibration

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

:CALibration:SOURce:STATe?

Controls the state of the 50 MHz alignment signal.

NOTE The alignment signal is internally switched to the **INPUT** for Agilent models E4401B and E4411B. For all other models, connect a cable between front panel connector **AMPTD REF OUT** and the **INPUT** connector before performing a calibration.

Factory Preset and *RST:	Off
Front Panel Access:	For Agilent ESA models E4401B and E4411B: **Input/Output (or Input), Amptd Ref (f=50 MHz) On Off** For all other Agilent ESA models: **Input/Output (or Input), Amptd Ref Out (f=50 MHz) On Off**

Calibrate the Tracking Generator

:CALibration:TG

:CALibration:TG?

Performs an alignment of the tracking generator assembly.

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

NOTE This command is only applicable on Agilent ESA models E4402B, E4403B, E4404B, E4405B, E4407B, and E4408B. Before executing this command, connect a cable between front panel connector **RF OUT** and the **INPUT** connector. The alignment will fail using command **CAL:TG** if the cable is not connected.

Front Panel
Access: **System, Alignments, Align Now, TG**

CONFigure Subsystem

CONFigure subsystem commands apply only to measurements found in the **MEASURE** menu. These commands stop the current measurement and set up the instrument for the specified measurement using the factory default instrument settings.

`:CONFigure:<measurement>` will always set **`:INITiate:CONTinuous OFF`** (single mode), and also places the measurement in the idle state.

`:CONFigure:SANalyzer` causes the present measurement to exit (functionally the same as pressing **MEASURE, Meas Off**), and places the analyzer in base instrument spectrum analyzer state.

The query **`:CONFigure?`** returns the current measurement name in quotes.

Configure the Adjacent Channel Power Measurements

`:CONFigure:ACPower`

Configure the Channel Power and Density Measurements

`:CONFigure:CHPower`

Configure the Emission Bandwidth Measurements

`:CONFigure:EBWidth`

Configure the Harmonic Distortion Measurements

`:CONFigure:HARMonics[:<number>]`

Configure the OBW and Transmit Frequency Error Measurements

`:CONFigure:OBWidth`

COUPle Subsystem

Some measurement settings are automatically coupled together to optimize speed and accuracy. These commands control that coupling.

COUPle the Function to Other Settings

`:COUPle ALL|NONE`

`:COUPle?`

The instrument can automatically couple instrument settings together for accurate measurements and optimum dynamic range. This command is used to override the coupling for special measurement needs.

Factory Preset and *RST:	All
Remarks:	The following list of analyzer functions can either be automatically coupled, or not coupled (manually set):

Residual bandwidth
Video bandwidth
Sweep time
Center frequency step
Attenuation
Marker count resolution
Source attenuation
Source amplitude step

`COUPle NONE` puts these functions into the manually set (not coupled) mode. **`COUPle ALL`** puts the functions into the auto coupled mode, and also puts the sweep coupling mode into SA (couple all).

Front Panel Access:	**Auto Couple**

DISPlay Subsystem

The DISPlay subsystem controls the selection and presentation of textual, graphical, and trace information. Within a display, information may be separated into individual windows.

Display Viewing Angle

:DISPlay:ANGLe <integer>

:DISPlay:ANGLe?

Changes the viewing angle for better viewing in different environments.

Factory Preset and *RST:	The factory default is 4. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.
Range:	Integer, 1 to 7
Front Panel Access:	**viewing angle keys**

Date and Time Display Format

:DISPlay:ANNotation:CLOCk:DATE:FORMat MDY|DMY

:DISPlay:ANNotation:CLOCk:DATE:FORMat?

Allows you to set the format for displaying the real-time clock. To set the date time use: SYSTem:DATE <year>, <month>, <day>.

Factory Preset and *RST:	The factory default is MDY. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.
Front Panel Access:	**System, Time/Date, Date Format MDY DMY**

Date and Time Display

:DISPlay:ANNotation:CLOCk[:STATe] OFF|ON|0|1

:DISPlay:ANNotation:CLOCk[:STATe]?

Turns on and off the display of the date and time on the spectrum analyzer screen.

Factory Preset and *RST:	The factory default is On. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.
Front Panel Access:	**System, Time/Date, Time/Date On Off**

Display Annotation Title Data

:DISPlay:ANNotation:TITLe:DATA <string>

:DISPlay:ANNotation:TITLe:DATA?

Enters the text that will be displayed in the user title area of the display.

Front Panel Access:	**Display, Title**
	Display, Title, Change Title
	Display, Title, Clear Title

Turn the Entire Display On/Off

:DISPlay:ENABle OFF|ON|0|1

Turns the display on or off. Having the display turned off may increase repetitive measurement rate.

Factory Preset and *RST: On

Remarks: The following key presses will turn display enable back on:

1. If in local, press any key
2. If in remote, press the local (system) key
3. If in local lockout, no key

Front Panel Access: **none**

Window Annotation

:DISPlay:WINDow:ANNotation[:ALL] OFF|ON|0|1

:DISPlay:WINDow:ANNotation[:ALL]?

Turns the screen annotation on or off for all windows.

Factory Preset and *RST: On

Front Panel Access:	**Display, Preferences, Annotation On Off**

Trace Graticule Display

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

:DISPlay:WINDow:TRACe:GRATicule:GRID[:STATe]?

Turns the graticule on or off.

Factory Preset and *RST:	On
Front Panel Access:	**Display, Preferences, Graticule On Off**

Trace X-Axis Scale Offset

:DISPlay:WINDow:TRACe:X[:SCALe]:OFFSet <freq>

:DISPlay:WINDow:TRACe:X[:SCALe]:OFFSet?

Specifies the frequency offset for all frequency readouts such as center frequency, except that it does not affect marker count.

Factory Preset and *RST:	0 Hz
Range:	–3 GHz to 500 THz
Default Unit:	Hz
Front Panel Access:	**FREQUENCY/Channel, Freq Offset**

Set the Display Line

:DISPlay:WINDow:TRACe:Y:DLINe <ampl>

:DISPlay:WINDow:TRACe:Y:DLINe?

Defines the level of the display line, in the active amplitude units if no units are specified.

Factory Preset and *RST:	2.5 divisions below the reference level
Range:	10 display divisions below the reference level to the reference level
Default Unit:	Current active units
Front Panel Access:	**Display, Display Line On Off**

Control the Display Line

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

:DISPlay:WINDow:TRACe:Y:DLINe:STATe?

Turns the display line on or off.

Factory Preset and *RST:	Off
Front Panel Access:	**Display, Display Line On Off**

Normalized Reference Level

:DISPlay:WINDow:TRACe:Y[:SCALe]:NRLevel <rel_ampl>

:DISPlay:WINDow:TRACe:Y[:SCALe]:NRLevel?

Sets the normalized reference level.

NOTE See command **:CALCulate:NTData[STATe] OFF|ON|0|1**

Factory Preset and *RST:	0 dB
Range:	–327.6 to 327.6 dB
Default Unit:	Current active units
Front Panel Access:	**View/Trace, Normalize, Norm Ref Lvl**

Normalized Reference Level Position

:DISPlay:WINDow:TRACe:Y[:SCALe]:NRPosition <rel_ampl>

:DISPlay:WINDow:TRACe:Y[:SCALe]:NRPosition?

Selects the position of the normalized reference level. The top and bottom graticule lines correspond to 10 and 0, respectively.

NOTE See command **:CALCulate:NTData[STATe] OFF|ON|0|1**

Factory Preset and *RST:	10
Range:	0 to 10
Front Panel Access:	**View/Trace, Normalize, Norm Ref Posn**

Reference Level Auto Ranging

:DISPlay:WINDow:TRACe:Y[:SCALe]:LOG:RANGe:AUTO OFF|ON|0|1

:DISPlay:WINDow:TRACe:Y[:SCALe]:LOG:RANGe:AUTO?

This command enables and disables auto ranging.

Factory Preset and *RST:	On
Remarks:	With Option 1DR, Narrow Resolution Bandwidths, the reference level is normally auto-ranged. This results in a wider dynamic range measurement at a slower measurement speed. Using this command to disable auto ranging results in faster measurements and better resolution but reduces the available dynamic range.

NOTE Internally to the instrument, the reference level may change, depending on instrument settings and input signals present. However, the reference level shown on the display does not change when this command is used.

Trace Y-Axis Amplitude Scaling

:DISPlay:WINDow:TRACe:Y[:SCALe]:PDIVision <rel_ampl>

:DISPlay:WINDow:TRACe:Y[:SCALe]:PDIVision?

Sets the per-division display scaling for the y-axis when y-axis units are set to amplitude units.

Factory Preset and *RST:	10 dB
Range:	0.1 to 20.0 dB
Default Unit:	dB
Front Panel Access:	**AMPLITUDE/Y Scale, Scale/Div**

Trace Y-Axis Frequency Scaling

:DISPlay:WINDow:TRACe:Y[:SCALe]:PDIVision:FREQuency <freq>

:DISPlay:WINDow:TRACe:Y[:SCALe]:PDIVision:FREQuency?

This command sets the per-division display scaling for the y-axis,when the y-axis units are set to frequency units, such as when looking at FM deviation with the command **[:SENSe]:DEMod:VIEW[:STATe] OFF|ON|0|1**.

Factory Preset and *RST:	20 kHz

Range:	1 kHz to 240 kHz
Default Unit:	Hz
Front Panel Access:	**AMPLITUDE/Y Scale, Scale/Div**

Trace Y-Axis Reference Level

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

`:DISPlay:WINDow:TRACe:Y[:SCALe]:RLEVel?`

Sets the amplitude value of the reference level for the y-axis.

Factory Preset and *RST:	0 dBm
Range:	With zero reference level offset:
	ESA E4401B, E4411B: –327.6 to 50 dBm
	ESA E4402B, E4403B: –327.6 to 55 dBm
	ESA E4404B: –327.6 to 55 dBm
	ESA E4405B: –327.6 to 55 dBm
	ESA E4407B, E4408B: –327.6 to 55 dBm
	–149.9 to 55 dBm with zero reference level offset and max mixer level = –10 dBm. In external mixing, the range is –327.5 to –10 dBm.
Default Unit:	current active units
Remarks:	The input attenuator setting may be affected. The minimum displayed value of reference level is –327.6 dBm, and the maximum displayed value is 327.6 dBm. See the remarks given for the command **`:DISPlay:WINDow:TRACe:Y[:SCALe]:RLEVel:OFFSet <rel_ampl>`**
Front Panel Access:	**Amplitude Y Scale, Ref Level**

Trace Y-Axis Reference Level Offset

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

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

Sets the amplitude level offset for the y-axis.

Factory Preset and *RST:	0 dB
Range:	–327.6 to 327.6 dB

Default Unit:	dB
Remarks:	The sum of (reference level offset + reference level) is clipped to the range –327.6 to 327.6 dB. The maximum limits are determined by the setting of the first of these two parameters, within the boundaries of their individual limits when initially set. For example, if the reference level is (first) set to –20 dBm, then the reference level offset can be set to values of –307.6 dB to 327.6 dB. In the case of a 327.6 dB reference level offset, the resultant reference level value changes to 307.6 dBm. The reference level value range can be intially set to values from –149.9 to 55 dBm. Setting the reference level offset value first yields the following: If the reference level offset is (first) set to –30 dB, then the reference level can be set to values of –327.6 to 25 dBm. The reference level is “clamped” at 25 dBm because its positive value of 55 dBm is reached at 25 dBm with an offset of –30 dB. Its own positive amplitude limit applies. If the reference level offset is (first) set to 30 dB, then the reference level can be set to values of –327.6 to 85 dBm. Again, the positive amplitude limit of reference level (alone) is factored in to the resultant combined limit.
Front Panel Access:	**Amplitude Y Scale, Ref Level Offst**

Vertical Axis Scaling

:DISPlay:WINDow:TRACe:Y[:SCALe]:SPACing LINear|LOGarithmic

:DISPlay:WINDow:TRACe:Y[:SCALe]:SPACing?

Specifies the vertical graticule divisions as log or linear units.

Factory Preset and *RST:	Logarithmic
Front Panel Access:	**AMPLITUDE/Y Scale, Scale Type Log Lin**

FETCh Subsystem

FETCh subsystem commands apply only to measurements found in the **MEASURE** menu. FETCh commands can only be used as queries.

FETCh commands put valid data into the output buffer, but do not initiate data acquisition. Use the **`:INITiate[:IMMediate]`** command to acquire data. You can only fetch results from the measurement that is selected, and when current measurement results are valid.

`:FETCh <meas>?` will return valid data only when the measurement is in one of the following states:

idle
initiated
paused

Return Main, Lower, and Upper Channel Power

`:FETCh:ACPower?`

This command returns scalar results of main channel power, lower channel power (relative), and upper channel power (relative).

Remarks:	The main channel power is returned in the current amplitude units, and the lower and upper channel results are always returned in dB. The results are returned in a comma-separated list.

Return Main Channel Power

`:FETCh:ACPower:MAIN?`

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

Front Panel Access:	**MEASURE, ACP**

Return Lower Channel Power

`:FETCh:ACPower:LOWer?`

This command returns the value of the lower channel power relative to the main channel power, in dB.

Return Upper Channel Power

:FETCh:ACPower:UPPer?

This command returns the value of the upper channel power relative to the main channel power, in dB.

Return Channel Power and Density

:FETCh:CHPower?

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

Remarks:	The main channel power is returned in the current amplitude units, and the density value is returned in current amplitude units/Hz. The results are returned in a comma-separated list.

Return Channel Power

:FETCh:CHPower:CHPower?

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

Front Panel Access:	**MEASURE, Channel Power**

Return Channel Power Density

:FETCh:CHPower:DENSity?

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

Return Emission Bandwidth

:FETCh:EBWidth?

:FETCh:EBWidth:EBWidth?

This command returns the value of emission bandwidth in Hz.

Front Panel Access:	**MEASURE, Emission BW**

Return Harmonic Amplitudes

:FETCh:HARMonics:AMPLitude:ALL?

Returns a comma-separated list of the amplitudes of the ten harmonics.

Range:	–180 dBm to 70 dBm

Default Unit:	dBm (fundamental); dBc (all others)
Remarks:	The first value (for the fundamental) is measured in dBm; the remaining harmonics are measured in dBc. If fewer than ten harmonics are measured, zero is returned for any harmonic not measured. The harmonic amplitude precision for SCPI and the display is two decimal places.
Front Panel Access:	**MEASURE, Harmonic Dist**

Return Harmonic N Amplitude

:FETCh:HARMonics:AMPLitude[n]?

Returns the amplitude of harmonic number n, measured in dBc from the fundamental, or in dBm if n=1.

Range:	–180 dBm to 70 dBm
Default Unit:	dBc (from fundamental); dBm (n=1)
Remarks:	The harmonic amplitude precision for SCPI and the display is two decimal places.
Front Panel Access:	**MEASURE, Harmonic Dist**

Return % Total Harmonic Distortion

:FETCh:HARMonics[:DISTortion]?

Returns the computed total harmonic distortion as a percentage.

Range:	0 to float64
Default Unit:	%
Remarks:	The total harmonic distortion precision for SCPI and the display is three significant digits.
Front Panel Access:	**MEASURE, Harmonic Dist**

Return Harmonic Frequency List

:FETCh:HARMonics:FREQuency:ALL?

Returns a comma-separated list of the frequencies of the ten harmonics, measured in Hz.

Range:	0 Hz to the maximum frequency range of the analyzer
Default Unit:	Hz

Remarks:	If fewer than ten harmonics are measured, zero is returned for any harmonic not measured. The harmonic frequency precision for SCPI and the display is four significant digits.
Front Panel Access:	**MEASURE, Harmonic Dist**

Return Harmonic N Frequency

:FETCh:HARMonics:FREQuency[n]?

Returns the frequency of harmonic number n (2 to 10), measured in Hz.

Range:	0 Hz to the maximum frequency range of the analyzer
Default Unit:	Hz
Remarks:	The harmonic frequency precision for SCPI and the display is four significant digits.
Front Panel Access:	**MEASURE, Harmonic Dist**

Return Fundamental Frequency

:FETCh:HARMonics:FUNDamental?

Returns the frequency of the fundamental, measured in Hz.

Range:	0 Hz to the maximum frequency range of the analyzer
Default Unit:	Hz
Remarks:	The harmonic frequency precision for SCPI and the display is four significant digits.
Front Panel Access:	**MEASURE, Harmonic Dist**

Return OBW and Transmit Frequency Error

:FETCh:OBWidth?

This command returns scalar results of occupied bandwidth, and transmit frequency error.

Remarks:	The results for both values are returned in Hz and in a comma-separated list.

Return Occupied Bandwidth

`:FETCh:OBWidth:OBWidth?`

This command returns the value of occupied bandwidth in Hz.

Front Panel Access:	**MEASURE, Occupied BW**

Return Transmit Frequency Error

`:FETCh:OBWidth:FERRor?`

This command returns the value of transmit frequency error in Hz.

FORMat Subsystem

The FORMat subsystem sets a data format for transferring numeric and array information. **`TRACe[:DATA]`** and **`TRACe[:DATA]?`** are affected by FORMat subsystem commands.

Byte Order

`:FORMat:BORDer NORMal|SWAPped`

`:FORMat:BORDer?`

This command selects the binary data byte order for data transfer. It controls whether binary data is transferred in normal or swapped mode. This command affects only the byte order for setting and querying trace data for the command **`:TRACe[:DATA]`** and query **`:TRACe[:DATA]?`**

NOTE Normal mode is when the byte sequence begins with the most significant byte (MSB) first, and ends with the least significant byte (LSB) last in the sequence: 1|2|3|4. Swapped mode is when the byte sequence begins with the LSB first, and ends with the MSB last in the sequence: 4|3|2|1.

Factory Preset and *RST:	Normal

Numeric Data format

`:FORMat[:TRACe][:DATA] ASCii|INTeger,32|REAL,32|REAL,64|UINTeger,16`

`:FORMat[:TRACe][:DATA]?`

This command changes the format of the trace data input and output. The command affects only the data format for setting and querying trace data for the command **`:TRACe[:DATA]`** and query **`:TRACe[:DATA]?`**

NOTE

This command specifies the formats used for trace data during data transfer across any remote port.

For corrected trace data (**:TRACe[:DATA]** with parameter **<trace_name>**), **REAL** and **ASCii** formats will provide trace data in the current amplitude units. **INTeger** format will provide trace data in mdBm. The fastest mode is **INTeger,32**.

For uncorrected trace data (**:TRACe[:DATA]** with parameter **RAWTRACE**), **UINTeger** and **INTeger** formats apply to **RAWTRACE** queries, and return uncorrected ADC values. The fastest mode is **UINTeger,16**.

For state data, the format cannot be changed. It is always in a machine readable format only (machine units).

Table 5-1

Corrected Trace Data Types :TRACe:DATA?<trace_name>	
Data Type	**Result**
ASCii	Amplitude Units
INT,32 (fastest)	Internal Units
REAL,32	Amplitude Units
REAL,64	Amplitude Units

Table 5-2

Uncorrected Trace Data Types :TRACe:DATA? RAWTRACE	
Data Type	**Result**
INT,32	Uncorrected ADC Values
UINT,16 (fastest)	Uncorrected ADC Values

ASCii - Amplitude values are in ASCII, in amplitude units, separated by commas.

INTeger, 32 - Binary 32-bit integer values in internal units (mdBm), in a definite length block.

REAL, 32 (or 64) - Binary 32-bit, or 64-bit, real values in amplitude units), in a definite length block.

`UINTeger`, 16 - Binary 16-bit unsigned integer uncorrected ADC values, in a definite length block.

Factory Preset
and *RST: **`ASCII`**

HCOPy Subsystem

The HCOPy subsystem controls the setup of plotting and printing to an external device.

Abort the Print

:HCOPy:ABORt

Aborts hard copy printout of results.

Front Panel Access:	**ESC** (with print in progress)

Printer Type

:HCOPy:DEVice:TYPE AUTO|CUSTom|NONE

:HCOPy:DEVice:TYPE?

Sets up the printer by selecting printer type.

AUTO - the instrument queries the printer to determine the printer type and automatically sets itself for that printer

CUSTom - allows you to select a custom printer if your printer cannot be auto-configured

NONE - tells the instrument that the hardcopy output device is not a printer

Factory Preset and *RST:	The factory default is AUTO. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.
Front Panel Access:	**Print Setup, Printer Type**

Color Hard Copy

:HCOPy:IMAGe:COLor[:STATe] OFF|ON|0|1

:HCOPy:IMAGe:COLor[:STATe]?

Selects between color and monochrome mode for hardcopy output.

Factory Preset and *RST:	The factory default is On. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.

Front Panel
Access: **Print Setup, Color On Off**

Print a Hard Copy

`:HCOPy[:IMMediate]`

The entire screen is output to the parallel port.

Front Panel
Access: **Print**

Form Feed the Print Item

`:HCOPy:ITEM:FFEed[:IMMediate]`

Sends the printer a command to form feed.

Front Panel
Access: **Print Setup, Eject Page**

Page Orientation

`:HCOPy:PAGE:ORIentation LANDscape|PORTrait`

`:HCOPy:PAGE:ORIentation?`

Specifies the orientation of the print.

NOTE Landscape mode is not presently supported for PCL-3 printers.

Factory Preset
and *RST: The factory default is Landscape. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.

Front Panel
Access: **Print Setup, Orientation, Landscape**

Print Setup, Orientation, Portrait

Number of Items Printed on a Page

`:HCOPy:PAGE:PRINts <integer>`

`:HCOPy:PAGE:PRINts?`

Sets the number of display print outputs sent to print on one piece of paper, before a form feed is sent.

Factory Preset and *RST:	The factory default is 1. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.
Range:	Integer, 1 or 2
Front Panel Access:	**Print Setup, Prints/Page 1 2**

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 command affects sweep if not in a measurement, and affects trigger when in a measurement. A “measurement” refers to any of the functions under the **MEASURE** key. This corresponds to continuous sweep or single sweep operation when not in a measurement, and continuous measurement or single measurement operation when in a measurement.

NOTE

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

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

When in a measurement, this command does the following:

- When ON at the completion of each trigger cycle, the trigger system immediately initiates another trigger cycle.
- When OFF, the trigger system remains in an “idle” state until CONTinuous is 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.
- The query returns 1 or 0 into the output buffer. 1 is returned when there is continuous triggering. 0 is returned when there is only a single trigger.

Factory Preset: Continuous

*RST: Continuous, or On

Front Panel Access: **Sweep, Sweep Cont Single**

Single

Meas Control, Measure Cont Single

Take New Data Acquisitions

`:INITiate[:IMMediate]`

This command initiates a sweep if not in a measurement. If in a measurement, it triggers the instrument, if external triggering is the type of trigger event selected. Otherwise, the command is ignored. A “measurement” refers to any function under the **MEASURE** key.

Remarks: See also the **`*TRG`** command

Use the **`:TRIGer[:SEQuence]:SOURce EXTernal`** command to select the external trigger. The instrument must be in the single measurement mode. If **`:INITiate:CONTinuous`** is **ON** then the command is ignored.

Use **`:FETCh?`** to transfer a measurement result from memory to the output buffer. Refer to individual commands in the FETCh subsystem for more information.

If the analyzer is in signal identification mode, two sweeps are required, as this mode relies on the acquisition of data from two successive sweeps. Therefore, if the analyzer is in single sweep mode, two sweep triggers are needed to generate the sweep pair. In image suppress mode, synchronization is ensured by first turning off signal identifiation, initiating a single sweep, then turning on signal identification followed by two single sweeps. See **`[:SENSe]:SIDentify`** for more information about signal identification state.

Front Panel Access: **Sweep, Sweep Cont Single**

Single

Meas Control, Measure Cont Single

Restart Measurement

`:INITiate:RESTart`

This command applies to measurements found in the **MEASURE** menu. Use this command to restart the present measurement from the “idle” state, regardless of its operating state.

Remarks:	This command is equivalent of sending an **`:ABORt`** command followed by an **`:INITiate[:IMMediate]`** command.
Front Panel Access:	**Meas Control, Restart**

INPut Subsystem

The INPut subsystem controls the characteristics of analyzer input ports.

Input Port Coupling

`:INPut:COUPling AC|DC`

`:INPut:COUPling?`

Selects ac or dc coupling for the front panel INPUT port. A blocking capacitor is switched in for the ac mode.

CAUTION Instrument damage can occur if there is a dc voltage present at the INPUT and dc coupling is selected.

Factory Preset and *RST:	ac
Remarks:	This command is available only on Agilent ESA spectrum analyzer models E4402B Option UKB, E4404B, or E4405B.
Front Panel Access:	**Input/Output (or Input), Coupling AC DC**

Select Internal or External Mixer

`:INPut:MIXer INTernal|EXTernal`

`:INPut:MIXer?`

This command selects either the internal or external input mixer.

Factory Preset and *RST:	INTernal
Remarks:	Selecting the external input mixer activates all the keys in the Input Mixer menu and changes the RF attenuation annotation readout on the display to “Ext Mix.”
Front Panel Access:	**Input/Output (or Input), Input Mixer Int Ext**

Select Mixer Type

`:INPut:MIXer:TYPE PRESelected|UNPReselect`

`:INPut:MIXer:TYPE?`

This command selects the type of mixer being used.

Factory Preset and *RST:	UNPReselect
Remarks:	Setting mixer type to Presel activates a tuning signal that is routed to the PRESEL TUNE OUTPUT connector on the analyzer rear panel. This signal drives the tune input of the HP/Agilent 11974-Series Preselected Mixers at 1.5V/GHz. The sweep rate in this mode is limited to 40 MHz/msec.

NOTE: Preselected Mixer Type is not allowed when AUTO harmonic and Ext Mix Band K, E, W, F, D, G, Y, or J is selected.

Front Panel Access:	**Input/Output (or Input), Input Mixer, Mixer Config, Mixer Type Presel Unpre**

Clear the Input Overload

`:INPut:PROTection:CLEar`

Resets the overload protection circuitry for the input connector. There is no query form of this command.

NOTE: This command is valid only for Agilent ESA models E4401B and E4411B.

The excessive input signal may have caused 15 dB of attenuation to be switched in, or it may have completely switched the input connector out so that it is connected to the internal reference signal.

INSTrument Subsystem

This subsystem includes commands for querying and selecting instrument measurement (personality option) modes.

Select Application

:INSTrument[:SELect] SA|(application specific mode)

:INSTrument[:SELect]?

Select the measurement application (mode) by enumerated choice. The actual available choices depends upon which applications (modes) are installed in the instrument. See the manual that was part of the installed option for the mode designator (if any) of that option.

Once the instrument mode is selected, only the commands that are valid for that mode can be executed.

NOTE If you are using the SCPI status registers and the analyzer mode is changed, the status bits should be read, and any error conditions resolved, prior to switching modes. Error conditions that exist prior to switching modes cannot be detected using the condition registers after the mode change. This is true unless they recur after the mode change, although transitions of these conditions can be detected using the event registers.

Changing modes resets all SCPI status registers and mask registers to their power-on defaults. Therefore, any event or condition register masks must be re-established after a mode change. Also note that the power up status bit is set by any mode change, since that is the default state after power up.

Factory Preset and *RST:	Persistent state with factory default of Spectrum Analyzer
Front Panel Access:	**Mode**

MEASure Subsystem

MEASure subsystem commands apply only to measurements found in the **MEASURE** menu. These commands primarily control measurements in single mode.

Use the single commands in this subsystem to quickly make measurements using the factory default instrument settings.

`MEASure` commands stop the present measurement and set up the instrument for the specified measurement using the factory default values. Other SCPI communication is blocked until the measurement is complete. After the data is valid, the scalar result is returned for the specified measurement.

If you need to change some of the measurement parameters from the factory default settings, set up the measurement with the CONFigure command. Use the commands in the [:SENSe]: subsystem to change the settings. Then use the **`:READ?`** command, or the **`:INITiate`** and **`:FETCh?`** commands to initiate the measurement and query the results.

Measure Main, Lower, and Upper Channel Power

`:MEASure:ACPower?`

This command returns scalar results of main channel power, lower channel power (relative), and upper channel power (relative).

Remarks: The main channel power is returned in the current amplitude units, and the lower and upper channel results are always returned in dB. The results are returned in a comma-separated list.

Measure Main Channel Power

`:MEASure:ACPower:MAIN?`

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

Front Panel Access: **MEASURE, ACP**

Measure Lower Channel Power

`:MEASure:ACPower:LOWer?`

This command returns the value of the lower channel power relative to the main channel power, in dB.

Measure Upper Channel Power

:MEASure:ACPower:UPPer?

This command returns the value of the upper channel power relative to the main channel power, in dB.

Measure Channel Power and Density

:MEASure:CHPower?

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

Remarks:	The main channel power is returned in the current amplitude units, and the density value is returned in current amplitude units/Hz. The results are returned in a comma-separated list.

Measure Channel Power

:MEASure:CHPower:CHPower?

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

Front Panel Access:	**MEASURE, Channel Power**

Measure Channel Power Density

:MEASure:CHPower:DENSity?

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

Measure Emission Bandwidth

:MEASure:EBWidth?

:MEASure:EBWidth:EBWidth?

This command returns the value of emission bandwidth in Hz.

Front Panel Access:	**MEASURE, Emission BW**

Return Harmonic Amplitudes

:MEASure:HARMonics:AMPLitude:ALL?

Returns a comma-separated list of the amplitudes of the ten harmonics.

Range:	–180 dBm to 70 dBm
Default Unit:	dBm (fundamental); dBc (all others)
Remarks:	The first value (for the fundamental) is measured in dBm; the remaining harmonics are measured in dBc. If fewer than ten harmonics are measured, zero is returned for any harmonic not measured. The harmonic amplitude precision for SCPI and the display is two decimal places.
Front Panel Access:	**MEASURE, Harmonic Dist**

Return Harmonic N Amplitude

:MEASure:HARMonics:AMPLitude[n]?

Returns the amplitude of harmonic number n, measured in dBc from the fundamental, or in dBm if n=1.

Range:	–180 dBm to 70 dBm
Default Unit:	dBc (from fundamental); dBm (n=1)
Remarks:	The harmonic amplitude precision for SCPI and the display is two decimal places.
Front Panel Access:	**MEASURE, Harmonic Dist**

Return % Total Harmonic Distortion

:MEASure:HARMonics[:DISTortion]?

Returns the computed total harmonic distortion as a percentage.

Range:	0 to float64
Default Unit:	%
Remarks:	The total harmonic distortion precision for SCPI and the display is three significant digits.
Front Panel Access:	**MEASURE, Harmonic Dist**

Return Harmonic Frequency List

:MEASure:HARMonics:FREQuency:ALL?

Returns a comma-separated list of the frequencies of the ten harmonics, measured in Hz.

Range:	0 Hz to the maximum frequency range of the analyzer.

Default Unit:	Hz
Remarks:	If fewer than ten harmonics are measured, zero is returned for any harmonic not measured. The harmonic frequency precision for SCPI and the display is four significant digits.
Front Panel Access:	**MEASURE, Harmonic Dist**

Return Harmonic N Frequency

:MEASure:HARMonics:FREQuency[n]?

Returns the frequency of harmonic number n (2 to 10), measured in Hz.

Range:	0 Hz to the maximum frequeny range of the analyzer
Default Unit:	Hz
Remarks:	The harmonic frequency precision for SCPI and the display is four significant digits.
Front Panel Access:	**MEASURE, Harmonic Dist**

Return Fundamental Frequency

:MEASure:HARMonics:FUNDamental?

Returns the frequency of the fundamental, measured in Hz.

Range:	0 Hz to the maximum frequency range of the analyzer
Default Unit:	Hz
Remarks:	The harmonic frequency precision for SCPI and the display is four significant digits.
Front Panel Access:	**MEASURE, Harmonic Dist**

Measure OBW and Transmit Frequency Error

:MEASure:OBWidth?

This command returns scalar results of occupied bandwidth, and transmit frequency error.

Remarks:	The results for both values are returned in Hz and in a comma-separated list.

Measure Occupied Bandwidth

:MEASure:OBWidth:OBWidth?

This command returns the value of occupied bandwidth in Hz.

Front Panel Access:	**MEASURE, Occupied BW**

Measure Transmit Frequency Error

:MEASure:OBWidth:FERRor?

This command returns the value of transmit frequency error in Hz.

MMEMory Subsystem

The purpose of the MMEMory subsystem is to provide access to mass storage devices such as internal or external disk drives.

NOTE Refer also to **:CALCulate** and **:TRACe** subsystems for more trace and limit line commands.

Agilent ESA analyzers use two types of mass storage devices:

- 3.5 inch disk drive (high density, 1.44 MBytes formatted) designated "A:"
- Part of flash memory and treated as a device designated "C:"

The MMEMory command syntax term **<file_name>** is a specifier having the form: drive:name.ext, where the following rules apply:

- "drive" is "A:" or "C:"
- "name" is a DOS file name of up to eight characters, letters (A-Z, a-z) and numbers (0-9) only (lower case letters are read as uppercase)
- "ext" is an optional file extension using the same rules as "name," but consists of up to three characters total

Catalog the Selected Memory Location

:MMEMory:CATalog? <drive>

where "drive" is "A:" or "C:"

Lists all files in the specified drive. The return data will be of the format: <mem_used>,<mem_free> {,<file_listing>}

Each <file listing> indicates the name, and size of one file in the directory list: <file_name>,,<file_size>

Example:	`Catalog drive C:`, which is in instrument memory: **CATalog? "C:"**
Front Panel Access:	**File**

Copy a File

:MMEMory:COPY <file_name1>,<file_name2>

To copy a file, the source file name is <file_name1> and the destination file name is <file_name2>.

Example: **:MMEM:COPY "C:oldname.sta","A:\newname.sta"**

Front Panel
Access: **File, File Manager, Copy**

Move Data to File

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

`:MMEMory:DATA? <file_name>`

Loads <**`definite_length_block`**> into the memory location <file_name>.

The query returns the contents of the <file_name> in the format of a definite length block. This command can be used for copying files out of the analyzer over the remote bus. Refer to chapter 3, Programming Examples, for more information.

Example: Load ”abcd” into C:source.txt:

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

Front Panel
Access: **none**

Delete a File

`:MMEMory:DELete <file_name>`

Delete a file.

Example: **`:MMEM:DEL "C:source.txt"`**

Remarks: If <file_name> does not exist, a “File Name Error” will occur.

Front Panel
Access: **File, File Manager, Delete**

Load a Corrections Table from a File

`:MMEMory:LOAD:CORRection`
`ANTenna|CABLe|OTHer|USER,<file_name>`

Loads the data in the file <file_name> to the specified correction set.

Example: **`:MMEM:LOAD:CORR ANT, “A:TEST5.CBL”`**

Front Panel
Access: **File, Load, Corrections**

Load a Limit Line from Memory to the Instrument

`:MMEMory:LOAD:LIMit LLINE1|LLINE2,<file_name>`

Loads a limit line, from the specified file in mass storage to the instrument. Loading a time limit line deletes any frequency limit lines. Similarly, loading a frequency limit line deletes any time limit lines.

Example: **`:MMEM:LOAD:LIM LLINE2,"C:mylimit.lim"`**

Remarks: There is no SCPI short form for parameters **`LLINE1|LLINE2.`**

Front Panel Access: **File, Load, Limits**

Load an Instrument State from a File

`:MMEMory:LOAD:STATe 1, <file_name>`

The contents of the state file are loaded into the specified register. To then load the register into the current instrument state use *RCL.

Example: **`:MMEM:LOAD:STAT 1,"C:mystate.sta"`**

Remarks: See also commands **`:MMEMory:LOAD:STATe`** and **`:MMEMory:STORe:STATe`**

If the revision of the state being loaded is newer than the revision of the instrument, no state is recalled and an error is reported.

If the revision of the state being loaded is equal to the revision of the instrument, all regions of the state will be loaded.

If the revision of the state being loaded is older than the revision of the instrument, the instrument will only load the older regions of the state.

Front Panel Access: **File, Load, State**

Load a Trace From a File to the Instrument

`:MMEMory:LOAD:TRACe <file_name>`

The contents of the file are loaded into TRACE1. The file name must have a file extension of .trc or .csv. The file extension determines whether a trace is loaded, or a trace with its state, are loaded. The .csv extension is for trace files using the CSV (comma-separated values) format. The .trc extension is for files that include both trace and state data.

Example: **`:MMEM:LOAD:TRAC "C:mytrace.trc"`**

Remarks: See also commands **`:MMEMory:LOAD:STATe`** and **`:MMEMory:STORe:STATe`**

If the revision of the state being loaded is newer than the revision of the instrument, no state is recalled and an error is reported.

If the revision of the state being loaded is equal to the revision of the instrument, all regions of the state will be loaded.

If the revision of the state being loaded is older than the revision of the instrument, the instrument will only load the older regions of the state.

Create a New Directory

`:MMEMory:MDIRectory <name>`

This command creates a new directory.

Example: **`:MMEM:MDIR "C:\myDir"`**

Front Panel Access: **File, Create Dir**

Delete a Directory

`:MMEMory:RDIRectory <name>`

This command deletes a directory.

Example: **`:MMEM:RDIR "C:\myDir"`**

Remarks: This command deletes the specified directory and all files and subdirectories within that directory.

Front Panel Access: **File, Delete**

Store a Corrections Table to a File

`:MMEMory:STORe:CORRection ANTenna|CABLe|OTHer|USER,<file_name>`

Stores the specified correction set to the file named <file_name>.

Example: **`:MMEM:STOR:CORR ANT, "A:TEST1.ANT"`**

Remarks: This command will fail if the **`<file_name>`** already exists.

Front Panel Access: **File, Save, Corrections**

Store a Limit Line in a File

:MMEMory:STORe:LIMit LLINE1|LLINE2,<file_name>

Stores the specified limit line to the specified file in memory.

Example: **:MMEM:STOR:LIM LLINE2,"C:mylimit.lim"**

Remarks: This command will fail if the **<file_name>** already exists. There is no SCPI short form for parameters **LLINE1|LLINE2.**

Store a Screen Image in a Graphic File

:MMEMory:STORe:SCReen <file_name>

Saves the current instrument screen image, as a graphic file, to the specified file in memory. The file must have a .gif or .wmf file extension. The specified file extension determines which file format the instrument will use to save the image.

Example: **:MMEM:STOR:SCR "C:myscreen.gif"**

Remarks: This command will fail if the **<file_name>** already exists.

Store an Instrument State in a File

:MMEMory:STORe:STATe 1,<file_name>

Saves the instrument state to the file in memory.

Example: **:MMEM:STOR:STAT 1,"C:mystate.sta"**

Remarks: This command will fail if the **<file_name>** already exists.

Store a Trace in a File

:MMEMory:STORe:TRACe <label>,<file_name>

Saves the specified trace to a file in memory. The file name must have a file extension of .trc or .csv. The file extension determines whether a trace is stored, or a trace with its state, are stored. The .csv extension is for trace files using the CSV (comma-separated values) format. The .trc extension is for files that include both trace and state data.

Example: **:MMEM:STOR:TRAC TRACE3,"C:mytrace.trc"**

Range: Trace labels are: TRACE1|TRACE2|TRACE3|ALL

Remarks: This command will fail if the **<file_name>** already exists.

OUTPut Subsystem

The OUTPut subsystem controls the characteristics of the tracking generator output port. Refer also to the “SOURce Subsystem” on page 5-109, which contains several commands that control the tracking generator output.

Turn Output On/Off

`:OUTPut[:STATe] OFF|ON|0|1`

`:OUTPut[:STATe]?`

Controls the tracking generator output.

Factory Preset and *RST:	Off
Front Panel Access:	**Source, Amplitude On Off**

READ Subsystem

READ subsystem commands apply only to measurements found in the **MEASURE** menu. READ commands do not preset the measurement to the factory default values, as do the MEASurement subsystem commands. Instead, they use settings from the last measurement.

Read commands initiate the measurement and put valid data into the output buffer. Other SCPI communication is blocked until the measurement is complete. However, 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.

Measure Main, Lower, and Upper Channel Power

:READ:ACPower?

This command returns scalar results of main channel power, lower channel power (relative), and upper channel power (relative).

Remarks: The main channel power is returned in the current amplitude units, and the lower and upper channel results are always returned in dB. The results are returned in a comma-separated list.

Measure Main Channel Power

:READ:ACPower:MAIN?

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

Front Panel Access: **MEASURE, ACP**

Measure Lower Channel Power

:READ:ACPower:LOWer?

This command returns the value of the lower channel power relative to the main channel power, in dB.

Measure Upper Channel Power

:READ:ACPower:UPPer?

This command returns the value of the upper channel power relative to the main channel power, in dB.

Measure Channel Power and Density

:READ:CHPower?

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

Remarks:	The main channel power is returned in the current amplitude units, and the density value is returned in current amplitude units/Hz. The results are returned in a comma-separated list.

Measure Channel Power

:READ:CHPower:CHPower?

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

Front Panel Access:	**MEASURE, Channel Power**

Measure Channel Power Density

:READ:CHPower:DENSity?

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

Measure Emission Bandwidth

:READ:EBWidth?

:READ:EBWidth:EBWidth?

This command returns the value of emission bandwidth in Hz.

Front Panel Access:	**MEASURE, Emission BW**

Return Harmonic Amplitudes

:READ:HARMonics:AMPLitude:ALL?

Returns a comma-separated list of the amplitudes of the ten harmonics.

Range:	–180 dBm to 70 dBm
Default Unit:	dBm (fundamental); dBc (all others)

Remarks:	The first value (for the fundamental) is measured in dBm; the remaining harmonics are measured in dBc. If fewer than ten harmonics are measured, zero is returned for any harmonic not measured. The harmonic amplitude precision for SCPI and the display is two decimal places.
Front Panel Access:	**MEASURE, Harmonic Dist**

Return Harmonic N Amplitude

`:READ:HARMonics:AMPLitude[n]?`

Returns the amplitude of harmonic number n, measured in dBc from the fundamental, or in dBm if n=1.

Range:	–180 dBm to 70 dBm
Default Unit:	dBc (from fundamental); dBm (n=1)
Remarks:	The harmonic amplitude precision for SCPI and the display is two decimal places.
Front Panel Access:	**MEASURE, Harmonic Dist**

Return % Total Harmonic Distortion

`:READ:HARMonics[:DISTortion]?`

Returns the computed total harmonic distortion as a percentage.

Range:	0 to float64
Default Unit:	%
Remarks:	The total harmonic distortion precision for SCPI and the display is three significant digits.
Front Panel Access:	**MEASURE, Harmonic Dist**

Return Harmonic Frequency List

`:READ:HARMonics:FREQuency:ALL?`

Returns a comma-separated list of the frequencies of the ten harmonics, measured in Hz.

Range:	0 Hz to the maximum frequency range of the analyzer
Default Unit:	Hz

Remarks:	If fewer than ten harmonics are measured, zero is returned for any harmonic not measured. The harmonic frequency precision for SCPI and the display is four significant digits.
Front Panel Access:	**MEASURE, Harmonic Dist**

Return Harmonic N Frequency

`:READ:HARMonics:FREQuency[n]?`

Returns the frequency of harmonic number n (2 to 10), measured in Hz.

Range:	0 Hz to the maximum frequency range of the analyzer
Default Unit:	Hz
Remarks:	The harmonic frequency precision for SCPI and the display is four significant digits.
Front Panel Access:	**MEASURE, Harmonic Dist**

Return Fundamental Frequency

`:READ:HARMonics:FUNDamental?`

Returns the frequency of the fundamental, measured in Hz.

Range:	0 Hz to the maximum frequency range of the analyzer
Default Unit:	Hz
Remarks:	The harmonic frequency precision for SCPI and the display is four significant digits.
Front Panel Access:	**MEASURE, Harmonic Dist**

Measure OBW and Transmit Frequency Error

`:READ:OBWidth?`

This command returns scalar results of occupied bandwidth, and transmit frequency error.

Remarks:	The results for both values are returned in Hz and in a comma-separated list.

Measure Occupied Bandwidth

`:READ:OBWidth:OBWidth?`

This command returns the value of occupied bandwidth in Hz.

Front Panel Access:	**MEASURE, Occupied BW**

Measure Transmit Frequency Error

`:READ:OBWidth:FERRor?`

This command returns the value of transmit frequency error in Hz.

SENSe Subsystem

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

SENSe subsystem commands used for measurements in the **MEASURE** and **Meas Setup** menus may only be used to set parameters of a specific measurement when the measurement is active. Otherwise, an error will occur. You must first select the appropriate measurement using the **`:CONFigure:<measurement>`** command. If a **`:SENSe`** command is used to change a parameter during a measurement (while not in its idle state), the measurement will be restarted.

[:SENSe]:ACPower Subsection

Set Adjacent Channel Power Number of Averages

[:SENSe]:ACPower:AVERage:COUNt <integer>

[:SENSe]:ACPower:AVERage:COUNt?

Set the adjacent channel power measurement number of averages.

Remarks:	This command is used for measurements in the **MEASURE** menu.
Front Panel Access:	**Meas Setup, Avg Number**

Adjacent Channel Power Averaging On/Off

[:SENSe]:ACPower:AVERage[:STATe] OFF|ON|0|1

[:SENSe]:ACPower:AVERage[:STATe]?

Turn on or off the adjacent channel power measurement averaging.

Factory Preset and *RST:	Off
Remarks:	This command is used for measurements in the **MEASURE** menu.
Front Panel Access:	**Meas Setup, Avg Number On Off**

Set Adjacent Channel Bandwidth

[:SENSe]:ACPower:BANDwidth|BWIDth:ACHannel <freq>

[:SENSe]:ACPower:BANDwidth|BWIDth:ACHannel?

Set the adjacent channel bandwidth.

Default Unit:	Hz
Remarks:	This command is used for measurements in the **MEASURE** menu.
Front Panel Access:	**Meas Setup, Adj Chan BW**

Set Main Channel Bandwidth

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

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

Set the main (center) channel bandwidth.

Remarks:	This command is used for measurements in the **MEASURE** menu.

Front Panel Access:	**Meas Setup, Main Chan BW**

Set Adjacent Channel Spacing

`[:SENSe]:ACPower:CSPacing <freq>`

`[:SENSe]:ACPower:CSPacing?`

Set the adjacent channel spacing.

Remarks:	This command is used for measurements in the **MEASURE** menu.
Front Panel Access:	**Meas Setup, Main Chan Spacing**

[:SENSe]:AVERage Subsection

Clear the Current Average

`[:SENSe]:AVERage:CLEar`

Re-start the trace averaging function.

NOTE Re-start the trace at the beginning of a sweep to obtain valid average data. To do this, remotely abort the sweep and initiate a single sweep.

Front Panel Access:	**none**

Set the Average Count

`[:SENSe]:AVERage:COUNt <integer>`

`[:SENSe]:AVERage:COUNt?`

Specifies the number of measurements that are combined.

Factory Preset and *RST:	100
Range:	1 to 8192
Front Panel Access:	**BW/Avg, Average On Off**

Turn Averaging On/Off

`[:SENSe]:AVERage[:STATe] OFF|ON|0|1`

`[:SENSe]:AVERage[:STATe]?`

This command toggles averaging off and on. Averaging combines the value of successive measurements to average out measurement variations.

Factory Preset and *RST:	Off
Remarks:	When a measurement under the front panel **MEASURE** key is started, this command is turned off for video averaging (**`[:SENSe]:AVERage:TYPE LPOWer)`**. If this command is turned on for video averaging when any of the **MEASURE** key measurements are in progress, that measurement will be stopped.
Front Panel Access:	**BW/Avg, Average On Off**

Type of Averaging for Measurements

`[:SENSe]:AVERage:TYPE LPOWer|POWer`

`[:SENSe]:AVERage:TYPE?`

This command selects the type of averaging to be performed.

`LPOWer` logarithmically averages the power of the video data (typical units are dBm). This command is equivalent to pressing front panel keys **BW/Avg, Average Type, Video.**

`POWer` averages the linear power of successive measurements (typical units are watts).

Factory Preset and *RST:	**`LPOWer`**
Front Panel Access:	**BW/Avg, Average Type, Video Power**

[:SENSe]:BANDwidth Subsection

Resolution Bandwidth

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

`[:SENSe]:BANDwidth|BWIDth[:RESolution]?`

Specifies the resolution bandwidth.

Factory Preset and *RST:	3 MHz
Range:	10 Hz to 5 MHz with Option 1DR, narrow resolution bandwidth; 1 kHz to 5 MHz without Option 1DR.
Default Unit:	Hz
Front Panel Access:	**BW/Avg, Resolution BW Auto Man**

Resolution Bandwidth Automatic

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

[:SENSe]:BANDwidth|BWIDth[:RESolution]:AUTO?

Couples the resolution bandwidth to the frequency span.

Factory Preset and *RST:	On

Video Bandwidth

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

[:SENSe]:BANDwidth|BWIDth:VIDeo?

Specifies the video bandwidth.

Factory Preset and *RST:	3 MHz
Range:	1 Hz to 3 MHz. This range is dependent upon the setting of **[:SENSe]:BANDwidth|BWIDth[:RESolution]**.
Default Unit:	Hz
Front Panel Access:	**BW/Avg, Video BW Auto Man**

Video Bandwidth Automatic

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

[:SENSe]:BANDwidth|BWIDth:VIDeo:AUTO?

Couples the video bandwidth to the resolution bandwidth.

Factory Preset and *RST:	On
Front Panel Access:	**BW/Avg, Video BW Auto Man**

Video to Resolution Bandwidth Ratio

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

[:SENSe]:BANDwidth|BWIDth:VIDeo:RATio?

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

Factory Preset and *RST:	1.0
Range:	0.00001 to 3.0e6
Front Panel Access:	**BW/Avg, VBW/RBW Ratio**

[:SENSe]:CHPower Subsection

Set Channel Power Number of Averages

`[:SENSe]:CHPower:AVERage:COUNt <integer>`

`[:SENSe]:CHPower:AVERage:COUNt?`

Set the channel power measurement number of averages.

Remarks:	This command is used for measurements in the **MEASURE** menu.
Front Panel Access:	**Meas Setup, Avg Number**

Channel Power Averaging On/Off

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

`[:SENSe]:CHPower:AVERage[:STATe]?`

Turn on or off the channel power measurement averaging.

Factory Preset and *RST:	Off
Remarks:	This command is used for measurements in the **MEASURE** menu.
Front Panel Access:	**Meas Setup, Avg Number On Off**

Set Channel Power Integration Bandwidth

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

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

Set the frequency span (bandwidth) over which to integrate power.

Remarks:	This command is used for measurements in the **MEASURE** menu.
Front Panel Access:	**Meas Setup, Integration BW**

Set Channel Power Span

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

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

Set the measurement channel power span of the analyzer.

Remarks:	This command is used for measurements in the **MEASURE** menu.

Front Panel Access:	**Meas Setup, Chan Pwr Span**

[:SENSe]:CORRection Subsection

Perform Amplitude Correction

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

[:SENSe]:CORRection:CSET:ALL[:STATe]?

Turns On or Off the amplitude corrections. When turned On, only the correction sets that were turned on are enabled. When turned Off, all of the correction sets are disabled.

Factory Preset and *RST:	Off
Remarks:	To turn On or Off an individual correction set, use: **[:SENSe]:CORRection:CSET[1]|2|3|4[:STATe]**.
Front Panel Access:	**AMPLITUDE/Y Scale, Corrections, Corrections On Off**

Set Amplitude Correction Data

[:SENSe]:CORRection:CSET[1]|2|3|4:DATA <freq>,<rel_ampl>{,<freq>,<rel_ampl>}

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

Sets the amplitude correction data. These frequency/amplitude corrections will be applied to the displayed data to correct for system losses/gains outside the analyzer. Four different sets of correction data can be stored.

Example:	**:CORR:CSET1:DATA 900E6,0.3,1.0E9,0.35,1.3E9,0.2**
Range:	200 points per set
Default Unit:	There are no units on the frequency and amplitude pairs. They must be entered in hertz (Hz) and decibels (dB).
Remarks:	CSET number equivalents to front panel access definitions are as follows: CSET1 is Antenna CSET2 is Cable CSET3 is Other CSET4 is User

Front Panel Access:	**AMPLITUDE/Y Scale, Corrections, Modify**
	AMPLITUDE/Y Scale, Corrections, Modify, Select, Antenna
	AMPLITUDE/Y Scale, Corrections, Modify, Select, Cable
	AMPLITUDE/Y Scale, Corrections, Modify, Select, Other
	AMPLITUDE/Y Scale, Corrections, Modify, Select, User
	AMPLITUDE/Y Scale, Corrections, Modify, Edit, Point
	AMPLITUDE/Y Scale, Corrections, Modify, Edit, Frequency
	AMPLITUDE/Y Scale, Corrections, Modify, Edit, Amplitude
	AMPLITUDE/Y Scale, Corrections, Modify, Edit, Delete Point

Merge Additional Values into the Existing Amplitude Correction Data

[:SENSe]:CORRection:CSET[1]|2|3|4:DATA:MERGe <freq>,<rel_ampl>{,<freq>,<rel_ampl>}

Adds the points with the specified values to the current amplitude correction data, allowing you to merge correction data. If too much data is merged, as many points as possible are merged into the existing data and then an error is reported.

- **<freq>** is the frequency (in Hz) where the correction should be applied; no unit is allowed in this parameter
- **<rel_ampl>** is the amount of relative amplitude correction (in dB) needed; no unit is allowed in this parameter

Remarks:	CSET number equivalents to front panel access definitions are as follows:
	CSET1 is Antenna
	CSET2 is Cable
	CSET3 is Other
	CSET4 is User

Delete Amplitude Correction

[:SENSe]:CORRection:CSET[1]|2|3|4:DELete

Deletes the specified correction set. If the set was On, it is turned Off.

Front Panel Access:	**AMPLITUDE/Y Scale, Corrections, Modify, Delete**

Set Amplitude Correction Frequency Interpolation

[:SENSe]:CORRection:CSET[1]|2|3|4:X:SPACing

LINear|LOGarithmi

Sets the frequency interpolation to linear or logarithmic for the specified correction set.

Remarks:	Logarithmic frequency scale corrections are linearly interpolated between correction points with respect to the logarithm of the frequency. Linear frequency scale corrections are interpolated along straight lines, connecting adjacent points on a linear scale.
Front Panel Access:	**AMPLITUDE/Y Scale, Corrections, Modify, Freq Interp Log Lin**

Perform Amplitude Correction

[:SENSe]:CORRection:CSET[1]|2|3|4[:STATe] OFF|ON|0|1

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

Turns the amplitude correction function on or off for the given set.

NOTE **[:SENSe]:CORRection:CSET:ALL[:STATe]** must be on for this command to function.

Factory Preset and *RST:	Off
Remarks:	CSET number equivalents to front panel access definitions are as follows:
	CSET1 is Antenna
	CSET2 is Cable
	CSET3 is Other
	CSET4 is User
Front Panel Access:	**AMPLITUDE/Y Scale, Corrections, Modify, Correction On Off**

Input Impedance Correction

[:SENSe]:CORRection:IMPedance[:INPut][:MAGNitude] <number>

[:SENSe]:CORRection:IMPedance[:INPut][:MAGNitude]?

Amplitude correction is applied to the display data to adjust for measurement situations where the unit under test has a different impedance than the 50Ω input impedance of the analyzer. Some Agilent ESA analyzers have Option 1DP, 75Ω input. In this case, you may want to convert the data to make measurements in a 50Ω system.

Factory Preset and *RST:	The factory default is the input impedance of the analyzer.
Range:	50 or 75 ohms
Default Unit:	ohms
Front Panel Access:	**Input, Input Z Corr 50 Ω 75 Ω**

External Amplifier Correction

[:SENSe]:CORRection:OFFSet[:MAGNitude] <rel_ampl>

[:SENSe]:CORRection:OFFSet[:MAGNitude]?

A single value of amplitude correction can be applied to the displayed trace data to compensate for signal losses or gains that are due to other devices in the measurement setup, rather than the unit under test.

Factory Preset and *RST:	0 dB
Range:	–81.9 to 81.9
Default Unit:	dB
Front Panel Access:	**AMPLITUDE/Y Scale, Ext Amp Gain**

[:SENSe]:DEMod Subsection

Type of Demodulation

[:SENSe]:DEMod AM|FM

[:SENSe]:DEMod?

Sets the type of demodulation.

Factory Preset and *RST:	AM
Front Panel Access:	**Det/Demod, Demod, AM**
	Det/Demod, Demod, FM

FM Deviation

[:SENSe]:DEMod:FMDeviation <freq>

[:SENSe]:DEMod:FMDeviation?

Sets the total FM frequency deviation for full screen demodulation.

Factory Preset and *RST:	100 kHz
Range:	5 kHz to 1.2 MHz
Default Unit:	Hz
Front Panel Access:	**Det/Demod, Demod, FM Deviation**

Demodulation Control

`[:SENSe]:DEMod:STATe OFF|ON|0|1`

`[:SENSe]:DEMod:STATe?`

Turns demodulation on or off.

Factory Preset and *RST:	Off
Front Panel Access:	**Det/Demod, Demod, Off**

Demod Time

`[:SENSe]:DEMod:TIME <time>`

`[:SENSe]:DEMod:TIME?`

Sets the time used for frequency domain demodulation.

Factory Preset and *RST:	500 ms
Range:	2 ms to 100 s
Default Unit:	seconds
Front Panel Access:	**Det/Demod, Demod, Demod Time**

Demod View

`[:SENSe]:DEMod:VIEW[:STATe] OFF|ON|0|1`

`[:SENSe]:DEMod:VIEW[:STATe]?`

This command causes the demodulated signal to be displayed. If FM Demod is on, then the display scales the y-axis in units of kHz. The scale/div is set with the command **`:DISPlay:WINDow:TRACe:Y [:SCALe]:PDIVision:FREQuency <freq>`** if FM Demod is on. If FM Demod is on, then several functions are not available; these include: Log/Lin (display is always in linear), Y-Axis Units, Marker Search functions, Normalize, Display Line, Peak Excursion, and Peak Threshold. There is no effect when AM demodulation is used (only applicable for FM demodulation).

Factory Preset and *RST:	Off
Remarks:	This command is not available when Demod is set to Off.
Front Panel Access:	**Det/Demod, Demod, FM, Demod View**

[:SENSe]:DETector Subsection

Type of Detection

[:SENSe]:DETector[:FUNCtion] NEGative|POSitive|SAMPle

[:SENSe]:DETector[:FUNCtion]?

Specifies the detection mode.

Negative peak detection displays the lowest sample taken during the interval being displayed.

Positive peak detection displays the highest sample taken during the interval being displayed.

Sample detection displays the first sample taken during the interval being displayed.

Factory Preset and *RST:	Positive
Front Panel Access:	**Det/Demod, Detector**
	Det/Demod, Detector, Peak
	Det/Demod, Detector, Sample
	Det/Demod, Detector, Negative Peak

[:SENSe]:EBWidth Subsection

Set Emission BW Number of Averages

[:SENSe]:EBWidth:AVERage:COUNt <integer>

[:SENSe]:EBWidth:AVERage:COUNt?

Set the emission bandwidth measurement number of averages.

Remarks:	This command is used for measurements in the **MEASURE** menu.
Front Panel Access:	**Meas Setup, Avg Number**

Emission BW Averaging On/Off

`[:SENSe]:EBWidth:AVERage[:STATe] OFF|ON|0|1`

`[:SENSe]:EBWidth:AVERage[:STATe]?`

Turn on or off the channel power measurement averaging.

Factory Preset and *RST:	Off
Remarks:	This command is used for measurements in the **MEASURE** menu.
Front Panel Access:	**Meas Setup, Avg Number On Off**

Set Emission BW Span

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

`[:SENSe]:EBWidth:FREQuency:SPAN?`

Set the emission bandwidth span. The analyzer span will retain this value throughout the measurement.

Remarks:	This command is used for measurements in the **MEASURE** menu.
Front Panel Access:	**Meas Setup, EBW Span**

Emission BW Trace Max Hold On/Off

`[:SENSe]:EBWidth:MAXHold OFF|ON|0|1`

`[:SENSe]:EBWidth:MAXHold?`

Turn on or off the trace max hold trace feature for emission bandwidth measurements.

Factory Preset and *RST:	On
Remarks:	This command is used for measurements in the **MEASURE** menu. Max hold on displays and holds the maximum responses of a signal; max hold off disables the feature.
Front Panel Access:	**Meas Setup, Max Hold On Off**

Set Emission BW X dB Value

`[:SENSe]:EBWidth:XDB <rel_ampl>`

`[:SENSe]:EBWidth:XDB?`

This commands allows you to set the dB value (X dB) below the maximum value on the signal at which to measure the emission bandwidth.

Range:	Only negative values are allowed.
Default Unit:	dB
Remarks:	This command is used for measurements in the **MEASURE** menu.
Front Panel Access:	**Meas Setup, Emiss BW X dB**

[:SENSe]:FREQuency Subsection

Center Frequency

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

`[:SENSe]:FREQuency:CENTer?`

Set the center frequency.

Factory Preset and *RST:	ESA E4401B, E4411B: 750 MHz
	ESA E4402B, E4403B: 1.5 GHz
	ESA E4404B: 3.35 GHz
	ESA E4405B: 6.6 GHz
	ESA E4407B, E4408B: 13.25 GHz
Range:	ESA E4401B, E4411B: –80 MHz to 1.58 GHz
	ESA E4402B, E4403B: –80 MHz to 3.10 GHz
	ESA E4404B: –80 MHz to 6.78 GHz
	ESA E4405B: –80 MHz to 13.3 GHz
	ESA E4407B, E4408B: –80 MHz to 27.0 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
Front Panel Access:	**FREQUENCY/Channel, CF Step Auto Man**

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:	Span/10
Range:	Maximum negative frequency to the maximum positive frequency. The maximum positive frequencies are:
	ESA E4401B, E4411B: –80 MHz to 1.58 GHz
	ESA E4402B, E4403B: –80 MHz to 3.10 GHz
	ESA E4404B: –80 MHz to 6.78 GHz
	ESA E4405B: –80 MHz to 13.3 GHz
	ESA E4407B, E4408B: –80 MHz to 27.0 GHz
Default Unit:	Hz
Front Panel Access:	**FREQUENCY/Channel, CF Step Man**

Frequency Span

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

`[:SENSe]:FREQuency:SPAN?`

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

Factory Preset and *RST:	ESA E4401B, E4411B: 1.5 GHz
	ESA E4402B, E4403B: 3.0 GHz
	ESA E4404B: 6.7 GHz
	ESA E4405B: 13.2 GHz
	ESA E4407B, E4408B: 26.5 GHz
Range:	ESA E4401B, E4411B: 0 Hz, 100 Hz to 1.58 GHz
	ESA E4402B, E4403B: 0 Hz, 100 Hz to 3.10 GHz
	ESA E4404B: 0 Hz, 100 Hz to 6.78 GHz

	ESA E4405B: 0 Hz, 100 Hz to 13.3 GHz
	ESA E4407B, E4408B: 0 Hz, 100 Hz to 27.0 GHz
Default Unit:	Hz
Front Panel Access:	**SPAN/X Scale, Span**
	SPAN/X Scale, Zero Span

Full Frequency Span

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

Set the frequency span to full scale.

Factory Preset and *RST:	ESA E4401B, E4411B: 1.5 GHz
	ESA E4402B, E4403B: 3.0 GHz
	ESA E4404B: 6.7 GHz
	ESA E4405B: 13.2 GHz
	ESA E4407B, E4408B: 26.5 GHz
Front Panel Access:	**SPAN/X Scale, Full Span**

Previous Frequency Span

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

Set the frequency span to the previous span setting.

Front Panel Access:	**SPAN/X Scale, Last Span**

Start Frequency

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

`[:SENSe]:FREQuency:STARt?`

Set the start frequency.

Factory Preset and *RST:	0 Hz
Range:	ESA E4401B, E4411B: –80 MHz to 1.58 GHz
	ESA E4402B, E4403B: –80 MHz to 3.10 GHz
	ESA E4404B: –80 MHz to 6.78 GHz
	ESA E4405B: –80 MHz to 13.3 GHz
	ESA E4407B, E4408B: –80 MHz to 27.0 GHz

Default Unit:	Hz
Front Panel Access:	**FREQUENCY/Channel, Start Freq**

Stop Frequency

[:SENSe]:FREQuency:STOP <freq>

[:SENSe]:FREQuency:STOP?

Set the stop frequency.

Factory Preset and *RST:	ESA E4401B, E4411B: 1.5 GHz
	ESA E4402B, E4403B: 3.0 GHz
	AESA E4404B: 6.7 GHz
	ESA E4405B: 13.2 GHz
	ESA E4407B, E4408B: 26.5 GHz
Range:	ESA E4401B, E4411B: –80 MHz to 1.58 GHz
	ESA E4402B, E4403B: –80 MHz to 3.10 GHz
	ESA E4404B: –80 MHz to 6.78 GHz
	ESA E4405B: –80 MHz to 13.3 GHz
	ESA E4407B, E4408B: –80 MHz to 27.0 GHz
Default Unit:	Hz
Front Panel Access:	**FREQUENCY/Channel, Stop Freq**

[:SENSe]:HARMonics Subsection

Harmonic Measurement Averages

[:SENSe]:HARMonics:AVERage:COUNt <integer>

[:SENSe]:HARMonics:AVERage:COUNt?

Set the number of averages for the harmonic measurement.

Factory Preset and *RST:	10
Range:	1 to 1,000
Remarks:	This command specifies the number of sweep averages over which the amplitude of each harmonic and the total harmonic distortion will be calculated. Intermediate averaged results will be displayed.

Front Panel Access:	**Meas Setup, Avg Number On**

Turn On or Off Harmonic Measurement Averaging

`[:SENSe]:HARMonics:AVERage[:STATe] OFF|ON|0|1`

`[:SENSe]:HARMonics:AVERage[:STATe]?`

Turn on or off harmonic measurement averaging.

Factory Preset and *RST:	Off
Front Panel Access:	**Meas Setup, Avg Number**

Set Number of Measured Harmonics

`[:SENSe]:HARMonics:NUMBer <integer>`

`[:SENSe]:HARMonics:NUMBer?`

Set the number of harmonics to be measured.

Factory Preset and *RST:	10
Range:	2 to 10, limited by the maximum frequency range of the analyzer
Remarks:	This command sets the number of harmonics to measure before computing the total harmonic distortion. The minimum number is two (the fundamental and the second harmonic).
Front Panel Access:	**Meas Setup, Harmonics**

Set Harmonic Measurement Sweep Mode

`[:SENSe]:HARMonics:SWEeptime:AUTO OFF|ON|0|1`

`[:SENSe]:HARMonics:SWEeptime:AUTO?`

Set auto sweep time per harmonic to off or on.

Factory Preset and *RST:	On
Remarks:	See **`[:SENSe]:HARMonics:SWEeptime <time>`**
Front Panel Access:	**Meas Setup, ST/Harmonic**

Set Harmonic Measurement Sweep Time

`[:SENSe]:HARMonics:SWEeptime <time>`

[:SENSe]:HARMonics:SWEeptime?

Set the sweep time used for measuring each harmonic.

Factory Preset and *RST:	200/resolution bandwidth (RBW)
Range:	10 ms to the maximum sweep time of the analyzer
Default Unit:	seconds
Remarks:	When set to 1 (On), the sweep time for measuring each harmonic will be set to 200/RBW, where RBW is the resolution bandwidth setting in Hz. When set to 0 (Off), you can specify any sweep time within the limits of the analyzer, with a minimum of 10 ms. This sweep time is used only for the zero span measuring of the harmonics, and not during the search for the fundamental. In this case, the sweep time used will be the sweep time set before the measurement began.
Front Panel Access:	**Meas Setup, ST/Harmonic**

Turn On or Off Fundamental Zoom

[:SENSe]:HARMonics:ZOOM OFF|ON|0|1

[:SENSe]:HARMonics:ZOOM?

Turn on or off the fundamental signal zoom before the frequency count occurs.

Factory Preset and *RST:	On
Remarks:	When counter zoom is On, the measurement will zoom in on the fundamental after the original marker search and before the frequency count is executed. This ensures the accuracy of the frequency count, and is necessary when the starting resolution bandwidth is very small or the starting span is very wide.
Front Panel Access:	**Meas Setup, Counter Zoom**

[SENSe]:MIXer Subsection

Select External Mixer Band

[:SENSe]:MIXer:BAND K|A|Q|U|V|E|W|F|D|G|Y|J|USER

[:SENSe]:MIXer:BAND?

This command allows the selection of one of the pre-defined bands corresponding to the external mixer currently in use.

Factory Preset and *RST:	Band A (26.5-40 Ghz)
Remarks:	If the mixer harmonic configuration (**`[:SENSe]:MIXer:HARMonic:AUTO <boolean>`**) is set to manual, then a query will return "**`USER`**."

NOTE Bands K, E, W, F, D, G, Y, and J are not available if Mixer Type is set to Presel.

Front Panel Access:	**Input/Output (or Input), Input Mixer, Ext Mix Band**

External Mixer Bias Adjust

`[:SENSe]:MIXer:BIAS <numeric>`

`[:SENSe]:MIXer:BIAS?`

This command allows the adjustment of an internal bias source for use with external mixers.

Factory Preset and *RST:	0
Range:	–10 mA to 10 mA
Default Unit:	mA
Remarks:	The bias signal is present on the center conductor of the IF INPUT connector on the front panel. See related command **`[:SENSe]:MIXer:BIAS[:STATe] OFF|ON|0|1`**.

NOTE Mixer Bias will be Off if AUTO Harmonic and Presel Mixer Type is selected.

Front Panel Access:	**Input/Output (or Input), Input Mixer, Mixer Config, Mixer Bias On Off**

Set External Mixer Bias On/Off

`[:SENSe]:MIXer:BIAS:STATe <boolean>`

`[:SENSe]:MIXer:BIAS:STATe?`

This command activates an internal bias source for use with external mixers.

Factory Preset and *RST:	Off

Remarks:	The bias signal is present on the center conductor of the IF Input connector on the front panel. See related command **[:SENSe]:MIXer:BIAS <numeric>**.

NOTE Mixer Bias will be Off if AUTO Harmonic and Presel Mixer Type is selected.

Front Panel Access:	**Input/Output (or Input), Input Mixer, Mixer Config, Mixer Bias On Off**

Set External Mixer LO Harmonic Value

[:SENSe]:MIXer:HARMonic <value>

[:SENSe]:MIXer:HARMonic?

This command allows you to set the LO harmonic value for mixers other than the HP/Agilent 11970-Series or 11974-Series Mixers. This is done after the mixer harmonic configuration (**[:SENSe]:MIXer:HARMonic:AUTO <boolean>**) is set to manual.

Factory Preset and *RST:	–8 (Band A, 26.5-40 GHz)
Range:	Any non-zero integer from –50 to 50, inclusive
Remarks:	The harmonic value with its associated sign is automatically determined from the external mixer band selected. This is the AUTO mode. For mixers other than the HP/Agilent 11970-Series or 11974-Series, an LO harmonic other than that determined in the AUTO mode may be required. This is achieved using this command and related command **[:SENSe]:MIXer:HARMonic:AUTO <boolean>**.

Set External Mixer LO Harmonic Mode

[:SENSe]:MIXer:HARMonic:AUTO <boolean>

[:SENSe]:MIXer:HARMonic:AUTO?

This command allows you to set the external mixer LO harmonic mode to either automatic or manual. The manual mode is used with mixers other than the HP/Agilent 11970-Series or 11974-Series Mixers to manually choose the required LO harmonic.

Factory Preset and *RST:	AUTO

Remarks: The harmonic value with its associated sign is automatically determined from the external mixer band selected. This is the AUTO mode. For mixers other than the HP/Agilent 11970-Series or 11974-Series, an LO harmonic other than that determined in the AUTO mode may be required. This is achieved using this command and related command `[:SENSe]:MIXer:HARMonic <value>`.

NOTE Manually selecting a harmonic will remove any restrictions on Mixer Bias and Mixer Type, and External Mixer Band will become USER. Returning the Harmonic to AUTO will reset the band to A, set Mixer Type to UNPReselect, and turn Mixer Bias Off.

Front Panel Access: **Input/Output (or Input), Input Mixer, Mixer Config, Harmonic Auto Man**

[:SENSe]:OBWidth Subsection

Set OBW Number of Averages

`[:SENSe]:OBWidth:AVERage:COUNt <integer>`

`[:SENSe]:OBWidth:AVERage:COUNt?`

Set the occupied bandwidth measurement number of averages.

Remarks: This command is used for measurements in the **MEASURE** menu.

Front Panel Access: **Meas Setup, Avg Number**

OBW Averaging On/Off

`[:SENSe]:OBWidth:AVERage[:STATe] OFF|ON|0|1`

`[:SENSe]:OBWidth:AVERage[:STATe]?`

Turn on or off occupied bandwidth averaging.

Factory Preset and *RST: Off

Remarks: This command is used for measurements in the **MEASURE** menu.

Front Panel Access: **Meas Setup, Avg Number On Off**

Set OBW Span

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

[:SENSe]:OBWidth:FREQuency:SPAN?

Set the occupied bandwidth span. The analyzer span will retain this value throughout the measurement.

Remarks:	This command is used for measurements in the **MEASURE** menu.
Front Panel Access:	**Meas Setup, OBW Span**

Set OBW % Power

[:SENSe]:OBWidth:PERCent <percent>

[:SENSe]:OBWidth:PERCent?

Set the occupied bandwidth power in percent.

Remarks:	This command is used for measurements in the **MEASURE** menu.
Front Panel Access:	**Meas Setup, Occ BW % Pwr**

[:SENSe]:POWer Subsection

Input Attenuation

[:SENSe]:POWer[:RF]:ATTenuation <rel_ampl>

[:SENSe]:POWer[:RF]:ATTenuation?

Set the input attenuator. This value is set at its auto value if input attenuation is set to auto.

Factory Preset and *RST:	10 dB
Range:	ESA E4401B, E4411B: 0 to 60 dB
	ESA E4402B, E4403B: 0 to 75 dB
	ESA E4404B: 0 to 75 dB
	ESA E4405B: 0 to 75 dB
	ESA E4407B, E4408B: 0 to 65dB
Default Unit:	dB
Front Panel Access:	**AMPLITUDE/Y Scale, Attenuation Auto Man**

Input Port Attenuator Auto

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

[:SENSe]:POWer[:RF]:ATTenuation:AUTO?

Select the input port attenuator range to be set either automatically or manually.

On – Input attenuation is automatically set as determined by the Reference Level Setting.

Off – Input attenuation is manually set

Factory Preset and *RST:	On
Range:	ESA E4401B, E4411B: 0 to 60 dB
	ESA E4402B, E4403B: 0 to 75 dB
	AESA E4404B: 0 to 75 dB
	ESA E4405B: 0 to 75 dB
	ESA E4407B, E4408B: 0 to 65dB
Front Panel Access:	**Input/Output (or Input), Input Atten**

Input Port Power Gain

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

[:SENSe]:POWer[:RF]:GAIN[:STATe]?

Turns the internal preamp on or off.

Factory Preset and *RST:	Off
Front Panel Access:	**AMPLITUDE/Y Scale, Int Preamp On Off**

Input Port Maximum Mixer Power

[:SENSe]:POWer[:RF]:MIXer:RANGe[:UPPer] <ampl>

[:SENSe]:POWer[:RF]:MIXer:RANGe[:UPPer]?

Specifies the maximum power at the input mixer.

Factory Preset and *RST:	–10 dBm
Range:	–100 dBm to –10 dBm
Default Unit:	dBm
Front Panel Access:	**AMPLITUDE/Y Scale, Max Mixer Lvl**

Optimize Preselector Frequency

`[:SENSe]:POWer[:RF]:PADJust <freq>`

`[:SENSe]:POWer[:RF]:PADJust?`

This command allows user-defined adjustment of the preselector frequency to optimize its response on the signal of interest.

Factory Preset and *RST:	0 Hz
Range:	–250 MHz to 250 MHz
Default Unit:	None. Use the MHz terminator in order for this command to work.
Remarks:	This command is available only on Agilent ESA models E4404B, E4405B, E4407B, and E4408B. Use this command for signals close to the noise level, multiple signals close together, or for other conditions when the preselector is not tuned to the frequency of interest.
Front Panel Access:	**AMPLITUDE/Y Scale, Presel Adjust**

Preselector Center

`[:SENSe]:POWer[:RF]:PCENter`

In internal mixing, this command centers the preselector filter at the signal of interest. In external mixing, the external preselector filter is adjusted to the peak of the filter response to maximize the amplitude at the active marker frequency. This command has no effect if it is activated in non-preselected bands.

NOTE This command is available only on Agilent ESA models E4404B, E4405B, E4407B, and E4408B. This command has no effect with markers set to less than 3 GHz.

Remarks:	A peak search will be done if no marker is on.
Front Panel Access:	**AMPLITUDE/Y Scale, Presel Center**

[:SENSe]:SIDentify Subsection

Set Mixer Signal Identification Mode

`[:SENSe]:SIDentify:MODE ISUPpress|IShift`

`[:SENSe]:SIDentify:MODE?`

This command lets you choose one of two types of signal identification methods when viewing multiple responses from non-preselected external mixers.

Factory Preset and *RST: ISUPpress

Front Panel Access: **Input/Output (or Input), Input Mixer, Signal ID Mode, Image Suppress**

Input/Output (or Input), Input Mixer, Signal ID Mode, Image Shift

Set Mixer Signal Identification State

`[:SENSe]:SIDentify[:STATe] <boolean>`

`[:SENSe]:SIDentify[:STATe]?`

This command activates an algorithm which either removes or aids with the identification of multiple responses. These responses are generated from a single input signal using non-preselected external mixers.

Factory Preset and *RST: Off

Remarks: This function takes control of and uses Trace 1 and Trace 2. Any data in these traces prior to activating signal identification will be lost

S

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

To synchronize in image shift mode, turn off signal identification, initiate a single sweep, and turn on signal identification. The results of the first sweep after signal identification is turned on must be ignored. The data from the second sweep is available in Trace 1 and the data from the third (shifted) sweep is available in Trace 2.

NOTE Signal identification is not available with signal track, resolution bandwidths ≤300 Hz, demod, or averaging. Signal identification will be turned off when input mixer is set to internal.

Front Panel Access:	**Input/Output (or Input), Input Mixer, Signal Ident On Off**

[:SENSe]:SWEep Subsection

Sweep Time

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

`[:SENSe]:SWEep:TIME?`

Specifies the time in which the instrument sweeps the display.

Factory Preset and *RST:	ESA E4401B, E4411B: 4ms
	ESA E4402B, E4403B: 5 ms
	ESA E4404B: 16.75 ms
	ESA E4405B: 33 ms
	ESA E4407B, E4408B: 265 ms
Range:	The range depends upon the installed options, number of sweep points, and firmware revision of your instrument. See "*Sweep Time Range*" in the Specifications Guide for details.
Default Unit:	seconds
Remarks:	A span value of 0 Hz causes the analyzer to enter zero span mode. In zero span the X-axis represents time rather than frequency. In this mode, the sweep time may be set to faster values when Option AYX is installed.
Front Panel Access:	**Sweep, Sweep Time Auto Man**

Automatic Sweep Time

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

`[:SENSe]:SWEep:TIME:AUTO?`

Automatically selects the fastest sweep time for the current settings.

Factory Preset and *RST:	On
Front Panel Access:	**Sweep, Sweep Time Auto Man**

Sweep Time Mode

[:SENSe]:SWEep:TIME:AUTO:MODE SRESponse|SANalyzer

[:SENSe]:SWEep:TIME:AUTO:MODE?

Specifies the type of automatic coupling for the fastest sweep time at the current settings.

Stimulus response

Spectrum analyzer

Factory Preset and *RST:	SANalyzer
Front Panel Access:	**Sweep, Sweep Coupling SR SA**

Time Gating Delay

[:SENSe]:SWEep:TIME:GATE:DELay <time>

[:SENSe]:SWEep:TIME:GATE:DELay?

Sets the delay time from when the gate trigger occurs to when the gate opens. This is for EDGE triggering only.

Factory Preset and *RST:	1 µs
Range:	0.3 µs to 429 seconds
Default Unit:	seconds
Front Panel Access:	**Sweep, Gate Setup, Edge Setup, Gate Delay**

Time Gate Length

[:SENSe]:SWEep:TIME:GATE:LENGth <time>

[:SENSe]:SWEep:TIME:GATE:LENGth?

Specifies the gate time length in seconds. For EDGE triggering only.

Factory Preset and *RST:	1 µs
Range:	0.3 µs to 429 seconds
Default Unit:	seconds
Front Panel Access:	**Sweep, Gate Setup, Edge Setup, Gate Length**

Time Gate Level

[:SENSe]:SWEep:TIME:GATE:LEVel HIGH|LOW

[:SENSe]:SWEep:TIME:GATE:LEVel?

Selects the level of the gate signal. This command is for LEVel triggering only.

Factory Preset and *RST:	High
Front Panel Access:	**Sweep, Gate Setup, Level Setup**

Time Gate Polarity

[:SENSe]:SWEep:TIME:GATE:POLarity NEGative|POSitive

[:SENSe]:SWEep:TIME:GATE:POLarity?

Selects the polarity of the gate signal. This command is for EDGE triggering only.

Factory Preset and *RST:	Positive
Front Panel Access:	**Sweep, Gate Setup, Edge Setup, Edge Pos Neg**

Preset Time Gate

[:SENSe]:SWEep:TIME:GATE:PRESet

Presets the time-gated spectrum analysis capability.

Remarks: This command resets gate parameters to default values, as follows:

Gate trigger type = edge

Gate polarity = positive

Gate delay = 1 μs

Gate length = 1 μs

Gate level = high

Control Time Gate

[:SENSe]:SWEep:TIME:GATE[:STATe] OFF|ON|0|1

[:SENSe]:SWEep:TIME:GATE[:STATe]?

Turns time gating on or off.

NOTE Time gate cannot be turned on if external trigger delay is on.

Factory Preset and *RST:	Off

Front Panel Access:	**Sweep, Gate On Off**

Time Gate Trigger Type

[:SENSe]:SWEep:TIME:GATE:TYPE LEVel|EDGE

[:SENSe]:SWEep:TIME:GATE:TYPE?

Selects between edge and level mode for time-gated spectrum analysis.

Level triggers the gate when the signal surpasses a specific level, set to either low or high.

Edge triggers the gate when the edge of a signal is encountered, set to either a negative-going edge or a positive-going edge.

Factory Preset and *RST:	Edge
Front Panel Access:	**Sweep, Gate Setup, Trig Type Edge Level**

SOURce Subsystem

The SOURce subsystem controls the signal characteristics of the tracking generator. Refer to the "OUTPut Subsystem" on page 5-73, which also contains commands that control the characteristics of the tracking generator.

Sets the Output Power Offset Correction

:SOURce:CORRection:OFFSet <rel_ampl>

:SOURce:CORRection:OFFSet?

Specifies an offset for the displayed output power level. An offset power level can be added to the displayed level to compensate for system losses (for example, cable loss) or gains (for example, preamplifier gain.) This offset does not change the power out of the source, it only changes the display so that it reads out the actual power delivered to the device under test.

Factory Preset and *RST:	0 dBm
Range:	–327.6 dBm to 327.6 dBm
Default Unit:	Currently selected source power units
Front Panel Access:	**Source, Amptd Offset**

Source Attenuation

:SOURce:POWer:ATTenuation <ampl>

:SOURce:POWer:ATTenuation?

Attenuates the source output level.

Factory Preset and *RST:	ESA E4401B, E4411B: 0 dB
	ESA E4402B, E4403B: 8 dB
	ESA E4404B: 8 dB
	ESA E4405B: 8 dB
	ESA E4407B, E4408B: 8 dB
Range:	ESA E4401B, E4411B: 0 dB to 60 dB in 10 dB steps
	ESA E4402B, E4403B: 0 dB to 56 dB in 8 dB steps
	ESA E4404B: 0 dB to 56 dB in 8 dB steps

	ESA E4405B: 0 dB to 56 dB in 8 dB steps
	ESA E4407B, E4408B: 0 dB to 56 dB in 8 dB steps
Default Unit:	dB
Front Panel Access:	**Source, Attenuation Auto Man**

Automatic Source Attenuation

:SOURce:POWer:ATTenuation:AUTO OFF|ON|0|1

:SOURce:POWer:ATTenuation:AUTO?

Selects if the source output level attenuator will be set automatically.

Factory Preset and *RST:	On
Front Panel Access:	**Source, Attenuation Auto Man**

Sets the Output Power

:SOURce:POWer[:LEVel][:IMMediate][:AMPLitude] <ampl>

:SOURce:POWer[:LEVel][:IMMediate][:AMPLitude]?

Specifies the source output power level. Use **:SOURce:POWer:SWEep** to set the change in power level across the sweep. Also see **:SOURce:POWer:STARt** and **OUTPut[:STATe]**.

Factory Preset and *RST:	–10 dBm
Range:	ESA E4401B, E4411B: –70 dBm to 3 dBm
	ESA E4402B, E4403B: –66 dBm to 3 dBm
	ESA E4404B: –66 dBm to 3 dBm
	ESA E4405B: –66 dBm to 3 dBm
	ESA E4407B, E4408B: –66 dBm to 3 dBm
Default Unit:	dBm
Front Panel Access:	**Source, Amplitude On Off**

Sets the Source Output Power Mode

:SOURce:POWer:MODE FIXed|SWEep

:SOURce:POWer:MODE?

Sets the source output to be at a single amplitude (fixed) or to sweep through a range of power levels.

Factory Preset and *RST:	Fixed
Front Panel Access:	**Source, Power Sweep On Off**

Set the Source Sweep Power Range

:SOURce:POWer:SPAN <rel_ampl>

:SOURce:POWer:SPAN?

Specifies the range of power levels through which the source output will sweep. Use **:SOURce:POWer:STARt** to set the power level at the start of the power sweep. This command is equivalent to **:SOURce:POWer:SWEep.**

Factory Preset and *RST:	0 dB
Range:	0 dB to 20 dB
Default Unit:	dB
Front Panel Access:	**none**

Set the Output Power at the Start of the Sweep

:SOURce:POWer:STARt <ampl>

:SOURce:POWer:STARt?

Specifies the source output power level at the start of the power sweep. Use **:SOURce:POWer:SPAN** to set the change in power level across the sweep. This command is equivalent to **:SOURce:POWer[:LEVel][:IMMediate][:AMPLitude]**.

Set the Output Power to Step Automatically

:SOURce:POWer:STEP:AUTO OFF|ON|0|1

:SOURce:POWer:STEP:AUTO?

Specifies the source power step size to be one vertical scale division.

Factory Preset and *RST:	On
Front Panel Access:	**Source, Amptd Step Auto Man**

Set the Output Power Step Size

:SOURce:POWer:STEP[:INCRement] <ampl>

:SOURce:POWer:STEP[:INCRement]?

Specifies the source power step size.

Default Unit:	dBm
Front Panel Access:	**Source, Amptd Step Auto Man**

Set the Source Sweep Power Range

:SOURce:POWer:SWEep <rel_ampl>

:SOURce:POWer:SWEep?

Specifies the range of power levels through which the source output will sweep. Use **:SOURce:POWer:STARt** to set the power level at the start of the power sweep. See also **:SOURce:POWer:SPAN**.

Factory Preset and *RST:	0 dB
Range:	0 dB to 20 dB
Default Unit:	dB
Front Panel Access:	**Source, Power Sweep On Off**

Output Power Tracking

:SOURce:POWer:TRCKing <number>

:SOURce:POWer:TRCKing?

Adjusts the tracking of the source output with the spectrum analyzer sweep in the present resolution bandwidth.

Factory Preset and *RST:	ESA E4401B, E4411B: none
	ESA E4402B, E4403B: 1800
	ESA E4404B: 2048
	ESA E4405B: 4000
	ESA E4407B, E4408B: 2378
Range:	Real, 0 to 4095
Remarks:	This command is not applicable for the 1.5 GHz tracking generator.

Front Panel Access:	**Source, Man Track Adj**

Output Power Tracking Peak

:SOURce:POWer:TRCKing:PEAK

Automatically adjusts the tracking of the source output with the spectrum analyzer sweep so that the power is maximized for the present resolution bandwidth.

Factory Preset and *RST:	none
Remarks:	This command is not applicable for the 1.5 GHz tracking generator.
Front Panel Access:	**Source, Tracking Peak**

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>|<non-decimal numeric>

:STATus:OPERation:ENABle?

This command determines which 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 Preset sets all bits in this enable register to 0. To have any Operation Events reported to the Status Byte Register, 1 or more bits must be set to 1.

Factory Preset and *RST:	0
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>|<non-decimal numeric>

:STATus:OPERation:NTRansition?

This command determines which 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
Range:	0 to 32767

Operation Positive Transition

:STATus:OPERation:PTRansition <number>|<non-decimal numeric>

:STATus:OPERation:PTRansition?

This command determines which 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)
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 ESE, and SRE Registers as described in IEEE Standard 488.2-1992, *IEEE Standard Codes, Formats, Protocols and Common Commands for Use with ANSI/IEEE Std 488.1-1987*. New York, NY, 1992.

STATus:QUEStionable Subsection

This subsection controls the SCPI-defined status-reporting structures.

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 which 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)
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 which 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
Range:	0 to 32767

Questionable Calibration Positive Transition

:STATus:QUEStionable:CALibration:PTRansition <number>

:STATus:QUEStionable:CALibration:PTRansition?

This command determines which bits in the Questionable Calibration Condition register will set the corresponding bit in the Questionable Calibration 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)
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 which 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. The Status Byte Event Register should be queried after each measurement to check the Questionable Status Summary (bit 3). If it is equal to 1, a condition during the test made the test results invalid. If it is equal to 0, this indicates that no hardware problem or measurement problem was detected by the analyzer.

Factory Preset and *RST:	0
Range:	0 to 32767

Questionable Event Query

:STATus:QUEStionable[:EVENt]?

This query returns the decimal value of the sum of the bits in the Questionable Event register.

NOTE The register requires that the 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 which 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)
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 which 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
Range:	0 to 32767

Questionable Frequency Positive Transition

:STATus:QUEStionable:FREQuency:PTRansition <number>

:STATus:QUEStionable:FREQuency:PTRansition?

This command determines which bits in the Questionable Frequency Condition register will set the corresponding bit in the Questionable Frequency 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)
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 which 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)
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 which 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
Range:	0 to 32767

Questionable Integrity Positive Transition

:STATus:QUEStionable:INTegrity:PTRansition <number>

:STATus:QUEStionable:INTegrity:PTRansition?

This command determines which bits in the Questionable Integrity Condition register will set the corresponding bit in the Questionable Integrity 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)
Range:	0 to 32767

Questionable Integrity Uncalibrated Enable

:STATus:QUEStionable:INTegrity:UNCalibrated:ENABle

:STATus:QUEStionable:INTegrity:UNCalibrated:ENABle?

This command determines which bits in the Questionable Integrity Uncalibrated Condition Register will set bits in the Questionable Integrity Uncalibrated Event register, which also sets the Data Uncalibrated Summary bit (bit 3) in the Questionable Integrity Register. The variable <number> is the sum of the decimal values of the bits you want to enable.

Factory Preset and *RST:	32767 (all 1's)
Range:	0 to 32767

Questionable Integrity Uncalibrated Event Query

:STATus:QUEStionable:INTegrity:UNCalibrated[:EVENt]?

This query returns the decimal value of the sum of the bits in the Questionable Integrity Uncalibrated Event register.

NOTE The register requires that the 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 Uncalibrated Negative Transition

:STATus:QUEStionable:INTegrity:UNCalibrated:NTRansition <number>

:STATus:QUEStionable:INTegrity:UNCalibrated:NTRansition?

This command determines which bits in the Questionable Integrity Uncalibrated Condition register will set the corresponding bit in the Questionable Integrity Uncalibrated Event register when 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

Range:	0 to 32767

Questionable Integrity Uncalibrated Positive Transition

:STATus:QUEStionable:INTegrity:UNCalibrated:PTRansition <number>

:STATus:QUEStionable:INTegrity:UNCalibrated:PTRansition?

This command determines which bits in the Questionable Integrity Uncalibrated Condition register will set the corresponding bit in the Questionable Integrity Uncalibrated Event register when 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)
Range:	0 to 32767

Questionable Negative Transition

:STATus:QUEStionable:NTRansition <number>

:STATus:QUEStionable:NTRansition?

This command determines which 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
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 which 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)
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 which 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
Range:	0 to 32767

Questionable Power Positive Transition

:STATus:QUEStionable:POWer:PTRansition <number>

:STATus:QUEStionable:POWer:PTRansition?

This command determines which bits in the Questionable Power Condition register will set the corresponding bit in the Questionable Power 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)
Range:	0 to 32767

Questionable Positive Transition

`:STATus:QUEStionable:PTRansition <number>`

`:STATus:QUEStionable:PTRansition?`

This command determines which bits in the Questionable Condition register will set the corresponding bit in the Questionable 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)
Range:	0 to 32767

SWEEp Subsystem

This subsystem is used to set the controls and parameters associated with the overall system sweep functions.

Sweep Points

`:SWEEp:POINts <number of points>`

`:SWEEp:POINts?`

This command sets the number of sweep points.

Factory Preset and *RST:	401
Example:	**`:SWE:POIN 401`**
Range:	101 to 8192
Remarks:	This command is only available on analyzers with firmware revision A.04.00 and later. The number of sweep points can be set only with Agilent ESA models E4401B, E4402B, E4404B, E4405B, and E4407B. The number of sweep points may be queried on all anlayzers with firmware revision A.04.00 and later.

Whenever the number of sweep points change, the following functions are affected:

- All trace data is erased
- Any traces in view mode will go to blank mode
- Sweep time is re-calculated
- Any limit lines that are on will be turned off

Front Panel Access:	**Sweep, Points**

SYSTem Subsystem

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

GPIB Address

`:SYSTem:COMMunicate:GPIB[1]|2|3|4[:SELF]:ADDRess <integer>`

`:SYSTem:COMMunicate:GPIB[1]|2|3|4[:SELF]:ADDRess?`

Sets and queries the GPIB address.

NOTE This command applies only to analyzers having Option A4H, HP-IB and Parallel I/O).

Factory Preset and *RST:	It is set to 18 by **`:SYSTem:PRESet:PERSistent`**, which sets the persistent state values to their factory defaults. This command is persistent. The term persistent means that the command retains the setting previously selected, even through a power cycle.
Range:	Integer, 0 to 30
Front Panel Access:	**System, Remote Port**

Serial Port DTR Setup

`:SYSTem:COMMunicate:SERial[1]|2|3|4|5|6|7|8:CONTrol:DTR OFF|ON|IBFull`

`:SYSTem:COMMunicate:SERial[1]|2|3|4|5|6|7|8:CONTrol:DTR?`

Sets the hardware pacing scheme. It controls the number of bytes in the input buffer.

Off - holds the DTR line in the unasserted (off) condition

On - holds the DTR line in the asserted (on) condition

IBFull - selects the input buffer full mode for the DTR line. The IBFull parameter sets the DTR line to indicate when the device is ready to receive. When the number of received bytes in the input buffer of the device reaches the stop threshold, the device will unassert the DTR line. When the number of bytes has been reduced to the start threshold, the device will assert DTR indicating that it can receive input again. The device will also monitor the state of CTS and will stop transmission if the line becomes unasserted.

There can potentially be four I/O cards installed (1, 2, 3, or 4). Each card may have up to two ports:

Card one uses port numbers 1 and 2

Card two uses port numbers 3 and 4

Card three uses port numbers 5 and 6

Card four uses port numbers 7 and 8

Factory Preset (no *RST):	The factory default is On. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.
Front Panel Access:	**none**

Serial Port RTS Setup

:SYSTem:COMMunicate:SERial[1]|2|3|4|5|6|7|8:CONTrol:RTS OFF|ON|IBFull

:SYSTem:COMMunicate:SERial[1]|2|3|4|5|6|7|8:CONTrol:RTS?

Sets the hardware pacing (hand-shaking) scheme. Many high speed asynchronous modems use this line (paired with CTS) as receive/transmit pacing.

Off - indicates that the RTS line should always be asserted

On - indicates that the RTS line should always be unasserted

IBFull - selects the input buffer full mode for the RTS line. IBFull sets the RTS line to indicate when the device is ready to receive. When the number of received bytes in the input buffer of the device reaches the stop threshold, the device will unassert the RTS line. When the number of bytes has been reduced to the start threshold, the device will assert RTS indicating that it can receive input again. RTS is sometimes called RFR (ready for receiving). The device will also monitor the state of CTS and will stop transmission if that line becomes unasserted.

If no optional serial port number is specified, port 1 is assumed. There can potentially be four I/O cards installed (1, 2, 3, or 4). Each card may have up to two ports:

Card one uses port numbers 1 and 2

Card two uses port numbers 3 and 4

Card three uses port numbers 5 and 6

Card four uses port numbers 7 and 8

Factory Preset (no *RST):	The factory default is IBFull. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.
Front Panel Access:	**none**

Serial Port Baud Rate Setup

:SYSTem:COMMunicate:SERial[1]|2|3|4|5|6|7|8[:RECeive]:BAUD <baud_rate>

:SYSTem:COMMunicate:SERial[1]|2|3|4|5|6|7|8[:RECeive]:BAUD ?

There can potentially be four I/O cards installed (1, 2, 3, or 4). Each card may have up two ports:

Card one uses port numbers 1 and 2

Card two uses port numbers 3 and 4

Card three uses port numbers 5 and 6

Card four uses port numbers 7 and 8

Factory Preset (no *RST):	The factory default is 9600. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.
Range:	Supported baud rates are 110 | 300 | 600 | 1200 | 2400 | 4800 | 9600 | 19200 | 38400 | 57600 | 115200
Front Panel Access:	**System, Remote Port**

Serial Port Receive Pace Setup

:SYSTem:COMMunicate:SERial[1]|2|3|4|5|6|7|8[:RECeive]:PACE XON|NONE

:SYSTem:COMMunicate:SERial[1]|2|3|4|5|6|7|8[:RECeive]:PACE ?

Set the receive pace to on or none for an instrument, with the RS-232 interface installed. If no optional serial port number is specified, port 1 is assumed.

There can potentially be four I/O cards installed (1, 2, 3, or 4). Each card may have up to two ports:

Card one uses port numbers 1 and 2

Card two uses port numbers 3 and 4

Card three uses port numbers 5 and 6

Card four uses port numbers 7 and 8

Factory Preset (no *RST):	The factory default is none. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.
Front Panel Access:	**none**

Serial Port Transmit Pace Setup

:SYSTem:COMMunicate:SERial[1]|2|3|4|5|6|7|8:TRANsmit:PACE XON|NONE

:SYSTem:COMMunicate:SERial[1]|2|3|4|5|6|7|8:TRANsmit:PACE?

Set the transmit pace to on or none for an instrument, with the RS-232 interface installed. If no optional serial port number is specified, port 1 is assumed.

There can potentially be four I/O cards installed (1, 2, 3, or 4). Each card may have up to two ports:

Card one uses port numbers 1 and 2

Card two uses port numbers 3 and 4

Card three uses port numbers 5 and 6

Card four uses port numbers 7 and 8

Factory Preset (no *RST):	The factory default is none. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.
Front Panel Access:	**none**

Hardware Configuration Query

:SYSTem:CONFigure:HARDware?

Returns string of information about the current hardware in the instrument.

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

Display the Hardware Configuration

:SYSTem:CONFigure:HARDware:STATe OFF|ON|0|1

:SYSTem:CONFigure:HARDware:STATe?

Shows the current hardware configuration of the instrument on the display.

Factory Preset and *RST:	Off
Front Panel Access:	**System, Show Hdwr**

System Configuration Query

:SYSTem:CONFigure[:SYSTem]?

Returns string of information about the configurations of the instrument.

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

Display System Configuration

:SYSTem:CONFigure[:SYSTem]:STATe OFF|ON|0|1

:SYSTem:CONFigure[:SYSTem]:STATe?

Shows the current system configuration of the instrument on the display.

Factory Preset and *RST:	Off
Front Panel Access:	**System, Show System**

Set Date

:SYSTem:DATE <year>,<month>,<day>

:SYSTem:DATE?

Sets the date of the real-time clock of the instrument.

Year is a 4-digit integer

Month is an integer 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**

Host Identification Query

:SYSTem:HID?

This command returns a string that contains the host identification. This ID is required in order to obtain the license key that enables a new application or option.

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

License Key – Install Application/Option

:SYSTem:LKEY <“option”>, <“license key”>

:SYSTem:LKEY? <“option”>

This command enters the license key required for installing the specified new application or option. The query returns a string that contains the license key for a specified application or option that is already installed in the instrument. The license key will also be returned if the application is not currently in memory, but had been installed at some previous time.

Example:	**:SYST:LKEY “BAC”, “123A456B789C”**
Remarks:	An option is a three character string that specifies the option or application that is to be installed, as found in the Ordering Guide (for example, BAH for GSM Measurement Personality). The option name must be enclosed in quotes. A license key is a 12-character hexadecimal string given with the option. The license key is unique to a specific option installed in the instrument with a specific host ID, as returned by **:SYST:HID?**. The license key must be enclosed in quotes.
Front Panel Access:	**System, Licensing**

Delete a License Key

`:SYSTem:LKEY:DELete <“option”>`

This command allows you to delete the license key from instrument memory for the selected option.

NOTE In general, deleting the license key number is not recommended. 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 works with one particular instrument host ID only.

Example: **`:SYST:LKEY:DEL “BAC”`**

Remarks: In the example above, BAC is the part of the command that describes the option. An option is a three-character string that specifies the option or application to be installed as found in the Ordering Guide (for example, BAH for GSM Measurement Personality). The option name must be enclosed in quotes.

Query Instrument Options

`:SYSTem:OPTions?`

Returns a list of the options that are installed.

It is a comma separated list such as: “1DS,1D6,A4H,A4J,1DN”

Power On Elapsed Time

`:SYSTem:PON:ETIMe?`

Returns the number of seconds that have elapsed since the instrument was turned on for the very first time.

Remarks:

Front Panel Access: **none**

Power On Time

`:SYSTem:PON:TIME?`

Returns the elapsed time since the analyzer was last turned on.

Front Panel Access: **none**

Power On Type

`:SYSTem:PON:TYPE PRESet|LAST`

`:SYSTem:PON:TYPE?`

Sets the defined instrument conditions after a power-on or **Preset**.

PRESet - The instrument settings at power-on will be either the factory preset or user preset, as set by **`:SYSTem:PRESet:TYPE FACTory|USER.`**

LAST - The instrument settings at power-on will be the settings at the time of power down.

Factory Preset and *RST:	The factory default is Preset. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.
Front Panel Access:	**System, Power On/Preset, Power On Last Preset**

Enable IF/Video/Sweep Output Ports

`:SYSTem:PORTs:IFVSweep:ENABle OFF|ON|0|1`

`:SYSTem:PORTs:IFVSweep:ENABle?`

This command enables or disables the IF, video, and sweep output ports for Agilent ESA analyzers having options A4J (IF, Sweep, and Video Ports) and AYX (Fast Time Domain Sweeps).

Factory Preset and *RST:	On
Example:	**`:SYST:PORT:IFVS:ENAB ON`**
Range:	On/Off
Remarks:	Disable the output ports for faster measurement times.

Preset

`:SYSTem:PRESet`

Returns the instrument to a set of defined conditions. The particular set is selected by **`:SYSTem:PRESet:TYPE`**. This command does not change any persistent parameters. The term persistent means that the command retains the setting previously selected, even through a power cycle.

Front Panel Access:	**Preset**

Persistent State Reset

`:SYSTem:PRESet:PERSistent`

Sets the persistent state values to their factory defaults. The term persistent means that the command retains the setting previously selected, even through a power cycle. Examples of persistent functions are: GPIB address, power-on type, and preset type.

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

Save User Preset

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

Saves the current instrument conditions as the *user preset* condition.

Front Panel Access:	**System, Power On/Preset, Save User Preset**

Preset Type

`:SYSTem:PRESet:TYPE FACTory|USER`

Selects the preset state to be either factory-defined or user-defined preset conditions.

Factory Preset and *RST:	The factory default is **`FACTory`**. This parameter is persistent, which means that it retains the setting previously selected, even through a power cycle.
Remarks:	**`:SYSTem:PRESet:USER:SAVE`** defines the *user preset*.
Front Panel Access:	**System, Power On/Preset, Preset Factory User**

Speaker Control

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

`:SYSTem:SPEaker[:STATe]?`

Turns the internal speaker on or off.

Factory Preset and *RST:	Off
Front Panel Access:	**Det/Demod, Speaker On Off**

Set Time

:SYSTem:TIME <hour>,<minute>,<second>

: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**

SCPI Version Query

:SYSTem:VERSion?

Returns the SCPI version number with which the instrument complies.

TRACe Subsystem

The TRACe subsystem controls access to the internal trace memory of the analyzer.

NOTE Refer also to **:CALCulate** and **:MMEMory** subsystems for more trace and limit line commands.

Copy Trace

:TRACe:COPY <source_trace>,<dest_trace>

Transfers the source trace to the destination trace and leaves the destination trace in VIEW mode.

Source traces are: TRACE1|2|3, LLINE1, LLINE2, and RAWTRACE

Destination traces are: TRACE1|2|3, LLINE1, LLINE2, and RAWTRACE

Example: **:TRAC:COPY TRACE2,TRACE1**

Front Panel Access: **View/Trace, Operations, 1 –> 3**

View/Trace, Operations, 2 –> 3

Transfer Trace Data

:TRACe[:DATA] <trace_name>|RAWTRACE,<definite_length_block>|<comma_separated_ASCII_data>

:TRACe[:DATA]? <trace_name> |RAWTRACE|LLINE1|LLINE2

This command transfers trace data from the controller to the instrument. The data format is set by the command **:FORMat [:TRACe][:DATA]**. The data is comma-separated ASCii values in ASCii formatting, and a definite length block in REAL, INTeger, and UINTeger formatting.

The query returns the current values of the designated trace. The data is terminated with <NL><END> (for GPIB that is newline, or linefeed, followed by EOI set true; for RS-232 this is newline only.)

LLINE1 and **LLINE2** can only be queried; they cannot be set.

<trace_name> is **TRACE1|2|3**

NOTE This command does not allow setting all trace points to the same amplitude value by sending just a single value. If you need to set all trace points to the same value, you must send all 401 values.

Rawtrace data is available with **UINT,16** or **INT,32** formatting. It is unitless, returns uncorrected ADC values, and is the fastest method of obtaining measurement data.

Example: **:TRAC:DATA TRACE1,#41604<binary trace data><LF-EOI>**

Remarks: Commands **:MMEM:STOR:TRAC** and **:MMEM:LOAD:TRAC** are used to transfer trace data to, or from, the internal hard drive or floppy drive of the instrument.

There are 401 points in a trace. The trace data format is determined by **:FORMat[:TRACe][:DATA]**, and the binary data byte order is determined by **:FORMat:BORDer**.

If the parameter to the query is **LLINE1** or **LLINE2**, a very large positive or negative value is returned at any point outside the range of limit values. A large positive number is returned for an upper limit, and a large negative value for lower limits. There is no SCPI short form for parameters **LLINE1|LLINE2.**

Exchange Traces

:TRACe:EXCHange <trace_1>,<trace_2>

Exchanges 2 traces, point by point and leaves both in VIEW mode.

Trace_1 choices are: TRACE1|2|3

Trace_2 choices are: TRACE1|2|3

Example: **:TRAC:EXCH TRACE3,TRACE2**

Front Panel Access: **View/Trace, Operations, 1 <–> 3**

View/Trace, Operations, 2 <–> 3

Trace Math Add

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

Adds the magnitudes of the two source traces together and places the result in the destination trace.

Destination traces are: TRACE1|2|3, LLINE1, LLINE2, and RAWTRACE

Source traces are: TRACE1|2|3, LLINE1, LLINE2, and RAWTRACE

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

is equivalent to : (trace 2 = trace 1 + trace 3)

Mean Trace Data

`:TRACe:MATH:MEAN? <trace>`

Returns the mean of the amplitudes of the trace amplitude elements in measurement units.

Traces are: TRACE1|2|3, LLINE1, LLINE2, and RAWTRACE

Query the Signal Peaks

`:TRACe:MATH:PEAK[:DATA]?`

Outputs the signal peaks by frequency or by amplitude. The sort mode is determined by the command **`:TRACe:MATH:PEAK:SORT`**. The commands **`:CALCulate:MARKer:PEAK:EXCursion`** and **`:CALCulate:MARKer:PEAK:THReshold`** are used to determine what is a signal peak. To get the number of signals found meeting the specified limits, use the query **`:TRACe:MATH:PEAK:POINts?`**

Query Number of Peaks Found

`:TRACe:MATH:PEAK:POINts?`

Outputs the number of signal peaks identified. The amplitude of the peaks can then be queried with **`:TRACe:MATH:PEAK:DATA?`**

Peak Sorting

`:TRACe:MATH:PEAK:SORT AMPLitude|FREQuency`

`:TRACe:MATH:PEAK:SORT?`

Determines if the signals in the **`:TRACe:MATH:PEAK:DATA?`** query are sorted by frequency or amplitude.

Amplitude sorts the identified peaks by descending amplitude.

Frequency sorts the identified peaks by increasing frequency.

Smooth Trace Data

:TRACe:MATH:SMOoth <trace>

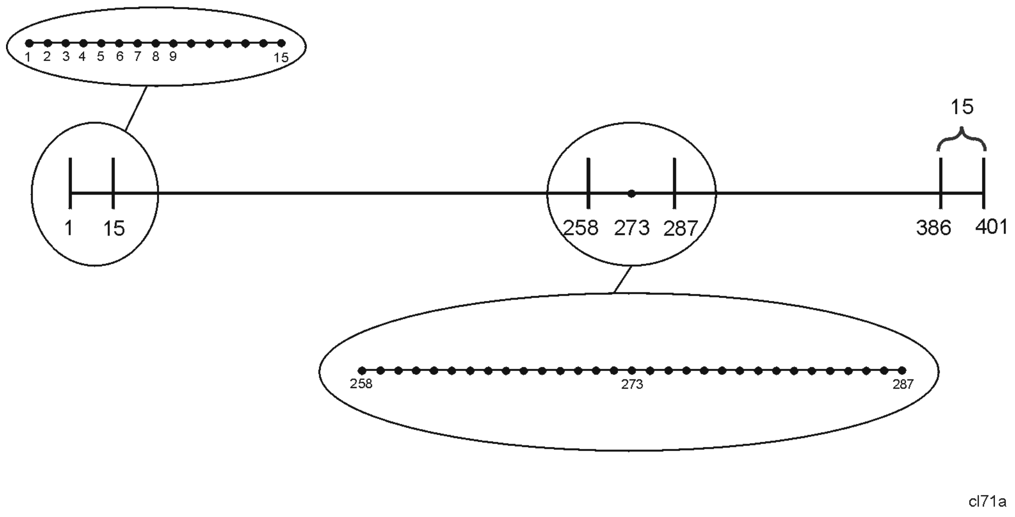
Smooths the trace according to the number of points specified in **:TRACe:MATH:SMOoth:POINts**. There is no equivalent front panel function.

Remarks: Traces are: TRACE1|2|3, and RAWTRACE commands.

The purpose of this function is to perform a spatial video averaging as compared to the temporal version supplied by the video-average command **[:SENSe]:AVERage:TYPE LPOWer**. The functions of **:TRACe:MATH:SMOoth <trace>** and **[:SENSe]:AVERage:TYPE LPOWer** are not interchangeable.

Each point value is replaced with the average of the values of the selected number of points, with half of those points located on each side of any particular point (when possible). Refer to Figure 5-1. This figure illustrates a 401 point trace with a smoothing number of 31. Think of the trace points as “buckets” of data. To smooth (arbitrary) point 273, the analyzer averages buckets 258 through 287 and applies that value to point 273.

Figure 5-1 Smoothing With 401 Trace Points and 31 Smoothing Points



Increasing the number of points increases smoothing at the cost of decreasing resolution.

The amount of smoothing decreases at the endpoints. Because **`:TRACe:MATH:SMOoth <trace>`** averages values that occur before and after the data point in time, display irregularities can be caused at the start and stop frequencies. To avoid possible irregularities (signal distortion) at the ends of the trace, use small values for the smooth parameter.

Refer to Figure 5-1 for a discussion of this end-point smoothing phenomena. With 31 smoothing points and a 401 point trace, point 16 will be the first point to have full 31-bucket smoothing. Likewise, point 385 will be the last point with full 31-bucket smoothing. Under the conditions stated, points 2 through 15 will be smoothed as follows: Point 2 is derived from averaging buckets 1 through 3. Point 3 is derived from averaging buckets 1 through 5, Point 4 is derived from averaging buckets 1 through 7, and so forth until point 16 is reached. The quantity of buckets used for the smoothing running average increases at the rate of 2 buckets per point, from point 1 to point ([smoothing number/2] + 1), at which time the full number of smoothing points is utilized. The same characteristic occurs at the end of the trace, beginning at point 386, when the number of averaging buckets begins to decrease until point 401 is reached.

By replacing the value of each point in a trace with the average of the values of a number of points centered about that point, any rapid variations in noise or signals are smoothed into more gradual variations. It thereby performs a function similar to reducing the video bandwidth without the corresponding changes in sweep time; as such, frequency resolution is decreased. Also, signal peaks are reduced with large smoothing values; and this can cause the amplitude to appear to be less than its actual value.

Number of Points for Smoothing

`:TRACe:MATH:SMOoth:POINts <integer>`

`:TRACe:MATH:SMOoth:POINts?`

Specifies the number of points that will be smoothed in **`:TRACe:MATH:SMOoth`**. See that command for an explanation of how smoothing is performed.

Increasing the number of points increases smoothing at the cost of decreasing resolution. If the number of points is an even number, then the number of points is increased by one. If the number of points is larger than the number of sweep points, then the number of sweep points is used, unless the number of sweep points is even, in which case the number of points will be the sweep points minus one. The number of points smoothed is always an odd number.

Range:	Integer, 101 to current number of sweep points

Trace Math Subtract

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

Subtracts the magnitude of the two source traces (trace 1 – trace 2) and places the result in the destination trace.

Destination traces are: TRACE1|2|3, LLINE1, LLINE2, and RAWTRACE

Source traces are: TRACE1|2|3, LLINE1, LLINE2, and RAWTRACE

Example: **:TRAC:MATH:SUBT TRACE3,TRACE3,TRACE2**

is equivalent to: (trace 3 = trace 3 – trace 2)

Trace Math Subtract From Display Line

:TRACe:MATH:SUBTract:DLINe <trace>

Subtracts the magnitude of the display line from the selected trace and places the result back in the selected trace.

Trace is: TRACE1|2|3, LLINE1, LLINE2, and RAWTRACE

Example: **:TRAC:MATH:SUBT:DLIN TRACE1**

is equivalent to: (trace1 = trace 1 – display line)

Front Panel Access: **View/Trace, Operations, 2 – DL –> 2**

Select Trace Display Mode

:TRACe1|2|3:MODE WRITe|MAXHold|MINHold|VIEW|BLANk

:TRACe1|2|3:MODE?

Selects the display mode for the selected trace.

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

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

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

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

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

Front Panel Access:	**View/Trace, Clear Write**
	View/Trace, Max Hold
	View/Trace, Min Hold
	View/Trace, View
	View/Trace, Blank

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.

External Trigger, Line and TV Trigger Delay Value

`:TRIGger[:SEQuence]:DELay <delay>`

`:TRIGger[:SEQuence]:DELay?`

This command sets the amount of trigger delay when using the rear panel external trigger input, the front panel input with TV trigger, or the line trigger.

Factory Preset and *RST:	1 µs
Range:	0.3 µs to 429 seconds
Default Unit:	seconds

External Trigger, Line and TV Trigger Delay Enable

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

`:TRIGger[:SEQuence]:DELay:STATe?`

This command allows you to turn on or off a delay, during which the analyzer will wait to begin a sweep after receiving an external trigger signal, a front panel TV trigger, or a line trigger.

Factory Preset and *RST:	Off
Default Unit:	seconds
Remarks:	Free-run activates the trigger condition that allows the next sweep to start as soon as possible after the last sweep. This function is not available when **Gate** is on.
Front Panel Access:	**Trig, Trig Delay On Off**

External Trigger Slope

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

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

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

Factory Preset and *RST:	Positive
Front Panel Access:	**Trig, External Pos Neg**

Trigger Offset

`TRIGger:SEQuence:OFFSet <64 bit floating point value>`

`TRIGger:SEQuence:OFFSet?`

This command sets the trigger offset.

Factory Preset and *RST: 0 seconds

Example: **`:TRIG:SEQ:OFFS 1.0s`**

Range: Hardware specific; dependent upon the ADC being used, current state and the number of sweep points.

Default Unit: seconds

Remarks: Trigger offset refers to the specified time interval before or after the trigger event from which data is to be written to the trace, and then displayed. Ordinarily, the trigger offset value is zero, and trace data is displayed beginning at the trigger event. A negative trigger offset value results in the display of trace data prior to the trigger event. A positive trigger offset value results in an effective delay in the display of trace data after the trigger event.

The trigger offset value used when the feature is enabled will depend on the following parameters:

- Nominal trigger offset value originally entered
- Specific instrument hardware in use
- Sweep time
- Number of sweep points

The effective trigger offset value will be re-calculated whenever any of these parameters change.

Front Panel Access:	**Trig, Trig Offset**

Trigger Offset On/Off

`TRIGger:SEQuence:OFFSet:STATe ON|OFF`

`TRIGger:SEQuence:OFFSet:STATe?`

This command enables or disables trigger offset.

Factory Preset and *RST:	Off
Example:	**`:TRIG:SEQ:OFFS`**
Range:	On/Off
Remarks:	Trigger offset can only be turned on when in zero span and the resolution bandwidth is 1 kHz or greater (non-digital bandwidths). Trigger offset is available for all trigger modes.
Front Panel Access:	**Trig, Trig Offset**

Trigger Source

`:TRIGger[:SEQuence]:SOURce IMMediate|VIDeo|LINE|EXTernal|TV`

`:TRIGger[:SEQuence]:SOURce?`

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

Immediate is free-run triggering

Video triggers on the video signal level

Line triggers on the power line signal

External allows you to connect an external trigger source

TV triggers on the selected line of a TV frame

Factory Preset and *RST:	Immediate (free-run triggering)
Front Panel Access:	**Trig, Free Run**
	Trig, Video
	Trig, Line
	Trig, External Pos Neg
	Trig, TV

Remarks: Free-run activates the trigger condition that allows the next sweep to start as soon as possible after the last sweep.

NOTE Trigger Delay is not available in Free Run, so turning Free Run on turns off Trigger Delay, but preserves the value of Trigger Delay.

Set TV Field Mode

:TRIGger[:SEQuence]:TV:FMODe ENTire|ODD|EVEN

:TRIGger[:SEQuence]:TV:FMODe?

This command allows the user to determine how the fields of the TV picture signal will be treated by the trigger system.

Factory Preset and *RST: ENTire (entire frame)

Range: **ENTire (entire frame)**

For formats NTSC-M, NTSC-Japan, and PAL-M, the minimum line is 1, and the maximum line is 525.

For formats PAL-B, D, G, H, I, PAL-N, PAL-N Combin, and SECAM-L, the minimum line is 1, and the maximum line is 625.

ODD (Field 1)

For formats NTSC-M, NTSC-Japan, and PAL-M, the minimum line is 1, and the maximum line is 263.

For formats PAL-B, D, G, H, I, PAL-N, PAL-N Combin, and SECAM-L, the minimum line is 1, and the maximum line is 313.

EVEN (Field 2)

For formats NTSC-M, NTSC-Japan, and PAL-M, the minimum line is 1, and the maximum line is 262.

For formats PAL-B, D, G, H, I, PAL-N, PAL-N Combin, and SECAM-L, the minimum line is 1, and the maximum line is 312.

NOTE Refer to chapter 6, Front-Panel Key Reference, in the Agilent ESA Spectrum Analyzers User’s Guide for a more detailed explanation of TV fields and command dependencies.

Front Panel Access: **Trig, TV Trig Setup, Field, Entire Frame**

Trig, TV Trig Setup, Field, Field One

Trig, TV Trig Setup, Field, Field Two

Set TV Line Number for Synchronization

:TRIGger[:SEQuence]:TV:LINE <line>

:TRIGger[:SEQuence]:TV:LINE?

This command allows you to set the TV line number to which the analyzer will synchronize its sweep.

Factory Preset and *RST:	17
Range:	The range is dependent upon the Field Mode, as described in the command **:TRIGger[:SEQuence]:TV:FMODe ENTire|ODD|EVEN**. The minimum value is the minimum line, and rolls over to the maximum value. The maximum value is the maximum line, and rolls over to the minimum value.
Remarks:	Refer to the command **:TRIGger[:SEQuence]:SOURce:TV**, which is used to activate TV triggering.

Set Analyzer for TV Picture Monitoring

:TRIGger[:SEQuence]:TV:MONitor OFF|ON|0|1

:TRIGger[:SEQuence]:TV:MONitor?

The currently-selected standard is used to determine the proper setup of the hardware state of the video digitization circuitry for presentation of the TV picture.

Factory Preset and *RST:	Off
Front Panel Access:	**Trig, TV Trig Setup,TV Monitor**

Set the Video Waveform Sync. Pulse Direction

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

:TRIGger[:SEQuence]:TV:SLOPe?

This command defines the direction of the sync pulse on the TV video waveform.

Factory Preset and *RST:	POSitive

Remarks:	Normal baseband video has sync pulses on the bottom of the waveform (use the **NEG** parameter for more negative voltage). However, when the analyzer is used to demodulate an NTSC or PAL TV RF carrier, the detected video waveform is "upside down" with sync pulses on the top of the waveform (use the **POS** parameter for more positive voltage). When the analyzer is used to demodulate a SECAM TV RF carrier, the detected video waveform has normal polarity (use the **NEG** parameter).
Front Panel Access:	**Trig, TV Trig Setup, Sync Pos Neg**

Select TV Signal Path

:TRIGger[:SEQuence]:TV:SOURce SANalyzer|EXTernal

:TRIGger[:SEQuence]:TV:SOURce?

This command is used to select between the internal spectrum analyzer signal path (the detected video is fed to the TV trigger), or the **EXT VIDEO IN** connector on the analyzer rear panel.

Factory Preset and *RST:	SANalyzer
Front Panel Access:	**Trig, TV Trig Setup, TV Source, SA**
	Trig, TV Trig Setup, TV Source, EXT Video In

Select TV Standard

:TRIGger[:SEQuence]:TV:STANdard MNTSc|JNTSc|MPAL|BPAL|NPAL|CPAL|LSEC

:TRIGger[:SEQuence]:TV:STANdard?

This command allows you to choose one of the various TV standards.

Factory Preset and *RST:	NTSC-M
Example:	`:TRIG:SEQ:TV:STAN MNTS`
Remarks:	Once this function is defined, the selected type is persistent. Persistent means that it retains the setting previously selected, even through a power cycle. The setting will change with "Load State."

As the TV standard is changed, the current line value is clipped as necessary to keep it valid for the chosen standard and field mode. For example, line 600 is selected in Entire Frame mode in PAL-N; if NTSC-M is selected, the line number is clipped to 525. Or, if line 313 is selected in Field 1 mode in PAL-N and NTSC-M is selected, the line number is clipped to 263. Changing back to the PAL-N standard will leave the line number at 263.

Front Panel Access: **Trig, TV Trig Setup, Standard, NTSC-M**

Trig, TV Trig Setup, Standard, NTSC-Japan

Trig, TV Trig Setup, Standard, PAL-M

Trig, TV Trig Setup, Standard, PAL-B,D,G,H,I

Trig, TV Trig Setup, Standard, PAL-N

Trig, TV Trig Setup, Standard, PAL-N Combin

Trig, TV Trig Setup, Standard, SECAM-L

Video Trigger Level Amplitude

`:TRIGger[:SEQuence]:VIDeo:LEVel <ampl>`

`:TRIGger[:SEQuence]:VIDeo:LEVel?`

Specifies the level at which a video trigger will occur.

Factory Preset and *RST: 2.5 divisions below reference level

Range: 10 display divisions below reference level to reference level

Default Unit: current amplitude units

Remarks: Video is adjusted using this command, but must also be selected using the command **`:TRIGger[:SEQuence]:SOURce VIDeo`**. When in FM Demod and Demod View is on, the Video Trigger level is adjusted/queried using the command **`:TRIGger[:SEQuence]:VIDeo:LEVel:FREQuency <freq>`**.

NOTE Trigger Delay is not available in Video trigger mode, so turning Video on turns off Trigger Delay, but preserves the value of Trigger Delay.

Front Panel Access: **Trig, Video**

Video Trigger Level Frequency

`:TRIGger[:SEQuence]:VIDeo:LEVel:FREQuency <freq>`

`:TRIGger[:SEQuence]:VIDeo:LEVel:FREQuency?`

This command is used to adjust the Video Trigger level when in FM Demod, and Demod View is on.

Default Unit:	Hz
Remarks:	Video is adjusted using this command, but must also be selected using the command **`:TRIGger[:SEQuence]:SOURce VIDeo.`** When not in FM Demod, the Video Trigger level is adjusted/queried using the command **`:TRIGger[:SEQuence]:VIDeo:LEVel <ampl>`**.

NOTE Trigger Delay is not available in Video trigger mode, so turning Video on turns off Trigger Delay, but preserves the value of Trigger Delay.

UNIT Subsystem

Select Power Units of Measure

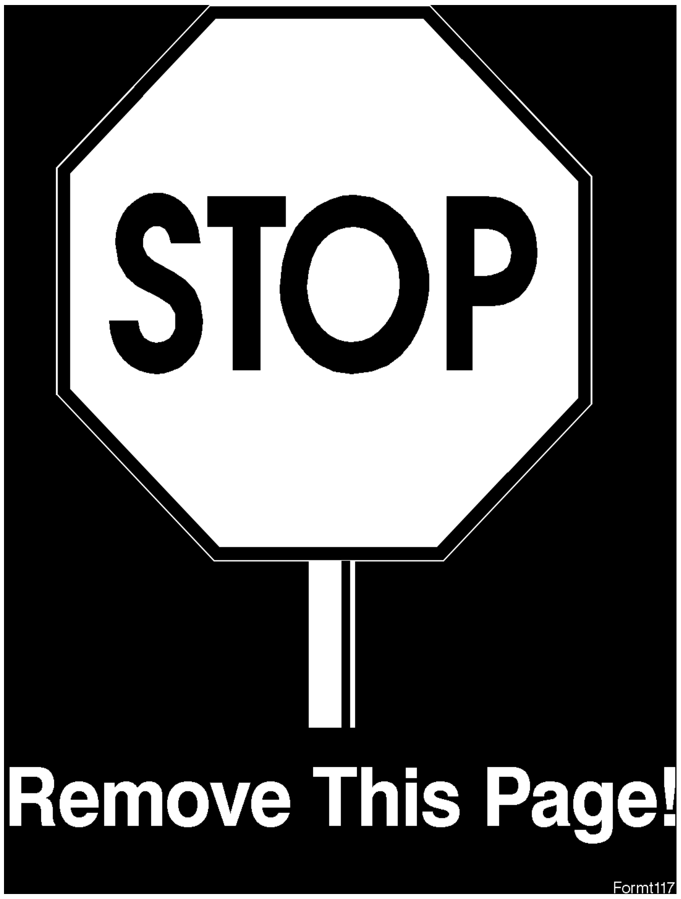
:UNIT:POWer DBM|DBMV|DBUV|V|W

:UNIT:POWer?

Specifies amplitude units for the input, output and display.

Factory Preset and *RST:	dBm in log amplitude scale
	volts in linear amplitude scale
Front Panel Access:	**AMPLITUDE/Y Scale, Amptd Units**
	AMPLITUDE/Y Scale, Amptd Units, dBm
	AMPLITUDE/Y Scale, Amptd Units, dBmV
	AMPLITUDE/Y Scale, Amptd Units, dBμV
	AMPLITUDE/Y Scale, Amptd Units, Volts
	AMPLITUDE/Y Scale, Amptd Units, Watts

6 Agilent 8590/ESA Spectrum Analyzers Programming Conversion Guide



NOTE Please remove this page and insert the wire-O bound Programming Conversion Guide here, Agilent Part Number E4401-90181.

7 Error Messages

Error Messages

The analyzer can generate various messages that appear on the display during operation. There are four types of messages.

- Status Messages appear on the right side of the analyzer display and/or set status bits in the SCPI Status Register system. These messages indicate a condition that may result in erroneous data being displayed. Most messages will only be displayed until the error condition is corrected. Multiple messages can be displayed and will be listed in the display area.
- Informational Messages provide information that requires no intervention. These messages appear in the status line at the bottom of the display, in green if you have a color display. The message will remain until you preset the analyzer, press **ESC**, or another message is displayed in the status line.
- User Error Messages appear when an attempt has been made to set a parameter incorrectly or an operation has failed (such as saving a file). These messages are often generated during remote operation when an invalid programming command has been entered. These messages appear in the status line at the bottom of the display, in yellow if you have a color display. The message will remain until you preset the analyzer, press **ESC**, or another message is displayed in the status line. A summary of the last 11 error messages may be viewed by pressing, **System** then **Show Errors**. When generated by activity on the remote interface, the messages are output to the remote bus. When output to the remote interface, they are preceded by an error number. Note that the error number is not displayed under the **System**, **Show Errors** key sequence.
- Pop-up Messages indicate a condition that may require intervention. They display in the middle of the display in a framed box. The message will remain until the appropriate intervention has taken place or the condition has been corrected.

Status Messages

The following messages indicate a condition that may result in erroneous data being displayed. In each case the name of the corresponding status bit is indicated in parenthesis. It will be noted if only a status bit is used (no message).

`* (Invalid Data)`

This indicator is displayed when data on the screen may not match the screen annotation, for example while analyzer settings are changing or when any trace is in view mode.

`50 MHz Osc Unlevel (50 MHz Osc Unleveled)`

The internal 50 MHz amplitude reference source has become unleveled. This condition must be corrected before a valid alignment can be performed.

`(ADC Align Failure)`

A status bit only, no message. The alignment routine was unable to align the analog-to-digital converter (ADC).

`Align Now All Needed (Align Needed)`

The instrument requires an **Align Now**, **All**. Restore the alignment by pressing **System**, **Alignments**, **Align Now**, **All**. On all Agilent Technologies ESA spectrum analyzer models except Agilent Technologies E4401B and Agilent Technologies E4411B you must connect the **AMPTD REF OUT** to the **INPUT** with the appropriate cable to perform this alignment.

`Align Now RF Needed (Align Now RF Needed)`

The instrument requires an **Align Now**, **RF**. Restore the alignment by pressing **System**, **Alignments**, **Align Now**, **RF (EXT Cable)**. On all Agilent Technologies ESA spectrum analyzer models except Agilent Technologies E4401B and Agilent Technologies E4411B, you must connect the **AMPTD REF OUT** to the **INPUT** with the appropriate cable to perform this alignment. *For Agilent Technologies E4401B and*

E4411B only: disconnect any signals from the **INPUT** prior to performing this procedure.

`Align RF Skipped (Align RF Skipped)`

The RF alignment has been skipped because a 50 MHz signal was detected at the INPUT; alignment will resume when the 50 MHz signal is removed. The alignment will not work when there is too much input power at 50 MHz. The instrument may not continue to measure properly. To remove the message, remove the 50 MHz input signal, then perform an **Align Now**, **RF**. Press **System**, **Alignments**, **Align Now**, **RF**. Be sure to connect the **AMPTD REF OUT** to the **INPUT** with the appropriate cable to perform the alignment.

`Ext Ref (no corresponding status bit)`

Indicates that the frequency reference is being supplied by an external 10 MHz source.

`Frequency Reference Error (Freq Ref Unlocked)`

The frequency reference has been tuned too far off of 10 MHz. This condition may be corrected by cycling power on the analyzer.

`(FM Demod Align Failure) status bit only, no message`

A failure has occurred during the FM Demod alignment. Measurement results may be invalid.

`(IF Align Failure) status bit only, no message`

A failure has occurred during the IF alignment. Measurement results may be invalid.

`IF Overload (IF/ADC Over Range)`

The IF section has been overloaded. Measurement results may be invalid.

`Input is internal (no corresponding status bit)`

This message applies to the Agilent Technologies E4401B and E4411B only. Indicates the **50 MHz Amptd Ref** selection is **On**. With the 50 MHz amplitude reference on, the input is routed through an internal signal path.

`(LO Align Failure) status bit only, no message`

A failure has occurred during the alignment of the local oscillator (LO). Measurement results may be invalid.

`LO Out Unlevel (LO Out Unleveled)`

Indicates the output of the local oscillator (LO) has become unleveled. This condition must be corrected to make valid measurements.

`LO Unlevel (LO Unleveled)`

Indicates the internal circuitry of the local oscillator (LO) has become unleveled. This condition must be corrected to make valid measurements.

`LO Unlock (Synth Unlocked)`

Indicates the phase locked circuitry of the local oscillator (LO) has become unlocked. This condition must be corrected to make valid measurements.

`Meas Uncal (Oversweep)`

The measurement is uncalibrated. Check the sweep time, span and bandwidth settings, or press **Auto Couple**.

`Overload: Reduce Signal and press <ESC> (Input Overload Tripped)`

This message applies to the Agilent Technologies E4401B and E4411B only. A signal has been applied to the input connector that caused the overload protection circuitry to engage. The input signal must be reduced. After the signal is reduced, press **ESC** to reset the overload detector so that you can continue using the analyzer.

CAUTION Exposing the analyzer to high levels of input power over a prolonged period of time can damage the internal circuitry.

`(RF Align Failure) status bit only, no message`

A failure has occurred during the alignment of the RF section. Measurement results may be invalid.

`Signal Ident On, Amptd Uncal (Signal Ident On)`

Indicates that the amplitude measurement could be uncalibrated because the signal identification feature is on.

`Source LO Unlevel (Source LO Unleveled)`

The internal circuitry of the local oscillator (LO) in the tracking generator has become unleveled. This condition must be corrected to make valid measurements.

`Source LO Unlock (Source Synth Unlocked)`

The phase-locked circuitry of the local oscillator (LO) in the tracking generator has become unlocked. This condition must be corrected to make valid measurements.

`Source Unlevel (Source Unleveled)`

Indicates the source power is set higher or lower than the analyzer can provide, the frequency span extends beyond the specified frequency range of the tracking generator, or the calibration data for the source is incorrect.

`(TG Align Failure) status bit only, no message`

A failure has occurred during the tracking generator (TG) alignment.

Informational Messages

The following messages provide information that requires no intervention. The information provided in brackets, for example <filename> or <directory>, is a variable that represents a specific input provided previously.

Informational messages are displayed at the bottom of the screen in the status line (green on color displays).

`<directoryname> directory deleted`

The directory indicated has been successfully deleted.

`<directoryname1> directory renamed to <directoryname2>`

Directory name1 has been successfully renamed to directory name2.

`<filename> file loaded`

The filename indicated has been successfully loaded.

`<filename> file saved`

The filename indicated has been successfully saved.

`<filename> file copied`

The filename indicated has been successfully copied.

`<filename> file deleted`

The filename indicated has been successfully deleted.

`<filename1> file renamed to <filename2>`

Filename1 has been successfully renamed to filename2.

`Atten auto set to 15 dB`

Indicates that an input signal has been detected which is of sufficient level to force the input attenuator to be autocoupled at 15 dB. If the signal level is reduced, the attenuator will stay at 15 dB. This overload protection occurs at an input power level of 13 dBm *(68 dBmV for Option 1DP)* and ± 7 dB when the input attenuation is autocoupled and set to <15 dB. To return to the original measurement setup, reduce the input signal level and press **Amplitude**. Then press **Attenuation (Auto)** .

Overload protection is only available in the Agilent Technologies E4401B and E4411B.

`Auto ranging...`

Displayed during autoranging.

`B7D and/or B7E not found. Code Domain not available.`

Digital Signal Processing and Fast Analog to Digital Converter (B7D) and/or RF Communications Hardware (B7E) are not installed options on your analyzer. Code domain is therefore not available.

`B7D and/or B7E not found. Mod Acc not available.`

Digital Signal Processing and Fast Analog to Digital Converter (B7D) and/or RF Communications Hardware (B7E) are not installed options on your analyzer. Modulation accuracy is therefore not available.

`Carrier Not Present.`

A carrier signal/burst is expected at the analyzer input. This signal cannot be found; however, the measurement will still run.

`Channel frequency outside device’s transmit band.`

Reset channel number or frequency.

`Default spur table values loaded.`

No spur table has been previously saved when the Out-of-Band Spurious measurement begins. Press **Meas Setup, Edit Table** to enter the frequency ranges of interest and press **Save Table** to save that information. This saved table will be loaded the next time the measurement is run.

`Device = Mobile. Code Domain not available.`

Code Domain measurement is greyed out when the device is set to mobile under the **Mode Setup** front-panel key. Code Domain measurement is only accessible when the device is set to base and Digital Signal Processing and Fast Analog to Digital Converter (*Option B7D*) or RF Communications Hardware (*Option B7E*) are installed.

`Device = Mobile. Mod Acc not available.`

Modulation accuracy measurement is greyed out when the device is set to mobile under **Mode Setup,** front-panel key. Modulation accuracy is only accessible when the device is set to base and Digital Signal Processing and Fast Analog to Digital Converter (*Option B7D*) or RF Communications Hardware (*Option B7E*) are installed.

`Directory already exists`

Each directory and file must have a unique name. The directory name you have entered is currently being used on the selected drive. You may either enter a new name or rename the directory currently existent. Refer to the *Agilent Technologies ESA Spectrum Analyzers User's Guide*.

`Entire trace is below the threshold level.`

The measurement cannot operate properly because the trace has fallen completely below the threshold level.

`Measurement halted. Press a measurement key to continue.`

This error occurs after you choose **Cancel** to refrain from setting the attenuator to 0 dB during the Receive Channel Power and Receive Spur measurements.

`Not enough frequency range to measure harmonics for channel.`

Selected harmonics are above the frequency range of the instrument.

`Option activated`

This message is displayed after entering the selected option's License Key.

`Please set RF input range (INPUT menu) to manual first.`

In order to manually set the reference level and/or the attenuation under the **Amplitude** front-panel key, the RF Input Range menu under the **Input** front-panel key must be set to **Man** (manual).

`Table loaded successfully.`

When the **Load Table** key was pressed on the second page of the edit table form while in the out-of-band spurious measurement, the file was present. The information has been loaded into the measurement where it may be edited again by the user. This message will also appear when the out-of-band spurious measurement is opened if a spur table has been previously saved.

`Table saved successfully.`

This message appears after the user presses the **Save Table** key on the second page of the edit table for the out-of-band spurious measurement. It indicates that the current spur table has been written successfully to disk and is available to be loaded by means of the **Load Table** key.

`Table saved successfully.`

This message appears after the user presses the **Save Table** key on the second page of the edit table for the out-of-band spurious measurement. It indicates that the current spur table has been written successfully to disk and is available to be loaded by means of the **Load Table** key.

`Volume <name> formatted`

The indicated disk has been successfully formatted.

`The calibration data is invalid, and has been cleared.`

A parameter has changed that affects calibration. Therefore the calibration data has been reset, and for best results recalibration is recommended.

`The file containing the list of cable types has been updated.`

The file update was successful.

`This operation requires a measurement to be active.`

The analyzer cannot perform this operation, as it requires a measurement to be running.

Error Queues

When a user-error condition occurs in the instrument as a result of SCPI (remote interface) activity, it is reported to both the front-panel display error-queue and the SCPI error queue. If it is a result of front-panel activity, it reports to the front panel display error queue, and may also report to the SCPI error queue depending on the error. These two queues are viewed and managed separately.

Table 7-1 Characteristics of the Error Queues

Characteristic	**Front-Panel Display Error Queue**	**SCPI Remote Interface Error Queue**
Capacity (number of errors)	11	30
Overflow Handling	Circular (rotating). Drops oldest error as new error comes in.	Linear, first-in/first-out. Replaces newest error with: `–350,Queue overflow`
Viewing Entries	Press: **System**, **Show Errors**	Use SCPI query `SYSTem:ERRor?`
Clearing the Queue	Press: **System**, **Show Errors**, **Clear Error Queue**	Power up. Send a `*CLS` command. Read last item in the queue.

Error Message Format

The system–defined error numbers are chosen on an enumerated (“1 of N”) basis. The error messages are listed in alphabetical order within each error message type section.

In this chapter, an explanation is included with each error to further clarify its meaning. The last error described in each class (for example, –400, –300, –200, –100) is a “generic” error. There are also references to the IEEE Standard 488.2-1992, *IEEE Standard Codes, Formats, Protocals and Common Commands for Use with ANSI/IEEE Std 488.1-1987*. New York, NY, 1992.

Error messages are displayed at the bottom of the screen in the status line (yellow on color displays).

Error Message Types

Events do not generate more than one type of error. For example, an event that generates a query error will not generate a device-specific, execution, or command error.

–499 to –400: Query Errors

These errors indicate that the instrument output queue control has detected a problem with the message exchange protocol described in IEEE 488.2, Chapter 6. Errors in this class set the query error bit (bit 2) in the event status register (IEEE 488.2, section 11.5.1). These errors correspond to message exchange protocol errors described in IEEE 488.2, 6.5. In this case:

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

–299 to –200: Execution Error Messages

These errors indicate that an error has been detected during instrument execution.

–199 to –100: Command Errors

These errors indicate that the instrument parser detected an IEEE 488.2 syntax error. Errors in this class set the command error bit (bit 5) in the event status register (IEEE 488.2, section 11.5.1). In this case:

- Either an IEEE 488.2 syntax error has been detected by the parser (a control-to-device message was received that 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.), or
- an unrecognized header was received. These include incorrect device-specific headers and incorrect or unimplemented IEEE 488.2 common commands.

201 to 799: Device-Specific Errors

These errors indicate that a device operation did not properly complete, possibly due to an abnormal hardware or firmware condition. These codes are also used for self-test response errors. Errors in this class set the device-specific error bit (bit 3) in the event status register (IEEE 488.2, section 11.5.1).

The <error_message> string for a positive error is not part of the SCPI standard. A positive error indicates that the instrument detected an error within the GPIB system, within the instrument firmware or hardware, during the transfer of block data, or during calibration.

Greater than 1000: Personality Specific Error Messages

These errors indicate that an error has been detected while executing measurements requiring *Option BAH*.

0:
No Error

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`.

–499 to –400: Query Errors

The instrument output queue control has detected a problem with the message exchange protocol described in IEEE 488.2, Chapter 6. Errors in this class set the query error bit (bit 2) in the event status register (IEEE 488.2, section 11.5.1). These errors correspond to message exchange protocol errors described in IEEE 488.2, 6.5.

In this case, either an attempt is being made to read data from the output queue when no output is either present or pending, or data in the output queue has been lost.

–430	`Query DEADLOCKED` Indicates that a SCPI output queue has filled, preventing further SCPI command execution, and there is no more room left in the corresponding SCPI input queue to accept a query to read from the output queue. The system automatically discards output to correct the deadlock.
–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.
–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.
–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.

–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).

–299 to –200: Execution Error Messages

An error number in the range [–299 to –200] indicates that an error has been detected during instrument execution.

Execution Error Message Descriptions

–221 `Settings conflict; parameter currently disabled`

This parameter is grayed out (unavailable) in the curent context. Check the individual parameter help/documentation for more information.

––230 `Data corrupt or stale.`

Possibly invalid data. A new measurement was started but not completed.

–199 to –100: Command Errors

The instrument parser detected an IEEE 488.2 syntax error. Errors in this class set the command error bit (bit 5) in the event status register (IEEE 488.2, section 11.5.1). In this case:

- Either an IEEE 488.2 syntax error has been detected by the parser (a control-to-device message was received that 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.), or
- an unrecognized header was received. These include incorrect device-specific headers and incorrect or unimplemented IEEE 488.2 common commands.

–160	`Block data error` This error, as well as errors –161 through –169, is generated when parsing a block 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.
–140	`Character data error` This error, as well as errors –141 through –149, is generated when parsing a character 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 than twelve characters (see IEEE 488.2, 7.7.1.4).

–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.
–110	`Command header error` An error was detected in the header. This message is used when the device cannot detect the more specific errors described for errors –111 through –119.
–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.
–123	`Exponent too large` The magnitude of an exponent was greater than 32000 (see IEEE 488.2, 7.7.2.4.1).
–170	`Expression data error` This error, as well as errors –171 through –179, is generated when parsing an expression data element. This particular 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.
–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.
–111	`Header separator error` A character which is not a legal header separator was encountered while parsing the header.

–114	`Header suffix out of range`	The value of a header suffix attached to a program mnemonic makes the header invalid.
–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.
–101	`Invalid character`	A syntactic command contains a character which is invalid for that type. For example, a header containing an ampersand, such as “SETUP&”. This error might be used in place of error numbers –114, –121, –141 and some others.
–141	`Invalid character data`	Either the character data element contains an invalid character or the particular element received is not valid for the header.
–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.
–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.
–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.

–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.
–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.
–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.
–120	`Numeric data error`
	This error, as well as errors –121 through –129, 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.
–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.
–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.
–112	`Program mnemonic too long`
	The header contains more than twelve characters (see IEEE 488.2, 7.6.1.4.1).

–150	`String data error` This error, as well as errors –151 through –159, is generated when parsing a string 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.
–130	`Suffix error` This error, as well as errors –131 through –139, is generated when parsing a suffix. 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).
–102	`Syntax error` An unrecognized command or data type was encountered. For example, a string was received when the device does not accept strings.
–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).
–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.

201 to 799: Device-Specific Errors

Some device operations did not properly complete, possibly due to an abnormal hardware or firmware condition. These codes are also used for self-test response errors. Errors in this class set the device-specific error bit (bit 3) in the event status register (IEEE 488.2, section 11.5.1).

The <error_message> string for a *positive* error is not defined by SCPI. A positive error indicates that the instrument detected an error within the GPIB system, within the instrument firmware or hardware, during the transfer of block data, or during calibration.

653 `Auto Align not available when using Calibration Defaults`

The Auto Alignment system cannot be used until an **Align Now All** is executed by pressing **System**, **Alignments**, **Align Now**, **All**. For Agilent Technologies E4402B, E4403B, E4404B, E4405B, E4407B, and E4408B only, you must connect the **AMPTD REF OUT** to the **INPUT** with the appropriate cable to perform this alignment.

614 `Bad or missing floppy disk`

The floppy is not inserted or the directory could not be read. Insert a known good disk and try again.

205 `Command not recognized`

Indicates that the command sent from the remote interface was not recognized. Check the programming guide for correct syntax.

205 `Command not recognized`

Indicates that the command sent from the remote interface was not recognized. Check the programming guide for correct syntax.

219	`Command not valid in this model`	Indicates that the command sent from the remote interface does not apply to this model number. For example, attempting to center the preselector in an analyzer without a preselector will generate this error.
222	`Command not valid when no measurement is active`	Indicates that the command sent from the remote interface must be issued while a measurement is running in the analyzer. .
772	`Cannot load a directory, please choose a file`	You have selected a directory instead of a file when attempting to perform the Load function under the **File** front-panel key.
652	`Connect Amptd Ref Output to Input`	*For Agilent Technologies E4402B, E4403B, E4404B, E4405B, E4407B, and E4408B only:* you must connect the **AMPTD REF OUTPUT** to the analyzer **INPUT** with the appropriate cable.
651	`Connect RF OUT to INPUT`	Attempt to align the tracking generator without its output connected. Connect the tracking generator RF OUT to the analyzer **INPUT**.
615	`Corrupted file`	The file that you were trying to load is corrupt.
610	`File access is denied`	The file is protected or hidden and cannot be accessed.
604	`File already exists`	Attempt to save to a file that already exists. Delete or rename the old file and try again.

607	`File Name Error` An invalid file name has been specified. Use filenames with a maximum of 8 characters (letters and digits only) and use a 3 character extension. Note that lowercase and uppercase are perceived as the same. This error will also occur if you attempt to delete a nonexistent file.
612	`File not found` The analyzer could not find the specified file.
613	`Flash memory is full` The internal flash memory is full. Clear some space by deleting unwanted files. You may also increase the flash memory size by purchasing *Option B72*.
602	`Floppy disk error` An unknown error has occurred while accessing the floppy disk.
601	`Floppy disk full` The floppy disk is full. Clear some space by deleting unwanted files.
618	`Illegal write access of Flash memory` Attempt to write to an unavailable area of internal flash memory.
727	`In <filename>: [DATA] header missing` This message indicates that the data section of a file did not begin with the token [**DATA**].
728	`In <filename>, line <nnn>: separator missing` The [**HEADER**] section of a file contains entries requiring an equal (=) sign, such as <keyword> = <value>. This message appears if the equal sign does not appear on the line.

729	`In <filename>: error reading file` Appears when loading data from a limit line or corrections disk file and a failure to the file occurs.
730	`In <filename>, line <numeric_value>: line too long` When loading data from a limit line or corrections disk file, this message will appear if the length of any line in the file exceeds 255 characters.
731	`In <command>: bad data count (<numeric_value>): expected multiple of <numeric_value>` This message indicates that the data sent to a corrections or limit table via the **DATA** or **MERGE** commands does not have the expected length for the table. For example, this message would appear if an attempt were made to merge 7 numeric values into a limit table, since each logical entry requires 3 values (frequency, amplitude, and connected).
732	`In <filename>, line <numeric_value>: error parsing tokens` This message may appear when loading data from a limit line or corrections disk file. It indicates a problem in the attempt to break a string of text into tokens. There may be too few tokens in the string. This typically happens when there are too few numeric values in the [**DATA**] section of a limit or corrections file.
733	`In <filename>, line <numeric_value>: <xxx> is not numeric` This message may appear when loading data from a limit line or corrections disk file. It indicates that a non-numeric token <xxx> was found where a numeric token was expected.
735	`In <filename>: bad amplitude unit <unit>` This message indicates that unit <unit> is not recognized or supported.

763	`Incorrect filename, allowable extensions are .gif or .wmf` Attempt to save a screen image to a file with an incorrect extension.
762	`Incorrect filename, allowable extensions are .trc or .csv` Attempt to save a trace to a file with an incorrect extension.
770	`Instrument mode requested is not supported` Instrument mode specified with :INST command is not valid. Refer to Chapter 5, “Instrument Subsystem” of *Agilent Technologies ESA Series Spectrum Analyzers Programmer's Guide* for more information.
751	`Instrument state may be corrupt, state has been reset to initial values` An error in the internal instrument state has been detected. The state has been reset to a default value.
734	`Interpolation error: cannot compute log of <negative_frequency_value>` Occurs when the frequency interpolation of a limit line is set to log and the start frequency of the instrument is negative. The <negative_frequency_value> is limited to – 80 MHz, so it may not match the frequency that caused the error.
216	`Invalid Baud Rate` Attempt to use invalid baud rate. Refer to Chapter 5, “Instrument Subsystem” of *Agilent Technologies ESA Series Spectrum Analyzers Programmer's Guide* for more information.

769	`Invalid instrument mode` You have attempted to switch to an instrument mode that is currently not installed. Confirm that the mode name (for `INST:SEL`) or number (for `INST:NSEL`) was entered correctly and that the requested personality is actually installed in the instrument.
221	`Invalid option, unable to uninstall package` You have attempted to remove a personality that is not currently installed. Verify command was entered correctly.
701	`Invalid printer response` In attempting to identify the printer an invalid response was received. Check that you are using a supported printer. Be sure you are using the proper cable and that it is securely fastened.
606	`Media is corrupt` A save was attempted to a corrupt device.
609	`Media is not writable` A save was attempted to a read-only device.
605	`Media is protected` A save was attempted to a write-protected device.
202	`No peak found` No signal peak was found.
201	`Option not installed` The desired operation cannot be performed because a required option is not installed. For example, pressing **Source** with no tracking generator installed in the analyzer will generate this error.

224	`Option not licensed.` The selected option requires a license. Refer to the installation procedures in the user's guide available for this particular option.
209	`Preselector centering failed` An attempt to center the preselector failed.
704	`Printer interface error` An error occurred while trying to print. Make sure the printer is turned on and properly connected.
705	`Printer Type is None` The current printer type is set to **None**, so no print operations are possible. Change the type in the **Print Setup** menu and try again.
211	`RBW limited to 1kHz when Span > 5MHz` In spans greater than 5 MHz, narrow (digital) resolution bandwidths, below 1 kHz, are not available.
217	`RS-232 Interface Error` An error occurred on the serial interface.
213	`Span limited to 5MHz when RBW < 1kHz` In narrow (digital) resolution bandwidths, below 1 kHz, spans greater than 5 MHz are not available.
771	`Store Ref trace before turning on Normalize` A reference trace must be available for the Normalize function to be activated. Refer to the *Agilent Technologies ESA Spectrum Analyzers User's Guide*, where the **Normalize** key function is explained in detail.

223	`Trigger Offset unavailable in swept spans`
	Trigger Offset is only available in Zero Span. *(Refer to the Agilent Technologies ESA Spectrum Analyzers User's Guide.)*
215	`TG start freq is less than 1/2 res bw`
	TG uncalibrated at start frequencies below 1/2 the current resolution bandwidth.
214	`TG start freq is less than 9kHz`
	TG uncalibrated below 9 kHz.
204	`TG Frequency Limit`
	The tracking generator has reached the limit of its allowable frequency range.
736	`Too many data values at <freq_or_time_value>`
	This message may appear when data is sent to a corrections or limit table using the **DATA** or **MERGE** commands. These tables limit the number of amplitudes associated with a frequency or time to 2 or less. This message will appear if an attempt is made to attach 3 or more values to a frequency or time.
206	`Unable to initialize flatness data`
	A failure occurred in setting the flatness data in the internal EEROM. Contact your local Agilent Technologies sales and service office.
762	`Unable to load file`
	A failure occurred while loading a file; the file was not loaded.
759	`Unable to load state saved from firmware Rev A.03.00`
	A saved state file from a newer firmware revision was attempted to be loaded into an older instrument.

752	`Unable to load state from file`	Loading of state from a file failed.
755	`Unable to load state from register`	Loading a state from an internal state register failed.
757	`Unable to load user state, factory preset was done`	An attempt to perform a **User Preset** failed, so the **Factory Preset** values were used. Save a valid state into **User Preset** and try again.
760	`Unable to query state`	Query of state over the remote interface was unsuccessful.
764	`Unable to save file`	A failure occurred while saving a file; the file was not saved.
756	`Unable to save state to register`	Saving of state to an internal register failed.
753	`Unable to save state to file`	Saving of state to a file failed.
758	`Unable to save user state`	An attempt to save to the **User Preset** state failed.
761	`Unable to set state`	Attempt to set the state over the remote interface was unsuccessful.

207	`Unable to store flatness data` A failure occurred in setting the flatness data in the internal EEROM. Contact your local Agilent Technologies sales and service office.
703	`Unknown printer` In attempting to identify the printer, a valid response was received but the printer is not known to the analyzer. Use the **Custom** printer menu under **Print Setup** to configure the printer.
702	`Unsupported printer` A printer which is recognized, but known to be unsupported was identified. This printer cannot be used with the analyzer. For example, a printer only supported by Microsoft Windows will generate this error.
617	`Wrong density floppy inserted` The floppy disk has the wrong density. It should be 1.44 MB.

Greater than 1000: Personality Specific Error Messages

An error detected with a number greater than 1000 indicates the instrument has detected an error relating to an installed personality. For more information on these errors, refer to the user’s guide for the personality in use.

10219	`Awaiting trigger` A trigger event from the selected trigger source has not been detected.
10164	`Band Measurement not defined for Out of Band.` User is attempting to monitor the band but has set the frequency outside the band. Reset the band for the particular standard for which you are testing or use the channel setting which does not require a frequency to be set. (**Meas Setup**, **Method (Channel)**).
10228	`Cannot correlate to input signal`. This error is normally generated because of one of the following reasons: 1. There is no carrier signal. 2. Walsh channels other than the pilot are active. 3. There is some other modulation problem that will prevent the measurement from being made. This problem must be corrected before the measurement can continue.
10163	`Cannot find the Power vs Time Limits File.` The limit line definition file for the GSM standards has been deleted. This message is displayed while the **Measure** key is greyed out. Reinstall the GSM measurement personality.
10166	`Cannot update the list of cable types.` The cable file may have been moved or deleted accidentally. Reinstall the GSM measurement personality.

10168 `Cannot update the list of cable types on drive C:`

The file update failed.

10179 `Carrier Present. Test Stopped!`

A carrier was found in the transmit band. Either disable the carrier or insert a bandpass filter for the receive bandwidth.

10153 `DSP algorithm timeout, aborting measurement`

The Digital Signal Processor demodulation algorithm timed-out for an unknown reason. This message normally indicates a problem with the modulated signal.

10230 `DSP timed out, resetting DSP.`

Digital Signal Processor was unable to finish the selected measurement within the given period of time. Restart the measurement.

10264 `Emmision bandwidth not found. Consider increasing span.`

This error is normally generated when attempting occupied bandwidth measurements. The “X dB” value you entered (**Meas Setup**, **Emis BW X dB**) to calculate the emission bandwidth is the difference between the highest point on the trace and the point “X dB” down on either side of the maximum. If the actual difference is less than the value entered, the emission bandwidth cannot be computed. Some responses to this situation are as follows:

1. Connect a signal to the input. (If there is no signal present, the difference between the trace minimum and maximum will generally be less than “X dB”.)
2. Increase the span. (If the signal is wide, the shoulders of the signal might not be present on the screen, and again, the difference between the trace minimum and maximum will be less than “X dB”.)

3. Center the signal. (There must be a point on the trace that is “X dB” down from the maximum on both sides of that maximum.)

10246 `Error reading file: CDMASTUN.CSV. Please reinstall cdmaOne.`

The file is missing or corrupt. Please reinstall the cdmaOne personality.

10247 `Error reading file: CDPDMDA. Please reinstall the cdmaOne.`

The file is missing or corrupt. Please reinstall the cdmaOne personality.

10248 `Error reading file: CDPPMCO. Please reinstall the cdmaOne.`

The file is missing or corrupt. Please reinstall the cdmaOne personality.

10249 `Error reading file: CDPPMDA. Please reinstall cdmaOne.`

The file is missing or corrupt. Please reinstall the cdmaOne personality.

10256 `Error reading file: OOBSTAB.CSV. Use Edit Table | Save Table.`

This error is generated when you try to load a table (using the **Load Table** key on page 2 of the edit table form menu) before a table has been saved. You must first save a table using the **Save Table** key before trying to load a table using the **Load Table** key.

10250 `Error reading file: RHODMDA. Please reinstall cdmaOne.`

The file is missing or corrupt. Please reinstall the cdmaOne personality.

10251	`Error reading file: RHOPMCO. Please reinstall cdmaOne.` The file is missing or corrupt. Please reinstall the cdmaOne personality.
10245	`Error reading file: SPCLIMIT.CSV. Cannot use custom limits.` The file could be missing or corrupt. Create a new limits file. Alternatively, the actual limits defined in the file might not allow the measurement to be executed. Redefine the limits or use the default limits. Restart the measurement.
10180	`Gate option not installed. Results may not be accurate.` This measurement method requires the use of the time-gate (option 1D6) in order to gate the spectrum during the 50-90% part of the burst. If the gate option is not installed, the measurement will still run although this warning will be displayed.
10218	`Hardkeys are disabled.` Some of the forms (for example Receiver Spurious in GSM) do not allow the user to close the form without either formally accepting or cancelling the form settings. For this reason, all of the hardkeys are disabled until the user terminates the form.
10233	`Level is low, results may degrade.` The signal being measured is of low power. The results may not be as accurate as they would be if the signal was stronger.
10152	`Lost trigger, aborting measurement.` The selected trigger source was present at the start of the measurement, but timed out before the measurement completed.

10161	`Lower Custom Mask is Invalid!` The user-specified lower custom mask cannot be resolved into a limit line.
10231	`Measurement failed for unknown reasons.` Check instrument settings and restart measurement.
10154	`Measurement not defined for Out of Band.` User has changed to an out-of-band frequency range. The band measurement only operates in the selected band.
10227	`Measurement suspended until carrier is turned off.` The receive channel power and the receive spur measurements are specified with the attenuation set to 0 dB. To prevent overload, the frequency spectrum of interest is monitored for signal levels which exceed a specified threshold before setting the attenuator to 0 dB. If a carrier is found, this message is displayed and the completion of the measurement will not occur until the carrier is removed. The carrier check may be turned off using the properties form under the front-panel **Mode Setup** key. You may also change the signal threshold which determines a carrier on the properties form.
10155	`No Fast ADC hardware installed. Meas unavailable.` The analyzer cannot use sweeptimes less than 5 msec when (*Option B7D* or *Option AYX*) is not installed. Therefore the measurement will not be executed.
10147	`Opt B7D bootrom requires upgrade.` The (*Option B7D*) bootrom revision is not supported by the curently loaded personality version. Refer to the user’s guide for the personality in use.

10149	`Opt BAH DSP algorithm code file requires upgrade.` The Digital Signal Processing algorithm code file revision is not supported by the currently loaded personality version. Refer to the user's guide for the personality in use for more information on installation/upgrade.
10150	`Opt BAH DSP algorithm coef. file requires upgrade.` The Digital Signal Processor algorithm coefficient file revision is not supported by the currently loaded personality version. Refer to the user's guide for the personality in use.
10151	`Opt BAH DSP algorithm files failed to load, aborting measure.` The Digital Signal Processor algorithm files required to perform the demodulation are corrupt and cannot be loaded properly.
10148	`Opt BAH DSP algorithm files not installed. Meas unavailable.` The Digital Signal Processor algorithm files required to perform the demodulation are not present in the analyzer.
10145	`Opt AYX hardware required. Meas unavailable.` *Option AYX* must be installed for this measurement to be enabled.
10146	`Opt B7D & B7E hardware required. Meas unavailable.` The (*Option B7E*) and (*Option B7D*) cards required to perform the demodulation are not present in the analyzer.

10239 `Opt Freq Ref setting does not match external reference.`

This message is generated if Source is set to External on the properties form under the front-panel **Mode Setup** key and the frequency on the same form is set to a frequency that does not match the frequency of the signal being used as the external reference.

10232 `RF Signal not found.`

This message is generated if there is no signal at the center frequency that is greater than 10 dB above the displayed average noise level.

10241 `RF Board could not detect any bursts in signal.`

This message is generated when the trigger is set to RF Burst and (*Option B7E*) cannot detect a burst.

10237 `RF Board LO Unlocked. Contact service center.`

This message occurs if the local oscillator on the (*Option B7E*) is in an unlocked state. This indicates broken hardware.

10240 `RF Board RF Osc Unlocked. Contact service center.`

This message occurs if the reference oscillator on the (*Option B7E*) is in an unlocked state. This indicates broken hardware.

10238 `RF Board SR Osc Unlocked. Contact service center.`

This message occurs if the sample rate oscillator on *Option B7E* is in an unlocked state. This indicates broken hardware.

10162 `Resolution BW<300kHz.`

This error message is a warning that the resolution bandwidth has been set below 300 kHz. The test results will not meet GSM specifications.

10172 `Sweep Time too fast(<2sec)`

The sweep time must be set to 2 seconds or longer for the results to be valid.

10141 `Sync word not found in frame (Burst Type)`

One or more active GSM bursts that match the selected Burst Type have been detected in the RF Input signal, but none contain the selected Training Sequence Code (TSC). The search was performed over the complete GSM frame.

10143 `Sync word not found in frame (Ref Burst)`

One or more active GSM bursts that match the selected Burst Type have been detected in the RF Input signal, but none contain the selected Training Sequence Code (TSC). The search was only performed using the Reference Burst type and Reference TSC settings over the complete GSM frame.

10142 `Sync word not found in specified timeslot (Burst Type)`

One or more active GSM bursts that match the selected Burst Type have been detected in the RF Input signal, but none contain the selected Training Sequence Code (TSC). The search was only performed over the specified timeslot setting.

10259 `Table could not be loaded.`

When trying to load a table, the previous table has been somehow corrupted. Use the **Save Table** key to save a valid table. Then edit the valid table, save it, and try to load it again.

10259 `Table could not be saved.`

This message occurs if C: drive is full or corrupt. Check the amount of space left on the drive.

10170	`The Cable Fault Measurement is active. Mode Setup is disabled.` Mode setup is not available in the cable fault utility.
10177	`There are no spurs to inspect.` The user has attempted to switch the **Inspect Spur** softkey to the **On** position after the measurement has finished, but found no spurs.
10229	`The regression portion failed.` This message occurs when (*Option B7D*) is not functioning properly. Demodulation measurements (modulation accuracy and code domain) might fail as a result of this error.
10157	`Tracking Generator hardware is not present. Meas unavailable.` The measurement requires a built-in tracking generator.
	`Unable to uninstall personality, file not deletable.` This message occurs when you try to delete a personality which has been marked as non-deletable. The personality is marked non-deletable at the factory. Contact your nearest service center for further problems.
10144	`Unknown demod status.` Demodulation is in an unknown state. Press **Preset**. If the error persists, contact your service center.
10160	`Upper Custom Mask is Invalid!` The user-specified upper custom mask cannot be resolved into a limit line. The format is incorrect.

10138	`Valid GSM burst not found in frame (Burst Type).` No active GSM bursts that match the selected Burst Type have been detected in the RF input signal. The search was performed over the complete GSM frame.
10140	`Valid GSM burst not found in frame (Ref Burst).` No active GSM bursts that match the selected Burst Type have been detected in the RF input signal. The search was performed using the Ref Burst type setting over the complete GSM frame.
10139	`Valid GSM burst not found in specified timeslot (Burst Type).` No active GSM bursts that match the selected Burst Type have been detected in the RF input signal. The search was only performed over the specified timeslot setting.

Index

Index