Reference Guide

Agilent Technologies PSA Spectrum Analyzers

This manual provides documentation for the following instrument:

Agilent Technologies PSA Series

E4440A (3 Hz – 26.5 GHz)

Agilent Technologies

Manufacturing Part Number: E4440-90010

Printed in USA
January 2001

Where to Find the Latest Information

Documentation is updated periodically. For the latest information about Agilent PSA spectrum analyzers, including firmware upgrades and application information, see: http://www.agilent.com/find/psa.

Contents

List of Commands

List of Commands

List of Commands

List of Commands

List of Commands

1 Using This Document

This chapter describes the organization of this reference guide.

1.1 Book Organization

This book is organized by front-panel functions. The functions are documented by blocks of keys as they are labeled on the instrument front panel. See the illustration below. Each block of keys is documented in its own chapter, see Table 1-1.

There are many terms used in this Reference Guide that are explained in detail in the Getting Started Guide. It is recommended that you read the Getting Started Guide first in order to become familiar with these terms and their definitions..

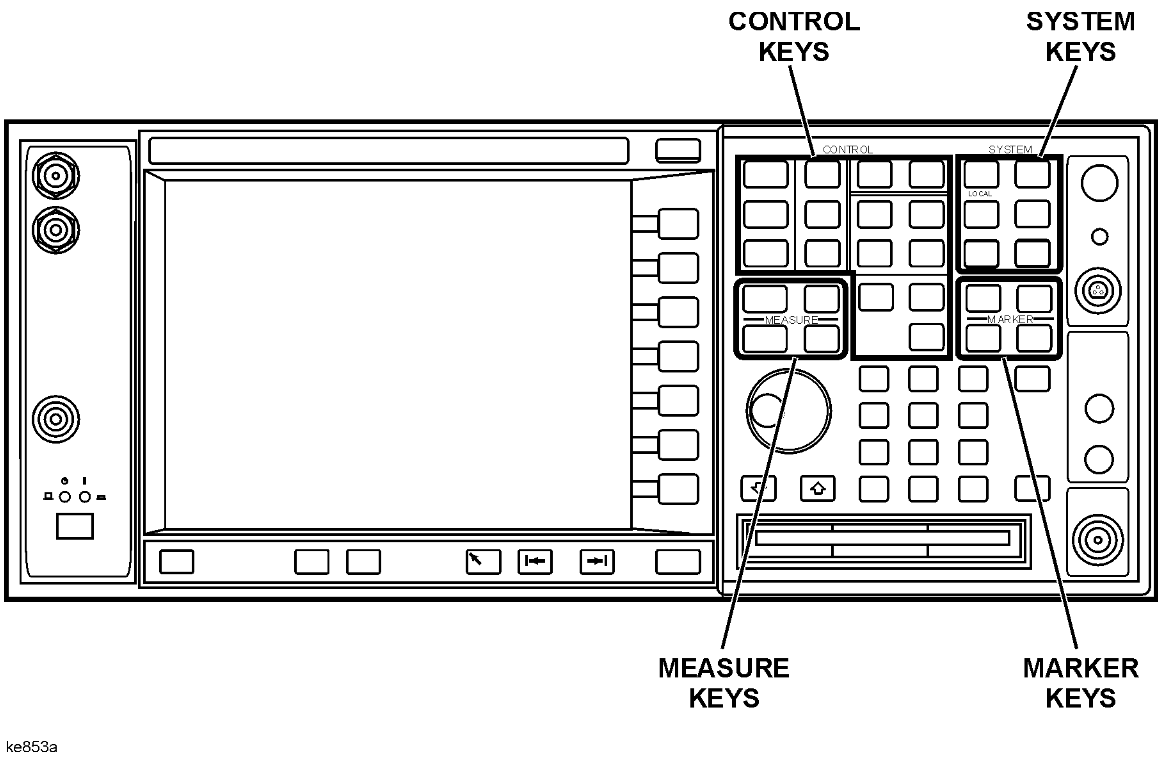


Table 1-1 Book Organization

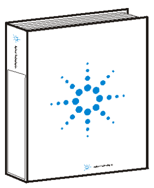
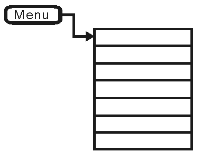
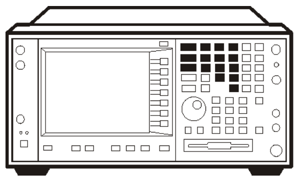
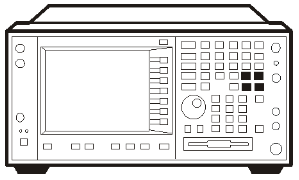
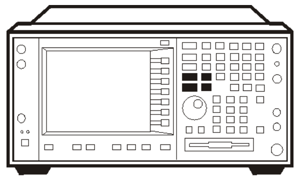
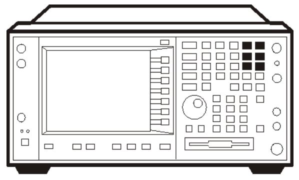
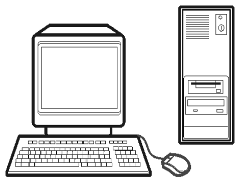
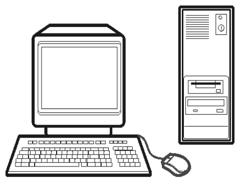
	**1. Using this Document** This chapter.	Describes the organization of this book.
	**2. Menu Maps** See page 29.	Illustrates the menu structure of the front-panel and lower-level keys. Refer to this chapter to identify the lower-level softkeys associated with the front-panel keys.

Table 1-1 Book Organization

	**3. Control Keys** See page 63.	Describes the instrument front-panel Control functions and programming commands.
	**4. Marker Keys** See page 119.	Describes the instrument front-panel Marker functions and programming commands.
	**5. Measure Keys** See page 141.	Describes the instrument front-panel Measure functions and programming commands.
	**6. System Keys** See page 163.	Describes the instrument front-panel System functions and programming commands.
	**7. Programming Fundamentals** See page 205.	Describes information on SCPI and C programming language basics and on using GPIB, RS-232, and LAN.
	**8. Status Commands** See page 265.	Describes the instruments internal status monitoring system with information on how to monitor the status using a remote program and descriptions of all the available commands.

NOTE The numeric keypad and alpha-numeric softkey fundamentals are described in the Getting Started guide.

1.1.1 Terms Used in This Book

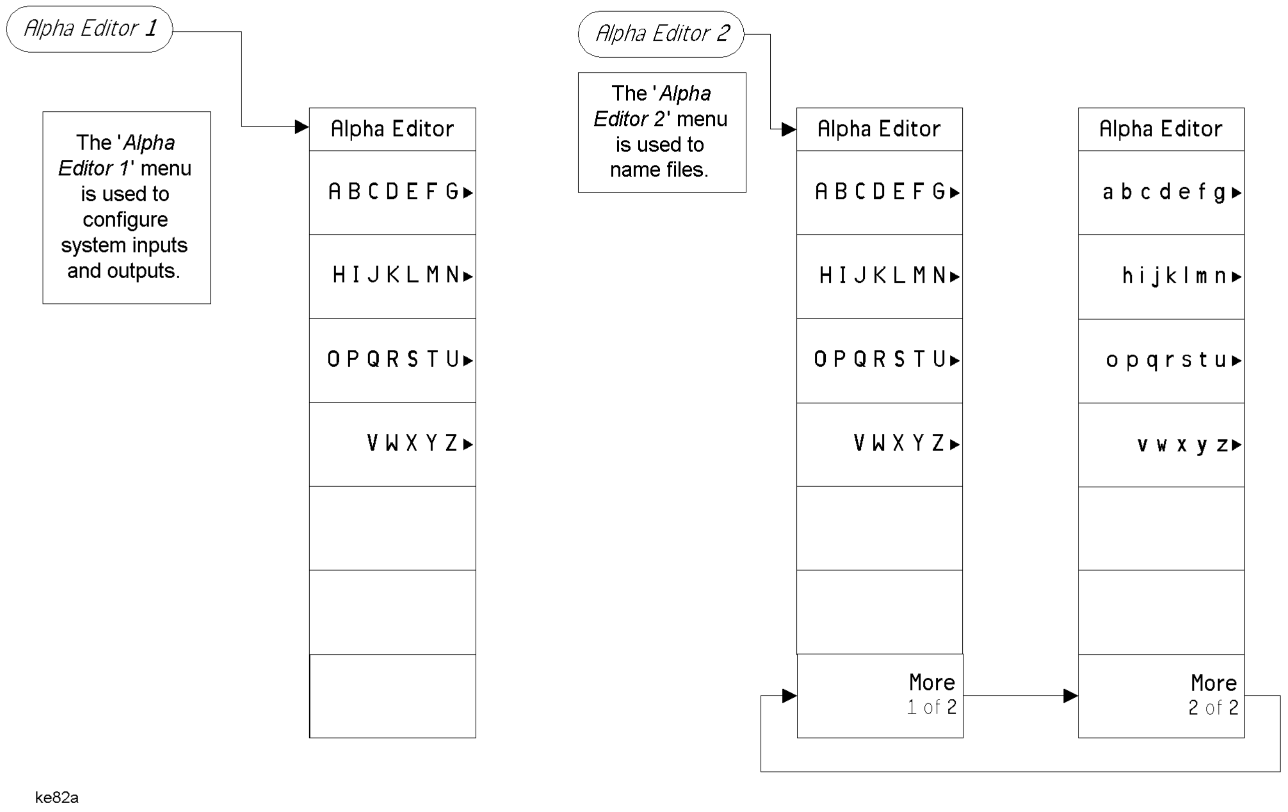
Information about each key is described using the following terms. Note that a particular key description my not use all the terms.

State Saved:	Indicates what happens to a particular function when the instrument state is saved (either to floppy disk or the internal c:\ drive). It also indicates whether your current setting of the function will be maintained if the instrument is powered on or preset using **Power On Last State** or **User Preset**.
Dependencies/ Couplings:	Describes dependencies or interactions to other functions and settings in the analyzer.
Factory Preset and *RST:	Describes the function settings after a **Factory Preset** or *RST command.
Maximum Value:	Describes the largest value to which this function can be set. If an attempt is made to go beyond this value the analyzer will default to this maximum value.
Minimum Value:	Describes the smallest value to which this function can be set. If an attempt is made to go below this value the analyzer will default to this minimum value.
Remote Command:	Shows the syntax requirements for each SCPI command.
Example:	Provides command examples using the indicated remote command syntax.

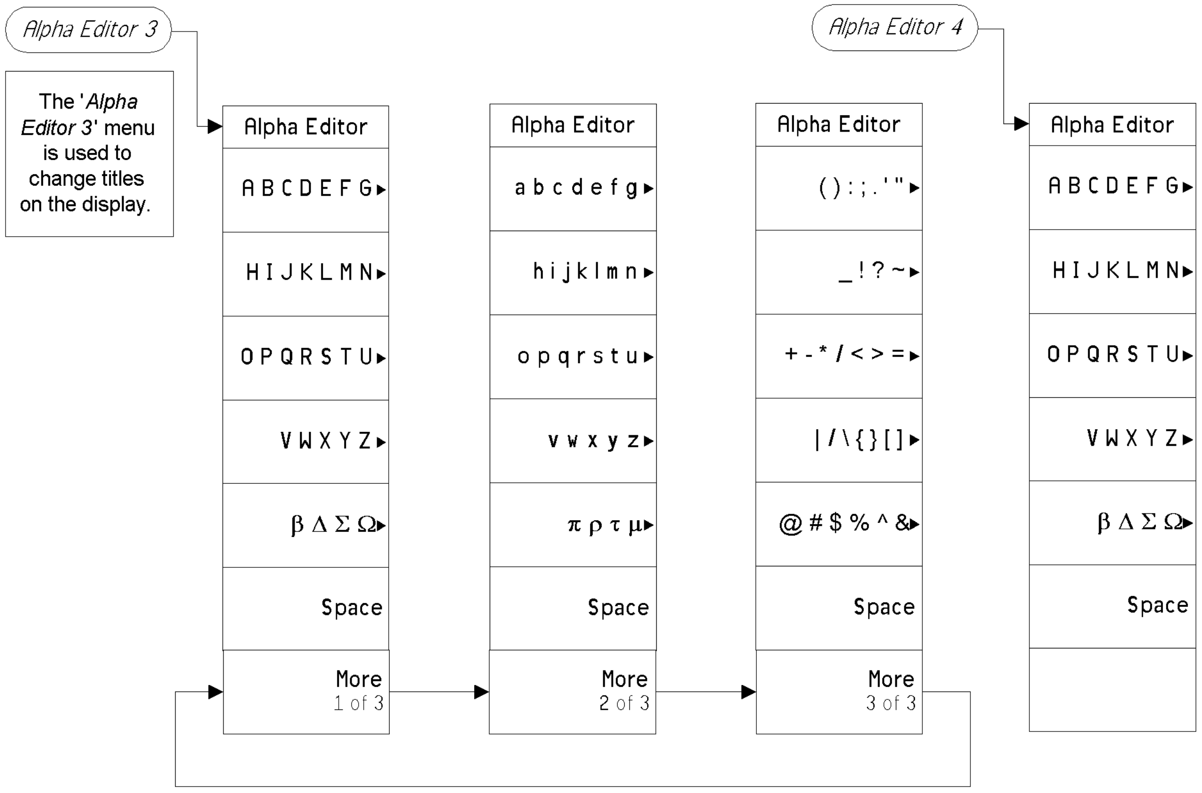
2 Menu Maps

These menu maps are in alphabetical order by the front panel key label or oval cross-reference label. You can locate detailed information about each key/function at the page number listed in the figure title for each menu.

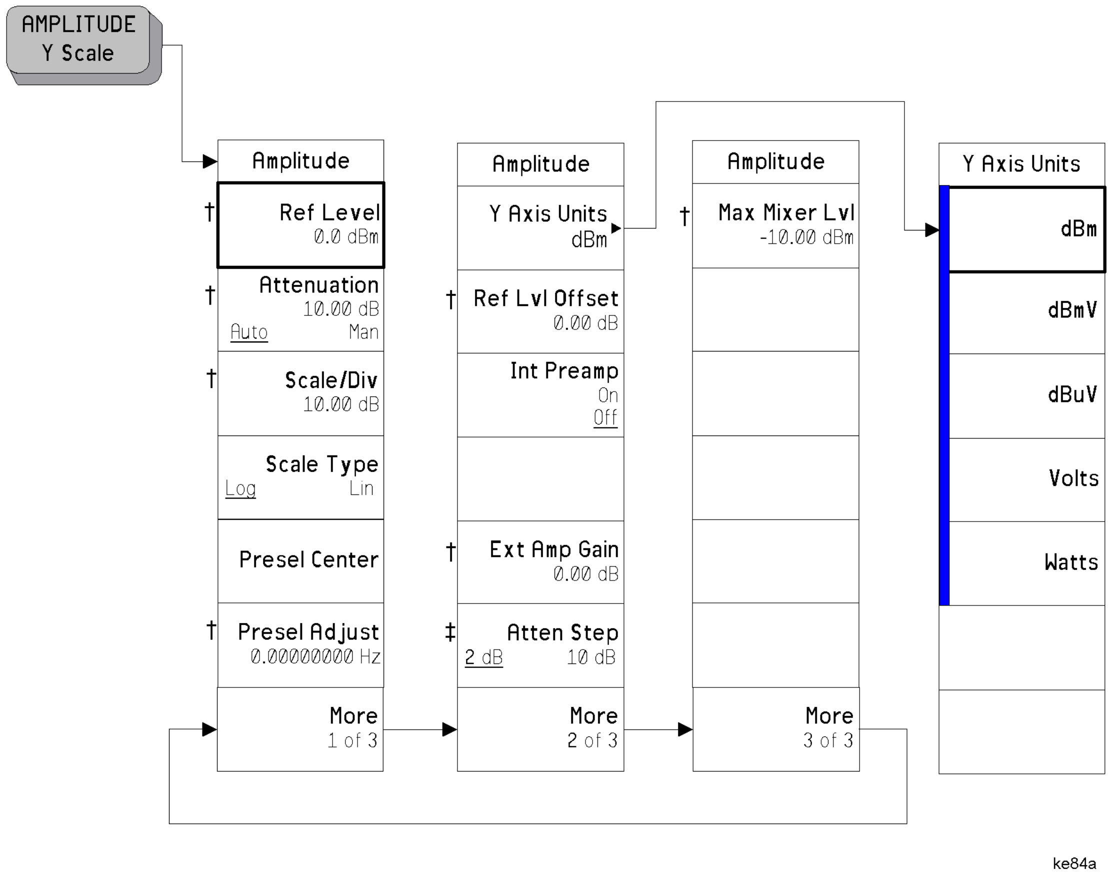
2.1 Alpha Editor Keys 1 of 2



Alpha Editor Keys 2 of 2



2.2 AMPLITUDE Y Scale (See page 65)

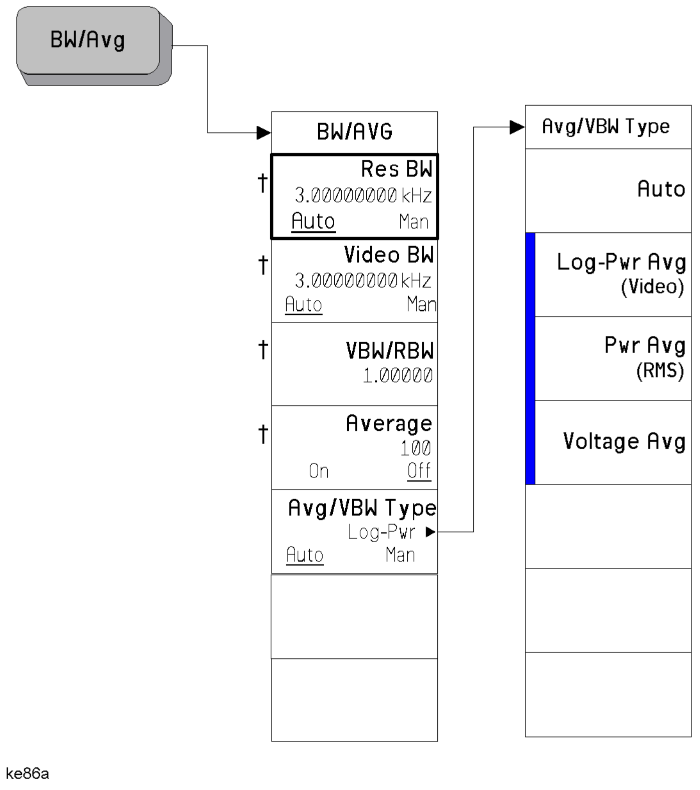


▌ A bar on the left of two or more softkeys indicates that the keys are a set of mutually exclusive choices.

† A dagger to the left of the softkey indicates that when the key is pressed this is an active function.

‡ A double-dagger to the left of the softkey indicates a function that is not always available. It is dependent on other instrument settings.

2.3 BW/Avg (See page 76)

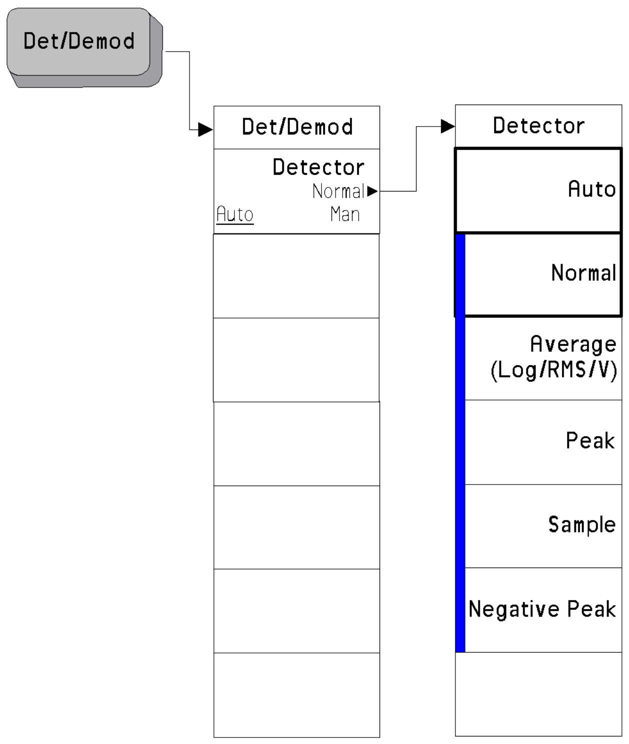


▮ A bar on the left of two or more softkeys indicates that the keys are a set of mutually exclusive choices.

† A dagger to the left of the softkey indicates that when the key is pressed this is an active function.

‡ A double-dagger to the left of the softkey indicates a function that is not always available. It is dependent on other instrument settings.

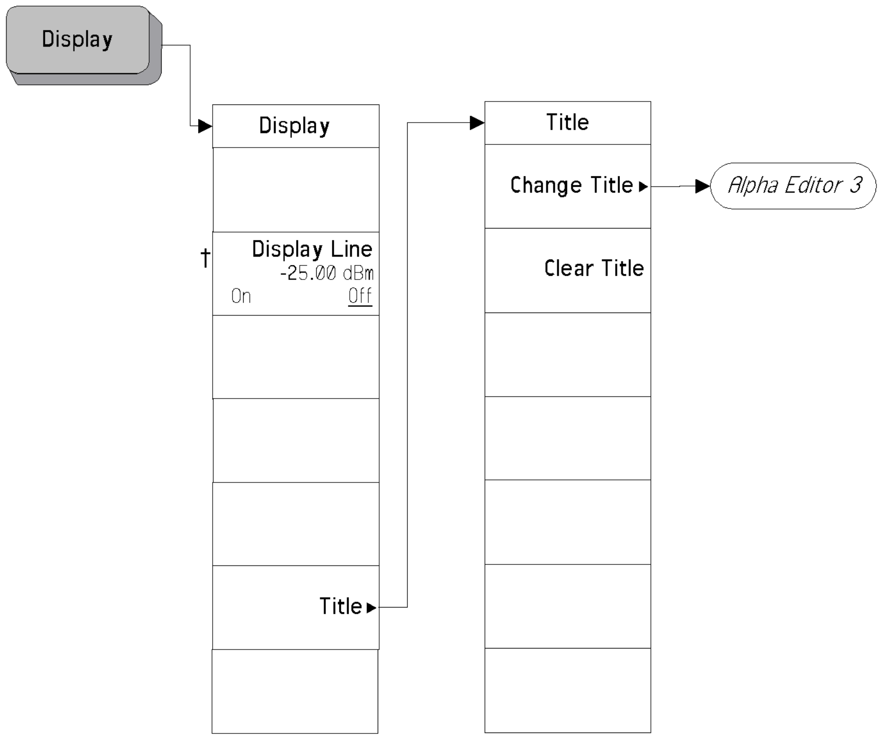
2.4 Det/Demod Key (See page 82)



ke87a

▮ A bar on the left of two or more softkeys indicates that the keys are a set of mutually exclusive choices.
† A dagger to the left of the softkey indicates that when the key is pressed this is an active function.
‡ A double-dagger to the left of the softkey indicates a function that is not always available. It is dependent on other instrument settings.

2.5 Display Key (See page 88)



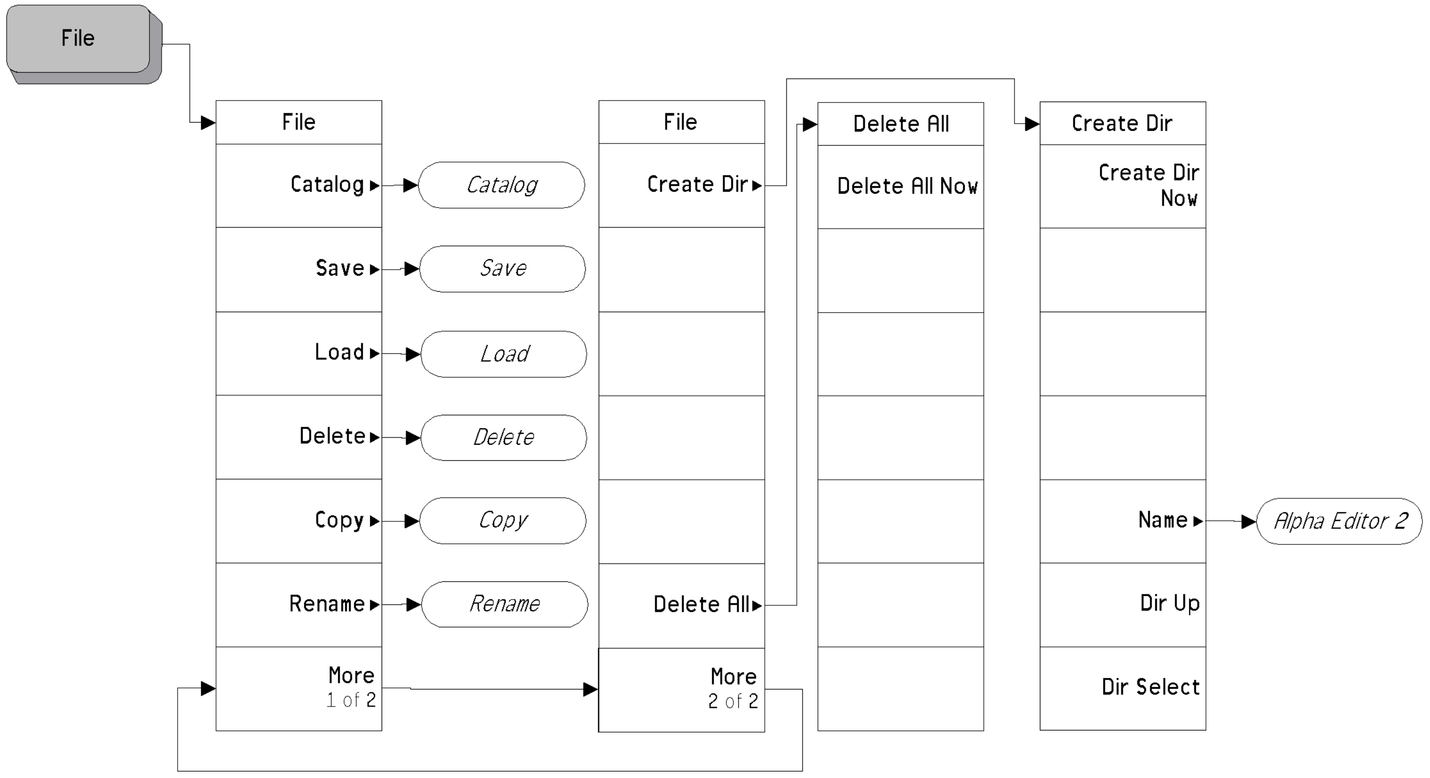
ke88a

▌ A bar on the left of two or more softkeys indicates that the keys are a set of mutually exclusive choices.

† A dagger to the left of the softkey indicates that when the key is pressed this is an active function.

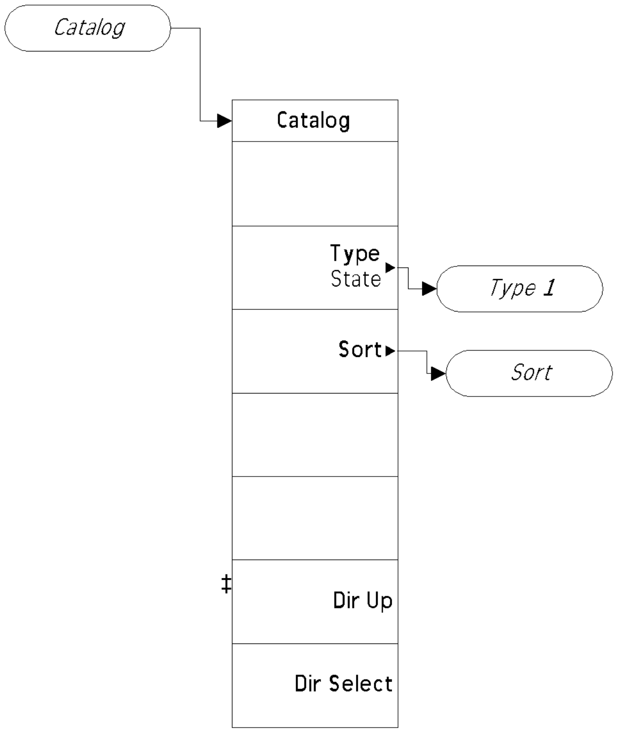
‡ A double-dagger to the left of the softkey indicates a function that is not always available. It is dependent on other instrument settings.

2.6 File Key 1 of 6 (See page 165)



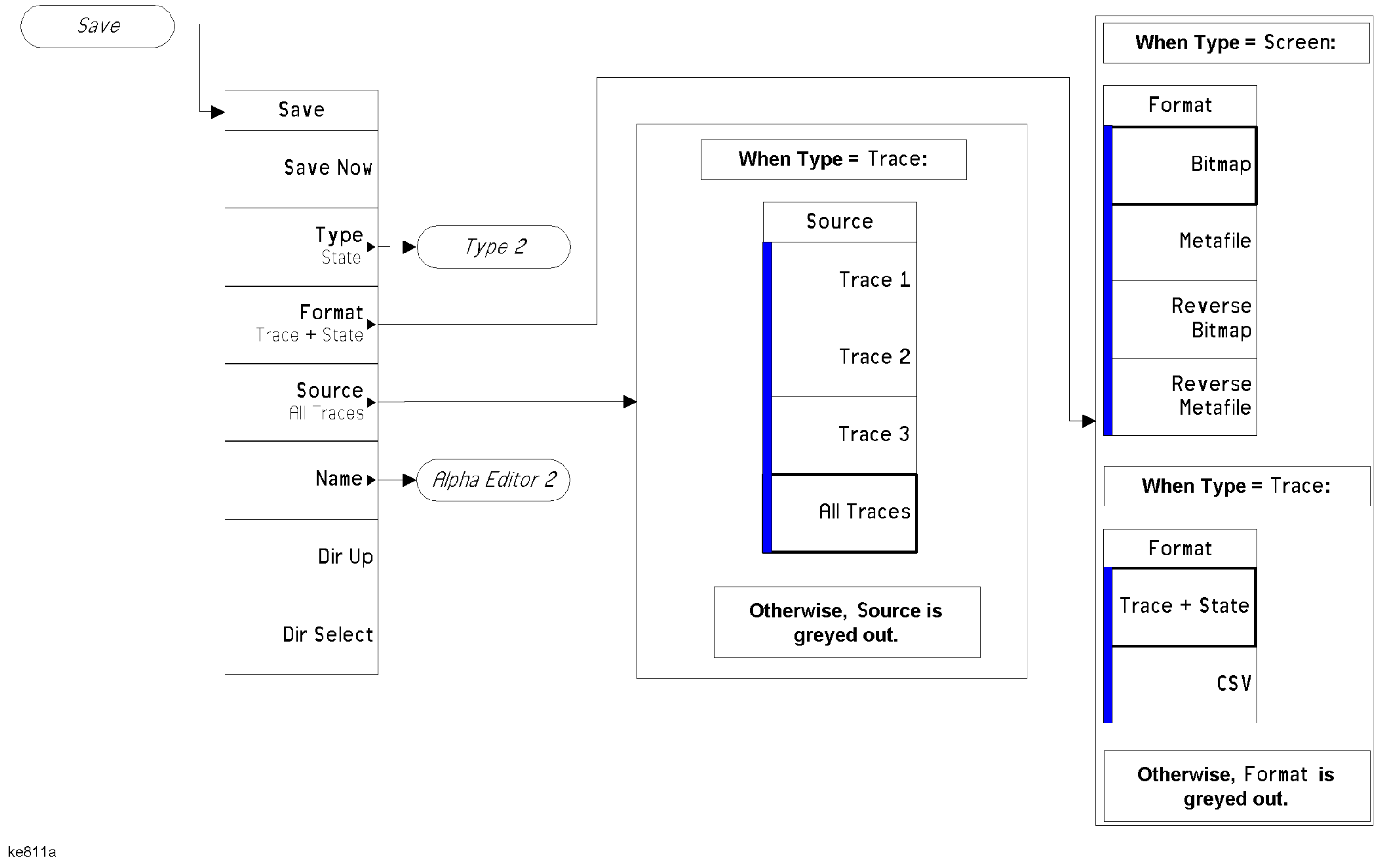
ke89a

File Key 2 of 6 (See page 165)



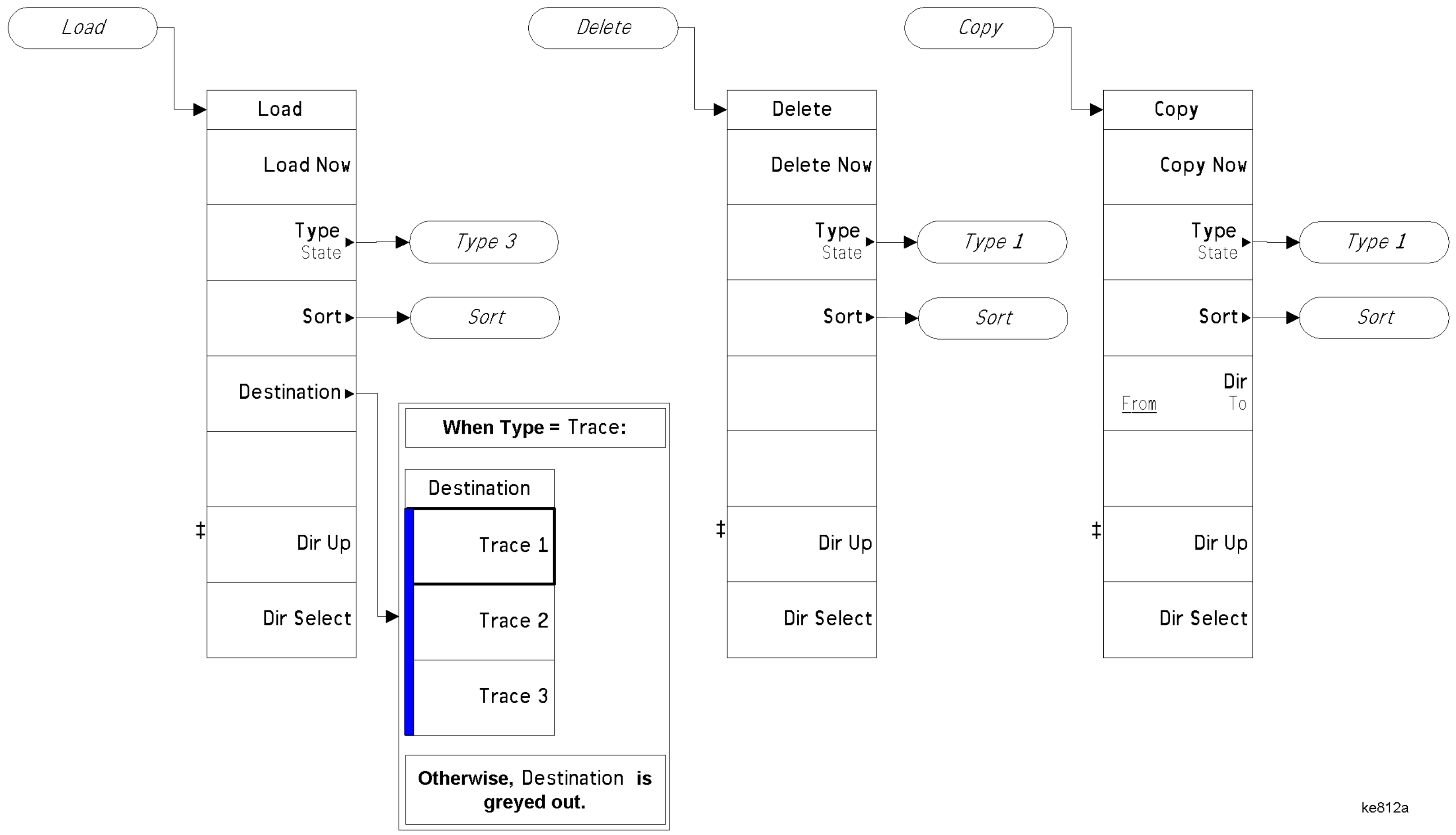
ke810a

File Key 3 of 6 (See page 165)



▮ A bar on the left of two or more softkeys indicates that the keys are a set of mutually exclusive choices.
† A dagger to the left of the softkey indicates that when the key is pressed this is an active function.
‡ A double-dagger to the left of the softkey indicates a function that is not always available. It is dependent on other instrument settings.

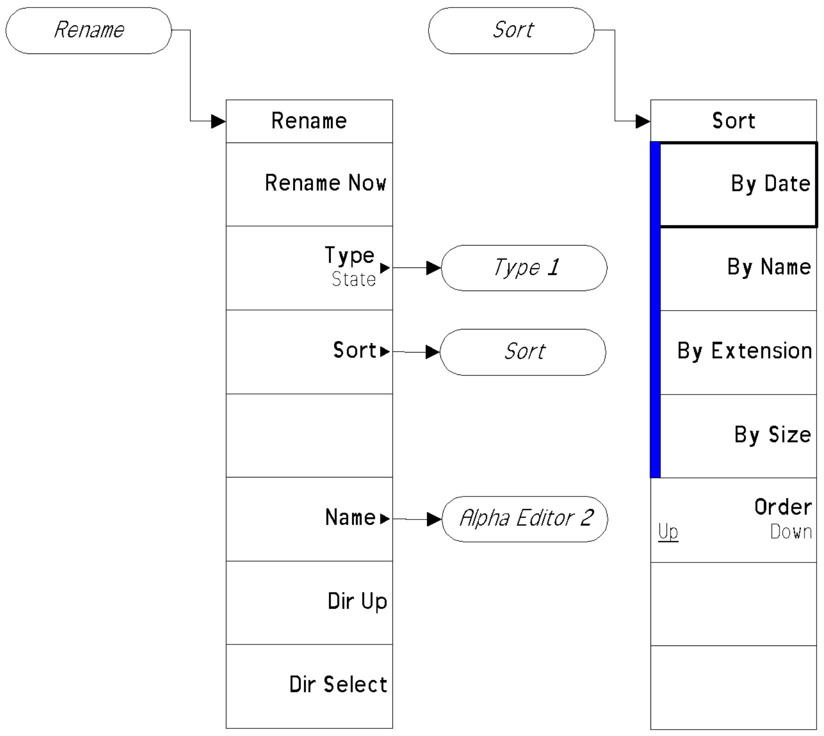
File Key 4 of 6 (See page 165)



🟦 A bar on the left of two or more softkeys indicates that the keys are a set of mutually exclusive choices.
† A dagger to the left of the softkey indicates that when the key is pressed this is an active function.
‡ A double-dagger to the left of the softkey indicates a function that is not always available. It is dependent on other instrument settings.

File Key 1 of 6 (See page 165)

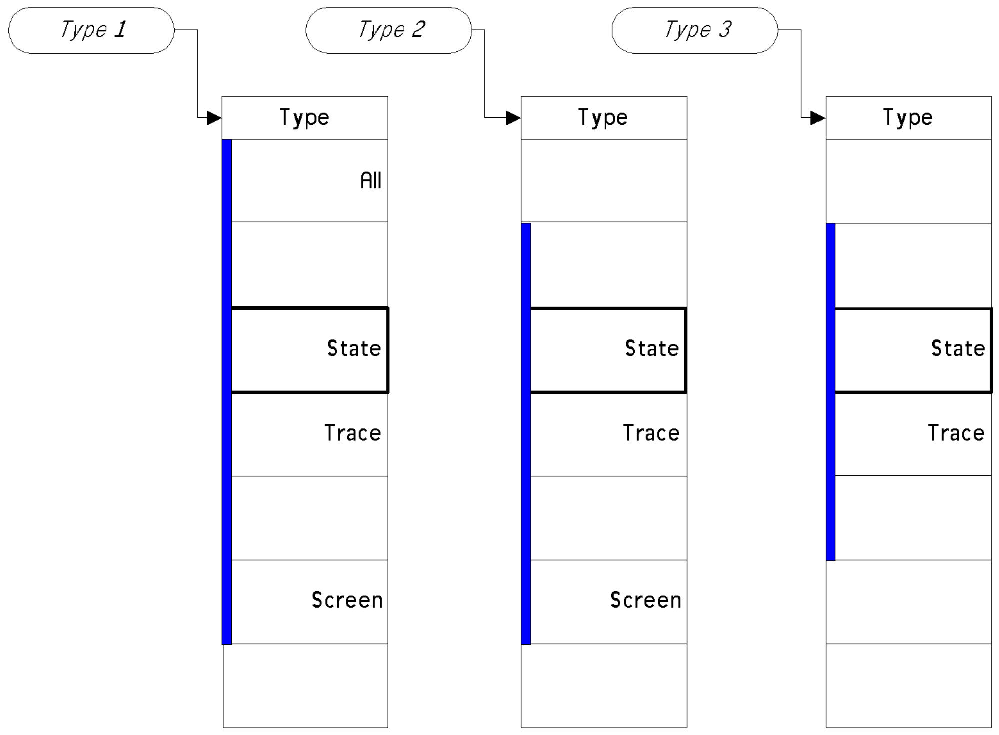
File Key 5 of 6 (See page 165)



ke813a

▮ A bar on the left of two or more softkeys indicates that the keys are a set of mutually exclusive choices.
† A dagger to the left of the softkey indicates that when the key is pressed this is an active function.
‡ A double-dagger to the left of the softkey indicates a function that is not always available. It is dependent on other instrument settings.

File Key 6 of 6 (See page 165)



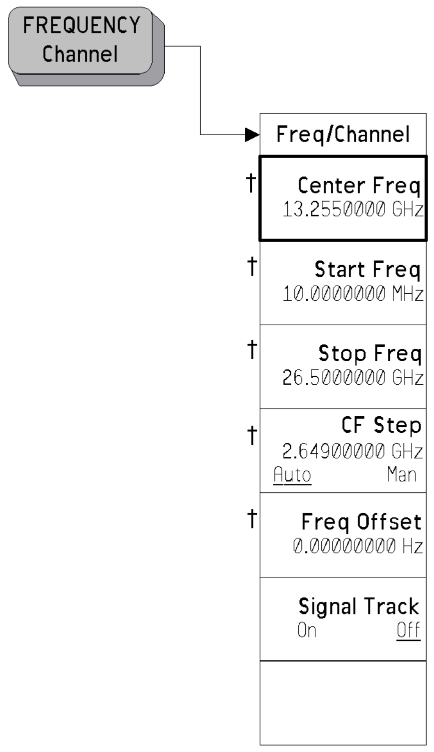
ke814a

▮ A bar on the left of two or more softkeys indicates that the keys are a set of mutually exclusive choices.

† A dagger to the left of the softkey indicates that when the key is pressed this is an active function.

‡ A double-dagger to the left of the softkey indicates a function that is not always available. It is dependent on other instrument settings.

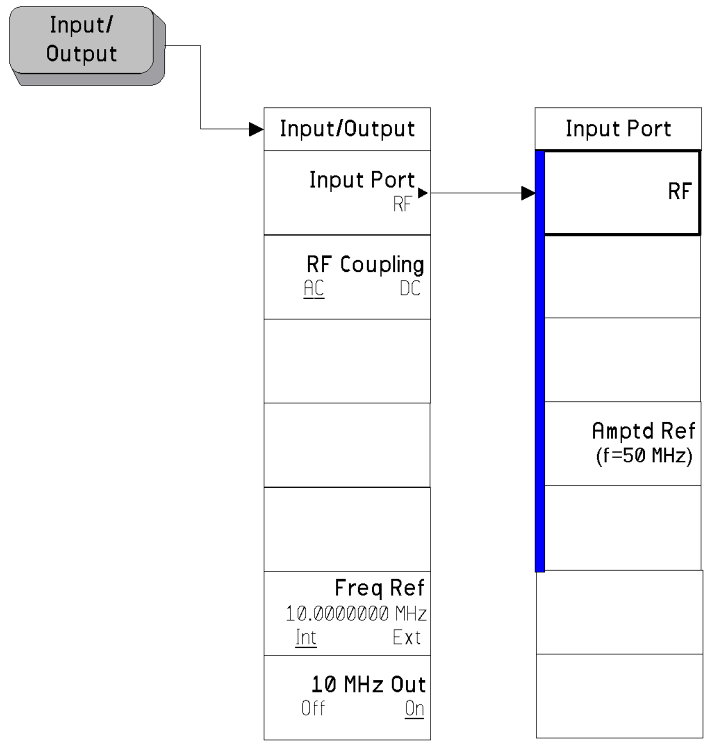
2.7 FREQUENCY Channel Key (See page 90)


FREQUENCY Channel
Freq/Channel
† Center Freq 13.2550000 GHz
† Start Freq 10.0000000 MHz
† Stop Freq 26.5000000 GHz
† CF Step 2.64900000 GHz Auto Man
† Freq Offset 0.00000000 Hz
Signal Track On Off

ke815a

▮	A bar on the left of two or more softkeys indicates that the keys are a set of mutually exclusive choices.
†	A dagger to the left of the softkey indicates that when the key is pressed this is an active function.
‡	A double-dagger to the left of the softkey indicates a function that is not always available. It is dependent on other instrument settings.

2.8 Input/Output Key (See page 94)



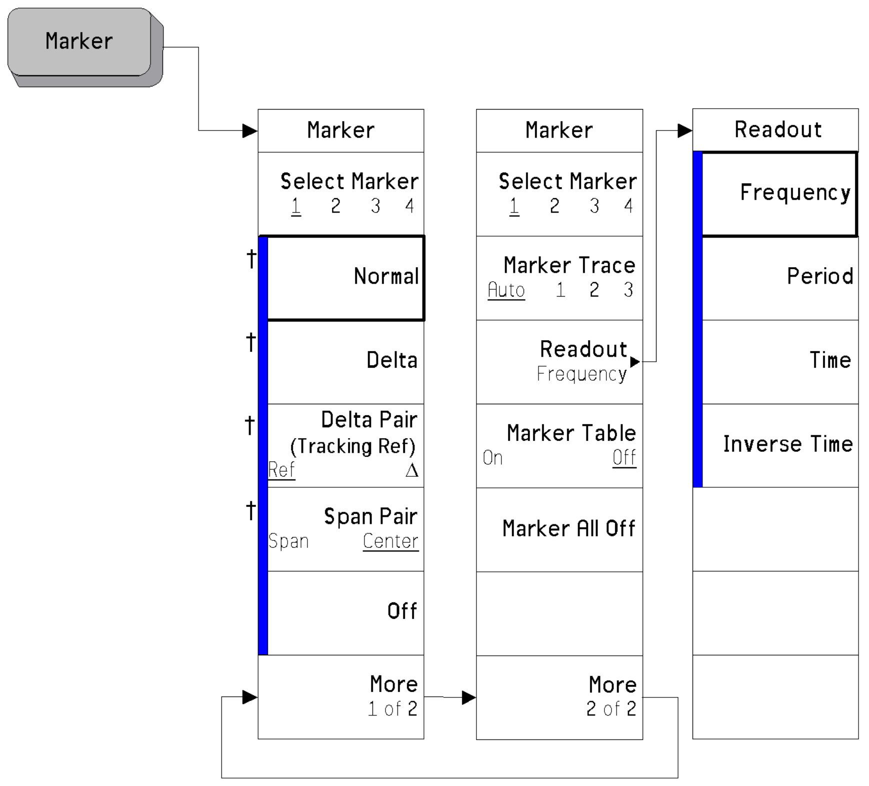
ke816a

▌ A bar on the left of two or more softkeys indicates that the keys are a set of mutually exclusive choices.

† A dagger to the left of the softkey indicates that when the key is pressed this is an active function.

‡ A double-dagger to the left of the softkey indicates a function that is not always available. It is dependent on other instrument settings.

2.9 Marker Key (See page 122)



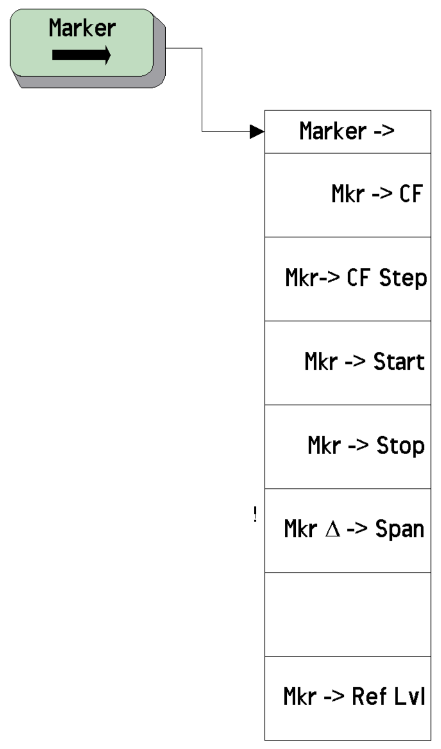
ke817a

▮ A bar on the left of two or more softkeys indicates that the keys are a set of mutually exclusive choices.

† A dagger to the left of the softkey indicates that when the key is pressed this is an active function.

‡ A double-dagger to the left of the softkey indicates a function that is not always available. It is dependent on other instrument settings.

2.10 Marker --> Key (See page 130)



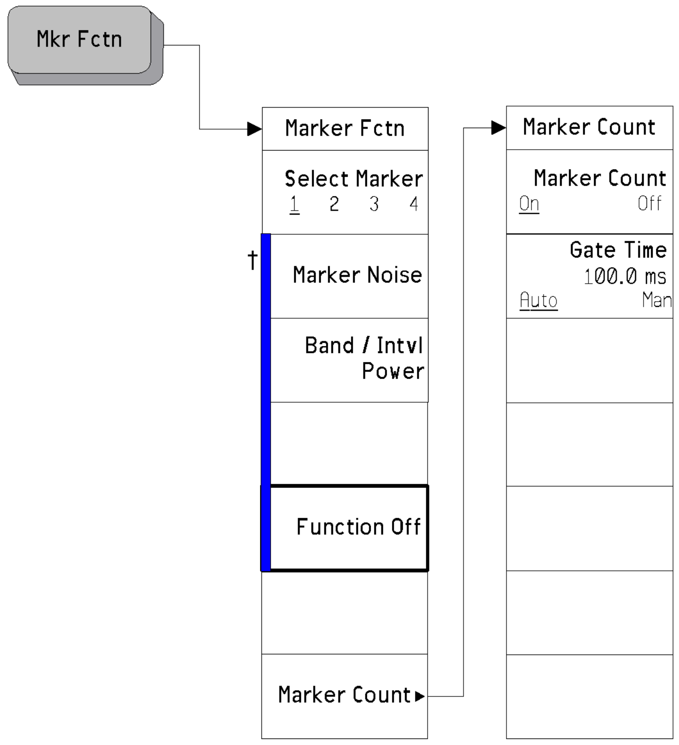
ke918a

▌ A bar on the left of two or more softkeys indicates that the keys are a set of mutually exclusive choices.

† A dagger to the left of the softkey indicates that when the key is pressed this is an active function.

‡ A double-dagger to the left of the softkey indicates a function that is not always available. It is dependent on other instrument settings.

2.11 Mkr Fctn Key (See page 132)



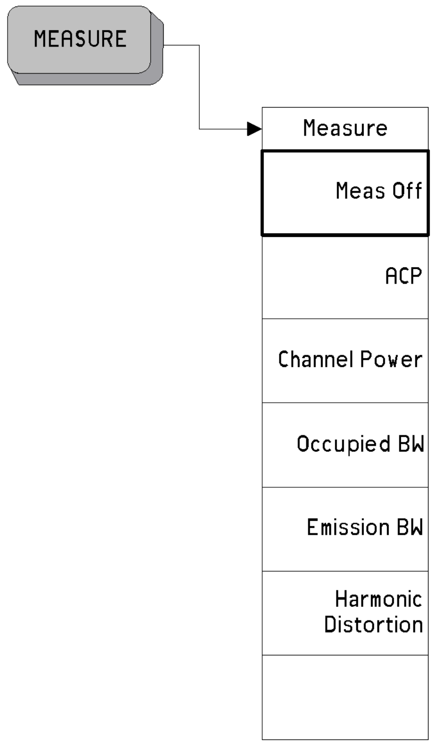
ke821a

▌ A bar on the left of two or more softkeys indicates that the keys are a set of mutually exclusive choices.

† A dagger to the left of the softkey indicates that when the key is pressed this is an active function.

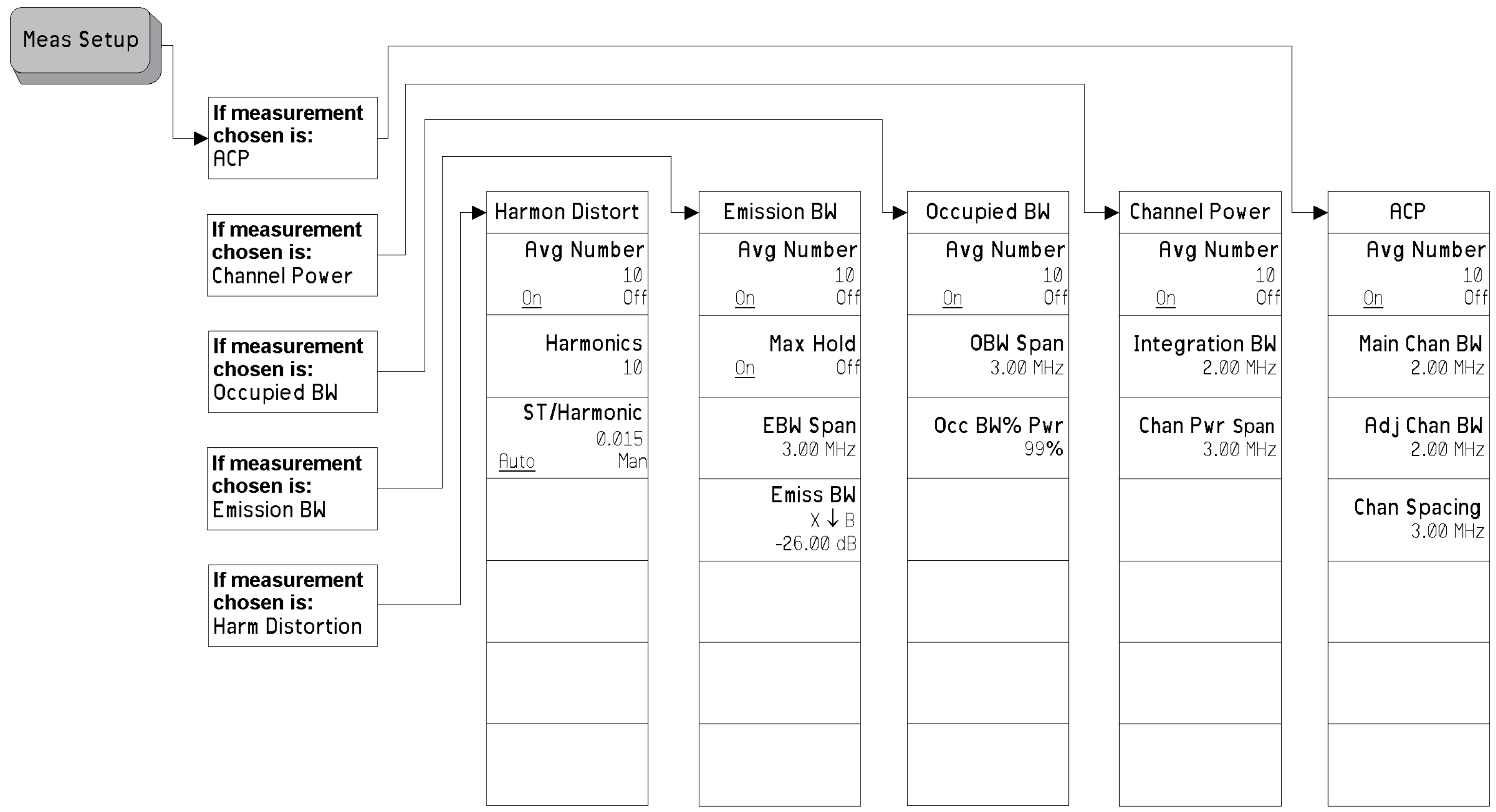
‡ A double-dagger to the left of the softkey indicates a function that is not always available. It is dependent on other instrument settings.

2.12 MEASURE Key (See page 143)



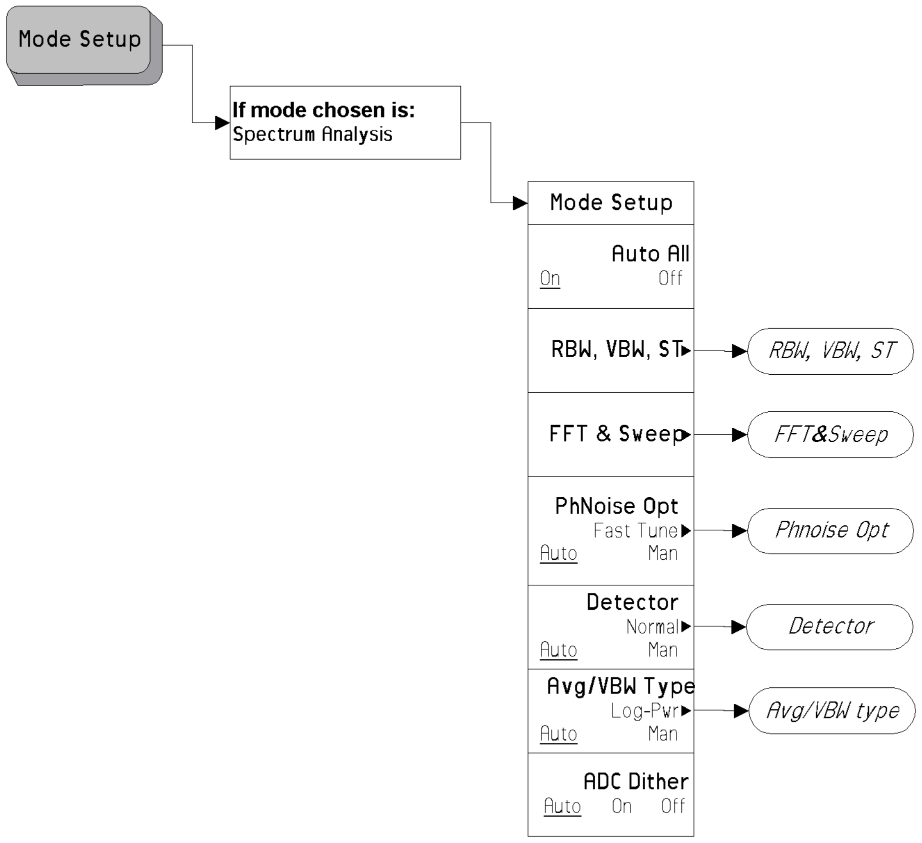
ke819a

2.13 Meas Setup Key (See page 150)



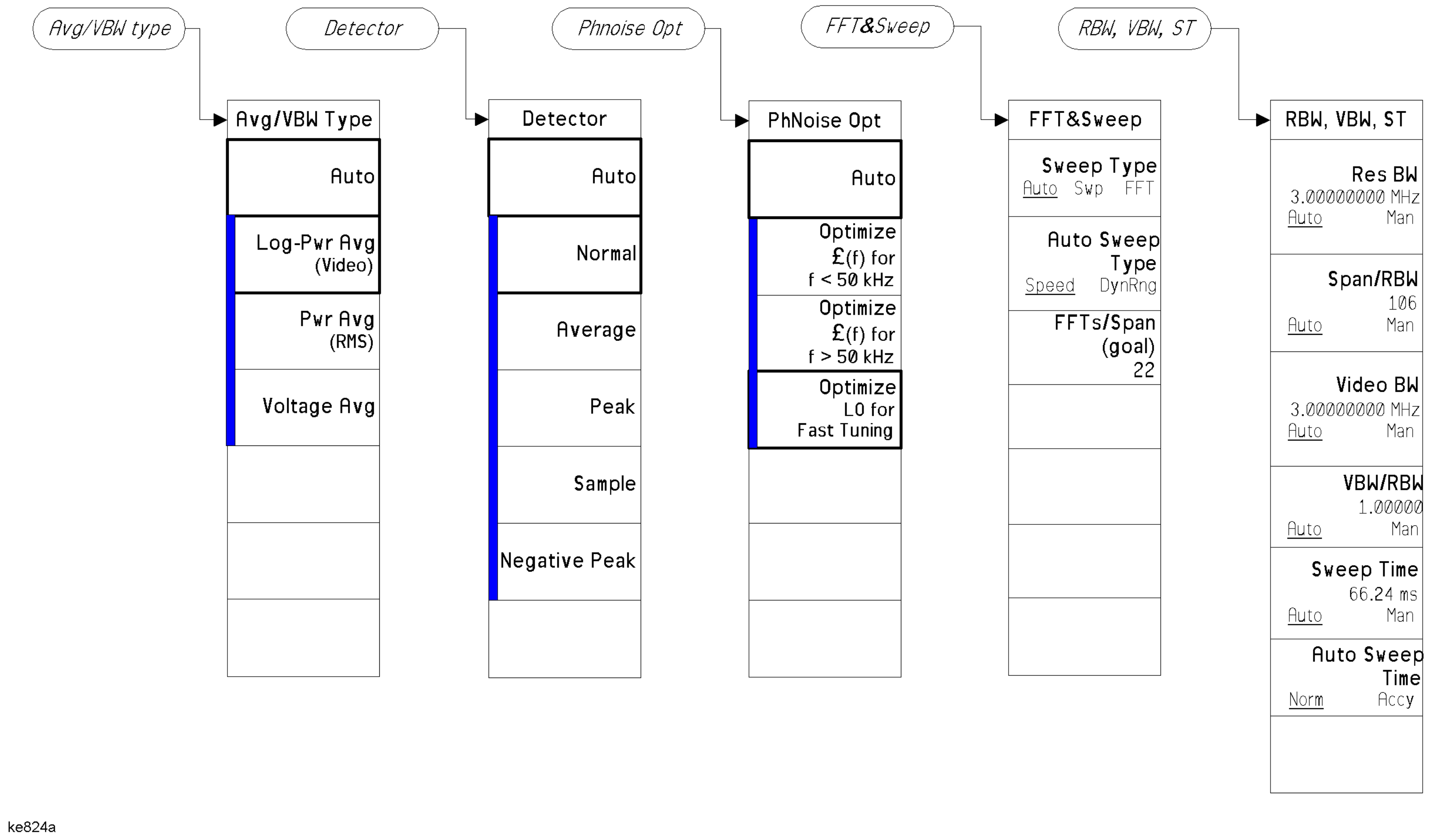
ke820a

2.14 Mode Setup Key 1 of 2 (See page 99)



ke823a

Mode Setup Key 2 of 2 (See page 99)

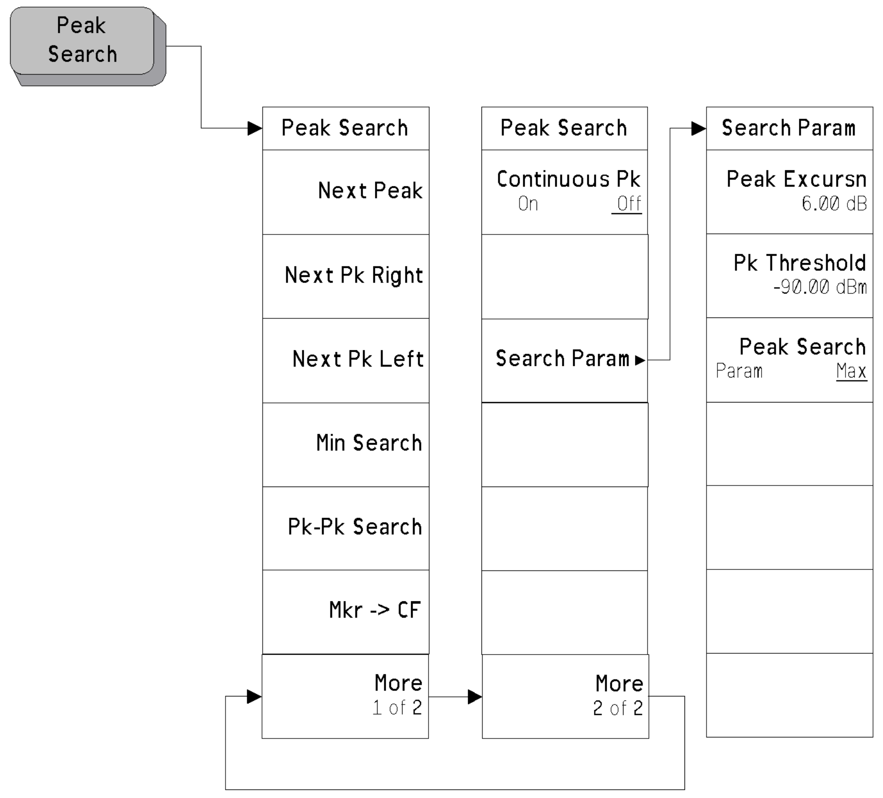


▌ A bar on the left of two or more softkeys indicates that the keys are a set of mutually exclusive choices.

† A dagger to the left of the softkey indicates that when the key is pressed this is an active function.

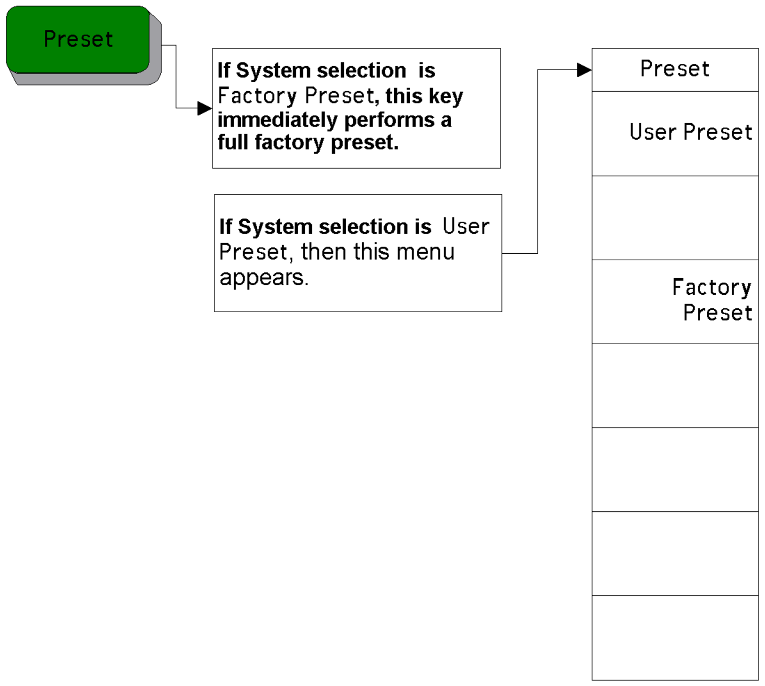
‡ A double-dagger to the left of the softkey indicates a function that is not always available. It is dependent on other instrument settings.

2.15 Peak Search Key (See page 136)



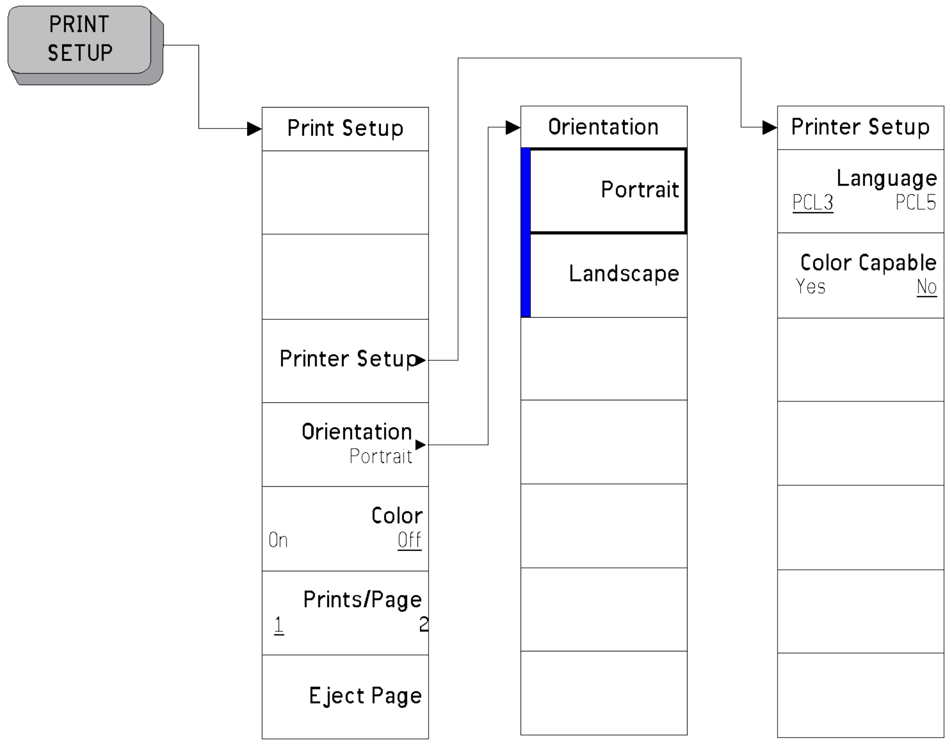
ke825a

2.16 Preset Key (See page 182)


Preset
If System selection is Factory Preset, this key immediately performs a full factory preset.
If System selection is User Preset, then this menu appears.
Preset
User Preset
Factory Preset

ke826a

2.17 Print Setup Key (See page 185)



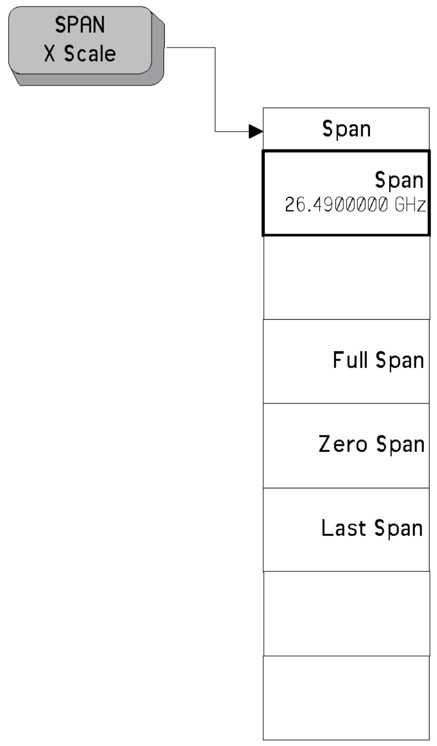
ke828a

▮ A bar on the left of two or more softkeys indicates that the keys are a set of mutually exclusive choices.

† A dagger to the left of the softkey indicates that when the key is pressed this is an active function.

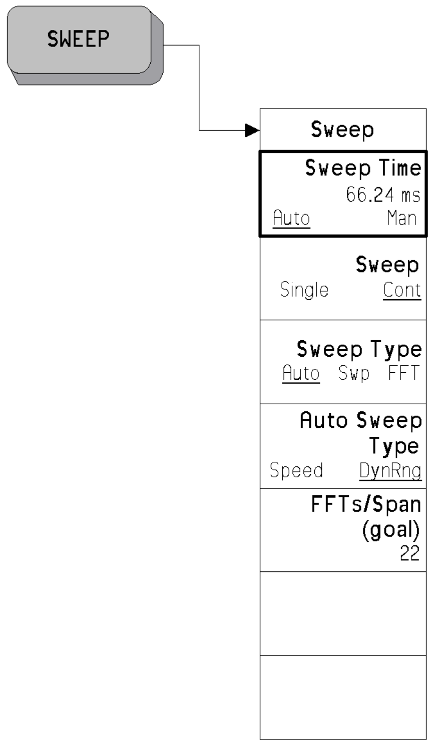
‡ A double-dagger to the left of the softkey indicates a function that is not always available. It is dependent on other instrument settings.

2.18 SPAN X Scale Key (See page 105)



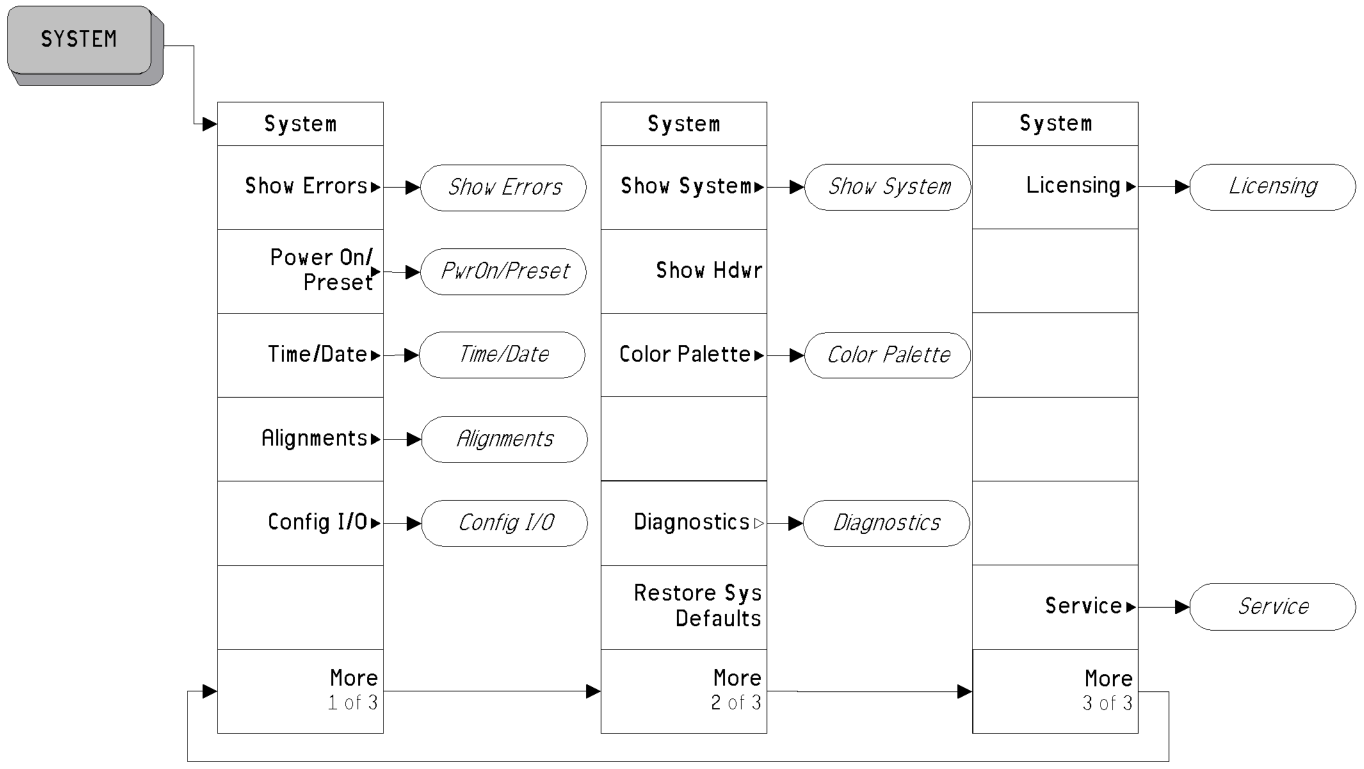
ke832a

2.19 Sweep Key (See page 107)



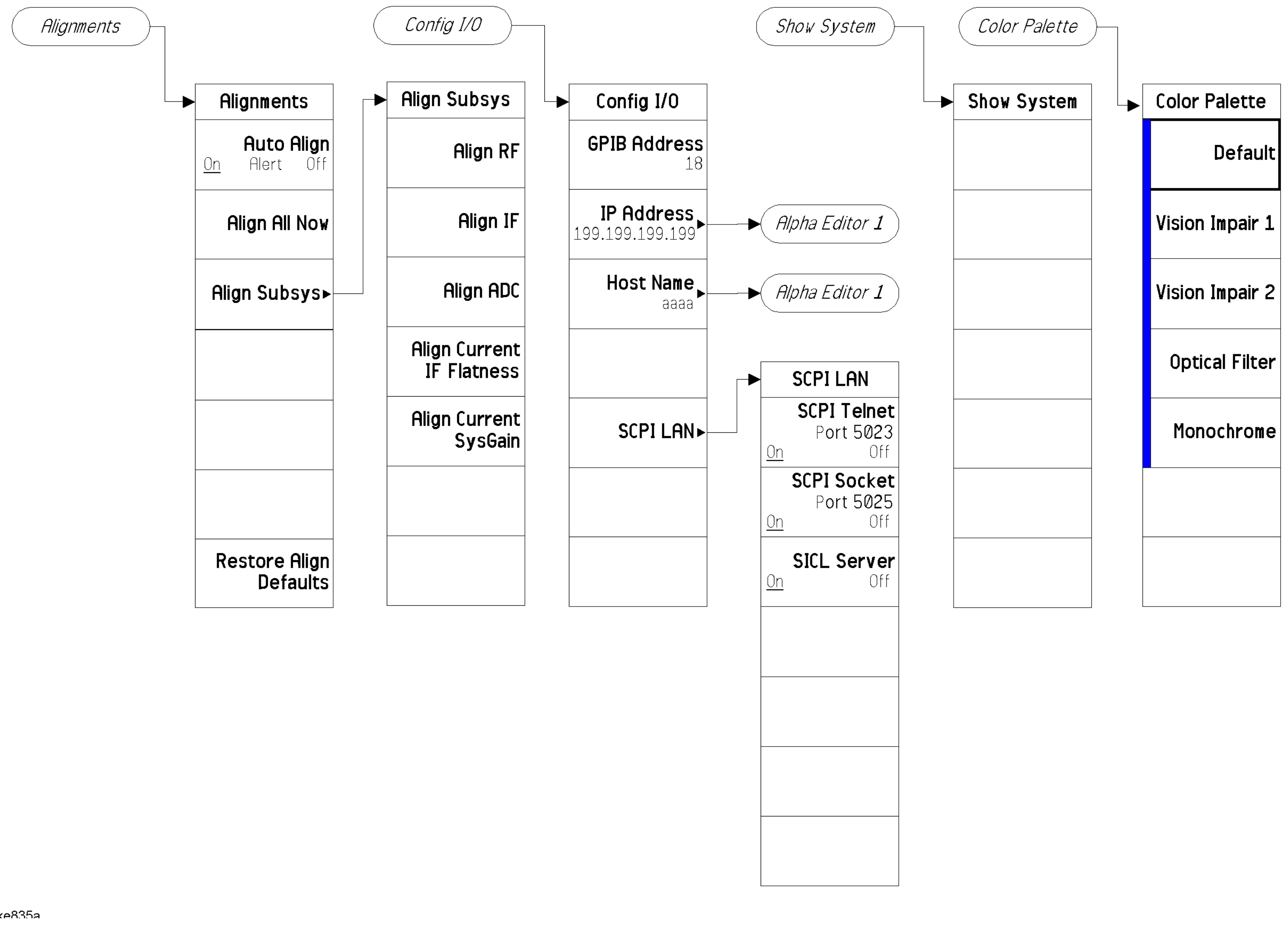
ke833a

2.20 System Key 1 of 4 (See page 191)



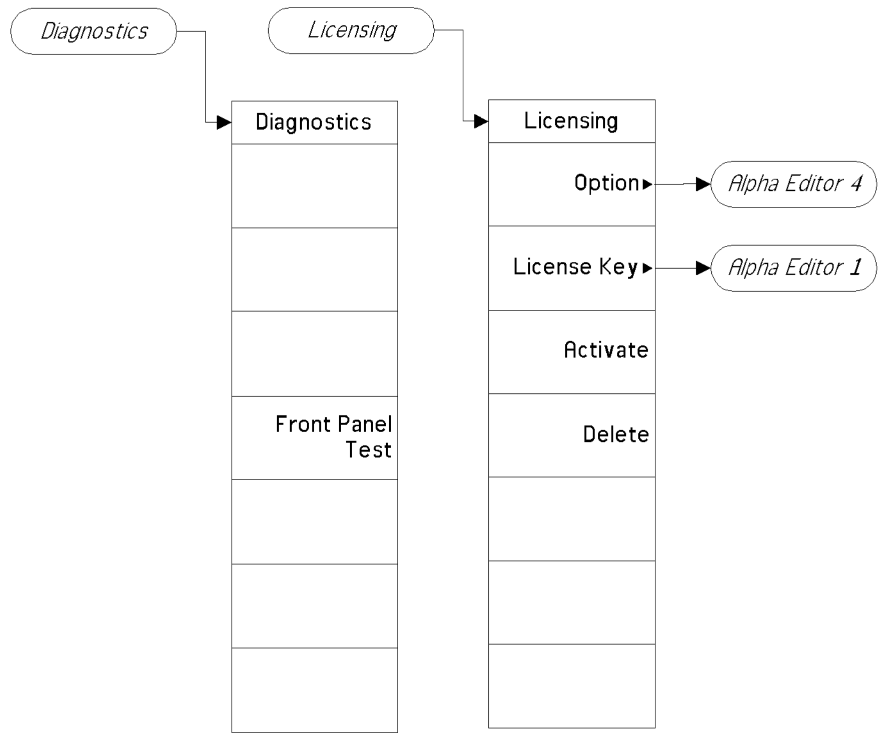
ke834a

System Key 2 of 4 (See page 191)



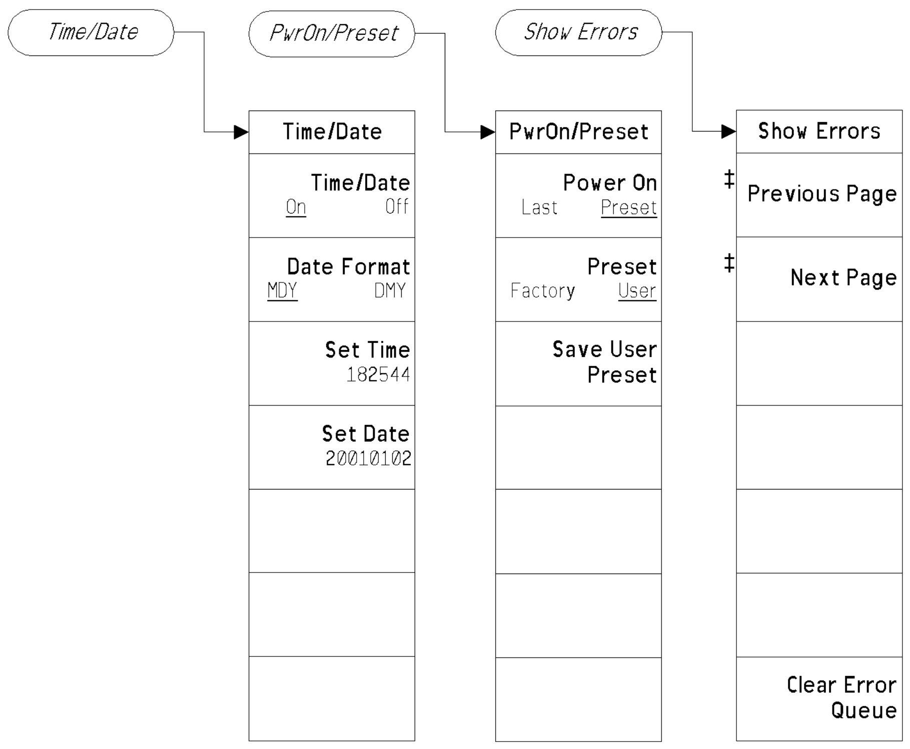
A bar on the left of two or more softkeys indicates that the keys are a set of mutually exclusive choices.

System Key 3 of 4 (See page 191)



ke836a

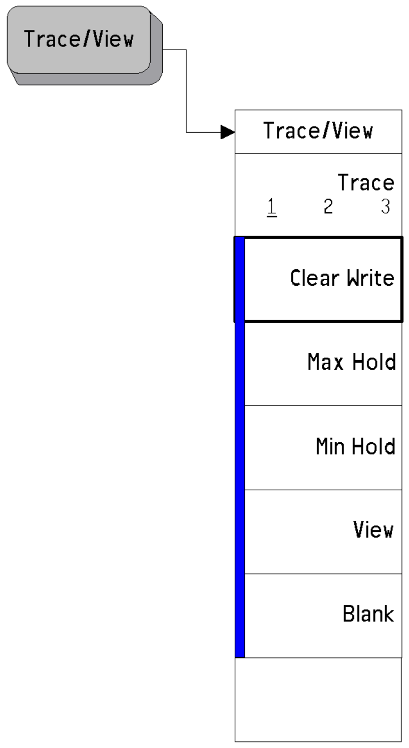
System Key 4 of 4 (See page 191)



ke839a

▌ A bar on the left of two or more softkeys indicates that the keys are a set of mutually exclusive choices.
† A dagger to the left of the softkey indicates that when the key is pressed this is an active function.
‡ A double-dagger to the left of the softkey indicates a function that is not always available. It is dependent on other instrument settings.

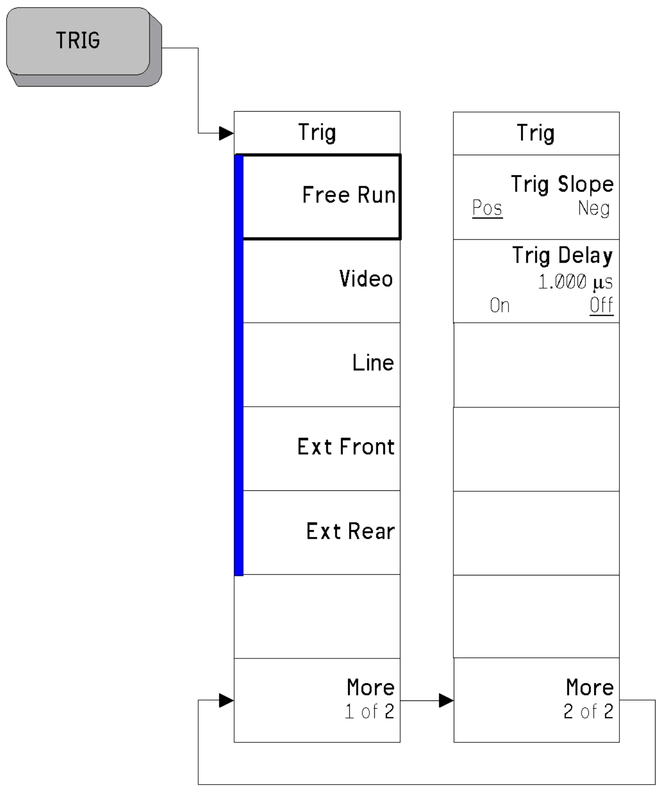
2.21 Trace/View Key (See page 111)



ke840a

▮	A bar on the left of two or more softkeys indicates that the keys are a set of mutually exclusive choices.
†	A dagger to the left of the softkey indicates that when the key is pressed this is an active function.
‡	A double-dagger to the left of the softkey indicates a function that is not always available. It is dependent on other instrument settings.

2.22 Trig Key (See page 114)



ke841a

▌ A bar on the left of two or more softkeys indicates that the keys are a set of mutually exclusive choices.

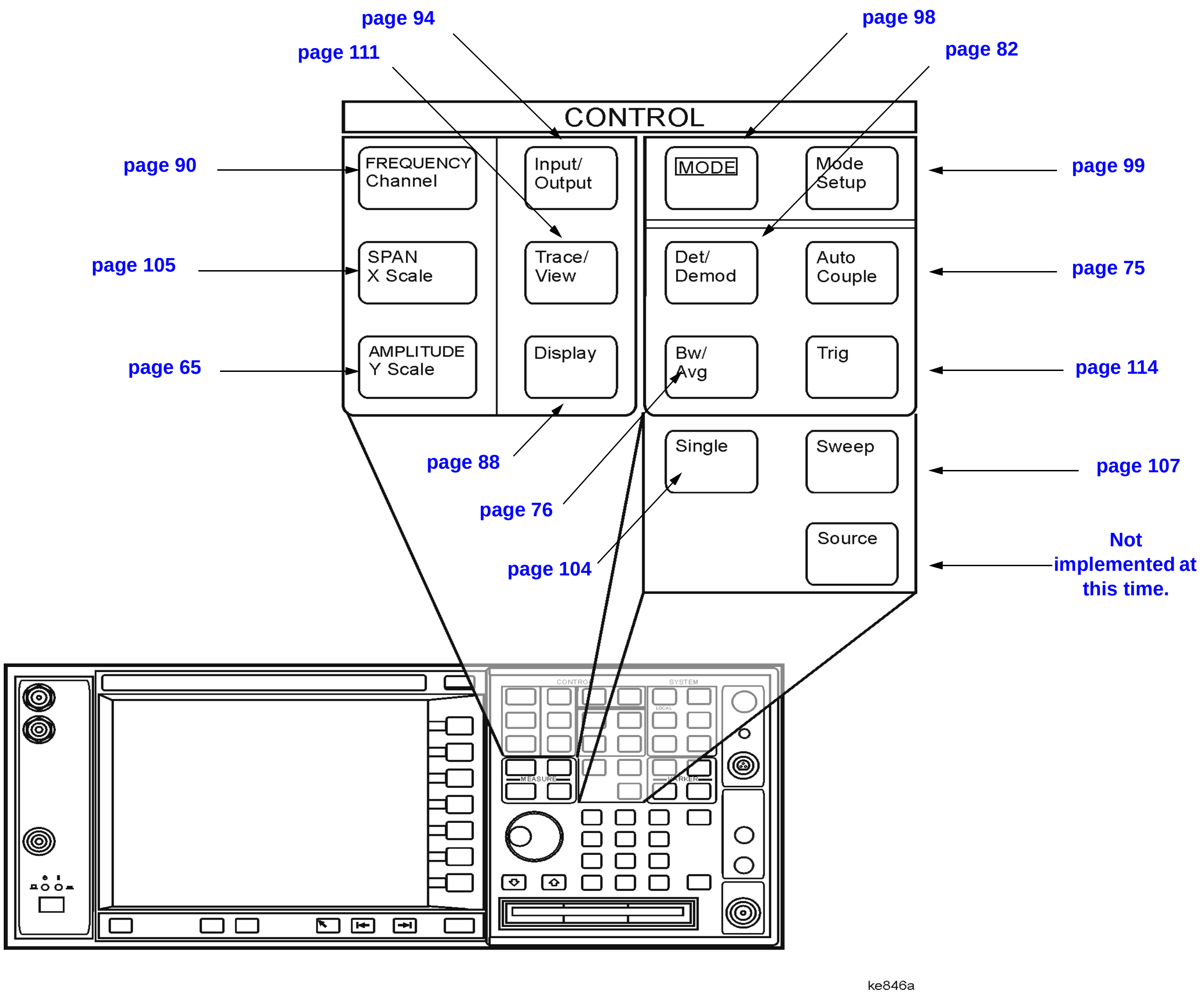
† A dagger to the left of the softkey indicates that when the key is pressed this is an active function.

‡ A double-dagger to the left of the softkey indicates a function that is not always available. It is dependent on other instrument settings.

3 Control Keys

This chapter provides key descriptions and programming information for the Control functions of your analyzer. The front-panel Control functions are listed alphabetically and are described with their associated menu keys. The lower-level menu keys are arranged and described as they appear in your analyzer.

Control Function Keys and Where to Find Them



3.1 AMPLITUDE / Y Scale

Activates the Reference Level function and accesses the Amplitude menu keys. These functions control how data on the vertical (Y) axis is displayed and corrected, and control instrument settings that affect the vertical axis.

3.1.1 Ref Level

Allows you to adjust the power or voltage represented by the top graticule line on the display (the reference level). The current value is indicated by **Ref** in the upper left corner of the display. The Reference Level units are determined by the **Amplitude**, **Y Axis Units** setting. The reference level can be changed using the step keys, the knob, or the numeric keypad.

If the Attenuation setting is lowered, the Reference Level may also have to be lowered by the analyzer to maintain the proper level at the top of the screen. If there is insufficient gain in the system to maintain the original level, the analyzer will automatically decrease the Ref Level by the required amount. If the Attenuation is then increased, the Ref Level does *not* increase back to its previous value.

State Saved:	Saved in instrument state.
Factory Preset and *RST:	0 dBm

Reference level range is determined by the settings of the input attenuator, reference level offset, external amplitude gain, and whether the preamp (Option 1DS) is on or off.

Examples:

–170 dBm to 30 dBm with zero reference level offset

–180 dBm to 20 dBm with 10 dB ext amp gain

–160 dBm to 40 dBm with 10 dB reference level offset

–170 dBm to 0 dBm with preamp on (Option 1DS)

NOTE The input attenuator setting may be affected when the reference level is changed. See **Attenuation**.

Remote Command:

```
:DISPlay:WINDow[1]:TRACe:Y:[SCALe]:RLEVel <ampl>
:DISPlay:WINDow[1]:TRACe:Y:[SCALe]:RLEVel?
```

Example:

```
:DISP:WIND:TRAC:Y:RLEV 20 dbm
```

Sets the reference level to 20 dBm, which then displays in the current Y-Axis Units. For example, if the Y-Axis Units are dBμV, 127 dBμV will be displayed.

3.1.2 Attenuation

Allows you to adjust the input attenuation. Press **Atten Step** to set the attenuation step so that attenuation will change in 2 dB or 10 dB increments. The analyzer input attenuator reduces the power level of the input signal delivered to the input mixer. If set manually, the attenuator is recoupled when **Attenuation (Auto)** is selected.

Attenuation is coupled to Reference Level, so adjusting the Reference Level may change the Attenuation. The analyzer selects an Attenuation setting that is as small as possible while keeping the Ref Level at or below the **Max Mixer Lvl** setting. The current value is indicated by **`Atten`** at the top of the display. A # appears in front of **`Atten`** when **Attenuation (Man)** is selected.

CAUTION To prevent damage to the input mixer, do not exceed a power level of +30 dBm at the input.

To prevent signal compression, keep the power at the input mixer below –3 dBm (10 MHz - 200 MHz), below 3 dBm (200 MHz - 6.6 GHz), and below 0 dBm (6.6 GHz - 26.5 GHz). With the attenuator set to Auto, a signal at or below the reference level results in a mixer level at or below –10 dBm.

State Saved: Saved in instrument state.

Factory Preset and *RST: 10 dB (for Ext Amp Gain of 0 dB), Auto Coupled

Maximum Value: 70 dB

Minimum Value: 0 dB

(To enter a value below 6 dB, you must use the numeric keypad.)

Remote Command:

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

Examples:

```
:POW:ATT 30 dB
:POW:ATT:AUTO ON
```

NOTE The Reference Level setting may be affected when the Attenuation is changed. See **Ref Level**.

3.1.3 Scale/Div

Sets the logarithmic units per vertical graticule division on the display. This function is only available when Scale Type (Log) is selected. When Scale Type (Lin) is selected, Scale/Div is grayed out.

State Saved: Saved in instrument state.

Factory Preset and *RST: 10

Maximum Value: 20 dB

Minimum Value: 0.1 dB

Remote Command:

```
:DISPlay:WINDow[1]:TRACe:Y:[SCALe]:PDIVision <rel_ampl>
:DISPlay:WINDow[1]:TRACe:Y:[SCALe]:PDIVision?
```

Example:

```
:DISP:WIND:TRAC:Y:PDIV 5 DB
```

3.1.4 Scale Type

Chooses a linear or logarithmic vertical scale for the display and for remote data readout.

When **Scale Type (Log)** is selected, the vertical graticule divisions are scaled in logarithmic units. The top line of the graticule is the Reference Level and each vertical division (all 10 divisions are calibrated) represents the value of **Scale/Div**.

When **Scale Type (Lin)** is selected, the vertical graticule divisions are linearly scaled with the reference level value at the top of the display and zero volts at the bottom. Each vertical division of the graticule represents one-tenth of the Reference Level.

The Y Axis Units used for each type of display are set by pressing **Y Axis Units**. The analyzer remembers the settings for both **Log** and **Lin**.

State Saved: Saved in instrument state.

Factory Preset and *RST: Log

Remote Command:

```
:DISPlay:WINDow[1]:TRACe:Y:[SCALe]:SPACing LINear|LOGarithmic
:DISPlay:WINDow[1]:TRACe:Y:[SCALe]:SPACing?
```

Example:

```
:DISP:WIND:TRAC:Y:SPAC LOG
```

3.1.5 Presel Center

Adjusts the centering of the preselector filter to optimize the amplitude accuracy at the active marker frequency. **Presel Center** only functions when measuring signals ≥2.85 GHz in band 1 and higher bands (the instrument is in band 1 when the stop frequency is >3 GHz). If no marker is on when **Presel Center** is pressed, the analyzer turns on the currently selected marker and places it on the peak signal. If a marker is already enabled, it must be placed at the signal peak before activating this function. The span must be less than 1.123 GHz, or the message, `Preselector centering failed. Narrow the span then try again`, will appear on the display alerting you to narrow the span.

NOTE If the signal is noise-like, the algorithm will not function properly.

Remote Command:

`[:SENSe]:POWer[:RF]:PCENter`

Example:

```
:POW:PCEN
```

3.1.6 Presel Adjust

Allows direct (manual) adjustment of the preselector filter frequency to optimize its response on the signal of interest. This function is only available when measuring signals ≥2.85 GHz in band 1 and higher bands.

State Saved: Saved in instrument state.

Factory Preset and *RST: 0 Hz

Maximum Value: 250 MHz

Minimum Value: –250 MHz

Remote Command:

`[:SENSe]:POWer[:RF]:PADJust <freq>`

`[:SENSe]:POWEr[:RF]:PADJust?`

Example:

```
:POW:PADJ 500
```

3.1.7 Y Axis Units

Accesses the menu keys that allow you to change the vertical (Y) axis amplitude units. The analyzer retains your entered **Y Axis Units** for both **Log** and **Lin** scale types. For example, if **Scale Type** has been set to **Log**, and you set **Y Axis Units** to **dBm**, pressing **Scale Type (Log)** will set your **Y Axis Units** to **dBm**. If **Scale Type** has been set to **Lin**, and you set **Y Axis Units** to **Volts**, pressing **Scale Type (Lin)** will set your **Y Axis Units** to **Volts**. Pressing **Scale Type (Log)** again will then set your Y Axis units back to dBm.

Y Axis Units, in conjunction with the **Scale Type**, affect how the data is read off the display, markers, and over the remote interface. When using the remote interface no units are returned, so you must know what the Y-Axis units are to interpret the results

Set the following:
  Scale Type (Log)
  Y Axis Units, dBm
  Scale/Div, 1 dB
  Ref Level, 10 dBm

This sets the top line to 10 dBm and each vertical division represents 1 dB. Thus, if a point on trace 1 is on the fifth graticule line from the top, it represents 5 dBm and will read out remotely as 5.

Set the following:
  Scale Type (Lin)
  Y Axis Units, Volts
  Ref Level, 100 Volts (10 V/div)

This sets the top line to 100 V and the bottom line to 0 V, so each vertical division represents 10 V. Thus, if a point on trace 1 is on the fifth graticule line from the top, it represents 50 V and will read out remotely as 50.

State Saved:	Saved in instrument state.
Factory Preset and *RST:	For **Scale Type (Log)** = dBm
	For **Scale Type (Lin)** = Volts

Remote Command:

```
:UNIT:POWer DBM|DBMV|DBUV|V|W
:UNIT:POWer?
```

Example:

```
:UNIT:POW dBmV
```

3.1.7.1 dBm

Sets the amplitude units to dBm. The units are displayed after the reference level value in the upper left corner of the display.

Remote Command:

See “Y Axis Units” on page 69.

Example:

```
:UNIT:POW DBM
```

3.1.7.2 dBmV

Sets the amplitude units to dBmV. The units are displayed after the reference level value in the upper left corner of the display.

Remote Command:

See “Y Axis Units” on page 69.

Example:

```
 :UNIT:POW DBMV
```

3.1.7.3 dBμV

Sets the amplitude units to dBμV. The units are displayed after the reference level value in the upper left corner of the display.

Remote Command:

See “Y Axis Units” on page 69.

Example:

```
:UNIT:POW DBUV
```

3.1.7.4 Volts

Sets the amplitude units to volts. The units are displayed after the reference level value in the upper left corner of the display.

Remote Command:

See “Y Axis Units” on page 69.

Example:

```
:UNIT:POW V
```

3.1.7.5 Watts

Sets the amplitude units to watts. Displayed after the reference level value in the upper left corner of the display.

Remote Command:

See "Y Axis Units" on page 69.

Example:

```
:UNIT:POW W
```

3.1.8 Ref Lvl Offset

Adds an offset value to the displayed reference level. The reference level is the amplitude power or voltage represented by the top graticule line on the display. Reference-level offsets are only entered by using the numeric keypad or programming commands. The knob and step keys are not active.

Offsets are used when gain or loss occurs between a device under test and the analyzer input. Thus, the signal level measured by the analyzer may be thought of as the level at the input of an external amplitude conversion device. Entering an offset does not affect the trace position or attenuation value.

The sum of the reference level offset and the reference level is clipped to the range –327.6 dB 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, the reference level value range can be initially set to values from –170 dBm to 30 dBm with no reference level offset. If the reference level is first set to –20 dBm, then the reference level offset can be set to values of –327.6 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.

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 –200 dBm to 0 dBm. The reference level is "clamped" at 0 dBm because its positive value of 30 dBm is reached at 0 dBm with an offset of –30 dB. Its own positive amplitude limit applies.

If the reference level offset is first set to 30 dB, the reference level can be set to values of –140 dBm to 60 dBm. Again, the positive amplitude limit of reference level alone, is factored into the resultant combined limit.

When an amplitude offset is entered, the offset value appears on the left side of the display under **`Offst`** (as opposed to frequency offsets which appear at the bottom of the display.) To eliminate an offset, press **Ref Lvl Offst**, **0**, and **dB**.

State Saved:	Saved in instrument state.
Factory Preset and *RST:	0.0 dB
Maximum Value:	327.6 dB
Minimum Value:	–327.6 dB

Remote Command:

`:DISPlay:WINDow[1]:TRACe:Y:[SCALe]:RLEVel:OFFSet <rel_power>` (in dB)

`:DISPlay:WINDow[1]:TRACe:Y:[SCALe]:RLEVel:OFFSet?`

Example:

```
:DISP:WIND:TRAC:Y:RLEV:OFFS 12.7
```

Sets the Ref Level Offset to 12.7 dB. The only valid suffix is dB. If no suffix is sent, dB will be assumed.

3.1.9 Int Preamp

(Option 1DS only.) Turns the internal preamp on and off. When the preamp is on, an automatic adjustment compensates for the gain of the preamp so that displayed amplitude readings still accurately reflect the value at the analyzer input connector. The preamp is switched off for frequencies above 3 GHz, and the correction is not applied.

State Saved: Saved in instrument state.

Factory Preset and *RST: Off

Remote Command:

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

`[:SENSe]:POWer[:RF]:GAIN[:STATe]?`

Example:

```
:POW:GAIN ON
```

3.1.10 Ext Amp Gain

Compensates for external gain/loss. The function is similar to the **Ref Lvl Offset** function, however Attenuation is coupled to the **Ext Amp Gain** function (10 dB of Attenuation is added for every 10 dB of **External Amp Gain**). The gain is subtracted from the amplitude readout so that the displayed signal level represents the signal level at the input of the external device.

Gains may only be entered with the numeric keypad or programming commands, not the knob or step keys.

State Saved: Saved in instrument state.

Factory Default: 0 dB

NOTE **Ext Amp Gain** is not affected by Factory Preset or power cycle. It can be reset to the factory default by pressing **System**, **Restore Sys Defaults**.

Maximum Value: 81.9 dB

Minimum Value: –81.9 dB

Remote Command:

`[:SENSe]:CORRection:OFFSet[:MAGNitude] <relative_power>` (in dB)

`[:SENSe]:CORRection:OFFSet[:MAGNitude]?`

Example:

```
:CORR:OFFS:MAGN 7.3 DB
```

Sets the Ext Amp Gain to 7.3 dB. The only valid suffix is dB. If no suffix is sent, dB is assumed.

3.1.11 Attn Step

Permits the selection of 2 dB or 10 dB step resolution for input attenuation.

State Saved:	Saved in instrument state.
Factory Preset and *RST:	2 dB

Remote Command:

`[:SENSe]:POWer[:RF]:ATTenuation:STEP[:INCRement] <integer>` (in dB)

`[:SENSe]:POWer[:RF]:ATTenuation:STEP[:INCRement]?`

Example:

```
:POW:ATT:STEP 2
```

Sets the Attenuation to 2 dB. The only valid suffix is dB. If no suffix is sent, dB is assumed.

If a value >5 is entered, 10 is used.

If a value ≤5 is entered, 2 is used.

3.1.12 Max Mixer Lvl

Enables you to set the input mixer level so that the highest signal that can be displayed is set at the reference level (top of screen). The level input to the mixer equals the reference level minus the attenuator setting. As the reference level changes, the input attenuator setting changes to ensure that a signal at the reference level does not exceed the Max Mixer Level.

State Saved:	Saved in instrument state.
Factory Preset and *RST:	–10 dBm
Maximum Value:	–10 dBm
Minimum Value:	–100 dBm

Remote Command:

`[:SENSe]:POWer[:RF]:MIXer:RANGe[:UPPer] <power>`

`[:SENSe]:POWer[:RF]:MIXer:RANGe[UPPer]?`

Example:

```
POW:MIX:RANG -15 dBm
```

3.2 Auto Couple

Coupled functions are functions that are linked. When Auto Couple is pressed, the analyzer automatically couples instrument settings for accurate measurements and optimum dynamic range. Changing one function changes the function that is coupled to it. When a function is coupled, it is in the **Auto** state. When it is uncoupled it is in the **Man** state. When a function is in the **Man** state, a # will appear next to its annotation on the display.

Coupled functions are affected depending on how they are coupled. VBW is *coupled to* RBW, for example, so changing RBW affects VBW, but changing VBW does not affect RBW. Changing VBW puts it in **Man** and causes a **#** to appear in front of the VBW annotation. When VBW is in **Man** it is unaffected by RBW. Pressing **Auto Couple** recouples VBW to RBW.

You can examine and change most of the coupled functions by pressing **Mode Setup**.

NOTE Although **Marker Count**, **Gate Time**, and **Marker Trace** have **Auto** settings, they are not affected by **Auto Couple**.

Dependencies/ Couplings: The following list of analyzer functions can be automatically coupled:

Resolution BW coupled to **Span**
Video BW coupled to **RBW**
Sweep Time coupled to **RBW**, **VBW**, **Detector**
Center Freq step coupled to **Span** in swept spans, to **RBW** in zero span
Attenuation coupled to **Ref Level**, **Ext Amp Gain**
Span/RBW ratio (approximately 106:1 in Auto)
Auto Sweep Time (Normal in Auto)
Sweep Type coupled to **RBW**
Auto Sweep Type (Dynamic Range in Auto)
PhNoise Opt (phase noise optimization) coupled to **RBW**, **Span**
Detector coupled to Marker functions, **Average On Off**, **Max Hold**, **Min Hold**
Avg/VBW Type coupled to Marker functions, **Detector**, **Scale Type**
ADC Dither coupled to **Sweep Type**, **Span**, **RBW**

Saved State: Saved in instrument state.

Factory Preset and *RST: All functions coupled

Remote Command:

```
:COUPle ALL|NONE
```

`NONE` sets all the functions to the manual (not coupled) mode.

`ALL` puts all the functions into the auto coupled mode.

Example:

```
:COUP ALL
```

3.3 BW/Avg

Activates the Resolution Bandwidth function and accesses the menu keys that control both the bandwidth and averaging functions.

3.3.1 Res BW

Allows you to select the 3.01 dB resolution bandwidth (RBW) of the analyzer from 1 Hz to 3 MHz in approximately 10% steps plus bandwidths of 4, 5, 6, or 8 MHz. If an unavailable bandwidth is entered with the numeric keypad, the closest available bandwidth is selected.

Sweep time is coupled to **RBW**. As the RBW is changed (if the sweep time is set to **Auto**), the sweep time is changed to maintain amplitude calibration. Video bandwidth (VBW) is also coupled to RBW. As the resolution bandwidth changes, the video bandwidth (if VBW is set to **Auto**) changes to maintain the VBW/RBW ratio set by **VBW/RBW**.

When **Res BW** is set to **Man**, the bandwidths are then entered by the user. These bandwidths will be used regardless of other analyzer settings. When **Res BW** is set to **Auto**, the bandwidths are autocoupled to span. The ratio of span to RBW is set by **Span/RBW**. (See the **Span/RBW** description under **Mode Setup**). The factory default for this ratio is approximately 106:1when auto coupled.

A # mark appears next to **`Res BW`** on the bottom of the analyzer display when it is not coupled. To couple the resolution bandwidth, press **Res BW (Auto)** or **Auto Couple**.

Saved State:	Saved in instrument state.
Factory Preset and *RST:	3 MHz, Auto
Maximum Value:	8 MHz
Minimum Value:	1 Hz

Remote Command:

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

Example:

```
BAND 1 kHz
BWID:AUTO On
```

3.3.2 Video BW

Allows you to change the analyzer post-detection filter from 1 Hz to 50 MHz in approximately 10% steps. A wide open video filter (VBW) may be chosen by selecting 50 MHz. **Video BW (Auto)** selects automatic coupling of the Video BW filter to the resolution bandwidth filter using the VBW/RBW ratio set by the **VBW/RBW** key.

NOTE Sweep Time is coupled to Video Bandwidth (VBW). As the VBW is changed, the sweep time is changed to maintain amplitude calibration (when **Sweep Time** is set to **Auto**.)

The Video BW filter is not actually “in-circuit” when the detector is set to Average. This circumstance occurs because of common hardware between the two circuits. However, because the purpose of the average detector and the VBW filter are the same, either can be used to reduce the variance of the result.

Even though the VBW filter is not “in-circuit” when using the average detector, reducing the VBW setting (when **Sweep Time** is set to **Auto**) increases the sweep time and thus, increases the averaging time, giving a lower-variance trace. Therefore, even though the VBW filter is not in-circuit, the Video BW key can have an effect on sweep time and is not disabled. However, when the average detector is used, and **Sweep Time** is set to **Man**, the VBW setting has *no* effect and hence is disabled (grayed out).

In zero span, using the average detector, the VBW filter is not "in-circuit", so the VBW key is disabled (grayed out).

Saved State: Saved in instrument state.

Factory Preset and *RST: 3 MHz, Auto Coupled

Maximum Value: 50 MHz

Minimum Value: 1 Hz

Remote Command:

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

Example:

```
:BAND:VID 1 kHz
:BWID:VID:AUTO ON
```

3.3.3 VBW/RBW

Selects the ratio between the video and resolution bandwidths used when VBW is auto coupled. This function has no effect when VBW is set to **Man**.

NOTE Video Bandwidth (VBW) wider than Resolution Bandwidth (VBW/RBW ratio greater than 1.000) is desirable for best peak measurements of signals such as wideband radar pulses. VBW narrower than RBW (VBW/RBW ratio less than 1.000) is desirable to reduce the variance of noise-like signals and to make spectral components close to the noise floor easier to view.

Saved State: Saved in instrument state.

Factory Preset and *RST: 1.0

Maximum Value: 3.0e6 (3,000,000)

Minimum Value: 0.00001

Remote Command:

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

Example:

```
:BAND:VID:RAT 2
```

3.3.4 Average

Initiates a digital averaging routine that averages the trace points in a number of successive sweeps resulting in trace "smoothing". The number of sweeps (average number) can be selected. Increasing the average number will further smooth the trace. The type of averaging used is selected by pressing **Avg/BW Type**.

The average is restarted when a new average number is entered, any measurement related parameter (e.g., Center Frequency) is changed, **Restart** is pressed, or **Single Sweep** is pressed. When in Single Sweep, the specified number of averages is taken, then the sweep stops. When in continuous sweep, the specified number of averages is taken, then the averaging continues with each new sweep averaged in with a weight of $\frac{1}{\text{Average Number}}$ and the old average reduced by multiplying it by $\left(\text{Average Number} - \frac{1}{\text{Average Number}}\right)$.

To turn off averaging, press **Average (Off)**. The number of sweeps can only be entered using the numeric keypad, not the knob or step keys.

Saved State: Saved in instrument state.

Factory Preset and *RST: Off, 100 averages

Maximum Value: Count: 8192

Minimum Value: Count: 1

Remote Command:

```
[:SENSe]:AVERage[:STATe] OFF|ON|0|1
[:SENSe]:AVERage[:STATe]?
[:SENSe]:AVERage:COUNt <integer>
[:SENSe]:AVERage:COUNt?
```

Example:

```
:AVER ON
:AVER:COUN 100
```

3.3.5 Avg/VBW Type

Accesses the functions to automatically or manually choose from one of the following averaging scales: log-power (video), power (RMS), or voltage averaging.

NOTE If log-power averaging is selected, the measurement results are the average of the signal level in logarithmic units (decibels). If the power average (RMS) is selected, all measured results are converted into power units before averaging and filtering operations, and converted back to decibels for displaying. The main point to remember is that there can be significant differences between the average of the log of power and the log of the average power.

Avg/VBW Type also affects video bandwidth filtering. There are actually four types of averaging processes within a spectrum analyzer. All of them are affected by this setting. They are:

1. Trace averaging (see **BW/Avg**).

   Averages signal amplitudes on a trace-to-trace basis.

2. Average detector (see **Detector, Average**).

   Averages signal amplitudes during the time or frequency interval represented by a particular measurement point.

3. Noise Marker (see **Marker Noise**)

   Averages signal amplitudes across 32 measurement points to reduce variations for noisy signals.

4. VBW filtering.

   Filtering the video is a form of averaging the video signal.

The **Avg/VBW Type** is shown on the left side of the display with a #, if **Avg/VBW Type (Man)** is selected. When **Avg/VBW Type (Auto)** is selected, the analyzer chooses the type of averaging to be used. When one of the average types is selected manually, the analyzer uses that type regardless of other analyzer settings and sets
Avg/VBW Type to **Man**.

Saved State:	Saved in instrument state.
Factory Preset and *RST:	Log Power, Auto

Remote Command:

```
[:SENSe]:AVERage:TYPE RMS|LOG|SCALar
[:SENSe]:AVERage:TYPE?
[:SENSe]:AVERage:TYPE:AUTO OFF|ON|0|1
[:SENSe]:AVERage:TYPE:AUTO?
```

Example:

`:AVER:TYPE:RMS`	Sets Power (RMS) averaging
`:AVER:TYPE:SCAL`	Sets Voltage averaging
`:AVER:TYPE:LOG`	Sets Log-Power (video) averaging

3.3.5.1 Auto

Chooses the optimum type of averaging for the current instrument measurement settings.

Auto selects Power (RMS) Averaging if **Marker Noise** is on, **Band/Intvl Power** is on, or **Detector** is set to **Man** and **Average**. It selects Voltage Averaging if **Amplitude, Scale Type** is set to **Lin**. For other conditions, **Auto** selects Log-Power Average.

Example:

```
:AVER:TYPE:AUTO ON
```

3.3.5.2 Log-Pwr Avg (Video)

Selects Log Power averaging, also known as Video averaging or Trace averaging. This type of averaging averages the data in successive traces point by point, but does so on a logarithmic scale. This scale is excellent for finding CW signals near noise, but its response to noise-like signals is 2.506 dB lower than the average power of those noise signals. This is compensated for in the Marker Noise function. When this type of averaging is selected, **LgAv** appears on the left side of the display.

The equation for trace averaging on the log-pwr scale is shown below, where N is the number of averages accumulated. (In continuous sweep mode, once N has reached the Average Number, N stays at that value, providing a running average.)

$$\text{New data} = \text{Old data} + \frac{\text{New data} - \text{Old data}}{\text{N}}$$

(Assumes all values in dB)

Remote Command:

See “Avg/VBW Type” on page 79.

Example:

```
:AVER:TYPE LOG
```

3.3.5.3 Pwr Avg (RMS)

In this Average type, all filtering and averaging processes work on the power (the square of the magnitude) of the signal, instead of its log or envelope voltage. This scale is best for measuring the true time power of complex signals. This scale is sometimes called “RMS” because the resulting voltage is proportional to the square root of the mean of the square of the voltage. When this type of averaging is selected, **PAvg** appears on the left side of the display.

The equation for averaging on this scale is shown below, where N is the number of averages accumulated. (In continuous sweep mode, once N has reached the Average Number, N stays at that value.)

$$\text{New data} = 10 \times \log\left(10^{\frac{\text{Old data}}{10}} + \left(\frac{10^{\frac{\text{New data}}{10}} - 10^{\frac{\text{Old data}}{10}}}{N}\right)\right)$$

(Assumes all values in dB)

Remote Command:

See “Avg/VBW Type” on page 79.

Example:

```
:AVER:TYPE RMS
```

3.3.5.4 Voltage Avg

In this Average type, all filtering and averaging processes work on the voltage of the envelope of the signal. This scale is good for observing rise and fall behavior of AM or pulse-modulated signals such as radar and TDMA transmitters, but its response to noise-like signals is 1.049 dB lower than the average power of those noise signals. This is compensated for in the Marker Noise function. When this type of averaging is selected, **VAvg** appears on the left side of the display.

The equation for averaging on this scale is shown below, where N is the number of averages accumulated. (In continuous sweep mode, once N has reached the Average Number, N stays at that value.)

$$\text{New data} = 20 \times \log\left(10^{\frac{\text{Old data}}{20}} + \left(\frac{10^{\frac{\text{New data}}{20}} - 10^{\frac{\text{Old data}}{20}}}{N}\right)\right)$$

(Assumes all values in dB)

Remote Command:

See “Avg/VBW Type” on page 79.

Example:

```
:AVER:TYPE SCAL
```

3.4 Det/Demod

Sets the controls and parameters associated with the detector modes.

3.4.1 Detector

Selects a specific detector, or uses the system to pick the appropriate detector (through **Auto**) for a particular measurement.

When discussing detectors, it is important to understand the concept of a trace "bucket." For every trace point displayed, there is a finite time during which the data for that point is collected. The analyer has the ability to look at all of the data collected during that time and present a single point of trace data based on the detector mode. We call the interval during which the data for that trace point is being collected, the "bucket." Thus, a trace is more than a series of single points. It is actually a series of trace "buckets." The data may be sampled many times within each bucket.

When the **Detector** choice is **Auto**, selecting trace averaging (**BW/Avg**, **Average (On)**) changes the detector. The **Auto** choice depends on marker functions, trace functions, and the trace averaging function. If a marker function or measurement is running, the **Auto** choice of detector is either **Average** or **Sample**. When one of the detectors (such as **Normal**) is manually selected instead of **Auto**, that detector is used regardless of other analyzer settings.

The **Sample** detector displays the instantaneous level of the signal at the center of the bucket represented by each display point.

The **Normal** detector displays the peak of CW-like signals and maximums and minimums of noise-like signals.

The **Peak** detector displays the peak of the signal within the bucket.

The **Negative Peak** detector displays the minimum of the signal within the bucket.

Neither average nor sample detectors measure amplitudes of CW signals as accurately as peak or normal, because they may not find a spectral component's true peak, but they do measure noise without the biases of peak detection.

The detector in use is indicated on the left side of the display. A # will appear next to it if the detector has been manually selected.

NOTE RMS Detection: To measure the average power (RMS voltage) in each display point, set **Detector** to **Average**, and verify that **Avg/VBW Type** is set to **Power.**

State Saved:	Saved in instrument state.
Factory Preset and *RST:	Normal, Auto Coupled

Remote Command:

`[:SENSe]:DETector[:FUNCtion] AVERage|NEGative|NORMal|POSitive|SAMPle|RMS`

`[:SENSe]:DETector[:FUNCtion]?`

The query returns a name that corresponds to the detector mode as shown by the following terms:

NORM:	Normal
AVER:	Average
POS:	Positive Peak
SAMP:	Sample
NEG:	Negative Peak

Example:

```
:DET POS
```

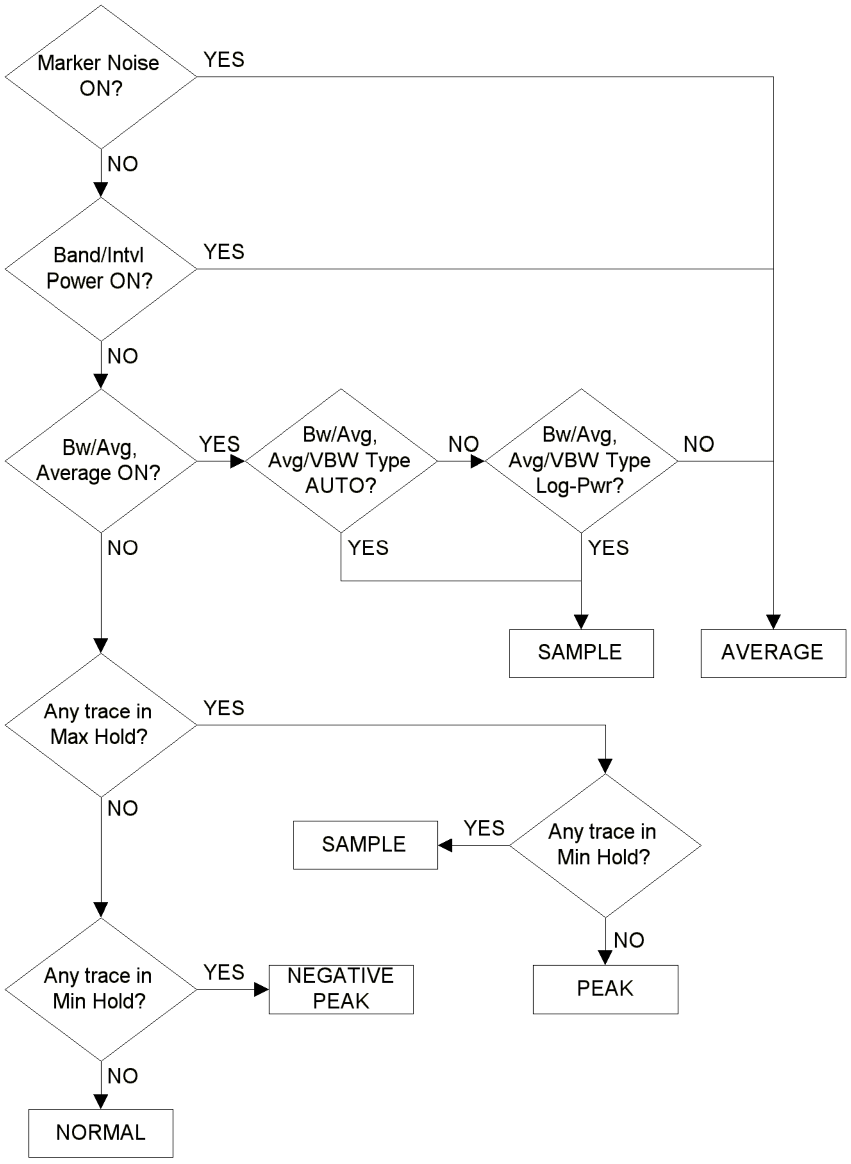
3.4.1.1 Auto

The system selects Normal detection as the default, but if a condition arises where a different type of detection scheme would be better utilized, the system will use the alternate scheme. For example, the Marker Noise function uses Average detection when in Auto mode because the system determines that the data will be more accurate for noise-type signals.

The **Auto** choice depends on marker functions, trace functions (**Max Hold**, **Min Hold**), and the **Avg/VBW** type.

The **Auto** rules for Detector are shown in Figure 3-1.

Figure 3-1 Auto Rules For Detector Selection



Remote Command:

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

Example:

```
:DET:AUTO ON
```

3.4.1.2 Normal

Displays the peak-detected level in the interval (bucket) being displayed when the signal is CW-like. If the signal is noise-like (within a bucket the signal both rose and fell), the even bucket shows the peak (maximum) within a two-bucket interval, and the odd bucket shows the negative peak (minimum). Gain is increased to compensate for the effects of faster sweep rates, to keep the displayed value accurate.

When **Normal** is selected, **`Norm`** appears on the left side of the display.

Remote Command:

See "Detector" on page 82.

Example:

```
:DET NORM
```

3.4.1.3 Average

For each interval (bucket) in the trace, Average detection displays the average of all the samples within the interval. The averaging can be done using any of three methods: the log method, the power method (also known as RMS), and the voltage envelope method. The method is controlled by the **BW/Avg**, **Avg/VBW Type** key. The combination of the average detector and the power method is equivalent to what is sometimes referred to as "RMS detection". When the method (**Avg/VBW Type**) is set to **Auto**, and **Detector** is set to **Average**, the RMS method is selected.

When **Average** is selected, **`Avg`** appears on the left side of the display.

Dependencies/ Couplings: Use of Average affects the VBW setting. See **BW/Avg**, **VBW**.
When in Average detection, video trigger is not available.

Remote Command:

See "Detector" on page 82.

Example:

```
DET:AVER
```

3.4.1.4 Peak

For each interval (bucket) in the trace, Peak detection displays the highest amplitude within the interval. Peak detection is used for CW measurements and some pulsed-RF measurements. For swept analysis, peak detection basically obtains the maximum video signal between the end of the last bucket and the start of the next one. Gain is increased to compensate for the effects of faster sweep rates, to keep the displayed value accurate. For FFT analysis, the highest spectral amplitude is displayed, even if that peak amplitude falls between samples of the spectrum computed in the FFT process.

When **Peak** is selected, **`Peak`** appears on the left side of the display.

Remote Command:

See “Detector” on page 82.

Example:

```
:DET POS
```

3.4.1.5 Sample

The sample detector displays the instantaneous level of the signal at the center of the interval (bucket) represented by each trace point.

Sample detection is primarily used to display noise or noise-like signals.

Sample detection is not best for amplitude measurements of CW-like signals for two reasons. First, the peak response to a signal can occur between samples, so unless the Span to RBW ratio is lower than usual, the highest sample can be well below the peak signal amplitude. Second, for the high sweep rates normally used, the peak response of the RBW filters is up to –0.5 dB. This sweeping error is compensated when using the peak and normal detectors by changing the overall gain. But the gain is not changed when in the sample detector, because to do so would cause errors in the response to noise.

When **Sample** is selected, **Samp** appears on the left side of the display.

Remote Command:

See “Detector” on page 82.

Example:

```
:DET SAMP
```

3.4.1.6 Negative Peak

For each interval (bucket) in the trace, **Negative Peak** detection displays the lowest sample within the interval. Negative peak detection is similar to peak detection, but selects the minimum video signal.

When **Negative Peak** is selected, **NPk** appears on the left side of the display.

Remote Command:

See “Detector” on page 82

Example:

```
:DET NEG
```

3.4.1.7 RMS (Remote Command Only)

Selects the Average Detector. If **Avg/BW Type** is set to **Auto** (or Power) this will yield the RMS voltage (average power) for each trace point. (See 3.4.1.3, Average)

There is no key selection for this setting, but you can access it by using **Average Detector** (see “Average” on page 85).

Remote Command:

See “Detector” on page 82.

Example:

```
:DET RMS
```

3.5 Display

Accesses menu keys that allow you to control certain items on the display of the analyzer.

3.5.1 Display Line

Activates an adjustable horizontal line that is used as a visual reference line. The line has an amplitude value that corresponds to its vertical position relative to the reference level. The value of the display line appears on the left side of the display below the label **Dl**. The display line can be adjusted using the step keys, knob, or numeric keypad. The units of Display Line are determined by the **Y-Axis Units** setting under Amplitude.

Saved State: Saved in instrument state.

Factory Preset and *RST: –25 dBm, Off

Maximum Value: +30 dBm

Minimum Value: –370 dBm

Remote Command:

:DISPlay:WINDow:TRACe:Y:DLINe <ampl>

:DISPlay:WINDow:TRACe:Y:DLINe?

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

:DISPlay:WINDow:TRACe:Y:DLINe:STATe?

Examples:

```
:DISP:WIND:TRAC:Y:DLIN -32 dBm
```

```
:DISP:WIND:TRAC:Y:DLIN:STAT OFF
```

3.5.2 Title

Accesses menu keys which allow you to change or clear a title on your display.

Saved State: Saved in instrument state.

Factory Preset and *RST: No title

3.5.2.1 Change Title

Allows you to write a title across the top of the display. Press Change Title to access the Alpha Editor Menus that contain available characters and symbols. You may also use the numeric keypad to enter numbers. Press Enter or Return to complete the entry. Press ESC to cancel the entry and preserve your existing title.

The display title will remain until you press **Change Title** again, or you recall a trace or state, or a **Factory Preset** is performed. A title can also be cleared by pressing **Title**, **Clear Title**.

Pressing this key cancels any active function.

Remote Command:

`:DISPlay:ANNotation:TITLe:DATA <string>`

`:DISPlay:ANNotation:TITLe:DATA?`

Example:

```
:DISP:ANN:TITL:DATA "This Is My Title"
```

3.5.2.2 Clear Title

Allows you to clear a title from the front-panel display. Once cleared, the title cannot be retrieved.

Remote Command:

There is no equivalent command, but the example below shows how to enter an empty title.

Example:

```
DISP:ANN:TITL:DATA ""
```

3.5.3 Display Enable (Remote Command Only)

Turns the display on/off. If enable is set to off, the display will appear to blank. This can make measurement run faster since the instrument doesn’t have to update the display after every data acquisition. There is often no need to update the display information when using remote operation.

State Saved:	Not saved in instrument state.
Factory Preset and *RST:	On

Remote Command:

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

`:DISPlay:ENABle?`

Example:

```
:DISP:ENAB OFF
```

3.6 FREQUENCY / Channel

Accesses the menu of frequency functions. The center frequency, or start and stop frequency values appear below the graticule on the display, depending on the **Frequency** entry mode. In Center/Span mode, the **Center Frequency** and **Span** are displayed and the default active function under **Frequency** is **Center Freq**. In Start/Stop mode, the **Start** and **Stop** frequencies are displayed and the default active function under **Frequency** is **Start Freq**.

NOTE Although the analyzer allows entry of frequencies greater than the specified frequency range, it is not recommended that the analyzer be used beyond its specified range.

Factory Preset and *RST: Center/Span mode

3.6.1 Center Freq

Activates the center frequency function which allows you to specify the frequency in the center of the display.

State Saved: Saved in instrument state.

Factory Preset and *RST: 13.25500000 GHz

Maximum Value: 26.50000000 GHz

Minimum Value: –99.99999500 MHz

Remote Command:

```
[:SENSe]:FREQuency:CENTer <frequency>
[:SENSe]:FREQuency:CENTer?
```

Example:

```
:FREQ:CENT 5 GHZ
```

3.6.2 Start Freq

Sets the frequency at the left side of the graticule. The left and right sides of the graticule correspond to the start and stop frequencies. **Start Freq** sets the Frequency entry mode to Start/Stop.

State Saved:	Saved in instrument state.
Factory Preset and *RST:	10 MHz
Maximum Value:	26.49999999 GHz
Minimum Value:	-100.0000000 MHz

Remote Command:

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

`[:SENSe]:FREQuency:STARt?`

Example:

```
FREQ:STAR 200 MHz
```

3.6.3 Stop Freq

Sets the frequency at the right side of the graticule. The left and right sides of the graticule correspond to the start and stop frequencies. **Stop Freq** sets the Frequency entry mode to Start/Stop.

State Saved:	Saved in instrument state.
Factory Preset and *RST:	26.5 GHz
Maximum Value:	26.50000000 GHz
Minimum Value:	-100.0000000 MHz

Remote Command:

`[:SENSe]:FREQuency:STOP <frequency>`

`[:SENSe]:FREQuency:STOP?`

Example:

```
:FREQ:STOP 1600
```

3.6.4 CF Step

Changes the step size for the center frequency function. Once a step size has been selected and the center frequency function is activated, the step keys (and the UP/DOWN parameters for Center Frequency from remote commands) change center frequency by the step-size value. The step size function is useful for finding harmonics and sidebands beyond the current frequency span of the analyzer. When auto-coupled, the center frequency step size is set to one horizontal graticule division (10 percent of the span).

State Saved:	Saved in instrument state.
Factory Preset and *RST:	2.64900000 GHz, Auto coupled
Maximum Value:	26.5 GHz
Minimum Value:	1 Hz

Remote Command:

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

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

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

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

Examples:

```
:FREQ:CENT:STEP 100 MHz
:FREQ:CENT:STEP:AUTO ON
```

3.6.5 Freq Offset

Allows you to input a frequency offset value that is added to the frequency readout of the marker, center frequency, start frequency and stop frequency, to account for frequency conversions external to the analyzer. Offsets may only be entered using the numeric keypad, not the knob or step keys. Offsets are not added to the span or frequency count readouts. Entering an offset does not affect the trace display. To eliminate an offset, perform a Factory Preset or set the frequency offset to 0 Hz.

State Saved:	Saved in instrument state.
Factory Preset and *RST:	0 Hz
Maximum Value:	500 THz
Minimum Value:	–3 GHz

Remote Command:

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

`[:SENSe]:FREQuency:OFFSet?`

Example:

```
:FREQ:OFFS 10 MHz
```

3.6.6 Signal Track

When a marker is placed on a signal and **Signal Track** is pressed, the marker will remain on the signal while the analyzer retunes the center frequency to the marker frequency, keeping the signal at the center of the display, as long as the amplitude of the signal does not change by more than 3 dB from one sweep to another.

If the signal is lost, an attempt will be made to find it again and continue tracking. If there are other signals on screen near the same amplitude, one of them may be found instead. Signals near 0 Hz cannot be tracked effectively as they cannot be distinguished from the LO feedthrough, which is excluded by intent from the search algorithm.

When **Signal Track** is **On** and the span is reduced, an automatic zoom is performed and the span is reduced in steps so that the signal remains at the center of the display. If the span is zero, signal track cannot be activated.

If no marker is active, pressing **Signal Track** to **On** will activate a marker, perform a peak search, and center the marker on the display.

NOTE This function is intended to track signals with a frequency that is changing, and an amplitude that is not changing.

State Saved:	If **On**, the fact is saved in instrument state.
Factory Preset and *RST:	Off

Remote Command:

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

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

Example:

```
:CALC:MARK1:TRCK ON
```

Turns on **Signal Track** using Marker 1.

3.7 Input/Output

Accesses keys which control the signal inputs and outputs of the analyzer.

3.7.1 Input Port

Sets the signal input path to be the RF input on the front panel or the internal amplitude reference.

State Saved:	Saved in instrument state.
Factory Preset and *RST:	RF

Remote Command:

`[:SENSe]:FEED RF|AREFerence`

`[:SENSe]:FEED?`

Example:

```
:FEED AREF
```

Selects the 50 MHz amplitude reference as the signal input.

3.7.1.1 RF

Selects the front panel RF Input port to be the analyzer signal input.

Remote Command:

See "Input Port" on page 94.

Example:

```
:FEED RF
```

3.7.1.2 Amptd Ref

Selects the 50 MHz, –25 dBm internal amplitude reference as the input signal.

Remote Command:

See “Input Port” on page 94.

Example:

```
:FEED AREF
```

3.7.2 RF Coupling

Specifies AC or DC coupling at the analyzer RF input port. Selecting AC coupling switches in a blocking capacitor that blocks any DC voltage present at the analyzer input. This decreases the input frequency range of the analyzer, but prevents damage to the analyzer’s input circuitry if there is a DC voltage present at the RF input.

In AC coupling mode, signals less than 10 MHz are not calibrated. You must switch to DC coupling to see calibrated frequencies of less than 10 MHz.

State Saved:	Saved in instrument state.
Factory Preset and *RST:	AC

Remote Command:

```
:INPut:COUPling AC|DC
:INPut:COUPling?
```

Example:

```
:INP:COUP DC
```

3.7.3 Freq Ref

Specifies the frequency reference as being internal or external. If the frequency reference is specified as internal, the frequency of the reference is automatically identified as being 10 MHz. If the frequency reference is specified as external, you must enter the frequency of the external reference being used. If External Reference is selected, **`Ext Ref`** will appear on the right side of the display.

The frequency of an external frequency reference is not automatically detected. If an external frequency source is selected, the frequency of the source must be entered.

If Ext is selected, and you press **Freq Ref**, Ext will remain selected and the Ext reference frequency will become the active function. If **Freq Ref** is pressed again, Int will become selected (at 10 MHz). The Ext reference frequency is remembered and will be used again if Ext is selected.

If the external reference is missing or out of range, or the frequency reference is unlocked, the message "**`External reference missing or out of range`**", will appear on the display.

State Saved: Not saved in instrument state. Neither the external reference frequency nor the state of this function (Int or Ext) are affected by factory preset or power cycle. Reset to the factory default (Int, 10 MHz) by pressing **System**, **Restore Sys Defaults**.

Default: Internal, 10 MHz

Maximum Value: 30 MHz

Minimum Value: 1 MHz

Remote Command:

`[:SENSe]:ROSCillator:SOURce INTernal|EXTernal`

`[:SENSe]:ROSCillator:SOURce?`

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

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

Examples:

You should specify the frequency of the external reference that you plan to use, before switching to the external reference source.

`:ROSC:EXT:FREQ 20 MHz`

Sets the external reference frequency to 20 MHz, but does not select the external reference.

`:ROSC:SOUR EXT`

Selects the external reference.

3.7.4 10 MHz Out

Switches the 10 MHz out signal on the rear panel of the analyzer on or off.

State Saved:	Not saved in instrument state. Not affected by factory preset or power cycle. Reset to the factory default (Off, 10 MHz) by pressing **System**, **Restore Sys Defaults**.

Remote Command:

`[:SENSe]:ROSCillator:OUTPut[:STATe] OFF|ON|0|1`

`[:SENSe]:ROSCillator:OUTPut[:STATe]?`

Examples:

```
:ROSC:OUT ON
```

3.8 MODE

Selects the measurement mode of your analyzer. Spectrum Analysis mode is currently the only mode available, but future modes will allow measurements of complex digital signals.

State Saved:	No save
Factory Preset and *RST:	Spectrum Analysis

Remote Command:

There is no remote command for this key.

3.8.1 Spectrum Analysis

Selects the spectrum analysis measurement mode for your analyzer.

3.9 Mode Setup

Allows you to change measurement settings common to all measurements. In Spectrum Analysis mode, allows you to modify settings for most Auto Couple functions. (Center Frequency Step, Marker Count Gate Time, and Marker Trace are not included under Mode Setup.)

3.9.1 Auto All

Auto-couples all coupled functions (see **Auto Couple**). If **Auto All (On)** is pressed, it is equivalent to pressing **Auto Couple**; all coupled functions are set to **Auto**. Setting any auto coupled function to **Man** (manual), sets this key to **Off**.

3.9.2 RBW, VBW, ST

Accesses measurement bandwidth and sweep time controls and allows you to turn off one or more coupled functions.

3.9.2.1 Res BW

See **Res BW** in the BW/Avg menu.

3.9.2.2 Span/RBW

Sets the ratio of span to resolution bandwidth. This parameter defines the coupling of RBW to Span. Higher ratios show more resolution. Lower ratios allow faster measurements.

Factory Preset and *RST: 106, Auto

Maximum Value: 10,000

Minimum Value: 2

Remote Command:

```
[:SENSe]:FREQuency:SPAN:BANDwidth[:RESolution]:RATio <integer>
[:SENSe]:FREQuency:SPAN:BANDwidth[:RESolution]:RATio?
[:SENSe]:FREQuency:SPAN:BANDwidth[:RESolution]:RATio:AUTO OFF|ON|0|1
[:SENSe]:FREQuency:SPAN:BANDwidth[:RESolution]:RATio:AUTO?
```

Examples:

```
:FREQ:BAND:RAT 50
:FREQ:BAND:RAT:AUTO ON
```

3.9.2.3 Video BW

See **Video BW** in the BW/Avg menu.

3.9.2.4 VBW/RBW

See **VBW/RBW** in the BW/Avg menu.

3.9.2.5 Sweep Time

See **Sweep Time** in the Sweep menu.

3.9.2.6 Auto Sweep Time

Selects the rules for sweep time selections when **Sweep Time** is set to **Auto**. When **Auto Sweep Time** is set to **Accy**, the sweep rate of swept analysis is slowed to ensure that even under worst case variations in the instrument, all amplitude accuracy specifications are met for CW signals. When **Auto Sweep Time** is set to **Norm**, the sweep rate is much higher, and an additional amplitude uncertainty (typically 0.05 dB) occurs. **Auto Sweep Time** is set to **Norm** when all functions are autocoupled by pressing **Auto Couple**.

State Saved:	Saved in instrument state.Saved in instrument state.
Factory Preset and *RST:	Normal

Remote Command:

```
[:SENSe]:SWEep:TIME:AUTO:RULes NORMal|ACCuracy
[:SENSe]:SWEep:TIME:AUTO:RULes?
```

Example:

```
:SWE:TIME:AUTO:RUL NORM
```

3.9.3 FFT & Sweep

Selects the FFT vs. Sweep key functions.

3.9.3.1 Sweep Type

See **Sweep Type** in the Sweep menu.

3.9.3.2 Auto Sweep Type

See **Auto Sweep Type** in the Sweep menu.

3.9.3.3 FFTs/Span

See **FFTs/Span** in the Sweep menu.

3.9.4 PhNoise Opt

Chooses the LO (local oscillator) phase noise behavior that is optimum for measurement accuracy. The selected value is displayed below the **£(f)** indicator on the left side of the screen. It is preceded by # if **PhNoise Opt (Man)** has been selected.

State Saved: Saved in instrument state.

Factory Preset and *RST: Fast

Remote Command:

[:SENSe]:FREQuency:SYNThesis <value>

1, selects optimization of functionality for frequencies <50 kHz

2, selects optimization of functionality for frequencies >50 kHz

3, selects optimization of LO for fast tuning

[:SENSe]:FREQuency:SYNThesis?

[:SENSe]:FREQuency:SYNThesis:AUTO OFF|ON|0|1

[:SENSe]:FREQuency:SYNThesis:AUTO?

Example:

```
:FREQ:SYNT 3, selects optimization for fast tuning
```

3.9.4.1 Auto

Allows the analyzer to automatically select an LO phase noise behavior that is optimum for the selected span and RBW. The **Auto** rules choose **Fast Tuning** whenever the span ≥2 MHz or the RBW >200 kHz. Otherwise, for spans >141.4 kHz, and for resolution bandwidths >9.1 kHz, the **Auto** rules choose **Optimize £(f) for f >50 kHz**. All remaining cases choose **Optimize £(f) for f <50 kHz**.

Factory Preset and *RST: On

Example:

```
:FREQ:SYNT:AUTO ON
```

3.9.4.2 Optimize £(f) for frequencies < 50 kHz

The LO phase noise is optimized for offsets less than 50 kHz from the carrier, at the expense of phase noise beyond 50 kHz offset.

Remote Command:

See "PhNoise Opt" on page 101.

Example:

```
:FREQ:SYNT 1
```

3.9.4.3 Optimize £(f) for frequencies > 50 kHz

Optimizes phase noise for offsets above 50 kHz from the carrier, especially those from 70 kHz to 300 kHz. Closer offsets are compromised and the throughput of measurements (especially remote measurements where the center frequency is changing rapidly), is reduced.

Remote Command:

See "PhNoise Opt" on page 101.

Example:

```
:FREQ:SYNT 2
```

3.9.4.4 Optimize LO for fast tuning

In this mode, the LO behavior compromises phase noise at all offsets from the carrier below approximately
2 MHz. This allows rapid measurement throughput when changing the center frequency or span.

Remote Command:

See "PhNoise Opt" on page 101.

Example:

```
:FREQ:SYNT 3
```

3.9.5 Detector

See Detector in Det/Demod menu.

3.9.6 Avg/VBW Type

See Avg/VBW Type in the BW/Avg menu.

3.9.7 ADC Dither

Turns the ADC dither on or off. **ADC Dither (On)** improves linearity on low level signals. **ADC Dither (Off)** yields a slightly decreased noise floor. **ADC Dither (Auto)** allows for the best compromise between the two.

When **ADC Dither** is set to **Auto**, it is turned On or Off depending on the span, RBW, and sweep type (FFT or Swp settings).

When **Sweep Type** is set to **Swp**, **ADC Dither** is turned off.

When **Sweep Type** is set to **FFT** and the RBW is ≤100 Hz, ADC Dither is turned off.

When **Sweep Type** is set to **FFT**, the RBW is >100 Hz, and the FFT width is less than 2 MHz, **ADC Dither** is turned on. The FFT width is given by the **Span** divided by the **FFTs/Span** parameter from the **Sweep** menu.

When **Sweep Type** is set to **FFT** and the FFT width is ≥2 MHz, **ADC Dither** is turned off. The FFT width is given by the **Span** divided by the **FFTs/Span** parameter from the **Sweep** menu.

When **ADC Dither** is **On**, the linearity of low-level signals is improved. Amplitude uncertainties are specified to be less than ±0.07 dB. By comparison, when **ADC Dither** is **Off**, signals below –70 dBm at the input mixer have amplitude uncertainties, sometimes as high as ±0.5 dB. When **ADC Dither** is **On**, however, the ADC dynamic range is reduced to make room for the dither. As a result, the noise floor of the analyzer is compromised. The noise increases by 2.5 dB nominally for the low band (0-3 GHz), and by less at higher frequencies. See the Specifications Guide for more information.

State Saved:	Saved in instrument state.
Factory Preset and *RST:	Auto

Remote Command:

`[:SENSe]:ADC:DITHer[:STATe] OFF|ON|AUTO`

`[:SENSe]:ADC:DITHer[:STATe]?`

Example:

```
:ADC:DITH OFF
```

3.10 Single

Changes the sweep control to single sweep if the analyzer is in continuous sweep mode, and executes a sweep after any trigger condition is met. If the analyzer is already in single sweep, pressing **Single** executes a new sweep after the trigger condition is met.

Some instrument settings require more than one sweep to complete the measurement (see **BW/Avg**, **Average**), or if you have selected a measurement from the functions under the **MEASURE** key, this function sets the trigger system to be initiated only once. In this case the trigger will be limited once and then all the necessary sweeps will be executed to make the measurement or complete the averaging function.

Factory Preset and *RST:	Continuous

Remote Command:

`:INITiate[:IMMediate]`

`*TRG`

Use the `:TRIGger[:SEQuence]:SOURce` command to select the trigger source.

See also the Sweep Single/Cont function in the Sweep key menu with the command INITiate:CONTinuous ON|OFF.

Examples:

```
:*TRG
```

```
:TRIG:IMM
```

3.11 SPAN / X Scale

Activates the Span function and accesses the menu of span functions. Pressing **SPAN / X Scale** sets the Frequency entry mode to Center/Span. (See **Frequency**)

Remote Command:

See the Span command below.

3.11.1 Span

Allows you to change the frequency range symmetrically about the center frequency. The frequency-span readout describes the total displayed frequency range. To determine frequency span per horizontal graticule division, divide the frequency span by 10. Setting the span to 0 Hz puts the analyzer into zero span and changes the horizontal axis from frequency to time. (See Zero Span, below). Pressing Span sets the Frequency entry mode to Center/Span. (See Frequency)

State Saved:	Saved in instrument state.
Factory Preset and *RST:	26.49 GHz
Maximum Value:	26.5 GHz
Minimum Value:	10 Hz

Remote Command:

[:SENSe]:FREQuency:SPAN <frequency>

[:SENSe]:FREQuency:SPAN?

Example:

```
:FREQ:SPAN 22 GHZ
```

3.11.2 Full Span

Changes the analyzer span to full span showing the full frequency range of the analyzer. Sets the maximum instrument span.

Remote Command:

[:SENSe]:FREQuency:SPAN:FULL

Example:

```
:FREQ:SPAN:FULL
```

3.11.3 Zero Span

Changes the frequency span to zero. This is equivalent to setting the span to 0 Hz. The detected video at the center frequency is displayed versus time. The instrument behavior is then similar to that of an oscilloscope with a detector (or log detector) installed in front of the oscilloscope.

NOTE The sweep time range changes in Zero Span. (See **Sweep**, **Sweep Time**)

NOTE Zero Span turns Signal Track off.

Remote Command:

There is no remote command for this key.

Example:

```
:SENSe:FREQ:SPAN 0 Hz
```

3.11.4 Last Span

Returns the instrument to the previously selected span setting.

Remote Command:

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

Example:

```
:FREQ:SPAN:PREV
```

3.12 SWEEP

Activates the Sweep Time function, and accesses the sweep function menu keys.

3.12.1 Sweep Time

Selects the length of time in which the spectrum analyzer sweeps the displayed frequency span. In swept spans, the sweep time varies from 1 millisecond to 2000 seconds. Reducing the sweep time increases the rate of sweeps. In zero span, the sweep time may be set from 1 µs to 6000 s. In FFT spans, the sweep time is not controlled by the user, but is an estimate of the time required to make FFT measurements. Sweep time is coupled to RBW and VBW, so changing those parameters may change the sweep time. When **Sweep Type** has been set to **FFT**, **Sweep Time** is disabled (grayed out).

State Saved: Saved in instrument state.

Factory Preset and *RST: Auto, 5 ms

Default Terminator: seconds

Maximum Value: 6000 s in zero span, 2000 s in swept spans

Minimum Value: 1 µs in zero span, 1 µs in swept spans

Remote Command:

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

Examples:

```
:SWE:TIME 500 ms
:SWE:TIME:AUTO OFF
```

3.12.2 Sweep

Switches the analyzer between continuous-sweep and single-sweep mode.

State Saved:	Save
Factory Preset and *RST:	Continuous

Remote Command:

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

Example:

```
:INIT:CONT OFF
```

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.

3.12.3 Sweep Type

Selects the type of "sweeping." This can be either FFT (Fast Fourier Transform) or conventional spectrum analyzer swept mode. At narrow resolution bandwidths, especially 10 kHz or less, FFT analysis can be much faster than swept analysis. This is because FFT analysis behaves as though hundreds of RBW filters are implemented to operate in parallel, while in swept analysis, only one RBW operates. But for wider resolution bandwidths, especially over 300 kHz, the computational overhead of FFTs makes FFT analysis much slower than swept analysis.

FFTs have a lower dynamic range for signals near the carrier, and slightly worse accuracy than swept analysis.

Dependencies/ Couplings: When **Auto Couple All** is invoked, **Sweep Type** is set to **Auto**.

When **Sweep Type** is set to **Auto**, the **Auto Sweep Type** key is active and the **FFTs/Span** key is grayed out.

When **Sweep Type** is set to **FFT**, the **Auto Sweep Type** and **Sweep Time** keys are inactive and grayed out, and the **FFTs/Span** key is active.

When **Sweep Type** is set to **Swp**, the **Auto Sweep Type** and **FFTs/Span** keys are inactive and grayed out.

State Saved: Saved in instrument state.

Factory Preset and *RST: Auto

Remote Command:

`[:SENSe]:SWEep:TYPE AUTO|FFT|SWEep`

`[:SENSe]:SWEep:TYPE?`

Example:

```
:SWE:TYPE FFT
```

3.12.4 Auto Sweep Type

Allows you to choose a sweep type optimized for speed or dynamic range.

When optimizing for speed, FFTs are used for resolution bandwidths ≤9.1 kHz, and swept mode for wide resolution bandwidths. The number of FFTs/Span (See “FFTs/Span” on page 110) is also automatically set to the minimum number of FFTs possible. The minimum number of FFTs is equal to INT(Span/10 MHz).

When optimizing for dynamic range, swept mode is used for resolution bandwidths > 500 Hz. For narrower resolution bandwidths, FFTs are used. But, the number of FFTs/Span is automatically set to (1 +INT(Span/5 kHz)) to optimize the dynamic range for signals near a carrier.

State Saved: Saved in instrument state.

Factory Preset and *RST: Dynamic Range

Remote Command:

`[:SENSe]:SWEep:TYPE:AUTO:RULes SPEed|DRANge`

`[:SENSe]:SWEep:TYPE:AUTO:RULes?`

Example:

```
:SWE:TYPE:AUTO:RUL SPE
```

3.12.5 FFTs/Span

Displays and controls the number of FFT segments used to measure the entire Span. This key is inactive and grayed out unless **Sweep Type** has been set to FFT. If **Sweep Type** is set to Auto and FFTs are selected, FFTs/Span is still grayed out, and the number of FFTs automatically selected is shown. If **Sweep Type** is set to **FFT**, **FFTs/Span** becomes active and an integer can be entered. The analyzer will try to use the number entered, but it may need to use a different number based on the rules for **Auto Sweep Type**.

FFT measurements require that the A/D converter in the IF remains at the same range for an entire FFT segment. This behavior leads to higher noise than in the swept case. In the swept case, the A/D converter can autorange as it sweeps through a signal, giving optimum dynamic range.

However, FFTs can be made higher dynamic range by cutting a span into many FFT segments and autoranging on each segment. This allows an almost continuous trade-off between high speed low dynamic range FFTs and slower, higher dynamic range swept mode.

An FFT can only be performed over a limited span or segment (also known as the FFT width). Several FFT widths may need to be combined to measure the entire span. The "FFT Width" is (Span)/(FFTs/Span), and affects the ADC Dither function. (See **Mode Setup**)

State Saved:	Saved in instrument state.
Factory Preset and *RST:	1
Maximum Value:	400000
Minimum Value:	1

Remote Command:

```
[:SENSe]:SWEep:FFT:SPAN:RATio <integer>
[:SENSe]:SWEep:FFT:SPAN:RATio?
```

Example:

```
:SWE:FFT:SPAN:RAT 20
```

3.13 Trace/View

Accesses menu keys that allow you to set how trace information is stored and displayed.

Factory Preset and *RST:	Trace 1: Clear Write
	Trace 2: Blank
	Trace 3: Blank

Remote Command:

:TRACe[1]|2|3:MODE WRITe|MAXHold|MINHold|VIEW|BLANk

WRITe = **Clear Write**

MAXHold = **Max Hold**

MINHold = **Min Hold**

VIEW = **View**

BLANk = **Blank**

:TRACe[1]|2|3:MODE?

3.13.1 Query Trace Data (Remote Command Only)

This query returns the current values of the designated trace amplitude values. The data is terminated with <NL><END>. (For GPIB this is newline, or linefeed, followed by EOI set true. For RS-232 this is newline only.)

Remote Command:

:TRACe[:DATA]? <trace_name>

<trace_name> is TRACE1|TRACE2|TRACE3

Example:

:TRAC? TRACE2 queries the analyzer for the contents of trace 2.

Commands :MMEM:STOR:TRAC and :MMEM:LOAD:TRAC are used to transfer trace data to/from the internal hard drive or floppy drive of the instrument. See “Save Now” on page 168 and “Load Now” on page 173.)

3.13.2 Trace

Determines which trace the menu keys will affect. Press Trace 1 2 3 until the number of the desired trace is underlined.

State Saved: Saved in instrument state for all traces. Saved in instrument state.

Remote Command:

There is no remote command for this function.

3.13.3 Clear Write

Erases any data previously stored in the selected trace and continuously displays signals during the sweep of the analyzer.

Remote Command:

See "Trace/View" on page 111.

Example:

```
:TRAC:MODE WRIT
```

3.13.4 Max Hold

Maintains the maximum level for each trace point of the selected trace (1, 2 or 3), and updates each trace point if a new maximum level is detected in successive sweeps.

Remote Command:

See "Trace/View" on page 111.

Example:

```
:TRAC:MODE MAXH
```

3.13.5 Min Hold

Maintains the minimum level for each trace point of the selected trace (1, 2 or 3), and updates each trace point if a new minimum level is detected in successive sweeps.

Remote Command:

See "Trace/View" on page 111.

Example:

```
:TRAC:MODE MINH
```

3.13.6 View

Holds and displays the amplitude data of the selected trace. The trace is not updated as the analyzer sweeps.

Remote Command:

See “Trace/View” on page 111.

Example:

```
:TRAC:MODE VIEW
```

3.13.7 Blank

Stores the amplitude data for the selected trace and removes it from the display. The selected trace register will not be updated as the analyzer sweeps.

Remote Command:

See “Trace/View” on page 111.

Example:

```
:TRAC:MODE BLAN
```

3.14 Trig

Accesses the menu of keys that allow you to select the trigger mode of a sweep or measurement. When in a trigger mode other than Free Run, the analyzer will only begin a sweep with the proper trigger condition.

State Saved: Saved in instrument state.

Factory Preset and *RST: Free Run

Remote Command:

`:TRIGger[:SEQuence]:SOURce IMMediate|VIDeo|LINE|EXTernal[1]|EXTernal2`

`:TRIGger[:SEQuence]:SOURce?`

IMM = Free Run
VID = Video
LINE = Line
Ext1 = External Front
Ext2 = External Rear

Example:

```
:TRIG:SOUR VID
```

Other trigger-related commands are found in the INITiate and ABORt subsystems.

3.14.1 Free Run

Sets the trigger to start a new sweep/measurement as soon as the last one has ended (continuous sweep mode) or immediately (single sweep mode).

Dependencies/ Couplings: Trigger Slope and Delay adjustments are not available with Free Run triggering.

Remote Command:

See “Trig” on page 114.

Example:

```
:TRIG:SOUR IMM
```

3.14.2 Video

Activates the trigger condition that allows the next sweep to start if the detected RF envelope voltage crosses a level set by the video trigger level. When Video is pressed, a line appears on the display. The analyzer triggers when the input signal exceeds the trigger level at the left edge of the display. You can change the trigger level using the step keys, the knob, or the numeric keypad. The line remains as long as video trigger is the trigger type.

Dependencies/ Couplings: Trigger Delay adjustment is not available with Video triggering.

Video triggering is not available when the detector is Average. Marker Functions which would set the detector to average (such as Marker Noise or Band/Intvl Power), which will conflict with video triggering, are not available when the video trigger is on.

Factory Preset and *RST: -25 dBm

Maximum Value: reference level

Minimum Value: 10 display divisions below the reference level when Scale Type is set to Log.

100 dB below the reference level when Scale Type is set to Lin.

Remote Command:

See “Trig” on page 114 for the command that sets trigger mode. The following commands set/read the trigger level.

`:TRIGger[:SEQuence]:VIDeo:LEVel <ampl>`

`:TRIGger[:SEQuence]:VIDeo:LEVel?`

Examples:

`:TRIG:SOUR VID` selects video triggering.

`:TRIG:VID:LEV -20 dBm` sets the video level to –20 dBm trigger.

3.14.3 Line

Sets the trigger to start a new sweep/measurement to be synchronized with the next cycle of the line voltage.

Remote Command:

See “Trig” on page 114.

Examples:

`:TRIG:SOUR LINE` selects line triggering.

3.14.4 Ext Front

Sets the trigger to start a new sweep/measurement whenever the external voltage (connected to EXT TRIGGER INPUT on the front panel) passes through approximately 1.5 volts. The external trigger signal must be a 0 V to +5 V TTL-type signal.

Remote Command:

See “Trig” on page 114.

Examples:

`:TRIG:SOUR EXT` to select front panel external triggering.

3.14.5 Ext Rear

Sets the trigger to start a new sweep/measurement whenever the external voltage (connected to TRIGGER IN on the rear panel) passes through approximately 1.5 volts. The external trigger signal must be a 0 V to +5 V TTL-type signal.

Remote Command:

See “Trig” on page 114.

Examples:

`:TRIG:SOUR EXT2` selects rear panel external triggering.

3.14.6 Trig Slope

Controls the trigger polarity, positive to trigger on a rising edge and negative to trigger on a falling edge.

Dependencies/ Couplings:	Not available for Free Run.
State Saved:	Saved in instrument state.
Factory Preset and *RST:	Positive (rising edge)

Remote Command:

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

`:TRIGger[:SEQuence]:SLOPe?`

Examples:

```
:TRIG:SLOP NEG
```

3.14.7 Trig Delay

Allows you to control a time delay during which the analyzer will wait to begin a sweep after receiving an external or line trigger signal. You can set the delay and turn it on or off.

Dependencies/ Couplings:	This function is not available when Trigger is Free Run or Video.
State Saved:	Saved in instrument state.
Factory Preset and *RST:	Off, 1 µs

Maximum Value: 500 µs

Minimum Value: 100 ns

Remote Command:

```
:TRIGger[:SEQuence]:DELay <time>
:TRIGger[:SEQuence]:DELay?
:TRIGger[:SEQuence]:DELay:STATe OFF|ON|0|1
:TRIGger[:SEQuence]:DELay:STATe?
```

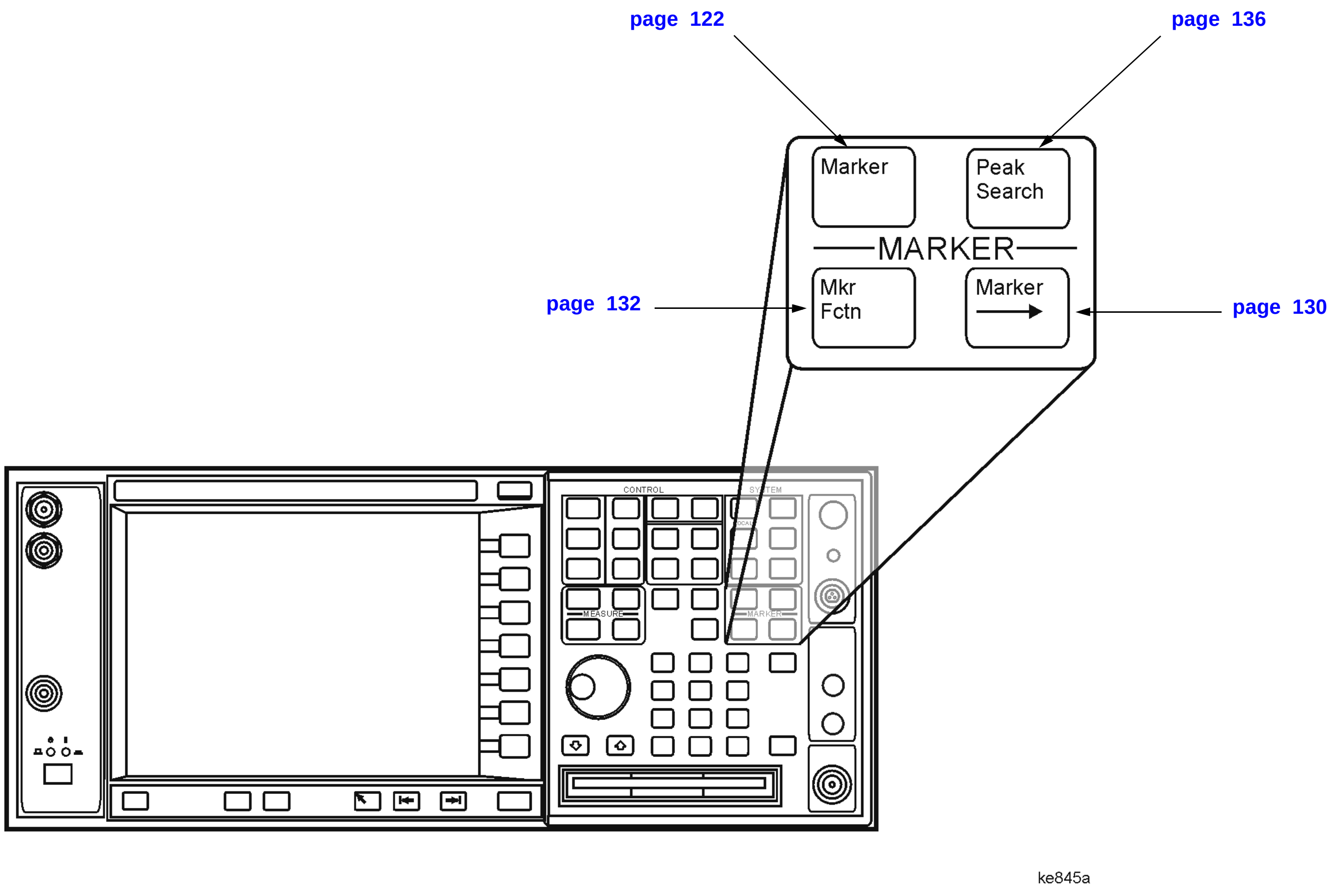
Examples:

```
:TRIG:DEL:STAT ON
:TRIG:DEL 100 ms
```

4 Marker Keys

This chapter provides key descriptions and programming information for the Marker functions of your analyzer. The front-panel Marker functions are listed alphabetically and are described with their associated menu keys. The lower-level menu keys are arranged and described as they appear in your analyzer.

Marker Function Keys and Where to Find Them



Your instrument stores data to a high degree of resolution and accuracy. It is often difficult to read the trace data directly from the screen to the desired accuracy. Markers are pointers that can be placed at any point on a trace to accurately read the data at that point. Markers may also be used in pairs to read the difference (or *delta*) between two data points. The data for the *active* marker (the one currently being controlled) appears in the upper-right corner of the display. In addition, when a marker is being actively controlled, the marker data appears in the active function area of the display. There are four markers in your instrument; each can be controlled as a single marker or as a reference/delta pair.

A *trace* is a connected series of points displayed on the instrument screen. The left-most point is point 0 and the right-most point (default) is point 600. You control markers by moving them from trace point to trace point. *Markers* are shaped like diamonds. The lowest point of the diamond shape represents the trace point that is being read. The marker number is indicated above the active marker. The same marker number is indicated with an R (for example, 1R) above the reference marker when in a delta mode (delta, delta pair, and span pair).

Marker Units

- Normal markers - the display shows the value of the marker's Y-axis position in the current Y-axis units. (See **Amplitude, Y Axis Units**.)
- **Delta**, **Delta Pair**, or **Span Pair** markers - the display shows the ratio (difference when expressed in dB) between two markers. If the Y-axis units are logarithmic (dBm, dBmV, dBuV) the ratio is expressed in dB. If the Y-axis units are linear (volts, watts) the ratio is expressed in percent (where 100% is the same as 0 dB difference). Note that the value when the Y-axis units are watts is the square of the value when the Y-axis units are volts. For example, when the percent ratio with Y-axis units in volts is 20% (0.2), the percent ratio with Y-axis units in watts will be 4% ($0.2^2 = 0.04$). When you read the value out remotely you have to know whether you are in log (dB) or linear (percent).
- Marker functions (**Marker Noise** and **Band/Intvl Power**) - the display shows the values with units that are dependent on the function and the Y-axis units. Refer to the individual function descriptions for more details about the units used. When you read the value out remotely you have to know what the expected units are.

4.1 Marker

Accesses the Marker Control menu. If no markers are active, **Marker** selects marker 1, sets it to **Normal** and places it at the center of the display. There are five control modes for the markers:

- **Normal (POSition) -** A single marker that can be moved to any point on the trace.
- **Delta (DELTa) -** A fixed reference marker and a moveable marker that you can place at any point on the trace
- **Delta Pair (BAND) -** Both a movable delta and a movable reference marker. You can independently adjust the position of each marker.
- **Span Pair (SPAN) -** A moveable reference and a movable delta marker. You can adjust the center point of the markers and the frequency span between the markers.
- **Off (Off) -** Turns off the active marker or marker pair.

State Saved:	The control mode for each marker pair, as well as the position of each marker, is saved in instrument state.
Factory Preset and *RST:	All Off.

Remote Command:

`:CALCulate:MARKer[1]|2|3|4:MODE POSition|DELTa|BAND|SPAN|OFF`

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

Sets or queries the marker control mode (see parameter list above).

`:CALCulate:MARKer[1]|2|3|4:X <param>`

Sets the marker X position to a specified point on the X axis in the current X-axis units (frequency or time). If the frequency or time chosen would place the marker off screen, the marker will be placed at the left or right side of the display, on the trace. This command will have no effect if the marker is **OFF**.

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

Queries the marker X position in the current x-axis units. The marker must be **ON** for the response to be valid.

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

Queries the marker Y value or delta in the current y axis units. Can also be used to read the results of marker functions such as **Marker Noise**. The marker must be **ON** for the response to be valid.

Example:

`:CALC:MARK:MODE POS` activates a normal marker (marker 1) at the center of the display

`:CALC:MARK2:X 20 GHZ` selects marker 2 and moves it to 20 GHz. (Marker 2 must first be turned on.)

4.1.1 Select Marker

Selects one of the four possible marker or marker pairs. Once a marker is selected, it can be set to any of the control modes, **Normal**, **Delta**, **Delta Pair**, **Span Pair**, or **Off**.

State Saved: The number of the selected marker is saved in instrument state.

Factory Preset and *RST: Marker 1

Remote Command:

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

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

Sets or queries the state of a marker. Setting a marker to state ON or 1 selects that marker. Setting a marker which is OFF to state ON or 1 puts it in **Normal** mode and places it at the center of the display. Setting a marker to state OFF or 0 selects that marker and turns it off. The response to the query will be 0 if OFF, 1 if ON.

Example:

`:CALC:MARK2:STAT ON` selects marker 2.

4.1.2 Normal

Sets the control mode for the selected marker to **Normal** (see “Marker” on page 122). If the marker is off, a single marker is activated at the center of the display. If you are in a marker pair mode, for example **Delta Marker**, the reference marker is turned off. You can then adjust the trace point of the marker.

Remote Command:

See “Marker” on page 122 for the mode command.

Example:

`:CALC:MARK:MODE POS` selects marker 1 and sets it to **Normal**.

4.1.3 Delta

Sets the control mode for the selected marker to **Delta** (see “Marker” on page 122). In **Delta** mode the display shows the difference between the active (**Delta**) marker and a reference marker. When **Delta** mode is selected the reference marker is placed at the current marker position, unless the marker was **OFF**, in which case both the active marker and the reference marker are placed at the center of the display. You can adjust the trace point of the active delta marker. Annotation in the active function block and in the upper-right corner of the display indicates the frequency or time difference and amplitude difference of the two markers.

Selecting **Delta** while already in **Delta** mode causes the reference marker to be reset to the current active (Δ) marker position.

The amplitude of the reference marker is fixed. In non-zero spans the frequency of the reference marker is fixed. If the center frequency of the analyzer is changed, so that the reference marker is off the screen, an arrow will appear with the marker number at the left or the right side of the display, indicating where the trace point is for the reference marker.

In **Zero Span** the reference marker remains fixed at the trace point on which it was placed. Also, changing **Center Frequency** does not move the reference marker while in **Zero Span**.

Remote Command:

See "Marker" on page 122 for the mode command.

Example:

`CALC:MARK4:MODE DELT` selects marker 4 as a delta marker and places a reference marker at the marker 4 position. If marker 4 is OFF it places both the active and the reference markers at the center of the display.

4.1.4 Delta Pair

Sets the control mode for the selected marker to **Delta Pair** (see "Marker" on page 122). In **Delta Pair** mode the display shows the difference between the delta marker and a reference marker.

There are four conditions that can occur when **Delta Pair** mode is selected.

- If marker mode is **Off**, the delta marker and reference marker are placed at the center of the display.
- If marker mode is **Normal**, the delta marker and reference marker are placed at the current marker position on the trace.
- If the marker mode is **Delta**, the current marker position remains unchanged and the reference marker is placed on the trace at the reference marker position.
- If the marker mode is **Span Pair**, the marker positions remain unchanged.

The difference between **Delta Pair** and **Delta** modes is that in **Delta Pair** mode the reference marker stays on the trace and you can adjust its trace point. The note `(Tracking Ref)` appears on the **Delta Pair** key because, in effect, the reference marker "tracks" the trace. (By comparison, in **Delta** mode the reference marker does not track changes in the trace results, it remains anchored in amplitude and frequency.)

Pressing the key again toggles between the two markers you are controlling. When `Ref` is underlined you are controlling the reference marker. When Δ is underlined you are controlling the delta marker.

Once positioned, the markers stay on the trace points you have selected. Changing the frequency or sweep time of the analyzer does not change the trace point of the markers. You cannot move the markers off the screen.

Factory Preset and *RST:	`Ref` is the active parameter.

Remote Command:

See "Marker" on page 122 for the command to select the control mode.

`:CALCulate:MARKer[1]|2|3|4:X:POSition:STARt <param>`

Sets the reference marker X position to a specified trace point.

`:CALCulate:MARKer[1]|2|3|4:X:POSition:STARt?`

Queries the reference marker trace point.

`:CALCulate:MARKer[1]|2|3|4:X:POSition:STOP <param>`

Sets the delta marker X position to a specified trace point.

`:CALCulate:MARKer[1]|2|3|4:X:POSition:STOP?`

Queries the delta marker trace point.

Example:

`CALC:MARK3:MODE BAND` selects marker 3 and sets it to **Delta Pair**.

`CALC:MARK4:X:POS:STAR 0` moves the reference marker 4 to the left edge of the display.

4.1.5 Span Pair

Sets the control mode for the selected marker to **Span Pair** (see "Marker" on page 122). In **Span Pair** mode the display shows the difference between the delta marker and a reference marker.

There are four conditions that can occur when **Span Pair** mode is selected.

- If marker mode is **Off**, the delta marker and reference marker are placed at the center of the display.
- If marker mode is **Normal**, the delta marker and reference marker are placed at the current marker position on the trace.
- If the marker mode is **Delta**, the current marker position remains unchanged and the reference marker is placed on the trace at the reference marker position.
- If the marker mode is **Delta Pair**, the marker positions remain unchanged.

The difference between **Span Pair** and **Delta** modes is that in **Span Pair** mode the reference marker stays on the trace and you can adjust its trace point.

Pressing the key again toggles between the two markers you are controlling. Adjusting the Span (`Span` is underlined) changes the difference between the two markers. Adjusting Center (`Center` is underlined) maintains the marker spacing and changes the midpoint of the markers.

Once positioned, the markers stay on the trace points on which they have been placed. Changing the frequency or time of the analyzer does not change the trace point of the markers, that is, they stay at the same horizontal position on the display.

You cannot move the markers off the screen. If you adjust either center or span to a value that would cause one of the markers to move off screen, the marker will be placed at the right or left side of the display, on the trace.

Factory Preset and *RST: `Center` is the active parameter.

Remote Command:

See "Marker" on page 122 for the command to select the control mode.

`:CALCulate:MARKer[1]|2|3|4:X:POSition:CENTer <param>`

Sets the mid point of the markers to a specific trace point.

`:CALCulate:MARKer[1]|2|3|4:X:POSition:CENTer?`

Queries the midpoint trace point.

`:CALCulate:MARKer[1]|2|3|4:X:POSition:SPAN <param>`

Sets the spacing between the markers to a specified number of trace points.

:CALCulate:MARKer[1]|2|3|4:X:POSition:SPAN?

Queries the spacing of the markers in trace points.

Example:

`:CALC:MARK4:X:POS:SPAN 200` sets the spacing between the markers to 200 trace points for marker pair 4.

`:CALC:MARK2:X:POS:CENT 300` sets the midpoint between the markers to the 300th trace point from the left of the display. For a 601 point trace this will be the middle of the display.

4.1.6 Off

Turns off the selected marker. In addition, **Off** also turns off functions related to the selected marker such as **Signal Track**, **Band/Intvl Power**, and **Marker Noise**.

Remote Command:

See “Select Marker” on page 123 for the command to select the control mode.

Example:

`:CALC:MARK3:STAT OFF` selects marker 3 and sets it to **Off**.

4.1.7 Marker Trace

Chooses which trace the selected marker will be placed on. You can pick **Marker Trace** 1, 2, or 3, or Auto. In **Auto** mode, the analyzer places the marker on the lowest-numbered trace which is in **Clear Write** mode. If no trace is in **Clear Write** mode, it places the marker on the lowest-numbered trace in **View/Trace** mode. If no trace is in **View/Trace** mode, it places the marker on the lowest-numbered trace in either **Max Hold** or **Min Hold** mode. For example, if trace 1 is in view, and trace 2 is in clear-write, any new marker is assigned to trace 2.

State Saved:	The **Marker Trace** for each marker is saved in instrument state.
Factory Preset and *RST:	Trace 1

Remote Command:

:CALCulate:MARKer[1]|2|3|4:TRACe 1|2|3

Puts the marker on the specified trace and turns **Auto** OFF for that marker.

:CALCulate:MARKer[1]|2|3|4:TRACe?

The query returns the number of the trace on which the marker currently resides, even if that marker is in **Auto** mode.

:CALCulate:MARKer[1]|2|3|4:TRACe:AUTO OFF|ON|0|1

Turning **Auto** OFF sets the **Marker Trace** value to the number of the trace on which the marker currently resides.

:CALCulate:MARKer[1]|2|3|4:TRACe:AUTO?

The response to the query will be 0 if OFF, 1 if ON.

Example:

`:CALC:MARK1:TRAC 2` places marker 1 on trace 2.

4.1.8 Readout

This affects how the x-axis information for the selected marker is displayed in the marker area (top-right of display) and the active function area of the display. It only affects the readout of the horizontal position information (for example, frequency) on the display.

NOTE It does not affect the way this information is sent remotely in response to the `:CALC:MARK:X?` command.

State Saved: In instrument state, for each marker.

Factory Preset and *RST: Frequency for non-zero spans. Time for **Zero Span**.

Remote Command:

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

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

Example:

`:CALC:MARK3:X:READ TIME` sets the marker 3 **Readout** to **Time**.

4.1.8.1 Frequency

Sets the marker readout to **Frequency**. This is the default in non-zero spans. **Frequency** readout is not available in **Zero Span**. Displays the absolute frequency of a normal marker or the frequency of the delta marker relative to the reference marker.

Remote Command:

See "Readout" on page 127 for this command.

Example:

`:CALC:MARK2:X:READ FREQ` sets the marker 2 **Readout** to **Frequency.**

4.1.8.2 Period

Sets the marker readout to Period. Displays the reciprocal of the frequency at the marker position, or the reciprocal of the frequency separation of the two markers in a delta-marker mode. Period readout is not available in **Zero Span**.

If the markers are at the same freedmen in a delta marker mode, the result will be the reciprocal of 0, which is infinitely large. The display will show a very large number.

Remote Command:

See "Readout" on page 127 for this command.

Example:

```
:CALC:MARK2:X:READ PER
```

4.1.8.3 Time

Sets the marker readout to Time. Time is the default in **Zero Span**. Displays the time interval between a normal marker and the start of a sweep or the time of the delta marker relative to the reference marker. While in **Zero Span** the time value is the time position relative to the start of the sweep. In a delta-marker mode it is the (sweep) time interval between the two markers.

Remote Command:

See "Readout" on page 127 for this command.

Example:

```
:CALC:MARK2:X:READ TIM
```

4.1.8.4 Inverse Time

Sets the marker readout to Inverse Time. This function is only available when in both zero span and in a delta-marker modes. Displays the reciprocal of (sweep) time between two markers.

If the markers are at the same x position, the time between them is 0, so the reciprocal of sweep time is infinitely large. The display will show a very large number.

Remote Command:

See "Readout" on page 127 for this command.

Example:

```
:CALC:MARK2:X:READ ITIM
```

4.1.9 Marker Table

When set to On the display is split into a measurement window and a marker data display window. For each marker pair, information is displayed in the data display window, which includes the marker number, trace number, marker type, X axis value, and the amplitude of the marker or the delta value, if a delta marker, or the function value, if in a marker function such as **Marker Noise** or **Band/Intvl Power**.

Factory Preset and *RST:	Off

Remote Command:

`:CALCulate:MARKer:TABLe:STATe OFF|ON|0|1`

`:CALCulate:MARKer:TABLe:STATe?` returns 1 if ON or 0 if OFF.

Example:

`:CALC:MARK:TABL:STAT ON` turns on the marker table.

4.1.10 Marker All Off

Turns off all markers, including markers used for signal track. This key also turns off marker related functions such as **Signal Track**, **Band Interval Power**, and **Marker Noise**.

NOTE	Selecting any measurement (including **Meas Off**) under **Measure**, turns off the marker table.

Remote Command:

`:CALCulate:MARKer:AOFF`

Example:

`:CALC:MARK:AOFF` turns off all markers.

4.2 Marker -->

Accesses menu keys which allow the current marker value to be copied into other instrument parameters (for example, **Center Frequency**).

Remote Command:

There is no remote command for this key.

4.2.1 Mkr->CF

Sets the center frequency of the analyzer to the frequency of the selected marker. The marker stays at this frequency, so it moves to the center of the display. This function is not available in **Zero Span**.

Remote Command:

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

Example:

`:CALC:MARK2:CENT` sets the CF of the analyzer to the value (or delta value) of marker 2.

4.2.2 Mkr->CF Step

Sets the center frequency (CF) step size of the analyzer to the marker frequency, or, in a delta-marker mode, to the frequency difference between the delta and reference markers. The step size can be verified by pressing **Frequency**, **Center Freq**. The step size is displayed in the third line of the active function area of the display.   This function is not available in **Zero Span**.

Remote Command:

`:CALCulate:MARKer[1]|2|3|4[:SET]:STEP`

Example:

`:CALC:MARK1:STEP` sets the CF step to the value (or delta value) of marker 1.

4.2.3 Mkr-->Start

Changes the start frequency to the frequency of the active marker. The marker stays at this frequency, so it moves to the left of the display. This function is not available in **Zero Span**.

Remote Command:

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

Example:

`:CALC:MARK1:STAR` sets the start frequency to the value (or delta value) of marker 1.

4.2.4 Mkr->Stop

Changes the stop frequency to the frequency of the active marker. The marker stays at this frequency, so it moves to the right of the display. This function is not available in **Zero Span**.

Remote Command:

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

Example:

`:CALC:MARK1:STOP` sets the stop frequency to the value (or delta value) of marker 1.

4.2.5 MkrΔ->Span

Sets the start and stop frequencies to the values of the delta markers. Only available in Delta mode, this function is not available in **Zero Span**.

Remote Command:

`:CALCulate:MARKer[1]|2|3|4[:SET]:SPAN`

Example:

`:CALC:MARK2:SET:SPAN` sets the start and stop frequencies to the values of the delta markers of marker 2.

4.2.6 Mkr->Ref Lvl

Sets the reference level to the amplitude value of the active marker.

In a delta-marker mode, sets reference level to the amplitude difference between the markers.

NOTE The reference level range is limited by the input attenuator setting.

Remote Command:

`:CALCulate:MARKer[1]|2|3|4[:SET]:RLEVel`

Example:

`:CALC:MARK2:RLEV` sets the reference level of the analyzer to the amplitude of marker 2.

4.3 Mkr Fctn

Accesses special marker functions.

State Saved: If a marker function (for example, **Marker Noise** or **Band/Intvl Power**) is on, that fact is saved in instrument state.

Factory Preset and *RST: Off

Remote Command:

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

4.3.1 Select Marker

See “Select Marker” on page 123.

4.3.2 Marker Noise

Activates a noise marker at the center of the display. Reads out the average noise level, normalized to a 1 Hz noise power bandwidth, around the active marker. If the detector is set to Auto, the average detector is chosen. Useful measurements can also be made with the peak detector if noise peaks are of interest. Also, the sample detector makes accurate noise measurements, though with a higher variance than the average detector. The noise marker averages 32 trace data values centered about the location of the marker on the frequency or time scale. The 32 values averaged (unless centered within the first 15 or last 14 values in the trace) commence with the 16th point to the left of the marker, and end with the 15th point to the right of the marker.

The data displayed (if the marker is in Normal mode) is the noise density around the marker. The value readout is followed by "(1Hz)" to remind you that display is normalized to a one Hz bandwidth.

To measure signal to noise ratio, be sure that **Marker Noise** is **Off**, and **Marker** is set to **Normal**. Place the marker on the signal peak, then select **Delta** mode. Now place the active (Δ) marker on the noise, then press **Marker Noise** to select **On**. In this case, the reference marker has units of amplitude and the data displayed is the ratio of the noise density at the delta marker to the reference marker power. The value readout is dB/Hz if the Y-axis units are logarithmic, and % if the Y-axis units are linear. It is understood, in this case, that % stands for the units $\%/\sqrt{\mathrm{Hz}}$ for volts units and 5/Hz for watts units.

To measure the ratio of the noise densities at two points, be sure that **Marker Noise** is **On**, and **Marker** is set to **Normal**. Select **Delta** mode, then move the active (Δ) marker to the second noise point. In this case both markers have units of noise density (for example, dBm/Hz), so the data displayed represents the ratio of the noise density at the delta marker to the noise density at the reference marker. The value readout is displayed as a ratio (dB or %).

To guarantee accurate data for noise-like signals, a correction for equivalent noise bandwidth is made by the analyzer. The **Marker Noise** function accuracy is best when the detector is set to **Average** or **Sample**, because neither of the detectors peak-biases the noise. The trade-off between sweep time and variance of the result is best when **Avg/VBW Type** is set to Power Averaging. **Auto** coupling, therefore, normally chooses the **Average** detector and **Power Averaging**. But, the Marker Noise function still works with all settings of detector and **Avg/VBW Type**.

Note that the value when the Y-axis units are watts is the square of the value when the Y-axis units are volts. For example, when the percent ratio with Y-axis units in volts is 20% (0.2), the percent ratio with Y-axis units in watts will be 4% ($0.2^2 = 0.04$). When you read the value out remotely you have to know whether you are in log (dB) or linear (percent).

Video triggering is not available when the detector is **Average**. Marker functions that would set the detector to **Average**, and thus conflict with video triggering, are not available when the **Video** trigger is **On**.

Remote Command:

See “Marker -->” on page 130 for the command to select a function.

Example:

`:CALC:MARK:Y?` returns the value of the **Marker Noise** function for marker 1 (if **Marker Noise** is ON for marker 1).

4.3.3 Band/Intvl Power

Measures the power in a bandwidth (non-zero span) or time interval (zero span) specified by the user. If no marker is on, this key activates Delta Pair mode. If the detector mode is set to **Auto**, the average detector is selected. If the **Avg/VBW** type is set to **Auto**, **Power Averaging** is selected. The active marker pair indicate the edges of the band. The measurement can be made on a single sweep or can continuously update at the end of each sweep. Only **Delta Pair** and **Span Pair** marker control modes can be used while in this function, selecting any other mode (for example, Normal or Delta) turns off this function.

Video triggering is not available when the detector is **Average**. Marker functions that would set the detector to **Average**, and thus conflict with video triggering, are not available when the **Video** trigger is **On**.

Remote Command:

See “Marker -->” on page 130 for the command to select the function.

Example:

`:CALC:MARK:Y? returns the value of the` **Band/Intvl Power** `function for marker 1 (if` **Band/Intvl Power** `is ON for marker 1).`

4.3.4 Function Off

Turns off marker functions (Band/Intvl Power and Marker Noise).

NOTE Delta markers will remain on screen.

Remote Command:

See "Marker -->" on page 130 for the command to select the function.

Example:

`:CALC:MARK2:FUNC OFF` turns the marker function to OFF.

4.3.5 Marker Count

Accesses the marker count menu.

4.3.5.1 Marker Count

Turns the marker frequency counter for the active marker on or off. The **Marker Count** key turns on the marker counter. If no marker is active before **Marker Count** is pressed, a marker is activated at the center of the display. Press **Marker Count** again to turn the marker counter off. **Marker Count** frequency readings are not affected by the frequency offset function.

The span to resolution bandwidth ratio must be less than 500 for the marker count function to work properly. If this is not the case, an error will be displayed on the screen (`Freq Count: Reduce Span/RBW ratio`). When you see this error, decrease the span or increase the resolution bandwidth until the error goes away.

In **Zero Span** the counter continues to function, counting any signal near the center frequency of the analyzer.

NOTE **Functions Off** does not turn **Marker Count** off.

State Saved: If **Marker Count** is on, that fact is saved in the instrument state.

Remote Command:

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

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

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

Example:

`:CALC:MARK:FCO:X?` returns the counted frequency. Returns 9e15 if **Marker Count** is **OFF**.

`:CALC:MARK2:FCO OFF`

4.3.5.2 Gate Time

Controls the length of time during which the frequency counter measures the signal frequency. For 10 ms and longer gate times, the counter resolution can be as low as 0.001 Hz.

Longer gate times allow for greater averaging of signals whose frequency is “noisy”, at the expense of throughput. If the gate time is an integer multiple of the length of a power-line cycle (20 ms for 50 Hz power, 16.67 ms for 60 Hz power), the counter rejects incidental modulation at the power line rate. The shortest gate time that rejects both 50 and 60 Hz modulation is 100 ms, which is the value chosen when gate time is in **Auto**.

State Saved: Saved in instrument state.

Factory Preset and *RST: Auto, 100 ms

Maximum Value: 500 ms

Minimum Value: 1 μs

Remote Command:

```
:CALCulate:MARKer[1]|2|3|4:FCOunt:GATetime:AUTO OFF|ON|0|1
:CALCulate:MARKer[1]|2|3|4:FCOunt:GATetime:AUTO?
:CALCulate:MARKer[1]|2|3|4:FCOunt:GATetime <val>
:CALCulate:MARKer[1]|2|3|4:FCOunt:GATetime?
```

Example:

`:CALC:MARK:FCO:GAT:AUTO On`

`:CALC:MARK2:FCO:GAT 1e-2` sets the gate time to 10^{-2} sec = 10 ms.

4.4 Peak Search

Places a marker on the highest peak and accesses the search menu. If Peak Search (Param) is set, then the peak found must meet the defined peak excursion and threshold values. (See "Search Param" on page 138.) Ignores peaks closer to 0 Hz than 1% of the current span. For example, if Span is 1 MHz, peaks will not be found between −10 kHz and +10 kHz. If no valid peak is found, an error (`No Peak Found`) is displayed. Press **Esc** to clear this message before attempting another search.

NOTE You can go into the Peak Search menu without actually performing a **Peak Search** by using the front-panel **Return** key (assuming you have previously accessed the Peak Search menu). Press **Return** to navigate through the previously accessed menus until you return to the Peak Search menu.

Remote Command:

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

Example:

`:CALC:MARK2:MAX` performs a peak search using marker 2.

`:CALC:MARK2:Y?` queries the marker amplitude (Y-axis) value for marker 2.

`:CALC:MARK2:X?` queries the marker frequency or time (X-axis) value for marker 2.

NOTE It is recommended that, in a command sequence such as that shown above, each command be placed on a separate line. This ensures that the peak has been performed properly before the data is read.

4.4.1 Next Peak

Places the marker on the next highest peak below the current peak. The peak must meet the defined peak excursion and threshold values. Ignores peaks closer to 0 Hz than 1% of the current span. If no valid peak is found, an error (`No Peak Found`) is displayed. Press **Esc** to clear this message before attempting another search.

Remote Command:

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

Example:

`:CALC:MARK2:MAX:NEXT` selects marker 2 and moves it to the next highest peak.

4.4.2 Next Pk Right

Moves the marker to the next peak to the right of the current marker. The peak must meet the defined peak excursion and threshold limits. Ignores peaks closer to 0 Hz than 1% of the current span. If no valid peak is found, an error (`No Peak Found`) is displayed. Press **Esc** to clear this message before attempting another search.

Remote Command:

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

Example:

`:CALC:MARK2:MAX:RIGH` selects marker 2 and moves it to the next peak to the right.

4.4.3 Next Pk Left

Moves the marker to the next peak to the left of the current marker. The peak must meet the defined peak excursion and threshold limits. Ignores peaks closer to 0 Hz than 1% of the current span. If no valid peak is found, an error "`No Peak Found`" is displayed. Press **Esc** to clear this message before attempting another search.

Remote Command:

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

Example:

`:CALC:MARK2:MAX:LEFT` selects marker 2 and moves it to the next peak to the left.

4.4.4 Min Search

Moves the active marker to the minimum detected amplitude value on the current trace.

Remote Command:

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

Example:

`:CALC:MARK:MIN` selects marker 1 and moves it to the minimum amplitude value.

4.4.5 Pk-Pk Search

Finds and displays the amplitude and frequency (or time, if in zero span) differences between the highest and lowest trace points by setting a reference marker on the peak signal and placing a Δ marker on the minimum signal.

Remote Command:

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

Example:

`:CALC:MARK:PTP`

`:CALC:MARK:Y?` queries the delta amplitude value for marker 1.

4.4.6 Mkr->CF

See "Mkr->CF" on page 130 for the command to select this function.

4.4.7 Continuous Pk

When a marker is placed on a signal and **Continuous Pk** is pressed, the marker will remain on the signal even if the signal frequency changes, as long as the amplitude of the signal does not change by more than 3 dB from one sweep to another.

If the signal is lost, an attempt will be made to find it again and continue tracking. If there are other signals on screen near the same amplitude, one of them may be found instead. Signals near 0 Hz cannot be tracked effectively as they cannot be distinguished from the LO feedthrough, which is excluded by intent from the search algorithm.

NOTE This function is intended to track signals with a frequency that is changing, and an amplitude that is not changing.

State Saved:	If **On**, the fact is saved in instrument state.
Factory Preset and *RST:	Off

Remote Command:

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

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

Example:

`:CALC:MARK:CPE ON`

4.4.8 Search Param

Accesses the Search Parameter menu that allows you to adjust the parameters for the peak search functions.

Remote Command:

There is no remote command for this key.

4.4.8.1 Peak Excursn

Sets the minimum amplitude variation of signals that the marker can identify as a separate peak. For example, if a value of 10 dB is selected, the marker Next Peak function moves only to peaks that rise more than 10 dB above the Peak Threshold and then fall back to the Peak Threshold. This applies to all traces.

Applies to **Next Peak**, **Next Peak Left**, and **Next Peak Right**. If **Peak Search** (Param) is set, it also applies to **Peak Search**.

State Saved: Saved in instrument state.

Factory Preset and *RST: 6.0 dB

Maximum Value: 0.0 dB

Minimum Value: 100 dB

Remote Command:

```
:CALCulate:MARKer[1]|2|3|4:PEAK:EXCursion <rel_amplitude>
:CALCulate:MARKer[1]|2|3|4:PEAK:EXCursion?
```

Example:

`:CALC:MARK:PEAK:EXC 40 dB` sets the peak excursion to 40 dB.

4.4.8.2 Pk Threshold

Specifies the minimum signal level for the analyzer's internal peak identification routine to recognize a signal as a peak. A signal must rise above the Peak Threshold by the value specified in **Peak Excursn**, then fall back to the Peak Threshold, to be considered a peak. This applies to all traces and all windows. Press ESC or select another active function to hide the threshold line.

Applies to **Next Peak**, **Next Peak Left**, and **Next Peak Right**. If **Peak Search** (Param) is set, it also applies to **Peak Search**.

State Saved: Saved in instrument state.

Factory Preset and *RST: -90 dBm

Maximum Value: Current reference level

Minimum Value: Bottom of the display range

Remote Command:

```
:CALCulate:MARKer[1]|2|3|4:PEAK:THReshold <ampl>
:CALCulate:MARKer[1]|2|3|4:PEAK:THReshold?
```

Example:

```
:CALC:MARK:PEAK:THR -20 dB
```

4.4.8.3 Peak Search

Sets the mode for Peak Search to either **Max** or **Param**.

- **Max** (Maximum mode) places a marker on the highest peak whenever a **Peak Search** is performed.
- **Param** (Parameter mode) searches only for peaks that meet the values set with **Peak Excursn** and **Pk Threshold**.

Applies to **Peak Search** only.

State Saved: No save

Factory Preset and *RST: Maximum

Remote Command:

```
:CALCulate:MARKer:PEAK[1]|2|3|4:SEARch:MODE PARameter|MAXimum
:CALCulate:MARKer:PEAK[1]|2|3|4:SEARch:MODE?
```

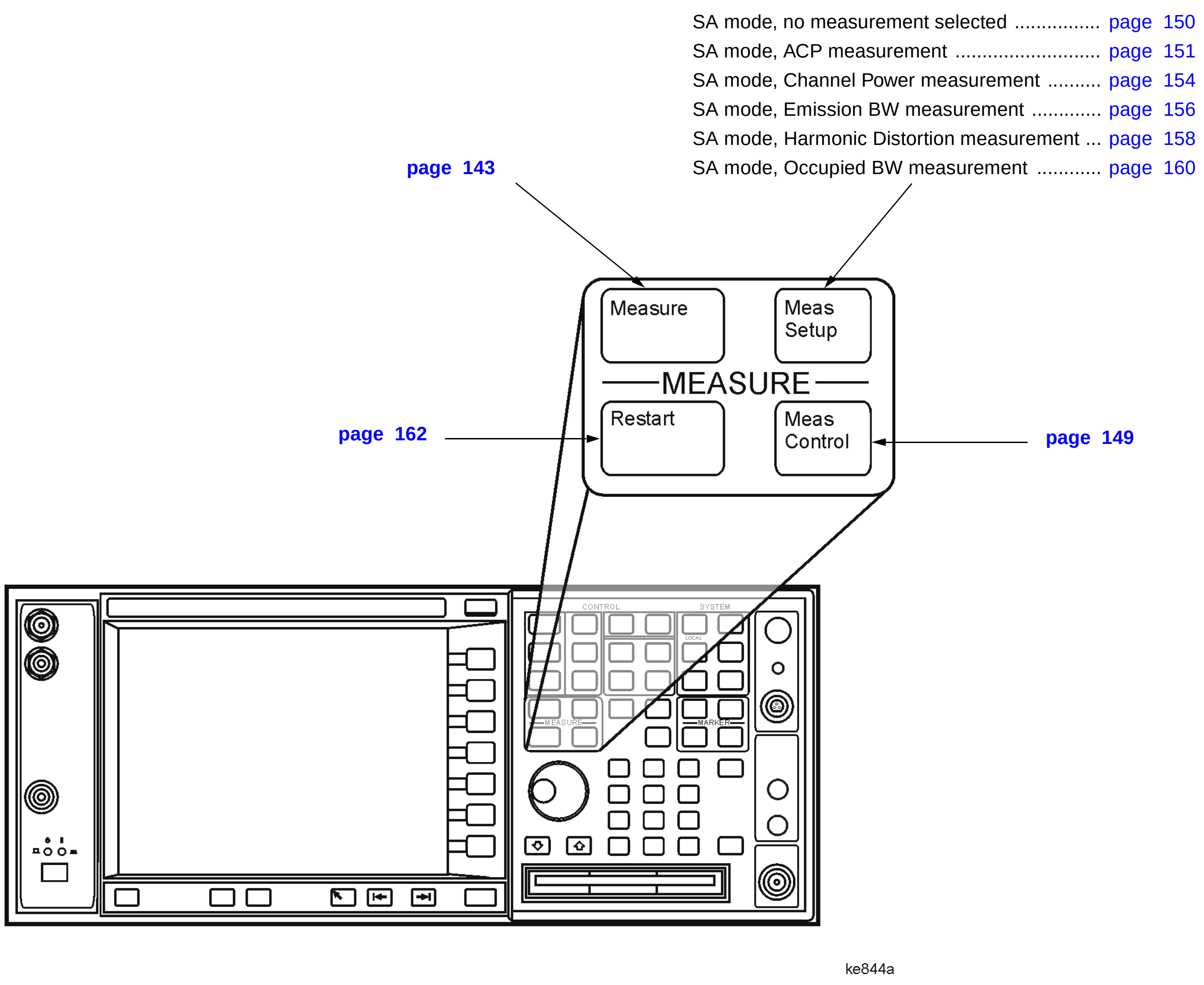
Example:

```
:CALC:MARK:PEAK:SEAR:MODE MAX
```

5 Measure Keys

This chapter provides key descriptions and programming information for the Measure functions of your analyzer. The front-panel Measure functions are listed alphabetically and are described with their associated menus keys. The lower-level menu keys are arranged and described as they appear in your analyzer menus with the corresponding front-panel key.

Measure Function Keys and Where to Find Them


SA mode, no measurement selected .... page 150
SA mode, ACP measurement .... page 151
SA mode, Channel Power measurement .... page 154
SA mode, Emission BW measurement .... page 156
SA mode, Harmonic Distortion measurement ... page 158
SA mode, Occupied BW measurement .... page 160
page 143
page 162
page 149
Measure
Meas Setup
MEASURE
Restart
Meas Control
CONTROL
SYSTEM
LOCAL
MEASURE
MARKER
ke844a

5.1 MEASURE (Spectrum Analysis Mode)

If you are in the Spectrum Analysis mode (see the **Mode** key), this key accesses a menu of keys that allow you to make transmitter power measurements such as adjacent channel power, occupied power bandwidth, and harmonic distortion measurements. If other modes are available and have been selected, the measurements for that particular mode will be displayed.

Remote Command:

There is no equivalent command.

5.1.1 Meas Off

Turns the active measurement function off.

Remote Command:

`:CONFigure:SANalyzer`

Example:

`CONF:SAN`

`:CONFigure:SANalyzer` causes the present measurement to exit and places the analyzer in base instrument general purpose spectrum analyzer state. The command `CONFigure:<measurement>` will always set `INITiate:CONTinuous OFF` (single measurement mode), and also places the measurement in the idle state.

5.1.2 ACP

Turns on the adjacent channel power measurement. The center frequency, reference level, and channel bandwidth must be set by the user. The span is set slightly wider than three channels, each designated by a pair of vertical lines and double-headed arrows and turns on the ACP measurement. The screen is split and the lower window displays the absolute power in the center channel in dBm and the power in each of the adjacent channels in dB relative to the center-channel power. Also displayed are center-channel bandwidth, adjacent-channel bandwidth, and channel spacing. The resolution bandwidth is set to nominally 1 to 3 percent of the center channel bandwidth, and the video bandwidth is set ten times wider than the resolution bandwidth to minimize its averaging effect.

The measurement settings may be configured by pressing **Meas Setup** after **ACP** has been selected. Pressing **Meas Control** allows you to pause or restart your measurement, or toggle between continuous and single measurement.

Remote Command:

`:CONFigure:ACP`

:FETCh:ACP?

:FETCh:ACP:MAIN|LOWer|UPPer?

:MEASure:ACP?

:MEASure:ACP:MAIN|LOWer|UPPer?

:READ:ACP?

:READ:ACP:MAIN|LOWer|UPPer?

Example:

FETC:ACP? or MEAS:ACP? or READ:ACP? commands return the scalar results of main channel power, lower channel power (relative), and upper channel power (relative)

FETC:ACP:MAIN? or MEAS:ACP:LOW? or READ:ACP:UPP? commands will return the single scalar result specified.

NOTE 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.

5.1.3 Channel Power

Turns on the channel power measurement and measures the power and power spectral density in the channel bandwidth that you specify. One marker pair on the display indicates the edges of the channel bandwidth. The center frequency, reference level, and channel bandwidth must be set by the user.

The measurement settings may be configured by pressing **Meas Setup** after **Channel Power** has been selected. The measurement can be made in single or continuous sweep mode. Pressing **Meas Control** allows you to pause or restart your measurement, or toggle between continuous and single measurement.

Remote Command:

:CONFigure:CHPower

:FETCh:CHPower?

:FETCh:CHPower:CHPower?

:FETCh:CHPower:DENSity?

:MEASure:CHPower?

:MEASure:CHPower:CHPower?

:MEASure:CHPower:DENSity?

:READ:CHPower?

:READ:CHPower:CHPower?

:READ:CHPower:DENSity?

Example:

FETC:CHP? or MEAS:CHP? or READ:CHP? command returns scalar results of main channel power and power density.

FETC:CHP:CHP? or MEAS:CHP:DENS? or READ:CHP:DENS? commands will return the single scalar result specified.

NOTE 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.

5.1.4 Occupied BW

Turns on the occupied bandwidth measurement and defaults to 99% of the occupied bandwidth power. Integrates the power of the displayed spectrum and puts markers at the frequencies between which a selected percentage of the power is contained. The power-bandwidth routine first computes the combined power of all signal responses contained in the trace. For 99% occupied power bandwidth, markers are placed at the frequencies on either side of 99% of the power. 1% of the power is evenly distributed outside the markers. The difference between the marker frequencies is the 99% power bandwidth and is the value displayed.

The occupied bandwidth function also indicates the difference between the analyzer center frequency and the center frequency of the channel. The measurement can be made in single or continuous sweep mode. The center frequency, reference level, and channel spacing must be set by the user.

The measurement settings may be configured by pressing **Meas Setup** after **Occupied BW** has been selected. Pressing **Meas Control** allows you to pause or restart you measurement, or toggle between continuous and single measurement.

Remote Command:

:CONFigure:OBW

:FETCh:OBW?

:FETCh:OBW:OBWidth|FERRor?

:MEASure:OBW?

:MEASure:OBW:OBWidth|FERRor?

:READ:OBW?

:READ:OBW:OBWidth|FERRor?

Example:

FETC:OBW? or MEAS:OBW? or READ:OBW? command returns scalar results of occupied bandwidth and transmit frequency error.

FETC:OBW:OBW? or MEAS:OBW:FERR? or READ:OBW:FERR? commands will return the single scalar result specified.

NOTE The results for both values are returned in Hz and in a comma-separated list.

5.1.5 Emission BW

Turns on the emission bandwidth measurement which measures the bandwidth between 2 points on a signal which are a specified number of dB below the highest point within the occupied bandwidth span.

For example: If the Emission BW X dB is set to –26 dB, and the Occupied BW Span is set to 10 MHz, then the peak signal power level is first determined from the 10 MHz wide trace sweep. The frequency of this trace maximum is designated as f0. Next, the analyzer determines the two frequencies furthest below and furthest above f0 at which the signal level is 26 dB below the peak level; these frequencies are designated as f1 and f2 respectively. The emission bandwidth = f2 – f1.

The measurement settings may be configured by pressing **Meas Setup** after **Emission BW** has been selected. Pressing **Meas Control** after allows you to pause or restart you measurement, or toggle between continuous and single measurement.

Remote Command:

:CONFigure:EBWidth

:FETCh:EBWidth?

:FETCh:EBWidth:EBWidth?

:MEASure:EBWidth?

:MEASure:EBWidth:EBWidth?

:READ:EBWidth?

:READ:EBWidth:EBWidth?

Example:

FETC:EBW? or FETC:EBW:EBW?

MEAS:EBW? or MEAS:EBW:EBW?

READ:EBW? or FETC:EBW:EBW?

All of these commands return the value of emission bandwidth in Hz.

5.1.6 Harmonic Dist

Turns on the harmonic distortion measurement that measures the harmonics of a single carrier signal and computes the total harmonic distortion. The carrier must be the largest amplitude peak (having a frequency
> 0 Hz, a peak excursion > 6 dB on both sides, and an amplitude ≥ –50 dBm) on the display at the time the measurement is started. The total harmonic distortion is then calculated from the measured harmonics.

When measuring the Nth harmonic, the analyzer will choose the narrowest resolution bandwidth that is ≤ N times the resolution bandwidth used to measure the fundamental. Widening the resolution bandwidth allows the measurement to capture all modulation on the harmonics. An asterisk (*) will appear next to the amplitude of measured harmonics for which the desired resolution bandwidth could not be set. The measurement will still be accurate as long as the signal has little or no modulation. The measurement precision is two decimal places for the amplitude results and four significant digits for the frequency results.

The measurement settings may be configured by pressing **Meas Setup** after **Harmonic Dist** has been selected. Pressing **Meas Control** allows you to pause or restart your measurement, or toggle between continuous and single measurement.

Remote Command:

```
:CONFigure:HARMonics

:FETCh:HARMonics:AMPL:ALL?
:FETCh:HARMonics:AMPL[n]?
:FETCh:HARMonics[:DISTortion]?
:FETCh:HARMonics:FREQ:ALL?
:FETCh:HARMonics:FREQ[n]?
:FETCh:HARMonics:FUNDamental?

:MEASure:HARMonics:AMPL:ALL?
:MEASure:HARMonics:AMPL[n]?
:MEASure:HARMonics[:DISTortion]?
:MEASure:HARMonics:FREQ:ALL?
:MEASure:HARMonics:FREQ[n]?
:MEASure:HARMonics:FUNDamental?

:READ:HARMonics:AMPL:ALL?
:READ:HARMonics:AMPL[n]?
:READ:HARMonics[:DISTortion]?
:READ:HARMonics:FREQ:ALL?
:READ:HARMonics:FREQ[n]?
```

:READ:HARMonics:FUNDamental?

Example:

`FETC:HARM:AMPL:ALL? or MEAS:HARM:AMPL:ALL or READ:HARM:AMPL:ALL`

Commands return a comma-separated list of the amplitudes of the first ten harmonics. The first value (for the fundamental) is measured in dBm. The remaining harmonics are measured in dBc from the fundamental. If fewer than ten harmonics are measured, zero is returned for any harmonic not measured.

`FETC:HARM:AMPL? or MEAS:HARM:AMPL2 or READ:HARM:AMPL9`

Commands return the amplitude of the specified harmonic number n. The FETCH example (where n=1) returns the amplitude of the fundamental in units of dBm The MEAS and READ examples return the second and ninth harmonics amplitude measured in dBc from the fundamental.

`FETC:HARM:DIST? or MEAS:HARM? or READ:HARM?`

Command returns the computed total harmonic distortion as a percentage.

`FETC:HARM:FREQ:ALL? or MEAS:HARM:FREQ:ALL or READ:HARM:FREQ:ALL`

Commands return a comma-separated list of the frequencies of the first ten harmonics in Hz. The first harmonic is the fundamental. If fewer than ten harmonics are measured, zero is returned for any harmonic not measured.

`FETC:HARM:FREQ2? or MEAS:HARM:FREQ3 or READ:HARM:FREQ10`

Commands return the amplitude of the specified harmonic number N. N = 2 to 10. These examples the frequency of the second, third and tenth harmonics in Hz.

`FETC:HARM:FUND? or MEAS:HARM:FUND? or READ:HARM:FUND?`

Commands return the frequency of the fundamental, measured in Hz.

5.1.7 Current Measurement Query (Remote Command Only)

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

Remote Command:

:CONFigure?

Example:

`:CONF?`

5.2 Meas Control

This functionality is not currently implemented from the front panel.

5.2.1 Pause the Measurement (Remote Command Only)

This command pauses the measurement that is currently running.

Remote Command:

`:INITiate:PAUSe`

Example:

```
:INIT:PAUS
```

5.2.2 Resume the Measurement (Remote Command Only)

This command resumes a measurement that has previously been paused.

Remote Command:

`:INITiate:RESume`

Example:

```
:INIT:RES
```

5.3 Meas Setup (SA with Measurements Off)

5.3.1 Measurement Setup

Displays the setup menu for the currently selected measurement. This menu is empty if no measurement is active. This could be because **Meas Off** is selected in the **Measure** menu.

Dependencies/ Couplings: Menu choices depend on the currently selected Mode and Menu

Remote Command:

There is no equivalent remote command.

5.4 Meas Setup (ACP Measurement)

If the ACP measurement has been selected in the Measure key menu of the Spectrum Analysis Mode, this key displays the appropriate measurement setup menu.

Remote Command:

There is no equivalent remote command.

5.4.1 Avg Number

Press Avg Number (On) to specify the number of measurements that will be averaged when calculating the measurement result. The average will be displayed at the end of each sweep.

State Saved: Saved in instrument state.

Factory Preset and *RST: 10 averages / Off

Minimum Value: 1

Maximum Value: 8192

Remote Command:

`[:SENSe]:ACP:AVERage:COUNt <integer>`

`[:SENSe]:ACP:AVERage:COUNt?`

`[:SENSe]:ACP:AVERage[:STATe] OFF|ON|0|1`

`[:SENSe]:ACP:AVERage[:STATe]?`

Example:

```
ACP:AVER:COUN 10
ACP:AVER OFF
```

5.4.2 Main Chan BW

When combined with Chan Spacing, Main Chan BW allows you to specify the range of integration used in calculating the power in the center reference channel. Changing Main Chan BW automatically changes Adj Chan BW and Chan Spacing to the same value.

State Saved: Saved in instrument state.

Factory Preset and *RST: 2 MHz

Default Terminator: Hz

Maximum Value: 7.5 GHz

Minimum Value: > adjacent channel BW, min of 50 Hz

Remote Command:

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

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

Example:

```
ACP:BWID:INT 5E6
```

5.4.3 Adj Chan BW

When combined with Chan Spacing, Adj Chan BW allows you to specify the range of integration used in calculating the power for the upper and lower adjacent channels. The two adjacent channels always have the same bandwidth.

NOTE The ratio of the measurement span to the adjacent channel integration bandwidth (**Adj Chan BW**) must be <50:1. The measurement span is determined by the channel spacing and the integration bandwidth.

State Saved: Saved in instrument state.

Factory Preset and *RST: 2 MHz

Default Terminator: Hz

Maximum Value: 30 MHz

Minimum Value: 75 kHz

Remote Command:

`[:SENSe]:ACP:BANDwidth|BWIDth:ACHannel <freq>`

`[:SENSe]:ACP:BANDwidth|BWIDth:ACHannel?`

Example:

```
ACP:BAND:ACH 10 MHZ
```

5.4.4 Chan Spacing

Allows you to specify the difference between the center frequency of the center channel and the center frequency of both adjacent channels. Adjacent channels must be symmetrical. Overlapping the channels is not allowed.

NOTE The ratio of the measurement span to the adjacent channel integration bandwidth (**Adj Chan BW**) must be <50:1. The measurement span is determined by the channel spacing and the integration bandwidth.

State Saved: Saved in instrument state.

Factory Preset and *RST: 3 MHz

Default Terminator: Hz

Maximum Value: dependent on main channel BW and adjacent channel BW

Minimum Value: 1/2(main channel BW) +1/2(adjacent channel BW)

Remote Command:

`[:SENSe]:ACP:CSPacing <freq>`

`[:SENSe]:ACP:CSPacing?`

Example:

```
ACP:CSP 5 MHz
```

5.5 Meas Setup (Channel Power)

When the channel power measurement has been selected in the Measure key menu of the Spectrum Analysis Mode, this key displays the appropriate measurement setup menu.

Remote Command:

There is no equivalent remote command.

5.5.1 Avg Number

Press **Avg Number (On)** to specify the number of measurement averages used when calculating the measurement result. The average will be displayed at the end of each sweep.

State Saved: Saved in instrument state.

Factory Preset and *RST: 10 averages/ Off

Minimum Value: 1

Maximum Value: 8192

Remote Command:

```
[:SENSe]:CHPower:AVERage:COUNt <integer>
[:SENSe]:CHPower:AVERage:COUNt?
[:SENSe]:CHPower:AVERage[:STATe] OFF|ON|0|1
[:SENSe]:CHPower:AVERage[:STATe]?
```

Example:

```
CHP:AVER:COUN 10
CHP:AVER OFF
```

5.5.2 Integration BW

Allows you to specify the range of integration used in calculating the power in the channel. (i.e. sets the main (center) channel bandwidth.)

State Saved: Saved in instrument state.

Factory Preset and *RST: 2 MHz

Default Terminator: Hz

Minimum Value: 100 Hz

Maximum Value: 26.5 GHz

Remote Command:

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

Example:

```
CHP:BAND:INT 1 MHz
```

5.5.3 Chan Pwr Span

Allows you to set the analyzer span for the channel power measurement.

State Saved: Saved in instrument state.

Factory Preset and *RST: 3 MHz

Default Terminator: Hz

Minimum Value: current integration bandwidth

Maximum Value: 10 times the integration bandwidth

Remote Command:

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

Example:

```
CHP:FREQ:SPAN 2 MHz
```

5.6 Meas Setup (Emission BW)

When the emission BW measurement has been selected in the Measure key menu of the Spectrum Analysis Mode, this key displays the appropriate measurement setup menu.

Remote Command:

There is no equivalent remote command.

5.6.1 Avg Number

Press **Avg Number (On)** to specify the number of measurement averages used when calculating the measurement result. The average will be displayed at the end of each sweep.

State Saved: Saved in instrument state.

Factory Preset and *RST: 10 averages / Off

Minimum Value: 1

Maximum Value: 8192

Remote Command:

`[:SENSe]:EBWidth:AVERage:COUNt <integer>`

`[:SENSe]:EBWidth:AVERage:COUNt?`

`[:SENSe]:EBWidth:AVERage[:STATe] OFF|ON|0|1`

`[:SENSe]:EBWidth:AVERage[:STATe]?`

Example:

```
EBW:AVER:COUN 100
EBW:AVER ON
```

5.6.2 Max Hold

Max Hold (On) displays and holds the maximum responses of a signal.

State Saved: Saved in instrument state.

Factory Preset and *RST: On

Remote Command:

`[:SENSe]:EBWidth:MAXHold ON|OFF|0|1`

`[:SENSe]:EBWidth:MAXHold?`

Example:

```
EBW:MAXH ON
```

5.6.3 EBW Span

Allows you to specify the measurement span over which to search for the peak level and X dB level transition points of the signal. The analyzer span will be set to the same value as the emission bandwidth span for the measurement. The emission bandwidth span should be set to approximately twice the expected emission bandwidth result.

State Saved: Saved in instrument state.

Factory Preset and *RST: 3 MHz

Default Terminator: Hz

Minimum Value: 100 Hz

Maximum Value: 26.5 GHz

Remote Command:

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

`[:SENSe]:EBWidth:FREQuency:SPAN?`

Example:

```
EBW:FREQ:SPAN 1 MHz
```

5.6.4 Emiss BW X dB

X dB is the amount of power (number of dBs) below the highest point in the signal at which to measure the emission bandwidth.

State Saved: Saved in instrument state.

Factory Preset and *RST: -26 dB

Default Terminator: dB

Minimum Value: -100.0 dB

Maximum Value: -0.10 dB

Remote Command:

`[:SENSe]:EBWidth:XDB <rel_amp>`

`[:SENSe]:EBWidth:XDB?`

Example:

```
EBW:XdB -30 dB
```

5.7 Meas Setup (Harmonic Distortion)

When the harmonic distortion measurement has been selected in the Measure key menu of the Spectrum Analysis Mode, this key displays the appropriate measurement setup menu.

Remote Command:

There is no equivalent remote command.

5.7.1 Avg Number

Press **Avg Number (On)** to specify the number of measurement averages used when calculating the measurement result. The average will be displayed at the end of each sweep.

State Saved: Saved in instrument state.

Factory Preset and *RST: 10 averages / Off

Minimum Value: 1

Maximum Value: 8192

Remote Command:

[:SENSe]:HARMonics:AVERage:COUNt <integer>

[:SENSe]:HARMonics:AVERage:COUNt?

[:SENSe]:HARMonics:AVERage[:STATe] OFF|ON|0|1

[:SENSe]:HARMonics:AVERage[:STATe]?

Example:

```
HARM:AVER:COUN 100
```

```
HARM:AVER ON
```

5.7.2 Harmonics

Harmonics indicates the number of harmonics to measure before computing the total harmonic distortion. The minimum number is 2 (only the fundamental and second harmonic will be measured). The maximum number is 10.

State Saved: Saved in instrument state.

Factory Preset and *RST: 10

Minimum Value: 2

Maximum Value: 10, limited by the maximum frequency range of the analyzer.

Remote Command:

```
[:SENSe]:HARMonics:NUMBer <integer>
[:SENSe]:HARMonics:NUMBer?
```

Example:

```
HARM:NUMB 5
```

5.7.3 ST/Harmonic

Sets the sweeptime used to measure each harmonic. The value is set to 200 divided by the resolution bandwidth, or 10 ms, whichever is greater when the measurement is started. This sweeptime is used only for measuring harmonics. The analyzer sweeptime before the measurement was started is used for finding the fundamental.

State Saved: Saved in instrument state.

Factory Preset and *RST: 10 ms / Auto

Default Terminator: seconds

Minimum Value: 10 ms

Maximum Value: maximum sweep time of the analyzer

Remote Command:

```
[:SENSe]:HARMonics:SWEeptime:AUTO OFF|ON|0|1
[:SENSe]:HARMonics:SWEeptime:AUTO?

[:SENSe]:HARMonics:SWEeptime <time>
[:SENSe]:HARMonics:SWEeptime?
```

Example:

```
HARM:SWE:AUTO OFF
HARM:SWE 100 ms
```

5.7.4 Counter Zoom

Turning this on ensures the correct operation of the internal frequency counter when the resolution bandwidth is relatively narrow compared to the starting span.

State Saved: Saved in instrument state.

Factory Preset and *RST: On

Remote Command:

There is no equivalent remote command.

5.8 Meas Setup (Occupied BW)

When occupied BW measurement has been selected in the Measure key menu of the Spectrum Analysis Mode, this key displays they appropriate measurement setup menu.

Remote Command:

There is no equivalent remote command.

5.8.1 Avg Number

Press **Avg Number (On)** to specify the number of measurement averages used when calculating the measurement result. The average will be displayed at the end of each sweep.

State Saved: Saved in instrument state.

Factory Preset and *RST: 10 averages / Off

Minimum Value: 1

Maximum Value: 8192

Remote Command:

`[:SENSe]:OBW:AVERage:COUNt <integer>`

`[:SENSe]:OBW:AVERage:COUNt?`

`[:SENSe]:OBW:AVERage[:STATe] OFF|ON|0|1`

`[:SENSe]:OBW:AVERage[:STATe]?`

Example:

```
OBW:AVER ON
```

5.8.2 OBW Span

Allows you to specify the range of integration used in calculating the total power from which the percent occupied bandwidth is then calculated. The analyzer span will be set to the same value as the OBW Span for the measurement. OBW Span should be set to approximately 2 times the expected occupied bandwidth result.

State Saved: Saved in instrument state.

Factory Preset and *RST: 3 MHz

Default Terminator: Hz

Minimum Value: 100 Hz

Maximum Value: 26.5 GHz

Remote Command:

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

`[:SENSe]:OBW:FREQuency:SPAN?`

Example:

```
OBW:FREQ:SPAN
```

5.8.3 Occ BW % Pwr

Allows you to change the percentage of signal power used when determining the occupied bandwidth.

State Saved:	Saved in instrument state.
Factory Preset and *RST:	99%

Minimum Value: 10%

Maximum Value: 99.9%

Remote Command:

`[:SENSe]:OBW:PERCent <percent>`

`[:SENSe]:OBW:PERCent?`

Example:

```
OBW:PERC 98
```

5.9 Restart

Restarts the current measurement, average, or sweep. Use this command to restart the current measurement from the “idle” state, regardless of its operating state.

Remote Command:

`:INITiate:RESTart`

Example:

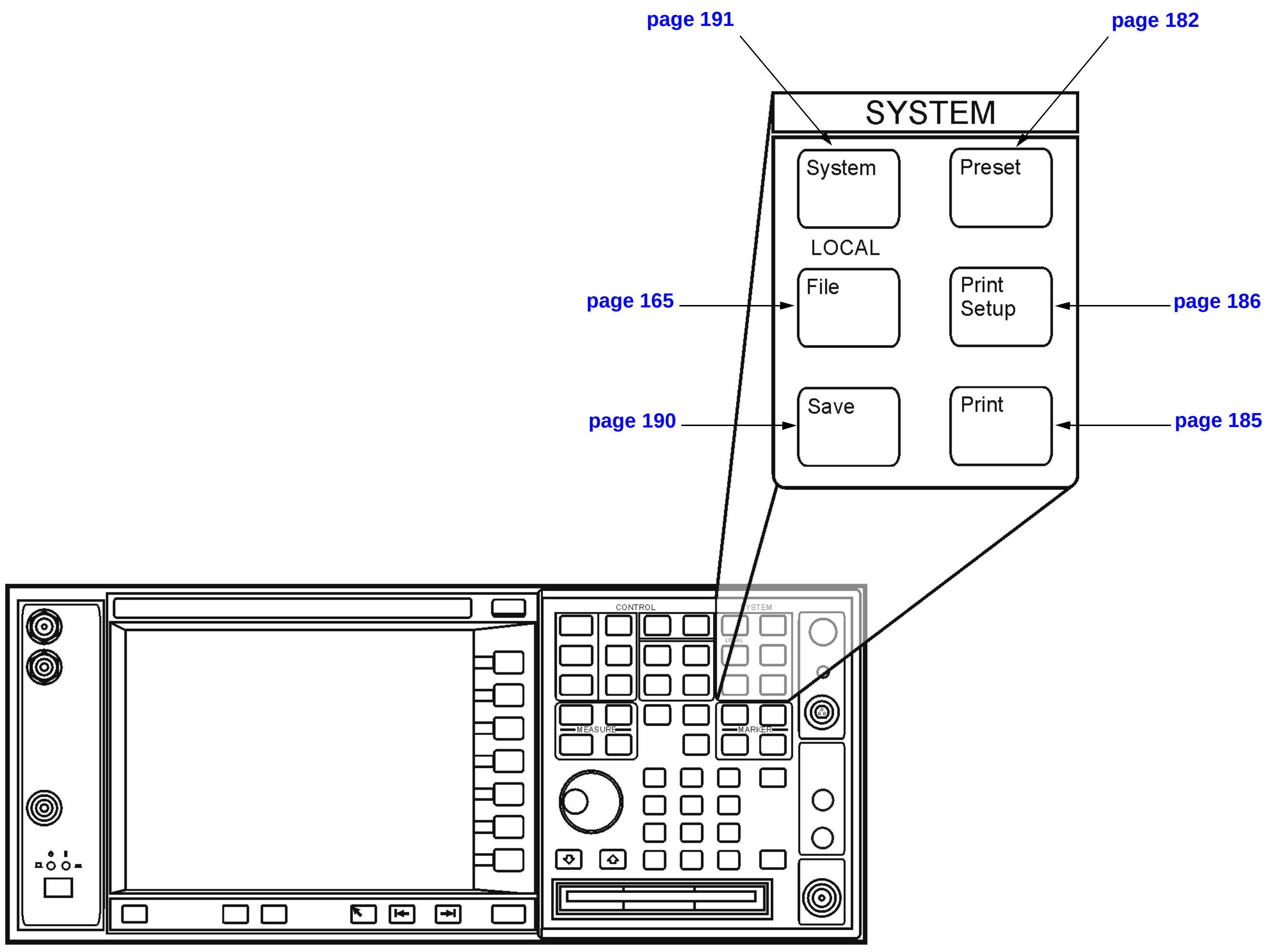
```
INIT:REST
```

This command is equivalent to sending an `:ABORt` command followed by an `:INITiate[:IMMediate]` command.

6 System Keys

This chapter provides key descriptions and programming information for the System functions of your analyzer. The front-panel System functions are listed alphabetically and are described with their associated menus keys. The lower-level menu keys are arranged and described as they appear in your analyzer menus with the corresponding front-panel key.

System Function Keys and Where to Find Them



ke843a

6.1 File

Accesses a menu of functions that enables you to load, save, and manage data on a floppy disk (A:) or the internal analyzer drive (C:); you can recall, save, copy, delete, or rename files of instrument states, trace data, and screen captures. The menu keys access dialog boxes appropriate for the selected function.

There are three types of files that you can save on the internal analyzer drive (C:) or a floppy disk (A:). They are described below.

- **State -** A file that contains a copy of the state of the analyzer at the time the file is saved. The settings of most analyzer functions are saved in the state files. When a **State** file is loaded into the analyzer, the analyzer is restored to the same state as when the file was saved. Some settings are not saved in the **State** files, for example the GPIB address; these settings are called "persistent". In this manual, each function describes whether that function is saved in "Instrument State" or is persistent.
- **Trace -** A file that contains a copy of the trace data for one or more traces. There are two formats for trace files, **Trace + State** and **CSV** files.

  Trace + State: A file that contains the trace data and a copy of the current analyzer state. The trace and state are stored in an internal data format. When a **Trace + State** file is loaded into the analyzer the trace data that was on the screen, when saved, is loaded into the analyzer. This enables you to view the trace as it looked when it was saved. Because the state data is also saved, the analyzer settings, including all the annotation on the screen, is restored as well. To preserve the trace data, the traces contained in the saved files are placed in **View** mode (see **Trace/View**, page 111) so that they are not immediately overwritten by new trace data. This means that you can save traces while making a measurement, and later load them back into the analyzer, and print them or transfer them to a computer for analysis.

  CSV: A file that contains trace data in comma-separated values format (CSV, standard PC spreadsheet format), to be read into a spreadsheet for analysis. Most spreadsheet programs support CSV format.   They cannot be loaded back into the analyzer.
- **Screen -** A file that contains an exact representation of the analyzer display at the time it was saved. You cannot extract data from **Screen** files as you can with **Trace** files, but you can print them or include them in other documents; **Screen** files look exactly as the display looked when the file was saved. They cannot be loaded into the analyzer. There are four formats for screen files, **Bitmap**, **Metafile**, **Reverse Bitmap**, and **Reverse Metafile**.

  Bitmap: A file that contains an exact bit representation of the screen. Stored in GIF format.

  Metafile: A file that contains information about the objects on the screen. Stored in WMF format, a format that can be read with Microsoft ™ Word and Microsoft ™ Excel, among others.

  Reverse Bitmap: Same as **Bitmap**, but the black display backgrounds are changed to white and the yellow traces are turned to green to preserve printer ink.

  Reverse Metafile: Same as **Metafile**, but the black display backgrounds are changed to white and the yellow traces are turned to green to preserve printer ink.

6.1.1 Catalog

Displays directories and files located on the selected drive, depending upon the preferences set under the Type (page 166) and Sort (page 167) keys. Catalog accesses menus to navigate the drives and to sort and select the files you wish to view.

NOTE The internal analyzer "drive" (C:\) is not an actual disk drive, but an area of nonvolatile (flash) memory which is presented as though it were a disk drive.

Remote Command:

```
:MMEMory:CATalog? <dir_name>
```

Example:

```
:MMEM:CAT? "C:\MYDIR\MYMEAS"
```

Query returns all files in the specified drive\path name.

The return data will be in the format: `<mem_used>,<mem_free> {,<file_listing>}`

Each `<file listing>` indicates the name and size in bytes of one file in the directory list in the form: "`<file_name>`,, `<file_size>`" for example, a file called "SCREN000.GIF" which is 21286 bytes in size, would list as `"SCREN000.GIF,,21286"`. Directories are indicated by square brackets, for example `"[MYDIR],,"`.

All files are listed, without regard to the preferences selected for the file catalog on the analyzer screen.

If you use lowercase characters, they are converted to uppercase in interpreting catalog commands.

6.1.1.1 Type

Selects the desired type of instrument-data files to be displayed. The catalog displays all files (if Type is set to All) or files of the currently selected file type. All directories are always displayed.

State Saved: Type is not saved in the instrument state

Factory Preset and *RST: Type survives **Factory Preset** and *RST, but is set to **State** at power on.

Remote Command:

There is no remote command for this key.

6.1.1.1.1 All

Displays all possible file types for display.

Displays all files located in the selected directory. If selected, it applies to **Catalog**, **Delete**, **Copy**, and **Rename**.

6.1.1.1.2 State Displays all state files (STA) in the selected directory. If selected, it applies to all File functions.

6.1.1.1.3 Trace

Displays all trace files (TRC and CSV) in the selected directory. If selected, it applies to all **File** functions.

6.1.1.1.4 Screen

Displays all screen (GIF and WMF) files in the selected directory.

6.1.1.2 Sort

Accesses the Sort menu keys that allow you to view your saved files according to a selected file attribute.The selections include, By Date, By Name, By Extension, By Size, and Order. Order (Up) sorts files in ascending order (for instance, A,B,C). Order (Down) sorts files in descending order (for instance, C,B,A).

The **Sort** setting applies to all of the **File** functions, except **Save**.

State Saved: The Sort order survives Preset, but is not saved in the instrument state.

Remote Command:

There is no remote command for this key.

6.1.1.2.1 By Date

Sorts and displays the current file catalog by the date of the files.

6.1.1.2.2 By Name

Sorts and displays the current file catalog in alphabetical order of the name of the files.

6.1.1.2.3 By Extension

Sorts and displays the current file catalog, in alphabetical order, by the file extension of the file names (for example, .TRC, .STA).

6.1.1.2.4 By Size

Sorts and displays the current file catalog by the size of the files.

6.1.1.2.5 Order

Changes the order of the display of the current file catalog. **Up** sorts the files in ascending order (a to z, 1 to 9), while **Down** sorts in descending order (z to a, 9 to 1).

6.1.1.3 Dir Up

Moves up one subdirectory level within a directory. If your position is in the top level of the drive already, it moves up to the drive level, wherein the current drive is highlighted (A: or C:).

6.1.1.4 Dir Select

Selects the drive or directory that the cursor is currently indicating highlighted by inverse video of all fields in that line.

If the top entry in the catalog has a ". ." indication, you are in a subdirectory. This entry type will be listed as **UP** and pressing **Dir Select** moves you up one directory level.

If your position is in the top level already, it moves up to the drive level.

6.1.2 Save

Accesses menu keys that allow you to save files to the floppy (A:) or internal (C:) drive.

CAUTION Never remove the floppy disk during a save operation. To do so could corrupt *all* data on the floppy disk.

NOTE Many errors can be generated by a bad **Save** operation. For this reason, if an `"Unable to Save file"` message is seen, you should check the error queue (**System**, **Show Errors**) for the source of the error.

NOTE You can press the front-panel **Save** key to immediately save a file using an automatically generated file name. The current **Save** parameters will be used, as though **Save Now** had been pressed.

Menus allow you to fill in data-entry fields for file name, type, format, source, and path (directory). Some fields may be blank depending on file type.

The catalog list box is active and can be used for selecting the directory in which to save the file. Saved files that match the current **Type** and **Format** are shown. The **Sort Order** is always **Down**, **By Date**.

NOTE If saving a **Screen**, the screen saved is that which was displayed before pressing **File**. For this reason, the screens seen while in the **File** menus cannot be saved.

6.1.2.1 Save Now

Executes the save function. When the save is complete, the message `XXXXXX file saved` (where XXXXXX is the filename) will appear in the status line on your display.

CAUTION Never remove the floppy disk during a save operation. To do so could corrupt *all* data on the floppy disk.

NOTE Many errors can be generated by a bad **Save** operation. For this reason, if an "Unable to Save file" message is seen, you should check the error queue (**System**, **Show Errors**) for the source of the error.

NOTE You are always safe pressing **Save Now** without entering a file name, because the auto-generated file name never conflicts with an existing file.

If the **`Path:`** field above the directory box is empty when pressing **Save Now**, the status line will display the error message: **`Unable to save file, invalid path`**. In this case, please select a drive.

While the file is being saved the popup message "`Saving file`" followed by "`Reading directory`" is displayed. After a successful save, the text message "`xxxxxx file saved`" (where xxxxxx is the file name) appears in the status line.

The analyzer will pick a filename for you based on the table below. ### represents a three-digit number which the analyzer has chosen to be the lowest number in the current sequence that does not conflict with an existing file name. (The number starts at `001` after a power cycle or **Restore Sys Defaults** and counts up with each attempted **Save**.) Or see **Name** (page 172) to enter your own file name.

Type	**Auto-Generated File Name**	**Extension**
State	STATE###	.STA
Trace	TRACE###	.TRC or .CSV
Screen	SCREN###	.GIF or .WMF

Remote Command:

`:MMEMory:STORe:SCReen <"file_name">`

`:MMEMory:STORe:STATe 1,<"file_name">`

`:MMEMory:STORe:TRACe <label>,<"file_name">`

Trace labels are: TRACE1|TRACE2|TRACE3|ALL
The file name must have a file extension of .trc or .csv. The file extension determines whether a trace is stored (.csv), or a trace with its state (.trc), are stored.

`*SAV <register#>`

Example:

The `MMEM:STOR:STAT` command saves the current instrument state to the specified file name. The *SAV command saves the current instrument state to a file name REGxxx, where `xxx` = the register number. The available register numbers are 0 to 127.

The `<"file_name">` must include the complete path, for example `"C:\MYTRACE.TRC"`. Lowercase characters are interpreted as uppercase.

These commands will fail if the `<"file_name">` already exists.

`:MMEM:STOR:STAT 1,"C:\mystate.sta"` The .sta extension is required.

`:MMEM:STOR:SCR "C:\myscreen.gif"` 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. Only **Bitmap** and **Metafile** are available (not **Reverse Bitmap** and **Reverse Metafile**).

`:MMEM:STOR:TRAC TRACE3,"C:\mytrace.trc"` Saves trace 3 to the trace + state file `C:\MYTRACE.TRC`

6.1.2.2 Type

Allows you to select the type of data you want to save.

See “Type” on page 166 for more information.

The file types available for saving are described below.

Type	Format	Source	Extension
State	State		STA
Trace	Trace + state	Trace 1, 2, 3, or all traces	TRC
	Comma separated trace values	Trace 1, 2, 3, or all traces	CSV
Screen	Bitmap		GIF
	Reverse bitmap		GIF
	Metafile		WMF
	Reverse metafile		WMF

NOTE **All** is not an option in **Save**, you have to specify the desired file type.

6.1.2.3 Format

When Type is set to Trace, Format allows you to choose between Trace + State and CSV formats. (See “File” on page 165.)

When **Type** is set to **Screen**, **Format** allows you to choose between **Bitmap**, **Metafile**, **Reverse Bitmap**, and **Reverse Metafile** formats. (See “File” on page 165.)

State Saved: **Format** is not saved in Instrument State.

Factory Preset and *RST: **Format** survives **Factory Preset** and *RST, but:

Trace file format is **Trace + State** at power on

Screen file format is **Bitmap** at power on

6.1.2.3.1 Trace + State

When the file type is **Trace**, this key selects the **Trace + State**, instrument-readable file format for your file.

6.1.2.3.2 CSV

When the file type is **Trace**, this key selects the trace data as comma-separated values (.CSV). The **CSV** format is readable by a spreadsheet on your computer (but the trace cannot be restored to the analyzer display).

6.1.2.3.3 Bitmap

When the file type is **Screen**, this key selects the bitmap file format (.GIF) for your saved data.

6.1.2.3.4 Metafile

When the file type is **Screen**, this key selects the metafile file format (.WMF) for your saved data.

6.1.2.3.5 Reverse Bitmap

When the file type is **Screen**, this key selects the inverse bitmap file format (.GIF) for your saved data.

6.1.2.3.6 Reverse Metafile

When the file type is **Screen**, this key selects the inverse metafile file format (.WMF) for your saved data.

6.1.2.4 Source

When the file type is **Trace**, this key allows you to save trace **1**, **2**, **3** or **All**. Saving trace **All** saves all traces in a single .TRC file.

For any other Save type, **Source** is disabled (grayed out).

State Saved:	**Source** is not saved in Instrument State.
Factory Preset and *RST:	**Source** survives **Factory Preset** and *RST, but is set to **All Traces** at power up.

6.1.2.4.1 Trace 1

Selects trace 1 to be saved.

6.1.2.4.2 Trace 2

Selects trace 2 to be saved.

6.1.2.4.3 Trace 3

Selects trace 3 to be saved.

6.1.2.4.4 All Traces

Selects all the traces to be saved.

6.1.2.5 Name

Accesses the Alpha Editor and allows you to enter a filename. The numeric keypad may also be used while entering file names. Press Enter or Return to complete the name entry.

NOTE Only capital letters (A-Z) and digits (0-9) may appear in file names (8 characters, maximum). Additionally, file names include a 3 digit extension which is automatically set by the instrument depending on the file type and format.

Remote Command:

The file name is entered as part of the directory/path name that is sent with the SCPI command. See “Save Now” on page 168.

6.1.2.6 Dir Up

Allows you to move up one directory level. If at the top level, Dir Up moves to the drive level, displaying the available disk drives. See “Dir Up” on page 168 for more information.

6.1.2.7 Dir Select

Accesses the highlighted directory on your display.

See “Dir Select” on page 168 for more information.

6.1.3 Load

Accesses the functions that load instrument-data files from your selected drive and directory back into the instrument. This function displays the file list box which shows the data-entry fields for the file name, type, destination, and path.

The catalog list box is active and can be used for selecting the file information in the data-entry fields. Only loadable files that match the current type are shown. Placing the cursor on a file name causes it to be loaded into the file name field.

6.1.3.1 Load Now

Loads the currently selected file. When the load is complete, the message `XXXXXX file loaded` (where XXXXXX is the filename) will appear in the status line on your display.

Displayed settings include name, type, destination, and path. While the file is being loaded a popup message is displayed “`Loading file`”. After a successful load, the text message “`xxxxxx file loaded`” (where xxxxxx is the file name) appears in the status line. When traces are loaded they always load in **View** mode.

Traces save in .TRC format can be loaded individually or as a group. When a trace is loaded, the state that existed when that trace was saved is loaded along with the trace. Also, the loaded trace(s) is/are placed in view mode.

NOTE If you wish to compare two saved traces from different saves, place traces in view mode before saving them. This prevents the trace from being rewritten based on a state change from subsequent loads.

Remote Command:

`:MMEMory:LOAD:STATe 1,<"file_name">`

The `MMEM:LOAD:STATe` command loads the specified state file into the current active state of the instrument.

`*RCL <register#>`

The `*RCL` command loads the state from the specified internal register into the current active state of the instrument. The available register numbers are 0 to 127.

`:MMEMory:LOAD:TRACe <label>,<"file_name">`

NOTE The <"file_name"> must include the complete path. Lowercase characters are read as uppercase.

- Trace labels are: `TRACE1|TRACE2|TRACE3|ALL`
- The file name must have a file extension of .trc.

Example:

`:MMEM:LOAD:STAT 1,"C:mystate.sta"` Loads the state file `C:\MYSTATE.STA`.

NOTE –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 the same as the revision of the instrument, all settings 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 settings of the state.

`:MMEM:LOAD:TRAC TRACE3,"C:mytrace.trc"`Loads the trace in file `C:\MYTRACE.TRC` into trace 3.

6.1.3.2 Type

Allows you to select the type of file you want to load. See “Type” on page 166 for more information.

The file types available for loading are described below.

Type	**Destination**	**Extension**
State		STA
Trace	Trace 1, 2, 3, or all traces	TRC

NOTE **All** is not an option in **Load**, you have to specify the desired file type.

6.1.3.3 Sort

Allows you to view your saved files according to a selected file attribute. See “Sort” on page 167 for more information.

6.1.3.4 Destination

When Type is set to Trace, Destination allows you to direct your data to Trace 1, Trace 2, or Trace 3 for a single-trace file. If the data is for all three traces (Source was All when they were saved), the data will be returned to the original trace registers, regardless of the Destination setting.

State Saved: Not saved in Instrument State.

Factory Preset and *RST: Trace file format, is All Traces at power on.

6.1.3.4.1 Trace 1

Selects trace 1 for the trace data to be loaded into.

6.1.3.4.2 Trace 2

Selects trace 2 for the trace data to be loaded into.

6.1.3.4.3 Trace 3

Selects trace 3 for the trace data to be loaded into.

6.1.3.5 Dir Up

Allows you to move up one directory level. If at the top level, Dir Up moves to the drive level, displaying the available disk drives. See “Dir Up” on page 168 for more information.

6.1.3.6 Dir Select

Accesses the highlighted directory on your display.

See “Dir Select” on page 168 for more information.

6.1.4 Delete

Accesses functions which delete instrument data files from your selected directory.

The catalog list box is active and can be used for selecting file information for the data-entry fields. Only files that match the current type are shown. Placing the cursor on a file name causes it to be loaded into the file name field.

6.1.4.1 Delete Now

Executes the delete function. If a directory is selected to be deleted, the message `WARNING: You are about to delete the contents of directory XXXXXX` (where XXXXXX is the full path and directory name) will appear on your display.

Press **Delete Now** again to perform the delete. After a successful delete, the message `XXXXXX file deleted` (where XXXXXX is the filename) will appear in the status line on your display.

While the file is being deleted, the popup message “`Deleting file`” followed by “`Reading directory`” are displayed. After a successful deletion, the text message “`xxxxxx file deleted`” (where xxxxxx is the file name) appears in the status line.

To quickly delete all of the file in a directory, select the file at the top of the list and press **Delete Now** repeatedly until all the files are deleted.

If the subdirectory that you want to delete is not empty, the following popup message is displayed “`WARNING: You are about to delete all of the contents of directory xxxxxx. Press Delete Now again to proceed or any other key to abort.`” (xxxxxx is the full path and directory name).

If <`"file_name"`> does not exist, a “`File Name Error`” occurs.

Remote Command:

`:MMEMory:DELete <"file_name">` to delete a file.

`:MMEMory:RDIRectory <"directory_name">` to delete a directory.

<`"file_name"`> and <`"directory_name"`> must include the complete path. Lowercase characters are read as uppercase.

Examples:

`:MMEM:DEL "C:\destinat.trc"` removes the file `C:\DESTINAT.TRC`.

`:MMEM:RDIR “C:\myDir”`

Removes directory `C:\MYDIR` and all files and subdirectories within that directory.

6.1.4.2 Type

Allows you to select the type of file you want to delete. See “Type” on page 166 for more information.

6.1.4.3 Sort

Allows you to view your saved files according to a selected file attribute. See “Sort” on page 167 for more information.

6.1.4.4 Dir Up

Allows you to move up one directory level. If at the top level, Dir Up moves to the drive level, displaying the available disk drives. See “Dir Up” on page 168 for more information.

6.1.4.5 Dir Select

Accesses the highlighted directory on your display. See “Dir Select” on page 168 for more information.

6.1.5 Copy

Accesses the functions to copy instrument data files in your selected directory to the directory and file name that you choose. This key also displays a catalog of the files that are currently saved in the selected directory and data-entry fields for the following: file name, type, and path location.

6.1.5.1 Copy Now

Accesses the functions to copy data files from one directory to another on one or more mass storage devices, using the currently displayed file settings.

If a copy is being done for a file that already exists in the “To” directory, the text message “`File already exists`” appears in the status line.

While the file is being copied, the “`Copying file`” followed by “`Reading directory`” popup message is displayed. After a successful copy, the green text message “`xxxxxx file copied`” (where xxxxxx is the file name) appears in the status line.

Remote Command:

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

The file names must include the complete file paths. Lowercase characters are read as uppercase.

Example:

`:MMEM:COPY “C:\oldname.sta”,”A:\newname.sta”` copies `C:\OLDNAME.STA` to `A:\NEWNAME.STA`.

6.1.5.2 Type

Allows you to select the type of file you want to copy. See “Type” on page 166 for more information.

6.1.5.3 Sort

Allows you to view your saved files according to a selected file attribute. See “Sort” on page 167 for more information.

6.1.5.4 Dir From/To

Allows you to select the source and destination directories for your copy on one or more drives.

Toggles between the two displayed directory list windows. It allows you to define the "From" and "To" locations for copying.

State Saved: Powers up with `C:\` as both the "From" and "To" drives. Not save in state. Survives **Factory Preset**.

6.1.5.5 Dir Up

Allows you to move up one directory level. If at the top level, Dir Up moves to the drive level, displaying the available disk drives. See “Dir Up” on page 168 for more information.

6.1.5.6 Dir Select

Accesses the highlighted directory on your display. See “Dir Select” on page 168 for more information.

6.1.6 Rename

Allows you to rename a file.

The catalog list box is active and can be used for selecting both the path and a file name. Only loadable files that match the current type are shown. Placing the cursor on a file name causes it to be loaded into the file name field.

6.1.6.1 Rename Now

Executes the rename function. When the rename is complete, the message XXXXXX `file renamed to` YYYYYY (where XXXXXX and YYYYYY are the filenames) will appear in the status line on your display.

Placing the cursor on a file name causes it to be loaded into the file name field.

If you try to rename a file with a name that already exists, the text message (`File already exists`) appears in the status line.

Remote Command:

`:MMEMory:MOVE <"file_name1">,<"file_name2">`

`<"file_name1">` must include the complete path, and the case *must* match that of the file to be renamed. `<"file_name2">` must contain the complete path of the destination, and the case of any directories in the path *must* match those of the directories in the destination path. The case of the destination file name is always interpreted as uppercase.

You can use this command to move files between directories and drives, even though there is no way to do this from the front panel.

Example:

```
:MMEM:MOVE "C:\STATE001.STA","C:\FREQ.STA"
```

6.1.6.2 Type

Allows you to select the type of file you want to rename. See "Type" on page 166 for more information.

6.1.6.3 Sort

Allows you to view your saved files according to a selected file attribute. See "Sort" on page 167 for more information.

6.1.6.4 Name

Accesses the Alpha Editor and allows you to enter the file name you want to rename the file to. The numeric keypad can also be used to enter a filename while the alpha editor is accessed. Complete your entry by pressing Return or Enter. See "Name" on page 172 for more information.

NOTE Only capital letters (A-Z) and digits (0-9) may appear in file names (8 characters, maximum). Additionally, file names include a 3 digit extension which is automatically set by the instrument.

6.1.6.5 Dir Up

Allows you to move up one directory level. If at the top level, Dir Up moves to the drive level, displaying the available disk drives. See "Dir Up" on page 168 for more information.

6.1.6.6 Dir Select

Accesses the highlighted directory on your display. See "Dir Select" on page 168 for more information.

6.1.7 Create Dir

Accesses the functions to create a new subdirectory in the currently selected directory.

6.1.7.1 Create Dir Now

Accesses the functions to create a new directory. While the directory is being created a popup message is displayed “`Creating directory`” followed by “`Reading directory`”. After the successful creation of a directory, the text message “`Directory xxxxxx created`” (where xxxxxx is the new directory name) appears in the status line.

If the creation of a new directory is being performed for a directory name that already exists, the text message “Directory already exists” appears in the status line.

Remote Command:

`:MMEMory:MDIRectory <"dir_name">`

`<"dir_name">` must contain the complete path for the new directory. Lowercase characters are interpreted as uppercase.

Example:

```
:MMEM:MDIR "C:\myDir"
```

Creates directory `MYDIR` on the `C:\` drive.

6.1.7.2 Name

Accesses the Alpha Editor and allows you to enter a filename. The numeric keypad can also be used to enter a filename while the alpha editor is accessed. Complete your entry by pressing Return or Enter.

NOTE Only capital letters (A-Z) and digits (0-9) may appear in file names (8 characters, maximum). Additionally, file names include a 3 digit extension which is automatically set by the instrument.

6.1.7.3 Dir Up

Allows you to move up one directory level. If at the top level, Dir Up moves to the drive level, displaying the available disk drives. See “Dir Up” on page 168 for more information.

6.1.7.4 Dir Select

Accesses the highlighted directory on your display. See “Dir Select” on page 168 for more information.

6.1.8 Delete All

Deletes all the files on a floppy disk; any information on the disk will be destroyed.

Remote Command:

There is no remote command for this key.

6.1.8.1 Delete All Now

Executes the **Delete All** function. After pressing **Delete All**, the following message will appear on the display: `WARNING: You are about to destroy ALL data on volume A: Press Delete All again to proceed or any other key to abort.`

While deleting, a popup message is displayed "`Deleting All`." After a successful floppy disk file deletion, the green text message "`Volume A: delete complete`", appears in the status line.

Remote Command:

There is no remote command for this key.

6.1.9 Move Data to a File (Remote Command Only)

This command loads a block of data in the format <definite_length_block> into the instrument memory location <"`file_name`">.

The query form of the command returns the contents of the file identified by <"`file_name`">, in the format of a definite length block of data. The query can be used for copying files out of the analyzer over the remote bus.

A definite length block of data starts with an ASCII header that begins with # and indicates how many additional data points are following in the block. Suppose the header is #512320.

- The first digit in the header (5) tells you how many additional digits/bytes there are in the header.
- The 12320 means 12,320 data bytes follow the header.
- Divide this number of bytes by your selected data format bytes/point, either 8 (for real 64), or 4 (for real 32). In this example, if you are using real 64 then there are 1540 points in the block.

Remote Command:

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

`:MMEMory:DATA? <"file_name">`

Example:

`:MMEM:DATA "C:\DEST.TXT","#14abcd"` Loads the data "abcd" into `C:\DEST.TXT`.

`:MMEM:DATA? "SCREN001.GIF` Initiates a transfer of data from file `C:\SCREN001.GIF`.

6.1.10 Set Data Byte Order (Remote Command Only)

Controls whether binary data is transferred in normal or swapped mode. In normal mode the most significant byte is sent first (in a 1,2,3,4 sequence). In swapped mode the least significant byte is sent first with the sequence reversed.

State Saved: Survives **Preset** but not power cycle. Not saved in Instrument State.

Factory Preset and *RST: Powers up in Normal.

Remote Command:

`:FORMat:BORDer NORMal|SWAPped`

`:FORMat:BORDer?`

Example:

```
:FORM:BORD NORM
```

6.1.11 Set Numeric Data File Format (Remote Command Only)

This command changes the format of the data output. It specifies the format used for trace data during data transfer across any remote port. REAL and ASCII formats will format trace data in the current amplitude units. The format of state data cannot be changed. It is always in a machine readable format only.

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.

ASCII - Amplitude values are in amplitude units separated by commas. ASCII format requires more memory than the binary formats. Handling large amounts of this type of data, takes more time and storage space.

Integer,16 - Binary 16-bit integer values in internal units (dBm), in a definite length block.

Real,32 (or 64) - Binary 32-bit, or 64-bit, real values in amplitude units, in a finite length block. Transfers of real data are done in a binary block format.

State Saved: Not saved in Instrument State.

Factory Preset and *RST: Survives Preset but not power cycle. Powers up Real, 32

Remote Command:

`:FORMat[:TRACe][:DATA] ASCii|INTeger,16|REAL,32|REAL,64`

`:FORMat[:TRACe][:DATA]?`

Example:

```
:FORM REAL,32
```

6.2 Preset

Provides a convenient starting point for making most measurements.

Depends on the preset (user vs. factory) setting in the System keys. If the preset type is set to **Factory**, pressing **Preset** results in an immediate instrument preset to the factory defaults. If it is set to **User**, pressing **Preset** accesses a menu that allows you choose your preset settings from either the factory default values or the settings you have previously defined as the **User** preset state.

SCPI Status Bits/ OPC Dependencies: Clears all pending OPC bits. The status byte is set to 0.

Factory Preset and *RST: The following table describes the conditions, of the analyzer, established by pressing **Preset** when the preset type is set to **Factory**. Coupled functions are indicated by an asterisk (*). All coupled functions are in the auto-coupled mode after a **Preset** is performed.

ADC Dither	Off *
Annotation and graticule	On
Attenuation	10 dB *
Atten Step	2 dB
Average	Off
Average number	100
Avg/VBW Type	Log Pwr *
Auto Swp Time	Normal *
Auto Swp Typ	DynRng *
Center frequency	13.255 GHz
CF Step	2.649 GHz
Coupled functions	All set to AUTO
Detector	Normal *
Display line	Off
Display line level	–25 dBm
Ext Amp Gain	0 dB
FFT/Span (Goal)	1 *
Frequency offset	0 Hz
Freq Ref	Int
Freq Ref	10 MHz
Gate Time	100 ms *

Input Port	RF
Log scale	10 dB/division
Marker Count	On
Markers	Off
Max Mixer Lvl	–10 dBm
PhNoise Opt	Fast Tune *
Pk Threshold	–90 dBm
Presel Adjust	0 Hz
Ref level	0 dBm
Reference Lvl Offset	0 dB
Resolution BW	3 MHz *
RF Coupling	AC
Scale Type	Log
Signal Track	Off
Span	26.49 GHz
Span/RBW Ratio	106 *
SRQ mask	40
Start Freq	10 MHz
Stop Freq	26.49 GHz
State registers	unaffected
Sweep	Cont
Sweep Time	66.24 ms *
Swp Type	Swp *
Title	Cleared
Trace 1	Clear-write
Trace 2	Blank, not cleared
Trace 3	Blank, not cleared
Trigger	Free run
VBW/RBW	1.00000 *
Video BW	3 MHz *
Y Axis Units	dBm

Remote Command:

```
:SYStem:PRESet
```

The :SYSTem:PRESet command sets instrument parameters to values dependent on the preset type (**Factory**, **User**).

If :SYSTem:PRESet:TYPE is set to USER, then a user preset will be performed. This presets the analyzer to a state saved by the user. See "Power On/Preset" on page 192. If preset type is FACTORY then preset does an immediate factory preset.

***RST**

The *RST command performs a **Factory Preset**.

Example:

```
:SYST:PRES
```

6.2.1 User Preset

Restores the analyzer to a user defined state. The state was defined from the System menu when the Power On/Preset function was selected and Save User Preset was pressed. If the you did not save a user state, then the current power-up state is stored as the user preset file for use when Preset is pressed.

Example:

:SYST:PRES:TYPE USER selects the user-preset mode.

6.2.2 Factory Preset

A full factory preset is executed so the instrument is returned to the factory default state. The preset type can be set to **Factory** from the **Power On/Preset** function in the System menu.

Examples:

:SYST:PRES:TYPE FACT selects factory as the type of preset.

6.3 Print

Initiates an output of the display data to the currently defined printer. The screen remains frozen (no further sweeps are taken) until the data transfer to the printer is complete. Refer to the Print Setup key description in this chapter for more information about the printer functions.

There must be a valid printer set up for the print function to work. The *Getting Started Guide* includes additional printer installation information.

If you need to abort a print in progress, use the **ESC** (escape) key.

Remote Command:

`:HCOPy[:IMMediate]`

Example:

```
:HCOPY
```

6.3.1 Abort the Printout (Remote Command Only)

This command aborts the print that is currently in process.

Remote Command:

`:HCOPy:ABORt`

Example:

```
:HCOP:ABOR
```

6.4 Print Setup

Accesses the functions which specify a particular printer and control its output.

Remote Command:

There is no remote command for this key.

6.4.1 Printer Setup

Allows you to define your printer by selecting its printer language and color capability.

Supported printers are equipped with a parallel interface. (A supported printer is one that accepts Printer Control Language Level 3 or 5). Your printer language can be found in its documentation or in the specifications found on the manufacturer's web page.

- PCL3 printers include most HP DeskJet printers.
- PCL5 printers include most HP LaserJet printers.

The table below lists some current Hewlett-Packard ™ printers and their settings.

Printer Models	**Language Type**	**Color Capable**
HP DeskJet 310	PCL3	yes
HP DeskJet 320	PCL3	yes
HP DeskJet 400	PCL3	yes
HP DeskJet 670C, 672C, 680C, 682C	PCL3	yes
HP DeskJet 720C, 722C	Windows only (not compatible)	
HP DeskJet 600C, 660C, 670C, 680C, 690C	PCL3	yes
HP DeskJet 820C	Windows only (not compatible)	
HP DeskJet 840C, 850C, 870C, 890C, 895C	PCL3	yes
HP DeskJet 935C, 970C	PCL3	yes
HP DeskJet 1120C	PCL3	yes
HP LaserJet 4L, 4P	PCL5	no
HP LaserJet 5, 5L, 5M, 5P, 5MP, 5N	PCL5	no
HP LaserJet 6, 6L, 6M, 6P, 6MP	PCL5	no
HP Professional Series 2500CM	PCL3	yes
HP DesignJet 755CM	PCL5	yes

Remote Command:

There is no remote command for this key.

6.4.1.1 Language

Lets you define your printer language as a PCL3 (Deskjet) or PCL5 (Laserjet) printer.

State Saved: Persistent, survives **Preset** and power cycle, but not saved in Instrument State.

Factory Default: PCL3

Remote Command:

`:HCOPy:DEVice:LANGuage PCL3|PCL5`

`:HCOPy:DEVice:LANGuage?`

Example:

```
:HCOP:DEV:LANG PCL5
```

6.4.1.2 Color Capable

Allows you to define whether you printer is color capable (Yes) or not (No).

NOTE **Color Capable** does not specify whether you want your printout in color. See “Color” on page 188 for information.

State Saved: Persistent, survives Preset and power cycle, but not saved in Instrument State.

Remote Command:

`:HCOPy:DEVice:COLor NO|YES`

`:HCOPy:DEVice:COLor?`

Example:

```
:HCOP:DEV:COL YES
```

6.4.2 Orientation

Allows you to select either Portrait or Landscape printing. Landscape is not available with a PCL3 (Deskjet) printer.

State Saved: Persistent, survives **Preset** and power cycle, but not saved in Instrument State.

Factory Preset and *RST: Portrait

Remote Command:

```
:HCOPy:PAGE:ORIentation LANDscape|PORTrait
:HCOPy:PAGE:ORIentation?
```

Example:

```
:HCOP:PAGE:ORI LAND
```

6.4.2.1 Portrait

Selects Portrait orientation for the printouts from the analyzer.

Remote Command:

See “Orientation” on page 187.

Example:

```
:HCOP:PAGE:ORI PORT
```

6.4.2.2 Landscape

Selects Landscape orientation for the printouts from the analyzer.

Remote Command:

See “Orientation” on page 187.

Example:

```
:HCOP:PAGE:ORI LAND
```

6.4.3 Color

Allows you to select between color or black and white printing on color-capable printers. This key is inactive (grayed out) if Color Capable is set to No, see page 187.

State Saved: Persistent, survives **Preset** and power cycle, but not saved in Instrument State.

Factory Preset and *RST: Off

Remote Command:

```
:HCOPy:IMAGe:COLor[:STATe] OFF|ON|0|1
:HCOPy:IMAGe:COLor[:STATe]?
```

Example:

```
:HCOP:IMAG:COL ON
```

6.4.4 Prints/Page

Selects the number of display prints per page when orientation is set to Portrait. The page will be ejected after the selected number of prints has been printed.

NOTE For Landscape printing, **Prints/Page** is always set to 1.

State Saved: Persistent, survives **Preset** and power cycle, but not saved in Instrument State.

Factory Preset and *RST: 1

Remote Command:

```
:HCOPy:PAGE:PRINts <integer>
:HCOPy:PAGE:PRINts?
```

Example:

```
:HCOP:PAGE:PRIN 2
```

6.4.5 Eject Page

Ejects your printed page.

Remote Command:

```
:HCOPy:ITEM:FFEed[:IMMediate]
```

Example:

```
:HCOP:ITEM:FFE
```

Ejects the page if prints per page is set to 2 and only 1 print has completed. Otherwise the page automatically ejects after the print is complete.

6.5 Save

Saves analyzer states, traces, and screen data to a floppy (A:) drive or internal flash memory (C:) drive, as configured by the **File** menu. For example, if you have configured the instrument to save a trace to the C: drive, every time you press **Save**, it will save the current trace to a file with a new default trace file name.

You must

first configure the save file **Type**, **Format**, **Source**, and **Destination** by using **File**, **Save** before pressing the front-panel **Save** key. Pressing the front-panel **Save** key will then be the same as pressing **File**, **Save**, **Save Now**.

6.6 System

Accesses the System menu keys to control overall System functions. This is also the GPIB "LOCAL" key. Pressing **System** after the analyzer has been placed in the remote GPIB mode returns it to the local mode and enables front-panel control. During GPIB operation, "R" appears in the upper-right corner of the display indicating the instrument is in remote mode. A "T", "L" or "S" may appear during remote operation, indicating talk, listen, or service request.

Remote Command:

There is no remote command for this key.

6.6.1 Show Errors

Enables a display of the error messages that have been reported to the front-panel error history queue, including the first time the message occurred, the last time, how many times, and the message. The most recent error will appear at the top of the list. Once an error is detected, it remains in the queue until cleared, even if the error no longer exists.

NOTE A continuous recurring error reappears in the queue even if it had been cleared.

Remote Command:

`:SYSTem:ERRor[:NEXT]?`

`*CLS`

Example:

The `SYSTem` command queries the earliest entry to the error queue and then deletes that entry.

`*CLS` clears the entire error queue.

`:SYST:ERR? returns <error number>,<"error string">`, for example `-113,"Undefined header"`.

6.6.1.1 Previous Page

Displays the previous page in the front-panel error history queue. This key is inactive (grayed out) if there is no previous page.

6.6.1.2 Next Page

Displays the next page in the front-panel error history queue. This key is inactive (grayed out) if there is no next page.

6.6.1.3 Clear Error Queue

Clears the front-panel error queue.

6.6.2 Power On/Preset

Accesses keys to allow you to define the power on state and a user preset state for the instrument.

6.6.2.1 Power On

Determines the state of the analyzer when it is powered on. If Power On is set to Preset, the power on state of the instrument is the same as it is after Preset is pressed. This may either be a factory default state or a user defined state. Use the Power On/Preset menu function to change the setting of the analyzer state which is recalled at power on.

If **Power On** is set to **Last**, then the instrument always returns to the state that it was in when it was powered off. The setting (**Last** or **Preset**) of **Power On** is not changed by pressing **Preset**.

State Saved: Survives Preset and power cycle, but not saved in Instrument State.

Remote Command:

`:SYSTem:PON:TYPE PRESet|LAST`

`:SYSTem:PON:TYPE?`

Example:

`:SYST:PON:TYPE LAST` defines the power on type as the last state the analyzer was in before power down.

6.6.2.2 Preset

Allows you to select what instrument state Preset will use when pressed. Choose between Factory defined preset or a User defined preset. Pressing Preset, with Factory preset type selected, presets the analyzer to the configuration originally set at the factory. Pressing Preset, with User preset type selected, sets the analyzer to the settings defined by Save User Preset.

Remote Command:

`:SYSTem:PRESet:TYPE FACTory|USER`

`:SYSTem:PRESet:TYPE?`

Example:

`:SYST:PRESET`

`:SYST:PRES:TYPE USER` defines the type of preset as the user preset.

`:SYST:PRES[:USER]:SAVE` saves the current state to be used as the preset user state.

6.6.2.3 Save User Preset

Saves the current state of the analyzer into the User Preset state for recall upon Preset. After you save a state here, you must go to the Preset (Factory User) key and select User in order to have this state used as the preset state.

Remote Command:

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

Example:

```
:SYST:PRES:SAVE
```

6.6.3 Time/Date

Accesses the Time/Date function menu keys used to set and display the real-time clock.

Remote Command:

There is no remote command for this key.

6.6.3.1 Time/Date

Turns the display of the real-time clock on or off.

State Saved: Survives **Preset** and power cycle, but not saved in Instrument State.

Factory Default: On (Restored by **System**, **Restore Sys Defaults**.)

Remote Command:

```
:DISPlay:ANNotation:CLOCk[:STATe] ON|OFF
:DISPlay:ANNotation:CLOCk[:STATe]?
```

Example:

```
DISP:ANN:CLOC ON
```

6.6.3.2 Date Format

Allows you to choose the display of the date from a month-day-year format to a day-month-year format. It is set to a month-day-year format when the instrument System Defaults are restored. Only affects display of date at the top of the screen, not in the file catalog.

State Saved: Survives **Preset** and power cycle, but not saved in Instrument State.

Factory Default: MDY (Restored by **System**, **Restore Sys Defaults**.)

Remote Command:

```
:DISPlay:ANNotation:CLOCk:DATE:FORMat MDY|DMY
:DISPlay:ANNotation:CLOCk:DATE:FORMat?
```

Example:

`:DISP:ANN:CLOC:DATE:FORM DMY`

6.6.3.3 Set Time

Allows you to set the time of the real-time clock. Enter the time in 24 hour HHMMSS format.

State Saved: Survives **Preset** and power cycle, but not saved in Instrument State.

Default Terminator: none

Maximum Value:

Hour (HH) maximum value is 23.

Minute (MM) maximum value is 59.

Second (SS) maximum value is 59.

Minimum Value:

Hour (HH) minimum value is 00.

Minute (MM) minimum value is 00.

Second (SS) minimum value is 00.

Remote Command:

`:SYSTem:TIME <hour>,<minute>,<second>`

`:SYSTem:TIME?`

Example:

`:SYST:TIME 12,42,00` Sets the clock to 12:42:00.

6.6.3.4 Adjust Time Setting (Remote Command Only)

Adjust the instruments internal time by the value entered.

Terminators: No units are allowed with the command.

Default Terminator seconds

Remote Command:

`:SYSTem:TIME:ADJust <seconds>`

Example:

`SYST:TIME:ADJ 3600` will advance the time one hour.

`SYST:TIME:ADJ -86400` will back the date up one day, without changing the time of day (minutes or seconds).

6.6.3.5 Set Date

Allows you to set the date of the real-time clock. Enter the date in the YYYYMMDD format.

State Saved: Survives **Preset** and power cycle, but not saved in Instrument State nor restored by **System**, **Restore Sys Defaults**.

Maximum Value: Year (YYYY) maximum value is 9999

Month (MM) maximum value is 12

Day maximum value is 31 (depending on the month)

Minimum Value: Year (YYYY) minimum value is 0000

Month (MM) minimum value is 01

Day minimum value is 01.

Remote Command:

`:SYSTem:DATE <year>,<month>,<day>`

`:SYSTem:DATE?`

Example:

`:SYST:DATE 2000,12,24` Sets the date to December 24, 2000

6.6.4 Alignments

Accesses functions that control the automatic alignment of the instrument and load default values for the alignment system.

6.6.4.1 Auto Align

Allows you to turn the instrument automatic alignment On or Off. or select Alert to be alerted that alignments are needed.

- **Off**, the instrument won’t initiate any alignments or alerts.
- **Alert**, a 3 degree (Celsius) temperature change or a time span of 24 hours since the last alignment will trigger an alert that alignments need to be done, but no alignments will be performed without user input.
- **On**, the instrument behaves like the **Alert**, but will automatically perform a full alignment when it is needed. The instrument will stop any measurement currently in process, perform the full alignment, then restart the measurement from the beginning (similar to pressing **Restart**). Also see “Align All Now” on page 196.

State Saved: Survives **Preset** and power cycle, but not saved in Instrument State.

Factory Default: On (Restored by System, Restore Sys Defaults.)

Remote Command:

`:CALibration:AUTO OFF|ON|ALERt`

`:CALibration:AUTO?`

Example:

```
:CAL:AUTO ON
```

6.6.4.2 Align All Now

Immediately executes an alignment cycle of all the subsystems (Align RF, Align IF, Align ADC, and Align Current Sysgain). The instrument will stop any measurement currently underway, perform the full alignment, then restart the measurement from the beginning (similar to pressing the Restart key). All other operations are stopped and the alignments will be visible on the display.

NOTE Connect a cable between the rear-panel connector `AMPTD REF OUT` to the front-panel `INPUT` connector before executing. The alignment will fail if the cable is not connected.

Remote Command:

`:CALibration[:ALL]` Performs a full alignment.

The following three commands perform a full alignment and return a number indicating the success of the alignment. A zero is returned if the alignment is successful. A one is returned if any part of the alignment fails.

`:CALibration[:ALL]?`

`*CAL?`

`*TST?`

Example:

```
:CAL?
```

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.

6.6.4.3 Align Subsys

Accesses keys that enable you to activate a partial alignment.

6.6.4.3.1 Align RF Activates an alignment on the RF circuitry.

Remote Command:

`:CALibration:RF`

`:CALibration:RF?`

Example:

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

```
:CAL:RF?
```

6.6.4.3.2 Align IF Activates an alignment on the IF circuitry.

Remote Command:

`:CALibration:IF`

`:CALibration:IF?`

Example:

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

```
:CAL:IF?
```

6.6.4.3.3 Align ADC Activates an alignment on the ADC circuitry.

Remote Command:

`:CALibration:ADC`

`:CALibration:ADC?`

Example:

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

```
:CAL:ADC
```

6.6.4.3.4 Align Current IF Flatness Activates an alignment of the current IF flatness.

Remote Command:

`:CALibration:FLATness:IF`

`:CALibration:FLATness:IF?`

Example:

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

```
:CAL:FLAT:IF
```

6.6.4.3.5 Align Current SysGain Activates a fine-tuning adjustment of the system gain.

Remote Command:

`:CALibration:GAIN:CSYStem`

`:CALibration:GAIN:CSYStem?`

Example:

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

```
:CAL:GAIN:CSYS
```

6.6.4.4 Restore Align Defaults

Loads the default values for the alignment system, turns on the frequency corrections, and resets the timebase to the factory values. Align All Now must be executed 3 times after pressing Restore Align Defaults to meet specifications.

Remote Command:

`:CALibration:DATA:DEFault`

Example:

```
:CAL:DATA:DEF
```

6.6.5 Config I/O

Accesses the keys and menus that enable you to identify and change the current GPIB address and LAN settings.

6.6.5.1 GPIB Address

Shows the current GPIB address and allows you to change this value using the numeric keyboard. The new value is displayed in the active function area. The knob and step keys are not active for this function.

State Saved: Survives **Preset** and power cycle, but not saved in Instrument State.

Factory Default: 18 (Reset by **System**, **Restore Sys Defaults**.)

Maximum Value: 30

Minimum Value: 0

Remote Command:

`:SYSTem:COMMunicate:GPIB[:SELF]:ADDRess <integer>`

`:SYSTem:COMMunicate:GPIB[:SELF]:ADDRess?`

Example:

```
:SYST:COMM:GPIB:ADDR 20
```

6.6.5.2 IP Address

Allows you to set the IP (internet protocol) address, domain name and node (host) name for the instrument. The IP address of the instrument can be changed by entering a numeric address composed of numbers and decimal points. Press ENTER to complete the entry.

State Saved: Survives **Preset** and power cycle, but not saved in Instrument State.

Factory Default: 199.199.199.199 (Not reset by **System**, **Restore Sys Defaults**.)

Remote Command:

`:SYSTem:COMMunicate:LAN[:SELF]:IP <string>`

`:SYSTem:COMMunicate:LAN[:SELF]:IP?`

Example:

```
:SYST:COMM:LAN:IP "150.222.50.52 mypsa"
```

Sets the IP address to 150.222.50.52 and sets the host name to mypsa.

6.6.5.3 Host Name

Allows you to set the host name for LAN identification. Pressing this key activates the alpha editor so you can change the host name. Press ENTER to complete the entry.

State Saved: Survives **Preset** and power cycle, but not saved in Instrument State.

Factory Default: aaaa (Not reset by **System**, **Restore Sys Defaults**.)

Remote Command:

See above command`:SYSTem:COMMunicate:LAN[:SELF]:IP <string>`

Example:

```
:SYST:COMM:LAN:IP "150.222.50.52 mypsa"
```

Sets the IP address to 150.222.50.52 and sets the host name to mypsa.

6.6.5.4 Host ID (Remote Command Only)

Allows you to query the host ID remotely. The current value of the host ID can be viewed on the display by pressing **System, Show System**.

State Saved: Survives **Preset** and power cycle, but not saved in Instrument State.

Factory Default: default is unique to your instrument (Not reset by **System**, **Restore Sys Defaults**.)

Remote Command:

`:SYSTem:HID?`

Example:

The host ID cannot be set remotely, it can only be queried.

```
:SYST:HID?
```

6.6.5.5 SCPI LAN

Accesses keys to enable SCPI functionality over LAN. There are a number of different ways to send SCPI remote commands to the instrument over the LAN. It can be a problem to have multiple users simultaneously accessing the instrument over the LAN. These keys allow you to limit that somewhat by disabling the telnet socket and/or SICL capability.

6.6.5.5.1 SCPI Telnet

Turns on/off the SCPI LAN telnet capability allowing you to limit SCPI access over LAN via telnet.

State Saved: Survives **Preset** and power cycle, but not saved in Instrument State.

Factory Default: On (Reset by System, Restore Sys Defaults.)

Remote Command:

:SYSTem:COMMunicate:LAN:SCPI:TELNet:ENABle OFF|ON|0|1

:SYSTem:COMMunicate:LAN:SCPI:TELNet:ENABle?

Example:

```
:SYST:COMM:LAN:SCPI:TELN:ENAB ON
```

6.6.5.5.2 SCPI Socket

Turns on/off the capability of establishing Socket LAN sessions. This allows you to limit SCPI access over LAN via socket sessions.

State Saved: Survives **Preset** and power cycle, but not saved in Instrument State.

Factory Default: On (Reset by System, Restore Sys Defaults.)

Remote Command:

:SYSTem:COMMunicate:LAN:SCPI:SOCKet:ENABle OFF|ON|0|1

:SYSTem:COMMunicate:LAN:SCPI:SOCKet:ENABle?

Example:

```
:SYST:COMM:LAN:SCPI:SOCK:ENAB ON
```

6.6.5.5.3 SICL Server

Turns on/off the SICL server capability allowing you to limit SCPI access over LAN via the SICL server.

State Saved: Survives **Preset** and power cycle, but not saved in Instrument State.

Factory Default: On (Reset by System, Restore Sys Defaults.)

Remote Command:

```
:SYSTem:COMMunicate:LAN:SCPI:SICL:ENABle OFF|ON|0|1
:SYSTem:COMMunicate:LAN:SCPI:SICL:ENABle?
```

Example:

```
:SYST:COMM:LAN:SCPI:SICL:ENAB ON
```

6.6.6 Show System

Displays the number and description of the options installed in your instrument. It also displays the instrument model number, product number, serial number, ethernet address, host ID, firmware revision, revision date, options, and system statistics.

Remote Command:

```
*IDN?
```

Returns four fields separated by commas:

- Manufacturer
- Model
- Serial number
- Firmware version

Example of returned string: `Agilent Technologies,E4440A,US00000123,A.01.01`

Example:

```
:*IDN?
```

6.6.7 Show Hdwr

Gives detailed information about the hardware installed on your instrument.

Remote Command:

```
:SYSTem:OPTions?
*OPT?
```

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

Example:

```
:*OPT
```

6.6.8 Color Palette

Accesses the Color Palette menu keys that set the display screen attributes.

State Saved: Not saved in Instrument State, survives **Preset**, but not a power cycle.

Factory Default: Default

Remote Command:

There is no remote command for this key.

6.6.8.1 Default

Selects the factory default color palette.

6.6.8.2 Vision Impair 1

Selects a special color scheme to accommodate color-deficient vision problems.

6.6.8.3 Vision Impair 2

Selects a special color scheme to accommodate color-deficient vision problems.

6.6.8.4 Optical Filter

Selects a special color scheme to accommodate protective goggles while viewing lasers.

6.6.8.5 Monochrome

Sets the color palette to single-color mode.

6.6.9 Restore Sys Defaults

Resets all the instrument settings including the “persistent” state variables to their defaults. Persistent values are things such as the auto alignment setting that are unaffected by a power cycle or a preset.

Remote Command:

`:SYSTem:PRESet:PERSistent`

Example:

```
:SYST:PRES:PERS
```

6.6.10 Licensing

Menu that will allow future measurement personality options to be installed in your analyzer.

6.6.11 Service

These functions are only used for servicing your analyzer. A password is required to access them. Refer to the Service Guide for more information.

6.6.12 Power On Elapsed Time (Remote Command Only)

Returns the number of seconds that have elapsed since the instrument was turned on for the very first time.

Remote Command:

```
:SYSTem:PON:ETIMe?
```

Example:

```
:SYST:PON:ETIM?
```

7 Programming Fundamentals

SCPI Language Basics

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

Topics covered in this chapter include:

For more information refer to:

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

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

Command Keywords and Syntax

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

Example: `SENSe:FREQuency:STARt 1.5 MHZ`

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

Example: Sens:Freq:Star 1.5 mhz

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

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

Creating Valid Commands

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

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

Special Characters in Commands

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

Special Character	Meaning	Example
	A vertical stroke between **keywords** indicates identical effects exist for both keywords. The command functions the same for either keyword. Only one of these keywords is used at a time.	Command: `SENSe:BANDwidth|BWIDth:OFFSet` Two identical commands are: Ex1: `SENSE:BWIDTH:OFFSET` Ex2: `SENSE:BAND:OFFSET`
[ ]	keywords in square brackets are optional when composing the command. These implied keywords will be executed even if they are omitted.	Command: `[SENSe:]BANDwidth[:RESolution]:AUTO` The following commands are all valid and have identical effects: Ex1: `bandwidth:auto` Ex2: `band:resolution:auto` Ex3: `sense:bandwidth:auto`
< >	Angle brackets around a word, or words, indicates they are not to be used literally in the command. They represent the needed item.	Command: `SENS:FREQ <freq>` In this command example the word <freq> should be replaced by an actual frequency. Ex: `SENS:FREQ 9.7MHz`.
{ }	Parameters in braces can optionally be used in the command either not at all, once, or several times.	Command: `MEASure:BW <freq>{,level}` A valid command is: `meas:BW 6 MHz, 3dB, 60dB`

Parameters in Commands

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

OFF|ON|0|1 (Boolean)	This is a two state boolean-type parameter. The numeric value 0 is equivalent to OFF. Any numeric value other than 0 is equivalent to ON. The numeric values of 0 or 1 are commonly used in the command instead of OFF or ON. Queries of the parameter always return a numeric value of 0 or 1.
keyword	The keywords that are allowed for a particular command are defined in the command syntax description.
Units	Numeric variables may include units. The valid units for a command depend on the variable type being used. See the following variable descriptions. The indicated default units will be used if no units are sent. Units can follow the numerical value with, or without, a space.
Variable	A variable can be entered in exponential format as well as standard numeric format. The appropriate range of the variable and its optional units are defined in the command description.

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

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

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

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

Variable Parameters

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

<power> <ampl>	Is a rational number followed by optional units. The default units are dBm. Acceptable units include: DBM, DBMV, W.
<rel_power> <rel_ampl>	Is a positive rational number followed by optional units. The default units are dB. Acceptable units include: DB.
<angle> <degrees>	Is a rational number followed by optional units. The default units are degrees. Acceptable units include: DEG, RAD.
<integer>	An integer value has no units.
<percent>	Is a rational number between 0 and 100, with no units.
<string>	Is a series of alpha numeric characters.

Block Program Data

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

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

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

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

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

- Hexadecimal, #Hdddd or #hdddd where ‘d’ represents a hexadecimal digit 0 to 9, or a to f.
- Octal, #Oddddd or #oddddd where ‘d’ represents an octal digit 0 to 7.
- Binary, #Bdddddddddddddddd or #bdddddddddddddddd where ‘d’ represents a 1 or 0.

<arbitrary block data> Consists of a block of data bytes. The first information sent in the block is an ASCII header beginning with #. The header can be used to determine how many bytes are in the data block. There are no units.

For example, suppose the header is #512320.

- The first digit in the header (5) tells you how many additional digits/bytes there are in the header.
- The 12320 means 12 thousand, 3 hundred, 20 data bytes follow the header.

- Divide this number of bytes by your current data format (bytes/point), either 8 (for real64), or 4 (for real32). For this example, if you’re using real64 then there are 1540 points in the block.

Programming in C Using VTL

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

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

Typical Example Program Contents

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

`visa.h`	This file is included at the beginning of the file to provide the function prototypes and constants defined by VTL.
`ViSession`	The `ViSession` is a VTL data type. Each object that will establish a communication channel must be defined as `ViSession`.
`viOpenDefaultRM`	You must first open a session with the default resource manager with the `viOpenDefaultRM` function. This function will initialize the default resource manager and return a pointer to that resource manager session.
`viOpen`	This function establishes a communication channel with the device specified. A session identifier that can be used with other VTL functions is returned. This call must be made for each device you will be using.
`viPrintf` `viScanf`	These are the VTL formatted I/O functions that are patterned after those used in the C programming language. The `viPrintf` call sends the IEEE 488.2 `*RST` command to the instrument and puts it in a known state. The `viPrintf` call is used again to query for the device identification (`*IDN?`). The `viScanf` call is then used to read the results.
`viClose`	This function must be used to close each session. When you close a device session, all data structures that had been allocated for the session will be de-allocated. When you close the default manager session, all sessions opened using the default manager session will be closed.

Linking to VTL Libraries

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

32-bit Version:

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

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

16-bit Version:

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

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

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

Compiling and Linking a VTL Program

32-bit Applications

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

For Microsoft Visual C++ version 2.0 compilers:

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

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

For Borland C++ version 4.0 compilers:

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

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

16-bit Applications

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

For Microsoft Visual C++ version 1.5:

- To set the memory model, do the following:
  1. Select `Options | Project`.
  2. Click on the `Compiler` button, then select `Memory Model` from the `Category` list.
  3. Click on the `Model` list arrow to display the model options, and select `Large`.
  4. Click on `OK` to close the `Compiler` dialog box.
- You may wish to add the include file and library file search paths. They are set under the `Options | Directories` menu selection:

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

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

Example Program

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

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

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

void main ()
{

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

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

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

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

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

     /* Close sessions */
```

```
    viClose (vi);
    viClose (defaultRM);
}
```

Including the VISA Declarations File

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

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

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

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

Opening a Session

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

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

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

Device Sessions

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

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

viOpenDefaultRM (*sesn*);

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

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

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

The following summarizes the parameters in the previous function calls:

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

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

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

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

Addressing a Session

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

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

The following describes the parameters used above:

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

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

The following are examples of valid symbolic names:

XI0::24::INSTR	Device at VXI logical address 24 that is of VISA type INSTR.
VXI2::128	Device at VXI logical address 128, in the third VXI system (VXI2).

GPIB-VXI0::24	A VXI device at logical address 24. This VXI device is connected via a GPIB-VXI command module.
GPIB0::7::0	A GPIB device at primary address 7 and secondary address 0 on the GPIB interface.

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

```
ViSession defaultRM, vi;

.
.

viOpenDefaultRM (&defaultRM);

viOpen (defaultRM, "GPIB0::23::INSTR", VI_NULL,VI_NULL,&vi);

.
.

viClose(vi);

viClose (defaultRM);
```

Closing a Session

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

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

Overview of the GPIB Bus

GPIB Instrument Nomenclature

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

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

GPIB Command Statements

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

The seven fundamental command functions are as follows:

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

- An output function that is used to send function commands and data commands from the controller to the addressed instrument (sometimes called: output, control, convert, image, iobuffer, transfer).
- An enter function that is the complement of the output function and is used to transfer data from the addressed instrument to the controller (sometimes called: enter, convert, image, iobuffer, on timeout, set timeout, transfer).

Using the LAN to Control the Analyzer

Using ftp for File Transfers

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

A sample ftp session from a MSDOS window on your PC, might be:

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

   You are now in the instrument `/rubicon` directory and can get files from the analyzer. The ftp commands in the following steps may not all be available from your controller. Typing `help` at the prompt will show you the ftp commands that are available on your system. Typing `quit` will end your ftp session.

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

4. `cd /userdir` (changes to the directory where data files are saved)
5. `ls` (lists all the available files, `ls -la` shows file permissions)
6. `bin` (changes to the binary file transfer mode)
7. `get myfilename` (<enter you file name> It is case sensitive.)

   This will "get" (copy) your file. Remember that the file will be copied to whatever location you were pointing to when you started the ftp process. Query your current location by entering `lcd .`(include the period). Change that location by entering `lcd C:\my path\mydir` (enter your path/directory location.)

The Standard UNIX FTP Command:

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

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

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

The following options are supported:

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

Table 7-1 lists the available user commands.

Table 7-1 **`ftp` Commands**

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

Table 7-1 ftp Commands

Command	**Description**
mget *remote_file [local_file]*	Copy *remote_file* to the local system. If *local_file* is unspecified, ftp uses the *remote_file* name as the *local_file* name.
mput *local_file [remote_file]*	Copies *local_file* to remote file. If *remote_file* is unspecified, ftp uses the *local_file* name as the *remote_file* name.
put *local_file* [*remote_file*]	Copies *local_file* to *remote file*. If *remote_file* is unspecified, ftp uses the *local_file* name as the *remote_file* name.
quit	Closes the connection to the host and exits ftp.

Using Telnet to Send Commands

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

NOTE If you need to control the bus using “device clear” or SRQ’s, you can use SICL LAN. SICL LAN provides control of your analyzer via IEEE 488.2 GPIB over the LAN. See “Using SICL LAN to Control the Analyzer” on page 228 in this chapter.

On unix or PC:

The syntax of the telnet command is:

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

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

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

You would type at the dos prompt

```
telnet
```

Unix Telnet Example:

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

```
telnet aaa 5023
```

When you connect to the instrument, it will display a welcome message and a command prompt. The instrument is now ready to accept your SCPI commands. As you type SCPI commands, query results appear on the next line. When you are done, break the telnet connection using your escape character, and type `quit`.

When the analyzer responds with the welcome message and the SCPI prompt, you can immediately enter programming (SCPI) commands. Typical commands might be:

```
CALC:MARK:MODE POS
CALC:MARK:MAX
CALC:MARK:X?
```

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

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

```
+2.50000000000E+010
```

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

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

The telnet connection closes and you see your regular prompt.

```
Connection closed.
```

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

Telnet Example:

```
Welcome to at42

Agilent Technologies,E4440A,US00000005,PROTOTYPE


SCPI>calc:mark:mode pos

SCPI>calc:mark:max

SCPI>calc:mark:x?

+2.5000000000000000E+010

SCPI>
```

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

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

The Standard UNIX TELNET Command:

Synopsis `telnet [host [port]]`

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

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

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

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

Using Socket LAN to Send Commands

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

Setting Up Your Analyzer for Socket Programming

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

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

NOTE LAN "device clear" capability has not been implemented in firmware revision A.01.xx.

Using SICL LAN to Control the Analyzer

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

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

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

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

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

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

Collecting SICL LAN Set-up Information

Before you set up your analyzer as a SICL LAN server, you will need to collect some information about your VISA/SICL LAN client application. The "value" of the following parameters can be found from the front panel **System** keys. They can then be used to set up your VISA/SICL LAN client application:

Emulated GPIB Name	The GPIB name is the name given to a device used to communicate with the analyzer. Your analyzer is shipped with gpib7 as its GPIB name. The GPIB name is the same as the remote SICL address.
Emulated GPIB Logical Unit	The logical unit number is a unique integer assigned to the device to be controlled using SICL LAN. Your analyzer is shipped with the logical unit number set to 8.
Emulated GPIB Address	The emulated GPIB address (bus address) is assigned to the device to be controlled using SICL LAN. The emulated GPIB address is automatically set to be the same as the current GPIB address. The instrument is shipped with the emulated GPIB address set to 18.

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

Configuring Your Analyzer as a SICL LAN Server

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

1. Identify the GPIB name.

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

2. Notice that the **Emulated GPIB Logical Unit** is set to **8**.
3. Notice that the **Emulated GPIB Address** is set the same as the GPIB address.

Configuring Your PC as a SICL LAN Client

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

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

These descriptions assume a LAN connection between your computer and network analyzer. They are not written for the GPIB to LAN gateway.

1. Install VISA revision G.02.02 or higher.
2. Run I/O configuration.
3. Select LAN Client from the available interface types.
4. Press Configure.

5. Enter an interface name, such as lan1.
6. Enter a logical unit number, such as 7.
7. Select Okay.
8. Select VISA LAN Client from the available interface types.
9. Press Configure.
10. Enter a VISA interface name, such as GPIB1.
11. Enter the hostname or IP address of your analyzer in the hostname field, such as my4440a.companyname.com
12. Enter a Remote SICL address, such as GPIB1.
13. Set the LAN interface to match the defined LAN client (lan1 in this example).
14. Select OK.
15. Close I/O Configuration by selecting OK.

Controlling Your Analyzer with SICL LAN and HP/Agilent VEE

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

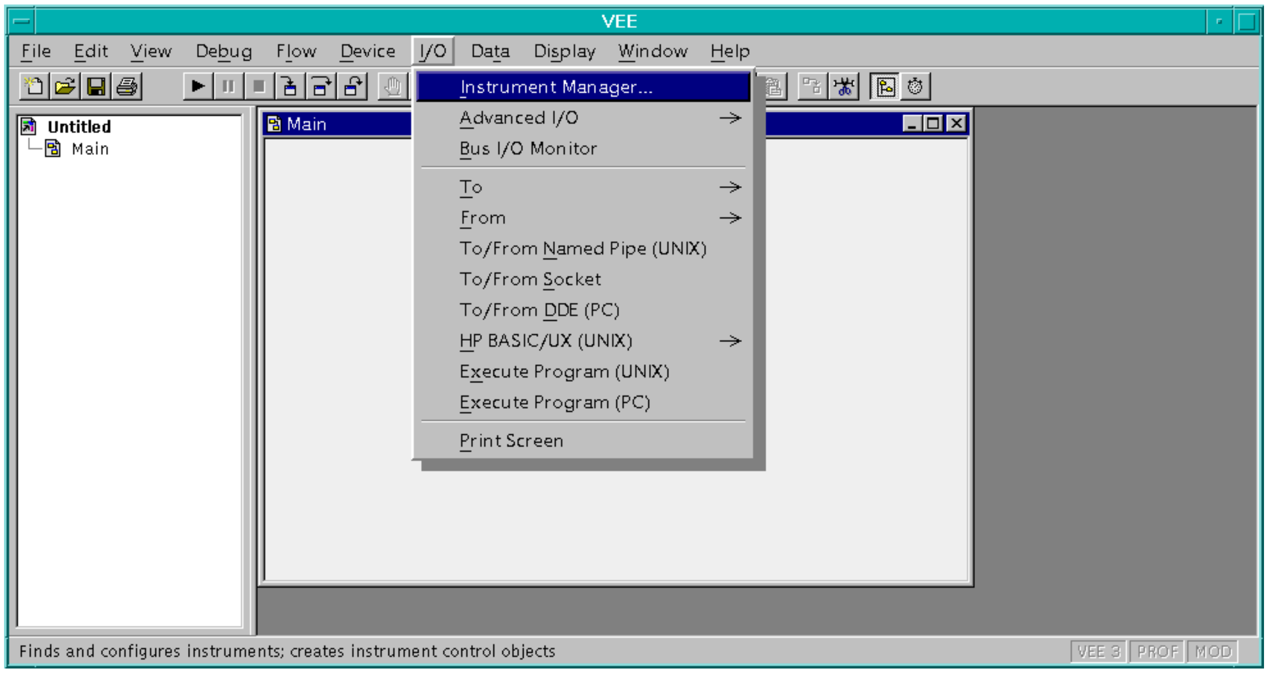
NOTE If you are using Agilent VEE and SICL LAN, the logical unit number is limited to the range of 0-8.

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

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

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

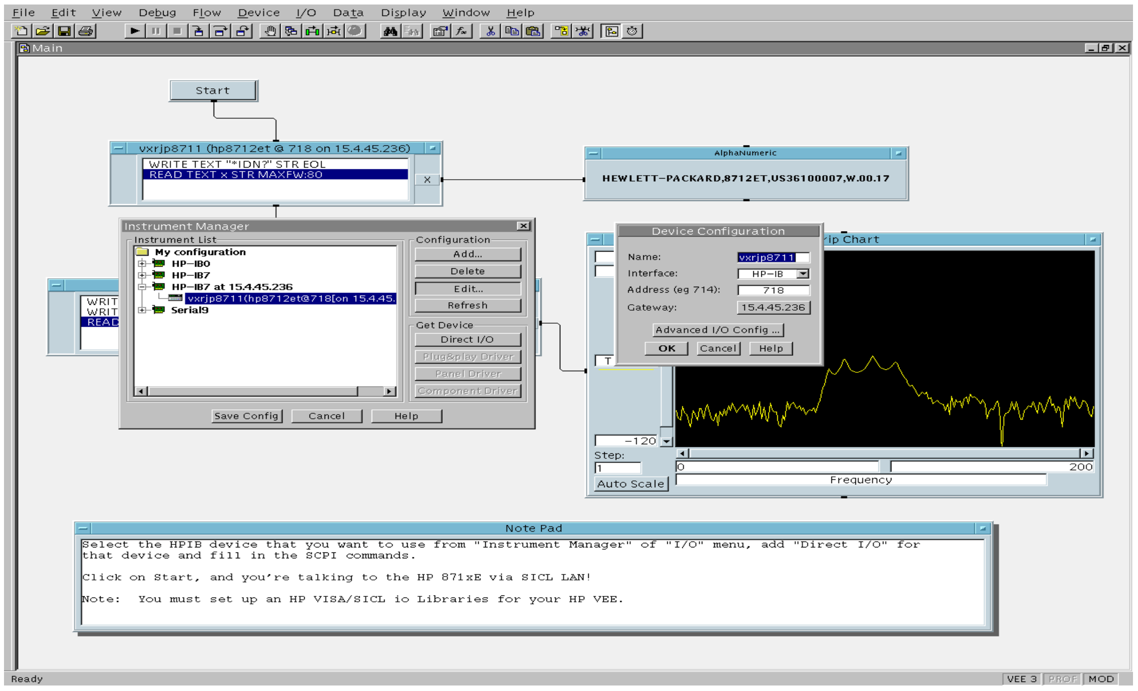
Figure 7-1 I/O | Instrument Manager Menu



Finds and configures instruments; creates instrument control objects

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

Figure 7-2 Adding Your Analyzer as a VEE Device



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

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

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

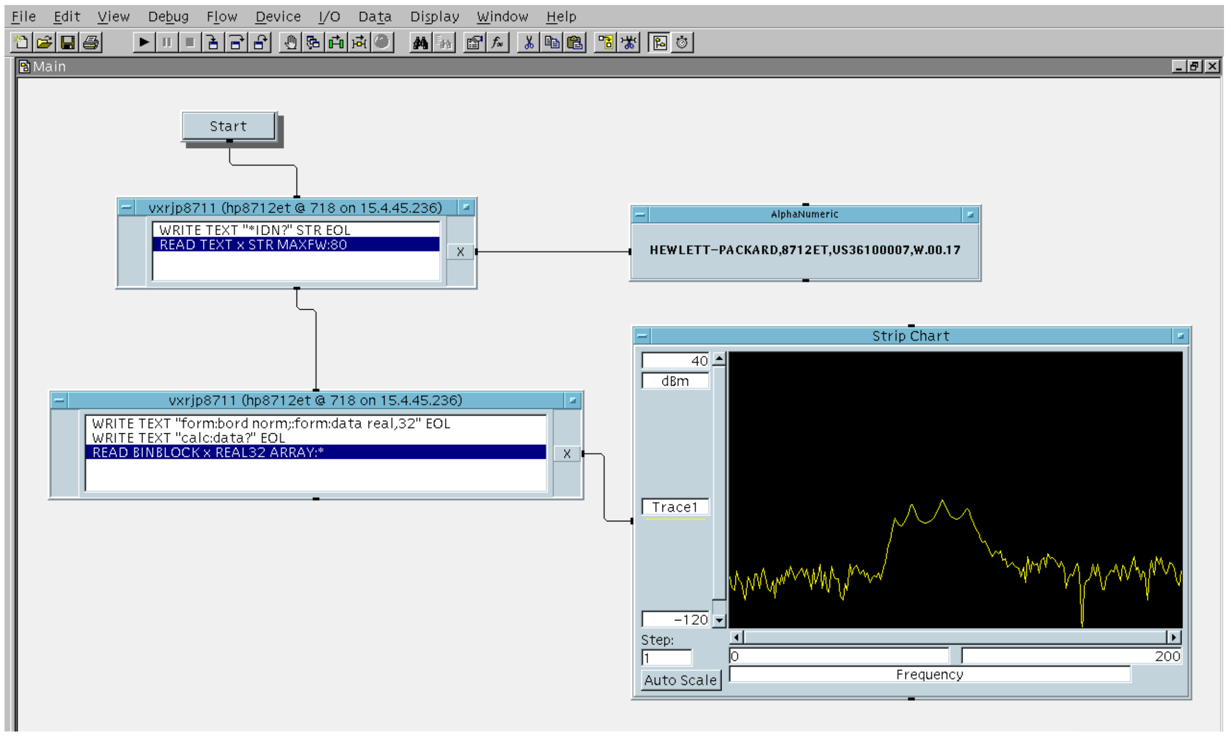
```
Select the HPIB device that you want to use from "Instrument Manager" of "I/O" menu, add "Direct I/O" for
that device and fill in the SCPI commands.

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

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

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

Figure 7-3 Sending SCPI Commands Directly to your Analyzer


File Edit View Debug Flow Device I/O Data Display Window Help
Main
Start
vxrjp8711 (hp8712et @ 718 on 15.4.45.236)
X
AlphaNumeric
Strip Chart
40
dBm
Trace1
-120
Step:
1
Auto Scale
0
200
Frequency

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

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

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

See the VEE example program for more details.

Controlling Your Analyzer with SICL LAN and Agilent BASIC for Windows

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

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

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

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

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

analyzer IP address **191.108.344.225**

analyzer GPIB name **inst0**

logical unit number **7**

timeout value (seconds) **30**

`LOAD` statement (all on a single line)

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

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

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

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

where 18 is the device address for the analyzer.

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

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

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

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

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

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

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

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

where 18 is the device address for the analyzer.

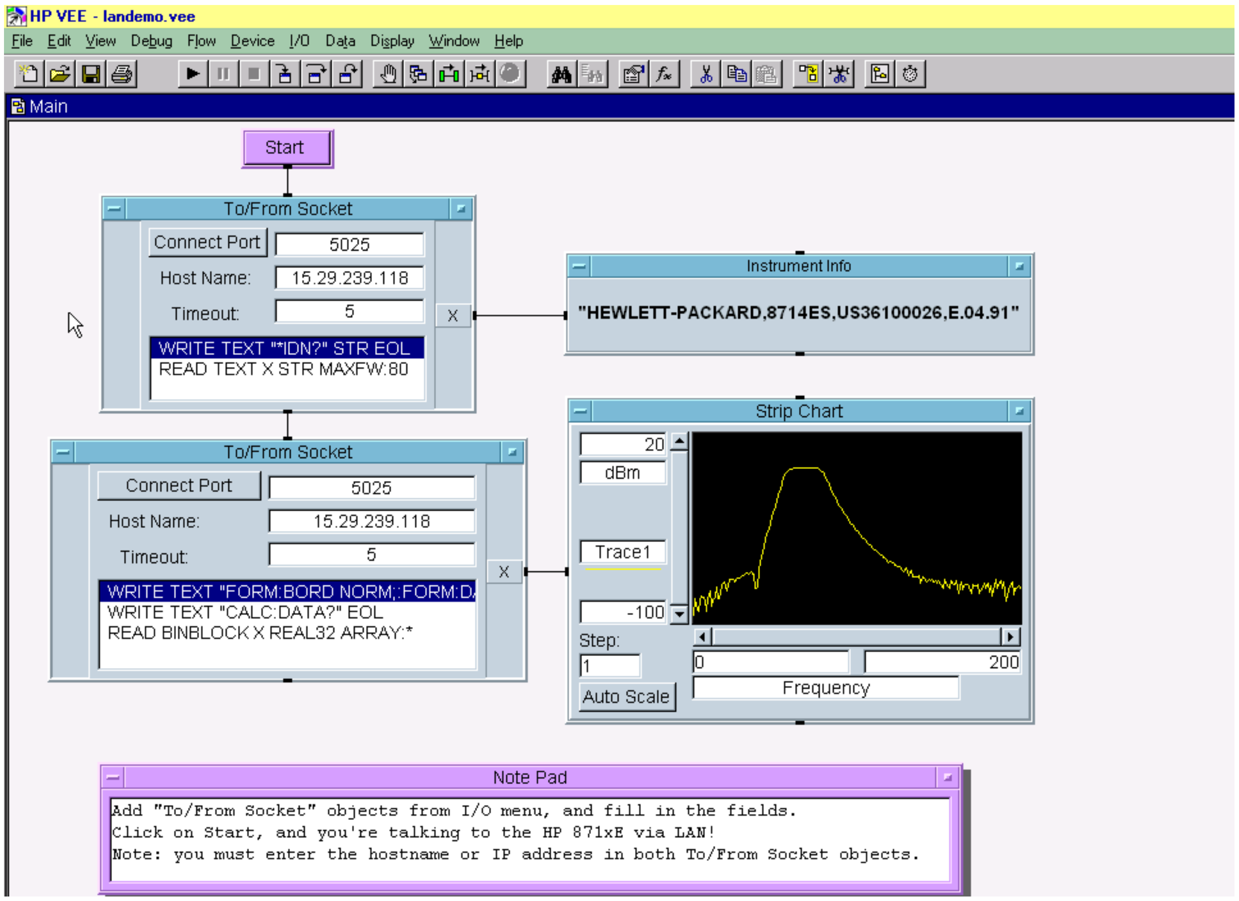
Using HP/Agilent VEE Over Socket LAN

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

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

For faster troubleshooting, you may want to set the timeout to a smaller number. If the hostname you enter doesn't work, try using the IP address of your analyzer (example: `191.108.43.5`). Using the IP address rather than the hostname may also be faster. See Figure 7-4 for an example of an VEE screen.

Figure 7-4 Sample VEE Screen



To/From Socket
Connect Port: 5025
Host Name: 15.29.239.118
Timeout: 5

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

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

To/From Socket
Connect Port: 5025
Host Name: 15.29.239.118
Timeout: 5

```
WRITE TEXT "FORM:BORD NORM;:FORM:D
WRITE TEXT "CALC:DATA?" EOL
READ BINBLOCK X REAL32 ARRAY:*
```

Note Pad

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

Using a Java™ Applet Over Socket LAN

The following example program demonstrates simple socket programming with Java. It is written in Java programming language, and will compile with Java compilers versions 1.0 and above.

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

Example Using Java Over Socket LAN

This programming example (ScpiDemo.java) will compile with Java compilers versions 1.0 and above.

Example:

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

// This is a SCPI Demo to demonstrate how one can communicate with the
// E4440A PSA  with a JAVA capable browser.  This is the
// Main class for the SCPI Demo.  This applet will need Socks.class to
// support the I/O commands and a ScpiDemo.html for a browser to load
// the applet.
// To use this applet, either compile this applet with a Java compiler
// or use the existing compiled classes.  copy ScpiDemo.class,
// Socks.class and ScpiDemo.html to a floppy.  Insert the floppy into
// your instrument.  Load up a browser on your computer and do the
// following:
//     1. Load this URL in your browser:
//        ftp://<Your instrument’s IP address or name>/int/ScpiDemo.html
//     2. There should be two text windows show up in the browser:
//        The top one is the SCPI response text area for any response
//        coming back from the instrument.  The bottom one is for you
//        to enter a SCPI command.  Type in a SCPI command and hit enter.
//        If the command expects a response, it will show up in the top
//        window.
public class ScpiDemo extends java.applet.Applet implements Runnable {
    Thread       responseThread;
    Socks        sck;
    URL          appletBase;
    TextField    scpiCommand = new TextField();
    TextArea     scpiResponse = new TextArea(10, 60);
    Panel        southPanel = new Panel();
    Panel        p;

    // Initialize the applets
    public void init() {

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

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

    // This routine is called whenever the applet is out of scope
    // i.e. minize browser
    public void stop() {
        // Close all local sockets
        sck.CloseSockets();
        // Kill the response thread
        StartResponseThread(false);
    }

    // Action for sending out scpi commands
    // This routine is called whenever a command is received from the
    // SCPI command panel.
    public boolean action(Event evt, Object what) {
        // If this is the correct target
        if (evt.target == scpiCommand) {
            // Get the scpi command
            String str = scpiCommand.getText();
            // Send it out to the Scpi socket
            sck.ScpiWriteLine(str);
            String tempStr = str.toLowerCase();
            // If command str is "syst:err?", don't need to send another one.
            if ( (tempStr.indexOf("syst") == -1) ||
                 (tempStr.indexOf("err") == -1)  ) {
                // Query for any error
                sck.ScpiWriteLine("syst:err?");
            }
            return true;
        }
        return false;
    }

    // Start/Stop a Response thread to display the response strings
    private void StartResponseThread(boolean start) {
        if (start) {
            // Start a response thread
            responseThread = new Thread(this);
            responseThread.start();
        }
        else {
            // Kill the response thread
            responseThread = null;
        }
    }

    // Response thread running
    public void run() {
        String str = "";  // Initialize str to null

        // Clear the error queue before starting the thread
        // in case if there’s any error messages from the previous actions
        while ( str.indexOf("No error") == -1 ) {
            sck.ScpiWriteLine("syst:err?");
            str = sck.ScpiReadLine();
        }

        // Start receiving response or error messages
        while(true) {
            str = sck.ScpiReadLine();
            if ( str != null ) {
                // If response messages is "No error", do no display it,
                // replace it with "OK" instead.
                if ( str.equals("+0,\"No error\"") ) {
                    str = "OK";
                }
                // Display any response messages in the Response panel
                scpiResponse.appendText(str+"\n");
            }
        }
    }

    // Set up and open the SCPI sockets
    private void SetupSockets() {
        // Get server url
        appletBase = (URL)getCodeBase();
        // Open the sockets
        sck = new Socks(appletBase);
    }

    // Set up the SCPI command and response panels
    private void SetupPanels() {
        // Set up SCPI command panel
        southPanel.setLayout(new GridLayout(1, 1));
        p = new Panel();
        p.setLayout(new BorderLayout());
        p.add("West", new Label("SCPI command:"));
        p.add("Center", scpiCommand);
        southPanel.add(p);

        // Set up the Response panel
        setLayout(new BorderLayout(2,2));
        add("Center", scpiResponse);
        add("South", southPanel);
    }

}


//  Socks class is responsible for open/close/read/write operations
//  from the predefined socket ports.  For this example program,
//  the only port used is 5025 for the SCPI port.
class Socks extends java.applet.Applet {
    // Socket Info
    // To add a new socket, add a constant here, change MAX_NUM_OF_SOCKETS
    // then, edit the constructor for the new socket.
    public final int SCPI=0;
    private final int MAX_NUM_OF_SOCKETS=1;

    // Port number
    // 5025 is the dedicated port number for E4440A Scpi Port
    private final int SCPI_PORT = 5025;

    // Socket info
    private URL appletBase;
    private Socket[] sock = new Socket[MAX_NUM_OF_SOCKETS];
    private DataInputStream[] sockIn = new DataInputStream[MAX_NUM_OF_SOCKETS];
    private PrintStream[] sockOut = new PrintStream[MAX_NUM_OF_SOCKETS];
    private int[] port = new int[MAX_NUM_OF_SOCKETS];
    private boolean[] sockOpen = new boolean[MAX_NUM_OF_SOCKETS];

    // Constructor
    Socks(URL appletB)
    {
        appletBase = appletB;

        // Set up for port array.
        port[SCPI] = SCPI_PORT;

        // Initialize the sock array
        for ( int i = 0; i < MAX_NUM_OF_SOCKETS; i++ ) {
            sock[i] = null;
            sockIn[i] = null;
            sockOut[i] = null;
            sockOpen[i] = false;
        }
    }

    //***** Sockects open/close routines
    // Open the socket(s) if not already opened
    public void OpenSockets()
    {
        try {
            // Open each socket if possible
            for ( int i = 0; i < MAX_NUM_OF_SOCKETS; i++ ) {
                if ( !sockOpen[i] ) {
                    sock[i] = new Socket(appletBase.getHost(),port[i]);
                    sockIn[i] = new DataInputStream(sock[i].getInputStream());
                    sockOut[i] = new PrintStream(sock[i].getOutputStream());
                    if ( (sock[i] != null) && (sockIn[i] != null) &&
                         (sockOut[i] != null) ) {
                        sockOpen[i] = true;
                    }
                }
            }
        }
        catch (IOException e) {
            System.out.println("Sock, Open Error "+e.getMessage());
        }
    }

    // Close the socket(s) if opened
    public void CloseSocket(int s)
    {
        try {
            if ( sockOpen[s] == true ) {
                // write blank line to exit servers elegantly
                sockOut[s].println();
                sockOut[s].flush();
                sockIn[s].close();
                sockOut[s].close();
                sock[s].close();
                sockOpen[s] = false;
            }
        }
        catch (IOException e) {
            System.out.println("Sock, Close Error "+e.getMessage());
        }
    }

    // Close all sockets
    public void CloseSockets()
    {
        for ( int i=0; i < MAX_NUM_OF_SOCKETS; i++ ) {
            CloseSocket(i);
        }
    }

    // Return the status of the socket, open or close.
    public boolean SockOpen(int s)
    {
        return sockOpen[s];
    }


    //************* Socket I/O routines.

    //*** I/O routines for SCPI socket
    // Write an ASCII string with carriage return to SCPI socket
    public void ScpiWriteLine(String command)
    {
        if ( SockOpen(SCPI) ) {
            sockOut[SCPI].println(command);
            sockOut[SCPI].flush();
        }
    }

    // Read an ASCII string, terminated with carriage return from SCPI socket
    public String ScpiReadLine()
    {
        try {
            if ( SockOpen(SCPI) ) {
                return sockIn[SCPI].readLine();
            }
        }
        catch (IOException e) {
            System.out.println("Scpi Read Line Error "+e.getMessage());
        }
        return null;
    }

    // Read a byte from SCPI socket
    public byte ScpiReadByte()
    {
        try {
            if ( SockOpen(SCPI) ) {
                return sockIn[SCPI].readByte();
            }
        }
        catch (IOException e) {
            System.out.println("Scpi Read Byte Error "+e.getMessage());
        }
        return 0;
    }

}
```

Using a C Program Over Socket LAN

The following example programs demonstrate simple socket programming. They are written in C, and compile in the HP-UX UNIX environment or the WIN32 environment.

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

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

Example Using C Over Socket LAN (UNIX)

This C programming example (socketio.c) compiles in the HP-UX UNIX environment. It is portable to other UNIX environments with only minor changes.

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

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

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

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

This program is also available on your documentation CD ROM.

Example:

```
/***************************************************************************
 *  $Header: socketio.c,v 1.5 96/10/04 20:29:32 roger Exp $
 *  $Revision: 1.5 $
 *  $Date: 96/10/04 20:29:32 $
 *
 *  $Contributor:     LSID, MID $
 *
 *  $Description:     Functions to talk to an Agilent E4440A spectrum
 *                    analyzer via TCP/IP.  Uses command-line arguments.
 *
 *                    A TCP/IP connection to port 5025 is established and
 *                    the resultant file descriptor is used to "talk" to the
 *                    instrument using regular socket I/O mechanisms. $
 *
 *
 *
 *  E4440A Examples:
 *
 *   Query the center frequency:
 *        lanio 15.4.43.5 'sens:freq:cent?'
 *
 * Query X and Y values of marker 1 and marker 2 (assumes they are on):
 *        lanio myinst 'calc:spec:mark1:x?;y?; :calc:spec:mark2:x?;y?'
 *
 *   Check for errors (gets one error):
 *        lanio myinst 'syst:err?'
 *
 *   Send a list of commands from a file, and number them:
 *        cat scpi_cmds | lanio -n myinst
 *
 ***************************************************************************
 *
 *  This program compiles and runs under
 *     - HP-UX 10.20 (UNIX), using HP cc or gcc:
 *           + cc -Aa    -O -o lanio  lanio.c
 *           + gcc -Wall -O -o lanio  lanio.c
 *
 *     - Windows 95, using Microsoft Visual C++ 4.0 Standard Edition
 *     - Windows NT 3.51, using Microsoft Visual C++ 4.0
 *           + Be sure to add  WSOCK32.LIB  to your list of libraries!
 *           + Compile both lanio.c and getopt.c
 *           + Consider re-naming the files to lanio.cpp and getopt.cpp
 *
 *  Considerations:
 *     - On UNIX systems, file I/O can be used on network sockets.
 *       This makes programming very convenient, since routines like
 *       getc(), fgets(), fscanf() and fprintf() can be used.  These
 *       routines typically use the lower level read() and write() calls.
 *
 *     - In the Windows environment, file operations such as read(), write(),
 *       and close() cannot be assumed to work correctly when applied to
 *       sockets.  Instead, the functions send() and recv() MUST be used.
 */

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

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

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

#ifdef WINSOCK

#include <windows.h>

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

#else /* UNIX with BSD sockets */

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

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

   typedef  int SOCKET;

#endif /* WINSOCK */

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

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

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


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



#ifdef WINSOCK
int init_winsock(void)
{
    WORD wVersionRequested;
    WSADATA wsaData;
    int err;
    wVersionRequested = MAKEWORD(1, 1);
    wVersionRequested = MAKEWORD(2, 0);

    err = WSAStartup(wVersionRequested, &wsaData);

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

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



/***************************************************************************
 *
 > $Function: openSocket$
 *
 * $Description:  open a TCP/IP socket connection to the instrument $
 *
 * $Parameters:  $
 *    (const char *) hostname . . . . Network name of instrument.
 *                                    This can be in dotted decimal notation.
 *    (int) portNumber  . . . . . . . The TCP/IP port to talk to.
 *                                    Use 5025 for the SCPI port.
 *
 * $Return:     (int)  . . . . . . . . A file descriptor similar to open(1).$
 *
 * $Errors:     returns -1 if anything goes wrong $
 *
 ***************************************************************************/
SOCKET openSocket(const char *hostname, int portNumber)
{
    struct hostent *hostPtr;
    struct sockaddr_in peeraddr_in;
    SOCKET s;


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

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

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

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

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

    return s;
}



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

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

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

    return NO_CMD_ERROR;
}


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

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

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

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

    }

    *ptr = '\0';

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



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

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


    /*********************************************************
     * Read response from analyzer
     ********************************************************/
    count = recv(sock, tmp_buf, 1, 0); /* read 1 char */
    ch = tmp_buf[0];

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

```
}

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

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

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

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

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

            /************************************************************
             * For LAN dumps, there is always an extra trailing newline
             * Since there is no EOI line.  For ASCII dumps this is
             * great but for binary dumps, it is not needed.
             ***********************************************************/
            if (resultBytes == numBytes)
            {
                char junk;
                count = recv(sock, &junk, 1, 0);
            }
        }
        else
        {
            /* indefinite block ... dump til we read only a line feed */
```

```
                do
                {
                    if (recv_line(sock, result, maxLength) == NULL) break;
                    if (strlen(result)==1 && *result == '\n') break;
                    resultBytes += strlen(result);
                    result += strlen(result);
                } while (1);
            }
        }
        else
        {
            /* ASCII response (not a binary block) */
            *result = (char)ch;
            if (recv_line(sock, result+1, maxLength-1) == NULL) return 0;

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

    return resultBytes;
}



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

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

        /*****************************************************************
         * Typical result_str:
         *     -221,"Settings conflict; Frequency span reduced."
         *     +0,"No error"
         * Don't bother decoding.
         *****************************************************************/
        if (strncmp(result_str, "+0,", 3) == 0) {
            /* Matched +0,"No error" */
            break;
        }
        puts(result_str);
    } while (1);

}


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

    /*********************************************************/
    /* if the command has a '?' in it, use queryInstrument.  */
    /* otherwise, simply send the command.                   */
    /* Actually, we must a little more specific so that      */
    /* marker value queries are treated as commands.         */
    /* Example:  SENS:FREQ:CENT (CALC1:MARK1:X?)             */
    /*********************************************************/
    if ( (query = strchr(cmd,'?')) != NULL)
    {
        /* Make sure we don't have a marker value query, or
         * any command with a '?' followed by a ')' character.
         * This kind of command is not a query from our point of view.
         * The analyzer does the query internally, and uses the result.
         */
        query++ ;       /* bump past '?' */
        while (*query)
        {
            if (*query == ' ') /* attempt to ignore white spc */
                query++ ;
            else break ;
        }

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

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

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

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

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

    /* now look for hostname and optional <command> */
    if (optind < argc)
    {
        destination = argv[optind++] ;
        strcpy(command, "");
        if (optind < argc)
        {
            while (optind < argc) {
                /* <hostname> <command> provided; only one command string */
                strcat(command, argv[optind++]);
                if (optind < argc) {
                    strcat(command, " ");
                } else {
                    strcat(command, "\n");
                }
            }
        }
        else
        {
            /* Only <hostname> provided; input on <stdin> */
            strcpy(command, "");

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

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

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

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

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

            strcat(charBuf, "\n");

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

            if ( isQuery(charBuf) )
            {
                long bufBytes;

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

    if (show_errs) {
        showErrors(instSock);
    }

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

    return 0;
}

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

Example Using C Over Socket LAN (Windows NT)

This C programming example (getopt.c) compiles in the Windows NT environment. In Windows, the routines send() and recv() must be used, since fread() and fwrite() may not work on sockets.

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

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

Example:

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

 getopt(3C)                                                      getopt(3C)


 NAME
      getopt - get option letter from argument vector

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

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

 DESCRIPTION
```

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

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

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

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


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

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

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

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

    optarg = NULL;

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

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

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

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

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

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

    return(c);
}
```

General LAN Troubleshooting

- "Troubleshooting the Initial Connection" on page 258.
- "Common Problems After You've Made the Connection" on page 259.
- "Pinging the Analyzer from Your Computer or Workstation" on page 261.
- "EIA/TIA 568B Wiring Information" on page 263.

Troubleshooting the Initial Connection

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

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

Either a hardware problem or a software problem can prevent the analyzer's remote file server from communicating over the LAN. The following common problems may be encountered:

Communications Not Established If you have just installed and configured the LAN interface and you have never been able to access the analyzer via ftp or telnet, go directly to "Pinging the Analyzer from Your Computer or Workstation" on page 261.

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

❑ Has any hardware been added or moved on your network? This includes adding or removing any workstations or peripherals, or changing any cabling.

❑ Have software applications been added to the network?

❑ Has the functionality been turned off from the front panel? Press **System, Config I/O, SCPI LAN**.

❑ Have any configuration files been modified? Pressing **System, Restore Sys Defaults** restores the original factory defaults and you will have to re-set the instrument IP address and hostname.

❑ Is the upper- and lower-case character usage in your hostname consistent?

❑ Have any of the following files been deleted or overwritten?

UNIX:

— /etc/hosts

— /etc/inetd.conf

— /etc/services

PCs:

— dependent network files

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

Timeout Errors

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

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

— The LAN connection has failed, or fails occasionally.

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

Packets Routinely Lost

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

Problems Transferring or Copying Files

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

Common Problems After You’ve Made the Connection

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

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

You cannot connect to the analyzer

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

— Make sure the analyzer is turned on.

— Check the physical connection to the LAN.

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

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

You cannot access the file system via ftp

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

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

- If you get a "connection timed out" message

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

You cannot telnet to the command parser port

- If you get a "connection refused" message

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

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

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

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

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

You get an "operation timed-out" message

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

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

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

If all else fails

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

Pinging the Analyzer from Your Computer or Workstation

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

From a UNIX workstation, type:

```
ping hostname 64 10
```

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

From a DOS or Windows environment, type:

```
ping hostname 10
```

where 10 is the number of echo requests.

Normal Response for UNIX

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

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

Normal Response for DOS or Windows

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

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

Error Messages

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

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

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

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

Check that the hostname and IP address are correctly entered in the node names database.

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

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

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

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

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

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

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

Options and Parameters

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

EIA/TIA 568B Wiring Information

Table 7-2 Straight-Through Cable (Unshielded-twisted-pair (UTP) cable with RJ-45 connectors)

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

Table 7-3 Cross-Over Cable (Unshielded-twisted-pair (UTP) cable with RJ-45 connectors)

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

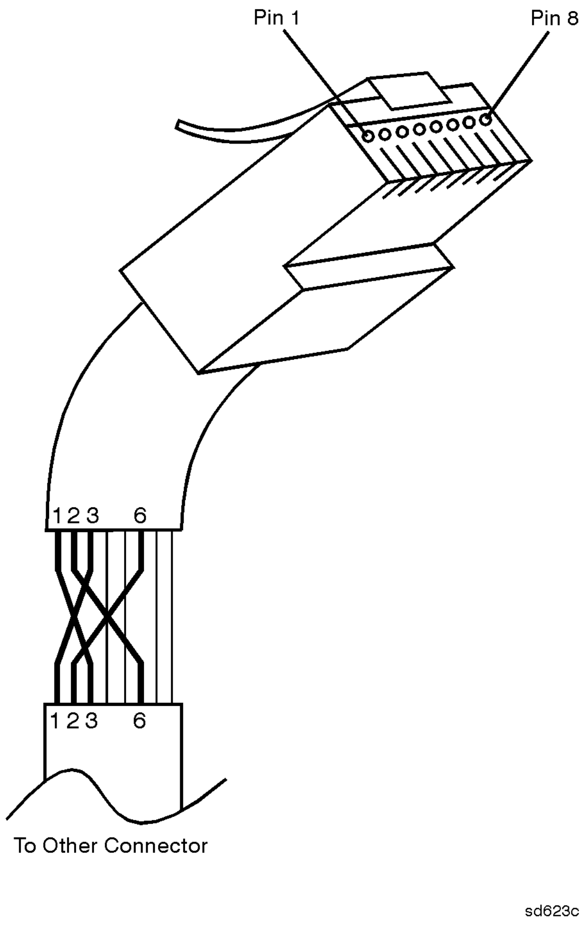
a. Either end of this cable can be used at the analyzer or LAN device. The connector names are a convention useful during cable construction only.

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

NOTE A convenient way to make a cross-over adapter is to use two RJ-45 *jacks* wired according to Table 7-3, above. Standard straight-through patch cables can then be used from the analyzer to the adapter, and from the adapter to other LAN devices. If you use a special-purpose adapter, you will avoid having a cross-over cable mistaken for a standard, straight-through patch cable.

NOTE Some commercially-available cross-over cables do not implement the cross-over wiring required for your analyzer. Please refer to Table 7-3, above, and verify all connections before using cables not made by Agilent Technologies.

Figure 7-5 Cross-Over Patch Cable Wiring (cross-over end)



8 Status Commands

When you are programming the instrument you may need to monitor instrument status to check for error conditions or monitor changes.You can determine the state of certain instrument events/conditions by programming the status register system. Figure 8.5.2 on page 273 shows the available instrument status registers and their hierarchy. IEEE common commands (those beginning with *) access the higher-level summary registers. To access the information from specific registers you would use the STATus commands.

8.1 Using the Instrument Status Registers

8.2 What are the Status Registers?

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

Most status registers are actually a family of five registers:

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

8.2.1 What are the Status Register SCPI Commands?

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

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

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

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

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

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

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

8.3 Why Would You Use the Status Registers?

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

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

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

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

Use the SRQ method when:

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

Use polling when:

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

To monitor a condition:

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

You can monitor conditions in different ways.

- Check the current instrument hardware and firmware status.

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

- Monitor for the occurrence of a particular condition (bit).

  Once you have enabled a bit, using the event enable register, the instrument will monitor that particular bit. If the bit becomes true in the event register, it will stay set until the event register is cleared. Querying the event register allows you to detect that this condition occurred even if the condition no longer exists. The event register can only be cleared by querying it or sending the *CLS command, which clears all event registers.

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

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

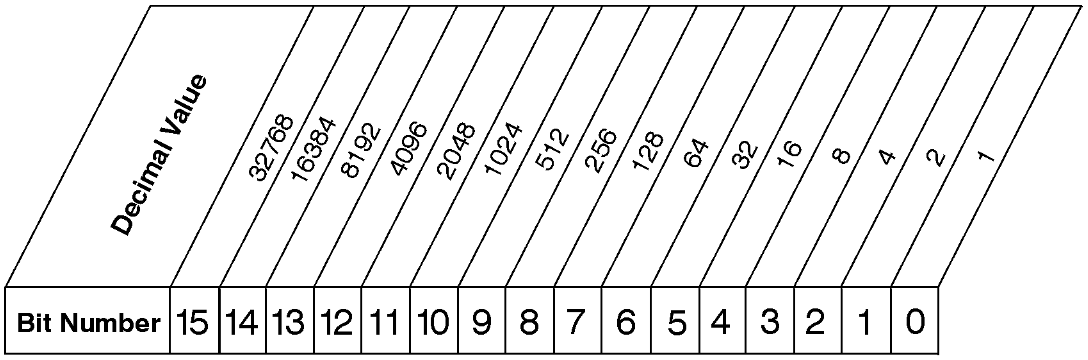
8.4 Using a Status Register

Each bit in a register is represented by a numerical value based on its location. See Figure 8-1 below. This number is sent with the command, to enable a particular bit. If you want to enable more than one bit, you would send the sum of all the bits that you are interested in.

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

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

Figure 8-1 Status Register Bit Values



NOTE Bit 15 is not used to report status.

8.5 Using the Service Request (SRQ) Method

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

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

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

8.5.1 Generating a Service Request

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

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

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

The SRQ process sets the GPIB SRQ line true. It also sets the status byte’s request service (RQS) bit to 1. Both actions are necessary to inform the controller that the instrument requires service. Setting the SRQ line only informs the controller that some device on the bus requires service. Setting the RQS bit allows the controller to determine which instrument requires service.

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

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

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

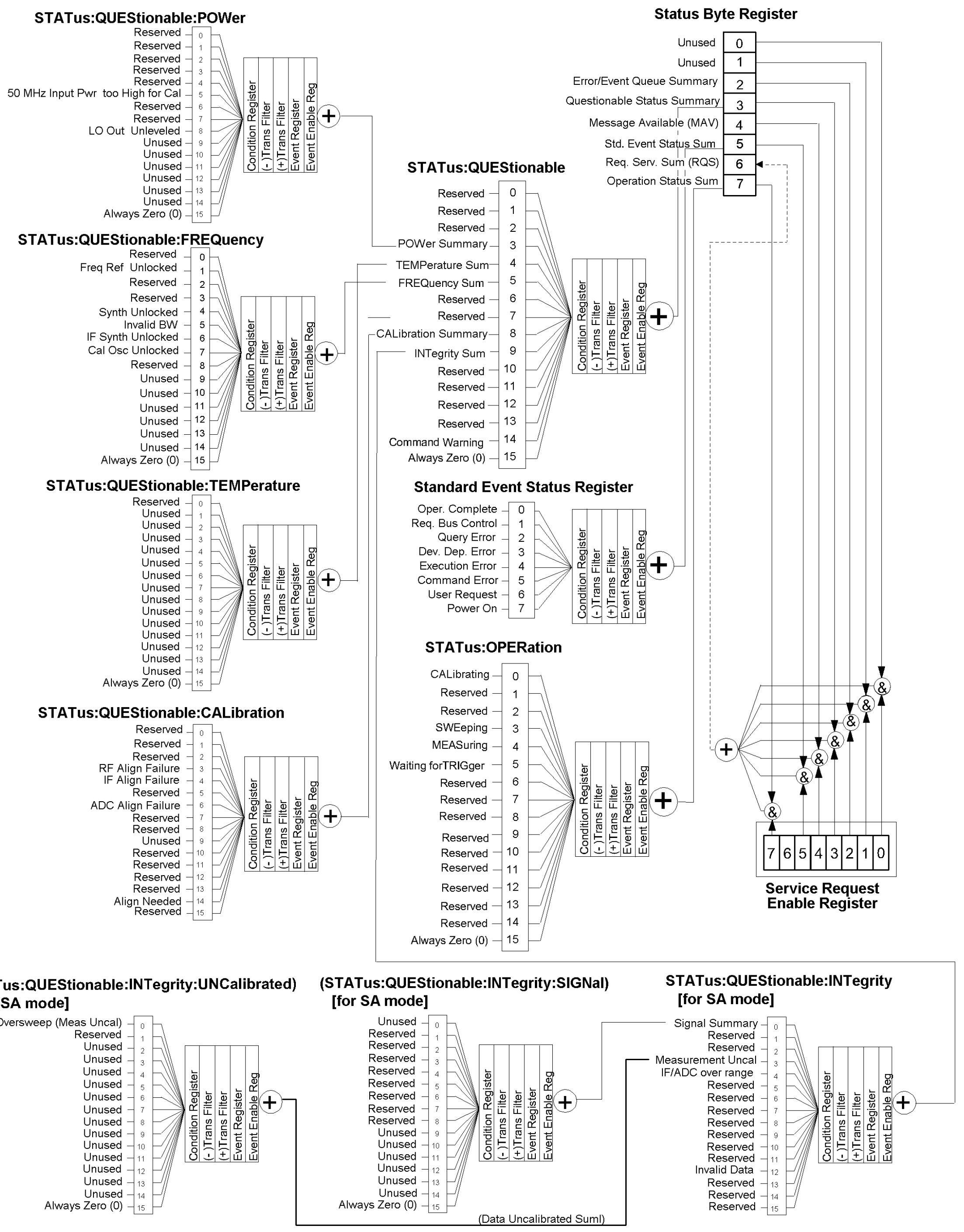
1. Set INITiate:CONTinuous off.
2. Set/enable the status registers.
3. Restart the measurement (send INIT).

8.5.2 Overall Status Register System

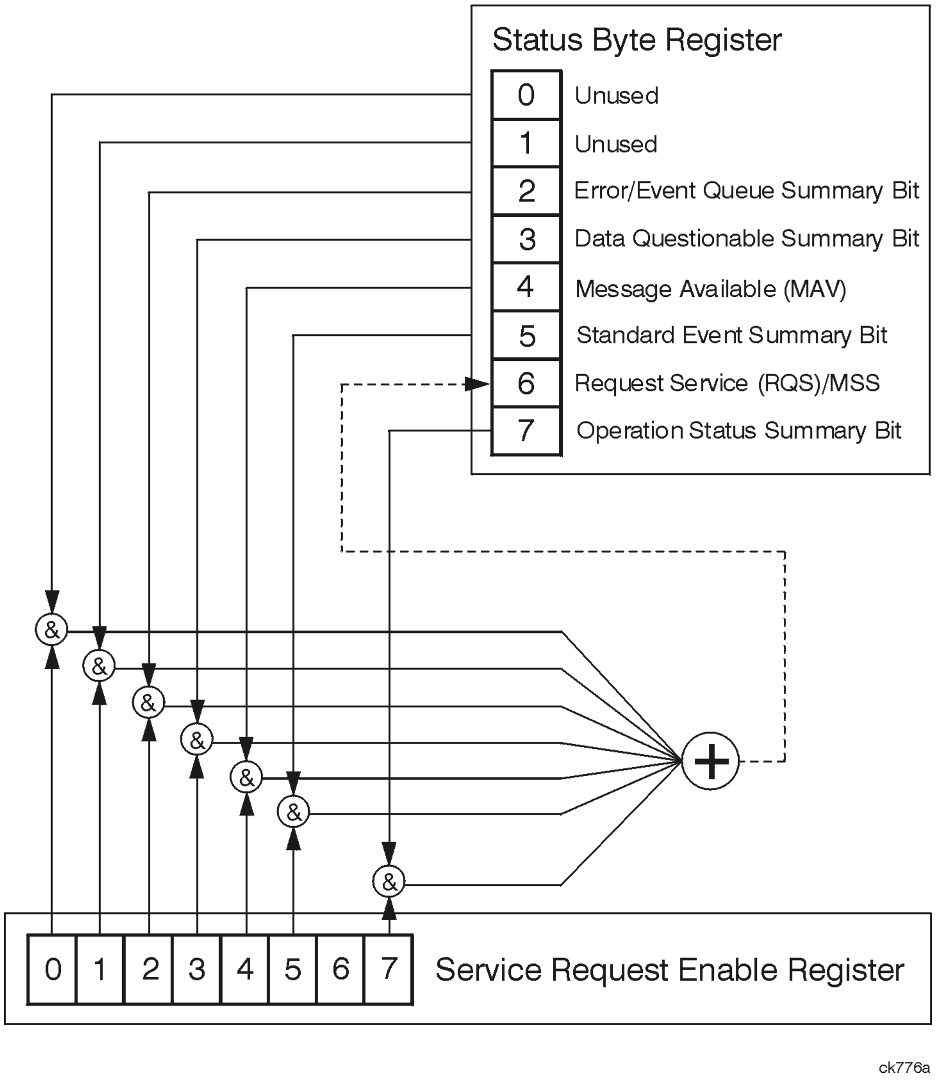
Preset Values

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

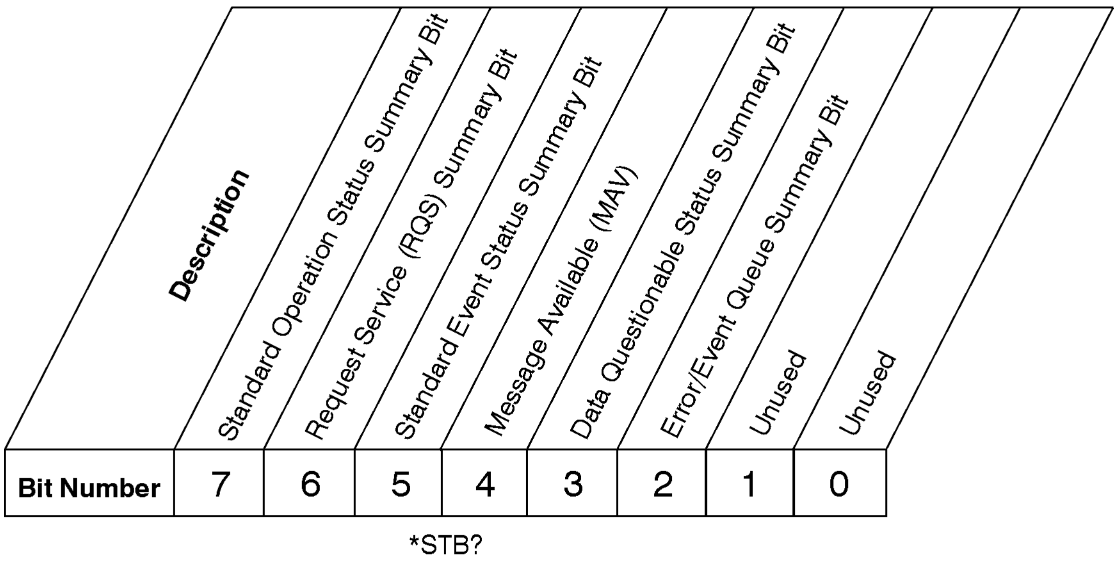
Unused: All unused bits = 0



8.6 Status Byte Register


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

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


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

Status Byte Register

ck725a

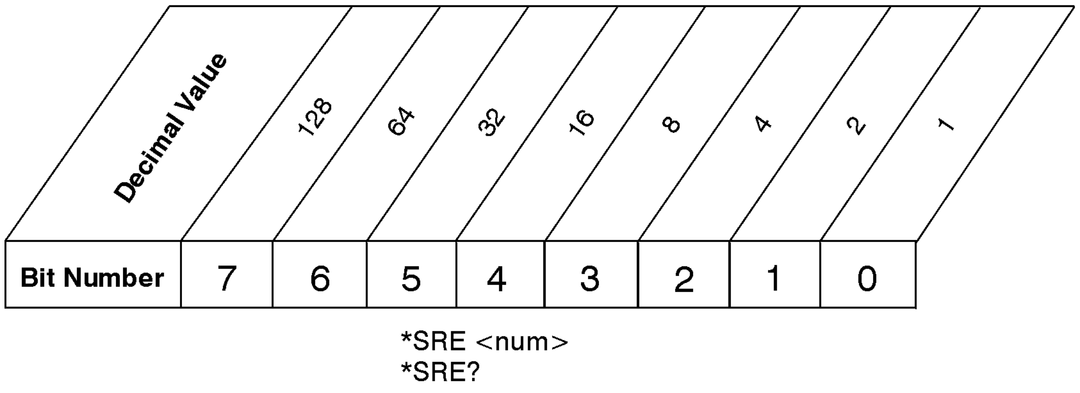
Bit	Description
0, 1	These bits are always set to 0.
2	A 1 in this bit position indicates that the SCPI error queue is not empty which means that it contains at least one error message.
3	A 1 in this bit position indicates that the data questionable summary bit has been set. The data questionable event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

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

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

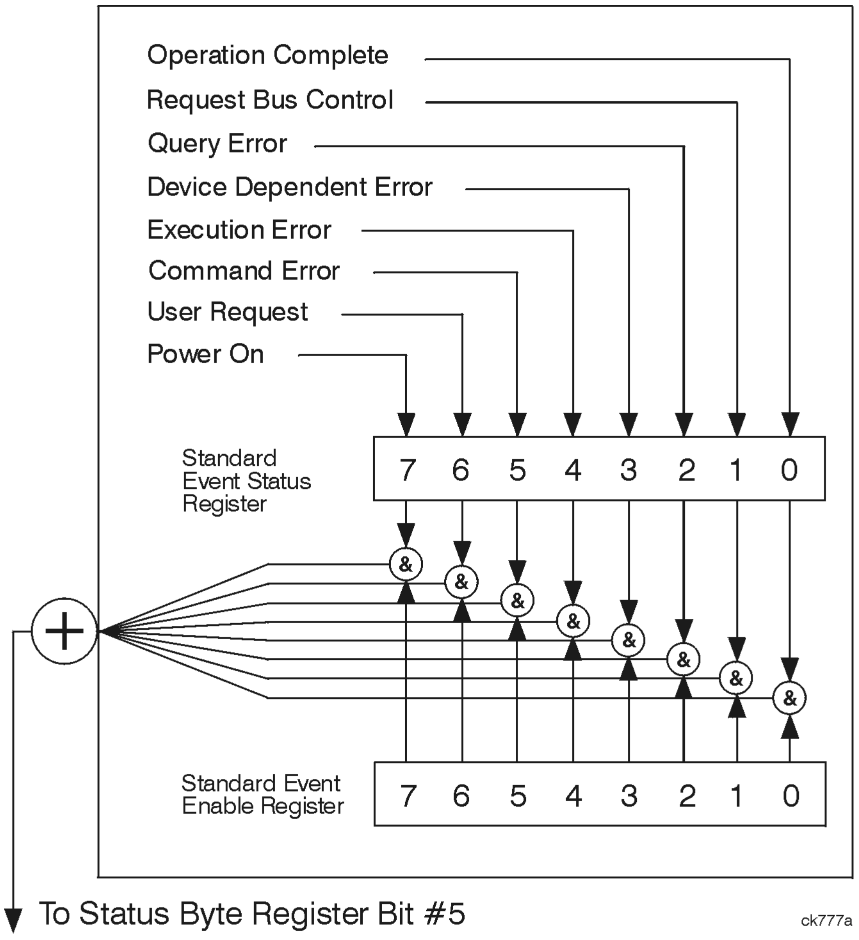
Send the `*SRE <number>` command where `<number>` is the sum of the decimal values of the bits you want to enable plus the decimal value of bit 6. For example, assume that you want to enable bit 7 so that whenever the standard operation status register summary bit is set to 1 it will trigger a service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

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

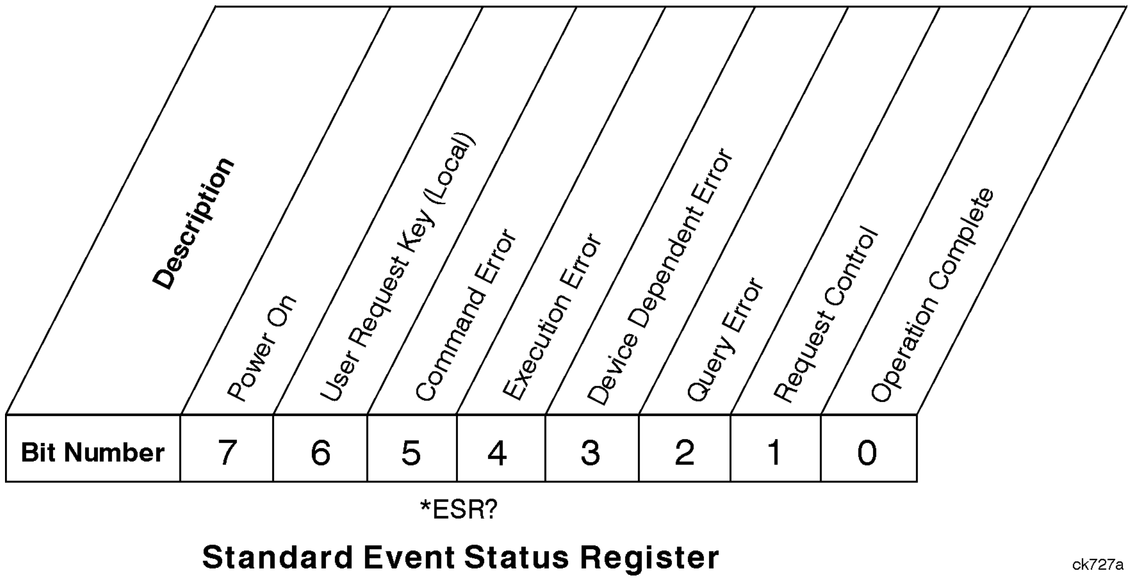


Service Request Enable Register

8.7 Standard Event Status Register



The standard event status register contains the following bits:

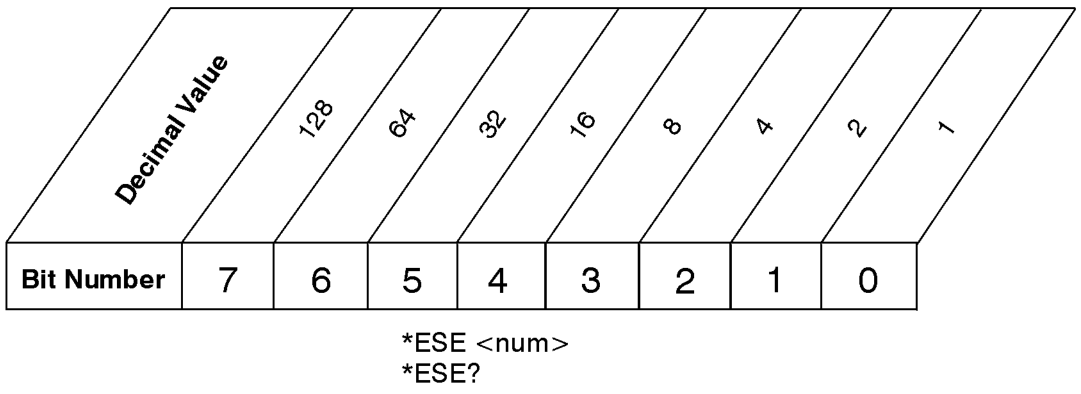


Bit	Description
0	A 1 in this bit position indicates that all pending operations were completed following execution of the *OPC command.
1	This bit is always set to 0. (The instrument does not request control.)
2	A 1 in this bit position indicates that a query error has occurred. Query errors have SCPI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

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

In addition to the standard event status register, the standard event status group also contains a standard event status enable register. This register lets you choose which bits in the standard event status register will set the summary bit (bit 5 of the status byte register) to 1. Send the `*ESE <number>` command where `<number>` is the sum of the decimal values of the bits you want to enable. For example, to enable bit 7 and bit 6 so that whenever either of those bits is set to 1, the standard event status summary bit of the status byte register will be set to 1, send the command `*ESE 192` (128 + 64). The command `*ESE?` returns the decimal value of the sum of the bits previously enabled with the `*ESE <number>` command.

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



Standard Event Status Enable Register

8.8 Operation and Questionable Status Registers

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

8.8.1 Operation Status Register

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

8.8.2 Questionable Status Register

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

8.9 SCPI Status Commands

IEEE (*) Commands	
Clear Status p283	
Standard Event Status Enable p283	
Standard Event Status Register Query p283	
Operation Complete Command p284	
Operation Complete Query p284	
Service Request Enable p284	
Read Status Byte Query p284	
Wait-to-Continue p284	

STATus Commands	
Operation Register p285	Operation Condition Query p285
	Operation Enable p285
	Operation Event Query p285
	Operation Negative Transition p286
	Operation Positive Transition p286
Preset the Status Bytes p286	
Status Register p286	Questionable Condition p286
	Questionable Enable p287
	Questionable Event Query p287
	Questionable Negative Transition p287
	Questionable Positive Transition p288
Calibration Register p288	Questionable Calibration Condition p288
	Questionable Calibration Enable p288
	Questionable Calibration Event Query p289
	Questionable Calibration Negative Transition p289
	Questionable Calibration Positive Transition p289
Frequency Register p290	Questionable Frequency Condition p290

STATus Commands	
	Questionable Integrity Enable p292
	Questionable Frequency Event Query p290
	Questionable Frequency Negative Transition p291
	Questionable Frequency Positive Transition p291
Integrity Register p291	Questionable Integrity Condition p291
	Questionable Integrity Event Query p292
	Questionable Integrity Event Query p292
	Questionable Integrity Negative Transition p292
	Questionable Integrity Positive Transition p293
Signal Integrity Register p293	Questionable Integrity Signal Condition p293
	Questionable Integrity Signal Enable p293
	Questionable Integrity Signal Event Query p294
	Questionable Integrity Signal Negative Transition p294
	Questionable Integrity Signal Positive Transition p294
Calibration Integrity Register p295	Questionable Calibration Integrity Condition p295
	Questionable Calibration Integrity Enable p295
	Questionable Calibration Integrity Event Query p295
	Questionable Calibration Integrity Negative Transition p295
	Questionable Calibration Integrity Positive Transition p296
Power Register p296	Questionable Power Condition p296
	Questionable Power Enable p296
	Questionable Power Event Query p297
	Questionable Power Negative Transition p297
	Questionable Power Positive Transition p297
Temperature Register p298	Questionable Temperature Condition p298
	Questionable Temperature Enable p298
	Questionable Temperature Event Query p298
	Questionable Temperature Negative Transition p299
	Questionable Temperature Positive Transition p299

8.10 Common IEEE Commands

These commands are specified in IEEE Standard 488.2-1992, *IEEE Standard Codes, Formats, Protocols and Common Commands for Use with ANSI/IEEE Std 488.1-1987*. New York, NY, 1992.

8.10.1 Clear Status

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.

Remote Command:

***CLS**

See also *STB?

Example:

*CLS

8.10.2 Standard Event Status Enable

Selects the desired bits from the standard event status enable register. This register monitors I/O errors and synchronization conditions such as operation complete, request control, query error, device dependent error, execution error, command error and power on. The selected bits are OR'd to become a summary bit (bit 5) in the status byte register which can be queried.

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

Minimum:	0
Maximum:	255

Remote Command:

***ESE <integer>**

***ESE?**

8.10.3 Standard Event Status Register Query

Queries and clears the standard event status event register. This is a destructive read.

Minimum:	0
Maximum:	255

Remote Command:

***ESR?**

8.10.4 Operation Complete Command

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

Remote Command:

***OPC**

8.10.5 Operation Complete Query

This 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.

Remote Command:

***OPC?**

8.10.6 Service Request Enable

This command sets the value of the service request enable register.

The query returns the value of the register.

Minimum:	0
Maximum:	255

Remote Command:

***SRE <integer>**

***SRE?**

8.10.7 Read Status Byte Query

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

Remote Command:

***STB?**

See also *CLS

8.10.8 Wait-to-Continue

***WAI**

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

8.11 STATus Subsystem Commands

8.11.1 Operation Register

8.11.1.1 Operation Condition Query

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.

Remote Command:

```
:STATus:OPERation:CONDition?
```

8.11.1.2 Operation Enable

This command determines what bits in the Operation Condition Register will set bits in the Operation Event register, which also sets the Operation Status Summary bit (bit 7) in the Status Byte Register. The variable <number> is the sum of the decimal values of the bits you want to enable.

Note: The preset condition is to have all bits in this enable register set to 0. To have any Operation Events reported to the Status Byte Register, 1 or more bits need to be set to 1. There is little reason to have any bits enabled for typical manufacturing tests. Enabling bits in this register would be of more value during test development.

Factory Preset and *RST: 0

Maximum Value: 32767

Minimum Value: 0

Remote Command:

```
:STATus:OPERation:ENABle <number>
:STATus:OPERation:ENABle?
```

8.11.1.3 Operation Event Query

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.

Note: The data in this register is latched until it is queried. Once queried, the data is cleared.

Remote Command:

```
:STATus:OPERation[:EVENt]?
```

8.11.1.4 Operation Negative Transition

This command determines what bits in the Operation Condition register will set the corresponding bit in the Operation Event register when that bit has a negative transition (1 to 0). The variable <number> is the sum of the decimal values of the bits that you want to enable.

Factory Preset and *RST: 0

Maximum Value: 32767

Minimum Value: 0

Remote Command:

```
:STATus:OPERation:NTRansition <number>
:STATus:OPERation:NTRansition?
```

8.11.1.5 Operation Positive Transition

This command determines what bits in the Operation Condition register will set the corresponding bit in the Operation Event register when that bit has a positive transition (0 to 1). The variable <number> is the sum of the decimal values of the bits that you want to enable.

Factory Preset and *RST: 32767 (all 1's)

Maximum Value: 32767

Minimum Value: 0

Remote Command:

```
:STATus:OPERation:PTRansition <number>
:STATus:OPERation:PTRansition?
```

8.11.2 Preset the Status Bytes

Sets bits in most of the enable and transition registers to their default state. It presets all the Transition Filters, Enable Registers, and the Error/Event Queue Enable. It has no effect on Event Registers, Error/Event QUEue, IEEE 488.2 ESE, and SRE Registers.

Remote Command:

```
:STATus:PRESet
```

8.11.3 Status Register

8.11.3.1 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.

Remote Command:

`:STATus:QUEStionable:CONDition?`

8.11.3.2 Questionable Enable

This command determines what bits in the Questionable Condition Register will set bits in the Questionable Event register, which also sets the Questionable Status Summary bit (bit3) in the Status Byte Register. The variable <number> is the sum of the decimal values of the bits you want to enable.

Note: The preset condition is to have all bits in this enable register set to 0. To have any Questionable Events reported to the Status Byte Register, 1 or more bits need to be set to 1. It is recommended that all bits be enabled in this register. The Status Byte Event Register should be queried after each measurement to check the Questionable Status Summary (bit 3). If it is equal to 1, there was some kind of condition during the test, that might make the test results invalid. If it is equal to 0, this indicates that no hardware problem, or measurement problem was detected by the analyzer.

Factory Preset and *RST:	0
Maximum Value:	32767
Minimum Value:	0

Remote Command:

`:STATus:QUEStionable:ENABle <number>`

`:STATus:QUEStionable:ENABle?`

8.11.3.3 Questionable Event Query

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.

Note: The data in this register is latched until it is queried. Once queried, the data is cleared.

Remote Command:

`:STATus:QUEStionable[:EVENt]?`

8.11.3.4 Questionable Negative Transition

This command determines what bits in the Questionable Condition register will set the corresponding bit in the Questionable Event register when that bit has a negative transition (1 to 0). The variable <number> is the sum of the decimal values of the bits that you want to enable.

Factory Preset and *RST: 0

Maximum Value: 32767

Minimum Value: 0

Remote Command:

```
:STATus:QUEStionable:NTRansition <number>
:STATus:QUEStionable:NTRansition?
```

8.11.3.5 Questionable Positive Transition

This command determines what bits in the Questionable Condition register will set the corresponding bit in the Questionable Event register when that bit has a 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)

Maximum Value: 32767

Minimum Value: 0

Remote Command:

```
:STATus:QUEStionable:PTRansition <number>
:STATus:QUEStionable:PTRansition?
```

8.11.4 Calibration Register

8.11.4.1 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.

Remote Command:

```
:STATus:QUEStionable:CALibration:CONDition?
```

8.11.4.2 Questionable Calibration Enable

This command determines what bits in the Questionable Calibration Condition Register will set bits in the Questionable Calibration Event register, which also sets the Calibration Summary bit (bit 8) in the Questionable Register. The variable <number> is the sum of the decimal values of the bits you want to enable.

Factory Preset and *RST: 32767 (all 1’s)

Maximum Value: 32767

Minimum Value: 0

Remote Command:

:STATus:QUEStionable:CALibration:ENABle <number>

:STATus:QUEStionable:CALibration:ENABle?

8.11.4.3 Questionable Calibration Event Query

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.

Note: The data in this register is latched until it is queried. Once queried, the data is cleared.

Remote Command:

:STATus:QUEStionable:CALibration[:EVENt]?

8.11.4.4 Questionable Calibration Negative Transition

This command determines what bits in the Questionable Calibration Condition register will set the corresponding bit in the Questionable Calibration Event register when that bit has a negative transition (1 to 0). The variable <number> is the sum of the decimal values of the bits that you want to enable.

Factory Preset and *RST: 0

Maximum Value: 32767

Minimum Value: 0

Remote Command:

:STATus:QUEStionable:CALibration:NTRansition <number>

:STATus:QUEStionable:CALibration:NTRansition?

8.11.4.5 Questionable Calibration Positive Transition

This command determines what bits in the Questionable Calibration Condition register will set the corresponding bit in the Questionable Calibration Event register when that bit has a 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)

Maximum Value: 32767

Minimum Value: 0

Remote Command:

```
:STATus:QUEStionable:CALibration:PTRansition <number>
:STATus:QUEStionable:CALibration:PTRansition?
```

8.11.5 Frequency Register

8.11.5.1 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.

Remote Command:

```
:STATus:QUEStionable:FREQuency:CONDition?
```

8.11.5.2 Questionable Frequency Enable

This command determines what bits in the Questionable Frequency Condition Register will set bits in the Questionable Frequency Event register, which also sets the Frequency Summary bit (bit 5) in the Questionable Register. The variable <number> is the sum of the decimal values of the bits you want to enable.

Factory Preset and *RST: 32767 (all 1's)

Maximum Value: 32767

Minimum Value: 0

Remote Command:

```
:STATus:QUEStionable:FREQuency:ENABle <number>
:STATus:QUEStionable:FREQuency:ENABle?
```

8.11.5.3 Questionable Frequency Event Query

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.

Note: The data in this register is latched until it is queried. Once queried, the data is cleared.

Remote Command:

```
:STATus:QUEStionable:FREQuency[:EVENt]?
```

8.11.5.4 Questionable Frequency Negative Transition

This command determines what bits in the Questionable Frequency Condition register will set the corresponding bit in the Questionable Frequency Event register when that bit has a negative transition (1 to 0). The variable <number> is the sum of the decimal values of the bits that you want to enable.

Factory Preset and *RST: 0

Maximum Value: 32767

Minimum Value: 0

Remote Command:

```
:STATus:QUEStionable:FREQuency:NTRansition <number>
:STATus:QUEStionable:FREQuency:NTRansition?
```

8.11.5.5 Questionable Frequency Positive Transition

This command determines what bits in the Questionable Frequency Condition register will set the corresponding bit in the Questionable Frequency Event register when that bit has a 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)

Maximum Value: 32767

Minimum Value: 0

Remote Command:

```
:STATus:QUEStionable:FREQuency:PTRansition <number>
:STATus:QUEStionable:FREQuency:PTRansition?
```

8.11.6 Integrity Register

8.11.6.1 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.

Remote Command:

```
:STATus:QUEStionable:INTegrity:CONDition?
```

8.11.6.2 Questionable Integrity Enable

This command determines what bits in the Questionable Integrity Condition Register will set bits in the Questionable Integrity Event register, which also sets the Integrity Summary bit (bit 9) in the Questionable Register. The variable <number> is the sum of the decimal values of the bits you want to enable.

Factory Preset and *RST: 32767 (all 1's)

Maximum Value: 32767

Minimum Value: 0

Remote Command:

```
:STATus:QUEStionable:INTegrity:ENABle <number>
:STATus:QUEStionable:INTegrity:ENABle?
```

8.11.6.3 Questionable Integrity Event Query

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.

Note: The data in this register is latched until it is queried. Once queried, the data is cleared.

Remote Command:

```
:STATus:QUEStionable:INTegrity[:EVENt]?
```

8.11.6.4 Questionable Integrity Negative Transition

This command determines what bits in the Questionable Integrity Condition register will set the corresponding bit in the Questionable Integrity Event register when that bit has a negative transition (1 to 0). The variable <number> is the sum of the decimal values of the bits that you want to enable.

Factory Preset and *RST: 0

Maximum Value: 32767

Minimum Value: 0

Remote Command:

```
:STATus:QUEStionable:INTegrity:NTRansition <number>
```

:STATus:QUEStionable:INTegrity:NTRansition?

8.11.6.5 Questionable Integrity Positive Transition

This command determines what bits in the Questionable Integrity Condition register will set the corresponding bit in the Questionable Integrity Event register when that bit has a 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)

Maximum Value: 32767

Minimum Value: 0

Remote Command:

:STATus:QUEStionable:INTegrity:PTRansition <number>

:STATus:QUEStionable:INTegrity:PTRansition?

8.11.7 Signal Integrity Register

8.11.7.1 Questionable Integrity Signal Condition

This query returns the decimal value of the sum of the bits in the Questionable Integrity Signal Condition register.

Note: The data in this register is continuously updated and reflects the current conditions.

Remote Command:

:STATus:QUEStionable:INTegrity:SIGNal:CONDition?

8.11.7.2 Questionable Integrity Signal Enable

This command determines what bits in the Questionable Integrity Signal Condition Register will set bits in the Questionable Integrity Signal Event register, which also sets the Integrity Summary bit (bit 9) in the Questionable Register. The variable <number> is the sum of the decimal values of the bits you want to enable.

Factory Preset and *RST: 32767 (all 1's)

Maximum Value: 32767

Minimum Value: 0

Remote Command:

:STATus:QUEStionable:INTegrity:SIGNal:ENABle <number>

:STATus:QUEStionable:INTegrity:SIGNal:ENABle?

8.11.7.3 Questionable Integrity Signal Event Query

This query returns the decimal value of the sum of the bits in the Questionable Integrity Signal Event register.

Note: The register requires that the equivalent PTR or NTR filters be set before a condition register bit can set a bit in the event register.

Note: The data in this register is latched until it is queried. Once queried, the data is cleared.

Remote Command:

```
:STATus:QUEStionable:INTegrity:SIGNal[:EVENt]?
```

8.11.7.4 Questionable Integrity Signal Negative Transition

This command determines what bits in the Questionable Integrity Signal Condition register will set the corresponding bit in the Questionable Integrity Signal Event register when that bit has a negative transition (1 to 0). The variable <number> is the sum of the decimal values of the bits that you want to enable.

Factory Preset and *RST: 0

Maximum Value: 32767

Minimum Value: 0

Remote Command:

```
:STATus:QUEStionable:INTegrity:SIGNal:PTRansition <number>
:STATus:QUEStionable:INTegrity:SIGNal:PTRansition?
```

8.11.7.5 Questionable Integrity Signal Positive Transition

This command determines what bits in the Questionable Integrity Signal Condition register will set the corresponding bit in the Questionable Integrity Signal Event register when that bit has a 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)

Maximum Value: 32767

Minimum Value: 0

Remote Command:

```
:STATus:QUEStionable:INTegrity:SIGNal:PTRansition <number>
:STATus:QUEStionable:INTegrity:SIGNal:PTRansition?
```

8.11.8 Calibration Integrity Register

8.11.8.1 Questionable Calibration Integrity Condition

This query returns the decimal value of the sum of the bits in the Questionable Calibration Integrity Condition register.

Note: The data in this register is continuously updated and reflects the current conditions.

Remote Command:

:STATus:QUEStionable:INTegrity:UNCalibrated:CONDition?

8.11.8.2 Questionable Calibration Integrity Enable

This command determines what bits in the Questionable Calibration Integrity Condition Register will set bits in the Questionable Integrity Signal Event register, which also sets the Integrity Summary bit (bit 9) in the Questionable Register. The variable <number> is the sum of the decimal values of the bits you want to enable.

Factory Preset and *RST: 32767 (all 1's)

Maximum Value: 32767

Minimum Value: 0

Remote Command:

:STATus:QUEStionable:INTegrity:UNCalibrated:ENABle <number>

:STATus:QUEStionable:INTegrity:UNCalibrated:ENABle?

8.11.8.3 Questionable Calibration Integrity Event Query

This query returns the decimal value of the sum of the bits in the Questionable Calibration 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.

Note: The data in this register is latched until it is queried. Once queried, the data is cleared.

Remote Command:

:STATus:QUEStionable:INTegrity:UNCalibrated[:EVENt]?

8.11.8.4 Questionable Calibration Integrity Negative Transition

This command determines what bits in the Questionable Calibration Integrity Condition register will set the corresponding bit in the Questionable Integrity Signal Event register when that bit has a negative transition (1 to 0). The variable <number> is the sum of the decimal values of the bits that you want to enable.

Factory Preset and *RST: 0

Maximum Value: 32767

Minimum Value: 0

Remote Command:

```
:STATus:QUEStionable:INTegrity:UNCalibrated:NTRansition <number>
:STATus:QUEStionable:INTegrity:UNCalibrated:NTRansition?
```

8.11.8.5 Questionable Calibration Integrity Positive Transition

This command determines what bits in the Questionable Calibration Integrity Condition register will set the corresponding bit in the Questionable Calibration 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)

Maximum Value: 32767

Minimum Value: 0

Remote Command:

```
:STATus:QUEStionable:INTegrity:UNCalibrated:PTRansition <number>
:STATus:QUEStionable:INTegrity:UNCalibrated:PTRansition?
```

8.11.9 Power Register

8.11.9.1 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.

Remote Command:

```
:STATus:QUEStionable:POWer:CONDition?
```

8.11.9.2 Questionable Power Enable

This command determines what bits in the Questionable Power Condition Register will set bits in the Questionable Power Event register, which also sets the Power Summary bit (bit 3) in the Questionable Register. The variable <number> is the sum of the decimal values of the bits you want to enable.

Factory Preset and *RST: 32767 (all 1's)

Maximum Value: 32767

Minimum Value: 0

Remote Command:

```
:STATus:QUEStionable:POWer:ENABle <number>
:STATus:QUEStionable:POWer:ENABle?
```

8.11.9.3 Questionable Power Event Query

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.

Note: The data in this register is latched until it is queried. Once queried, the data is cleared.

Remote Command:

```
:STATus:QUEStionable:POWer[:EVENt]?
```

8.11.9.4 Questionable Power Negative Transition

This command determines what bits in the Questionable Power Condition register will set the corresponding bit in the Questionable Power Event register when that bit has a negative transition (1 to 0). The variable <number> is the sum of the decimal values of the bits that you want to enable.

Factory Preset and *RST: 0

Maximum Value: 32767

Minimum Value: 0

Remote Command:

```
:STATus:QUEStionable:POWer:NTRansition <number>
:STATus:QUEStionable:POWer:NTRansition?
```

8.11.9.5 Questionable Power Positive Transition

This command determines what bits in the Questionable Power Condition register will set the corresponding bit in the Questionable Power Event register when that bit has a 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)

Maximum Value: 32767

Minimum Value: 0

Remote Command:

```
:STATus:QUEStionable:POWer:PTRansition <number>
:STATus:QUEStionable:POWer:PTRansition?
```

8.11.10 Temperature Register

8.11.10.1 Questionable Temperature Condition

This query returns the decimal value of the sum of the bits in the Questionable Temperature Condition register.

Note: The data in this register is continuously updated and reflects the current conditions.

Remote Command:

```
:STATus:QUEStionable:TEMPerature:CONDition?
```

8.11.10.2 Questionable Temperature Enable

This command determines what bits in the Questionable Temperature Condition Register will set bits in the Questionable Temperature Event register, which also sets the Temperature Summary bit (bit 4) in the Questionable Register. The variable <number> is the sum of the decimal values of the bits you want to enable.

Factory Preset and *RST: 32767 (all 1's)

Maximum Value: 32767

Minimum Value: 0

Remote Command:

```
:STATus:QUEStionable:TEMPerature:ENABle <number>
:STATus:QUEStionable:TEMPerature:ENABle?
```

8.11.10.3 Questionable Temperature Event Query

This query returns the decimal value of the sum of the bits in the Questionable Temperature Event register.

Note: The register requires that the equivalent PTR or NTR filters be set before a condition register bit can set a bit in the event register.

Note: The data in this register is latched until it is queried. Once queried, the data is cleared.

Remote Command:

```
:STATus:QUEStionable:TEMPerature[:EVENt]?
```

8.11.10.4 Questionable Temperature Negative Transition

This command determines what bits in the Questionable Temperature Condition register will set the corresponding bit in the Questionable Temperature Event register when that bit has a negative transition (1 to 0). The variable <number> is the sum of the decimal values of the bits that you want to enable.

Factory Preset and *RST: 0

Maximum Value: 32767

Minimum Value: 0

Remote Command:

```
:STATus:QUEStionable:TEMPerature:NTRansition <number>
:STATus:QUEStionable:TEMPerature:NTRansition?
```

8.11.10.5 Questionable Temperature Positive Transition

This command determines what bits in the Questionable Temperature Condition register will set the corresponding bit in the Questionable Temperature Event register when that bit has a 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)

Maximum Value: 32767

Minimum Value: 0

Remote Command:

```
:STATus:QUEStionable:TEMPerature:PTRansition <number>
:STATus:QUEStionable:TEMPerature:PTRansition?
```

Index

Index

Index

Index

Index

Index