Measurement Guide and Programming Examples

Agilent Technologies ESA Series Spectrum Analyzers

This guide documents firmware revision A.09.xx

This manual provides documentation for the following instruments:

Agilent Technologies ESA-E Series

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

and

Agilent Technologies ESA-L Series

E4411B (9 kHz- 1.5 GHz)
E4403B (9 kHz - 3.0 GHz)
E4408B (9 kHz - 26.5 GHz)

Agilent Technologies

Manufacturing Part Number: E4401-90425
Supersedes: E4401-90403 and E4401-90407
Printed in USA
June 2002

Notice

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

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

Safety Information

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

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

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

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

The instruction documentation symbol. The product is marked with this symbol when it is necessary for the user to refer to the instructions in the documentation.

This symbol is used to mark the on position of the power line switch.

This symbol is used to mark the standby position of the power line switch.

This symbol indicates that the input power required is AC.

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

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

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

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

Warranty

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

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

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

LIMITATION OF WARRANTY

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

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

Should Agilent have a negotiated contract with the User and should any of the contract terms conflict with these terms, the contract terms shall control.

EXCLUSIVE REMEDIES

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

Where to Find the Latest Information

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

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

Contents

1 Making Basic Measurements

What is in This Chapter

This chapter demonstrates basic analyzer measurements with examples of typical measurements; each measurement focuses on different functions. The measurement procedures covered in this chapter are listed below.

To find descriptions of specific analyzer functions, refer to the *Agilent Technologies ESA Series Spectrum Analyzers User's Guide.*

Test Equipment

Test Equipment	Specifications	Recommended Model
Signal Sources		
Signal Generator (2)	0.25 MHz to 4.0 GHz Ext Ref Input	E4433B or E443XB series
Adapters		
Type-N (m) to BNC (f) (3)		1250-0780
Termination, 50 Ω Type-N (m)		908A
Cables		
(3) BNC, 122-cm (48-in)		10503A
Miscellaneous		
Directional Bridge		86205A
Bandpass Filter	Center Frequency: 200 MHz Bandwidth: 10 MHz	
Lowpass Filter (2)	Cutoff Frequency: 300 MHz	0955-0455
RF Antenna		08920-61060

Comparing Signals

Using the analyzer, you can easily compare frequency and amplitude differences between signals, such as radio or television signal spectra. The analyzer delta marker function lets you compare two signals when both appear on the screen at one time or when only one appears on the screen.

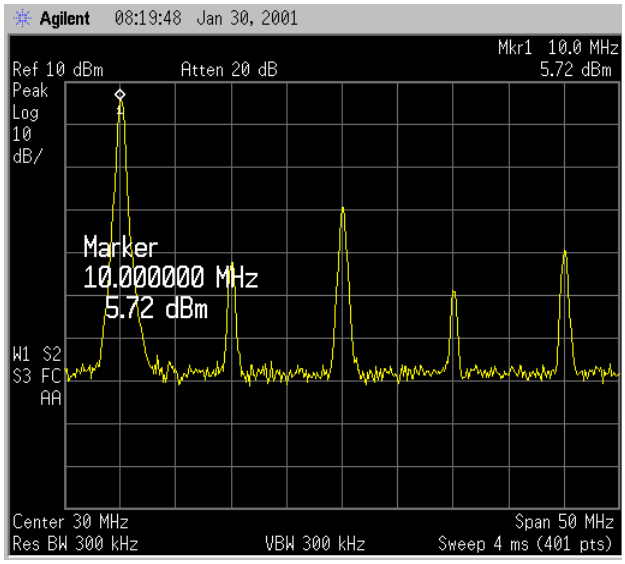
Signal Comparison Example 1:

Measure the differences between two signals on the same display screen.

1. Perform a factory preset by pressing **Preset**, **Factory Preset** (if present).
2. Connect the 10 MHz REF OUT from the rear panel to the front-panel INPUT.
3. Set the center frequency to 30 MHz by pressing **FREQUENCY**, **Center Freq**, 30, **MHz**.
4. Set the span to 50 MHz by pressing **SPAN, Span**, 50, **MHz**.
5. Set the reference level to 10 dBm by pressing **AMPLITUDE**, **Ref Level**, 10, **dBm**.

   The 10 MHz reference signal appears on the display.
6. Press **Peak Search** to place a marker at the highest peak on the display. (The **Next Pk Right** and **Next Pk Left** softkeys are available to move the marker from peak to peak.) The marker should be on the 10 MHz reference signal. See Figure 1-1.

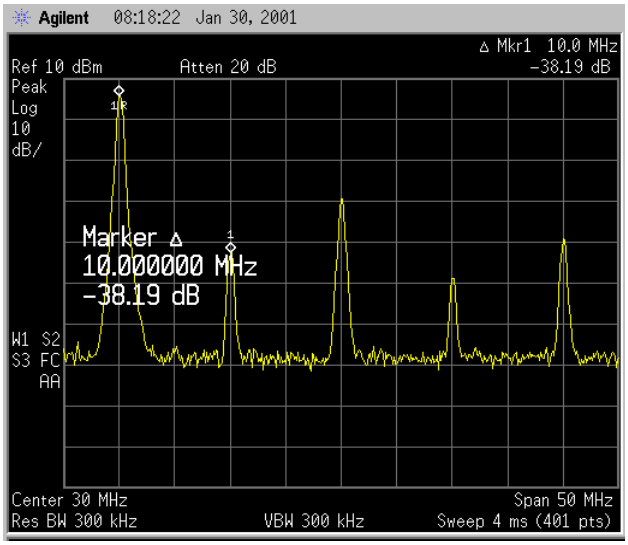
Figure 1-1 Placing a Marker on the 10 MHz Signal



7. Press **Marker**, **Delta**, to activate a second marker at the position of the first marker.
8. Move the second marker to another signal peak using the front-panel knob, or by pressing **Peak Search** and then either **Next Pk Right** or **Next Pk Left**. Next peak right is shown in Figure 1-2.

   The amplitude and frequency difference between the markers is displayed in the active function block and in the upper right corner of the screen. See Figure 1-2.
9. The resolution of the marker readings can be increased by turning on the frequency count function. For more information refer to "Making Better Frequency Measurements" on page 23.
10. Press **Marker**, **Off** to turn the markers off.

Figure 1-2 Using the Marker Delta Function



Signal Comparison Example 2:

Measure the frequency and amplitude difference between two signals that do not appear on the screen at one time. (This technique is useful for harmonic distortion tests when narrow span and narrow bandwidth are necessary to measure the low level harmonics.)

1. Perform a factory preset by pressing **Preset**, **Factory Preset** (if present).
2. Connect the 10 MHz REF OUT from the rear panel to the front-panel INPUT.
3. Set the center frequency to 10 MHz by pressing **FREQUENCY**, **Center Freq**, 10, **MHz**.
4. Set the span to 5 MHz by pressing **SPAN,** 5, **MHz**.
5. Set the reference level to 10 dBm by pressing **AMPLITUDE**, **Ref Level**, 10, **dBm**.

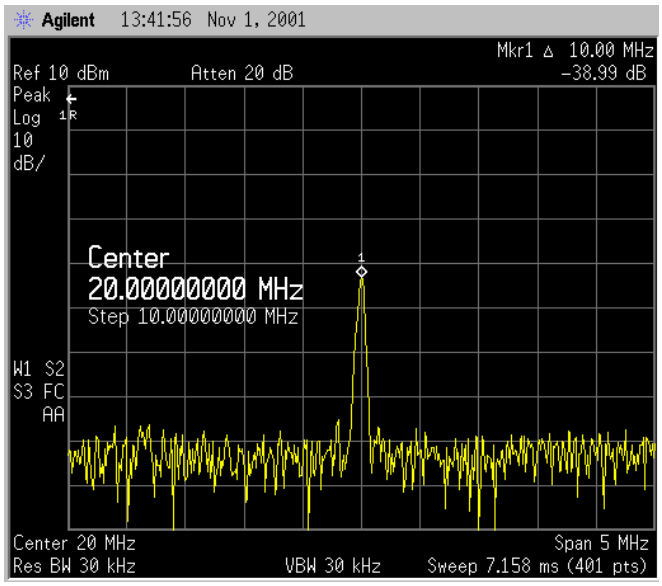
   The 10 MHz reference signal appears on the display.
6. Press **Peak Search** to place a marker on the peak.
7. Press **Marker**→, **Mkr**→**CF Step** to set the center frequency step size equal to the frequency of the fundamental signal.
8. Press **Marker**, **Delta** to anchor the position of the first marker and activate a second marker.
9. Press **FREQUENCY**, **Center Freq**, and the (↑) key to increase the center frequency by 10 MHz. The first marker moves to the left edge of the screen, at the amplitude of the first signal peak. See Figure 1-3.

10. Press **Peak Search** to place the second marker on the highest signal with the new center frequency setting. See Figure 1-3.

    The annotation in the upper right corner of the screen indicates the amplitude and frequency difference between the two markers.

11. To turn the markers off, press **Marker**, **Off**.

Figure 1-3 Frequency and Amplitude Difference Between Signals



Resolving Signals of Equal Amplitude

Two equal-amplitude input signals that are close in frequency can appear as a single signal trace on the analyzer display. Responding to a single-frequency signal, a swept-tuned analyzer traces out the shape of the selected internal IF (intermediate frequency) filter. As you change the filter bandwidth, you change the width of the displayed response. If a wide filter is used and two equal-amplitude input signals are close enough in frequency, then the two signals will appear as one signal. If a narrow enough filter is used, the two input signals can be discriminated and will appear as separate peaks. Thus, signal resolution is determined by the IF filters inside the analyzer.

The bandwidth of the IF filter tells us how close together equal amplitude signals can be and still be distinguished from each other. The resolution bandwidth function selects an IF filter setting for a measurement. Typically, resolution bandwidth is defined as the 3 dB bandwidth of the filter. However, resolution bandwidth may also be defined as the 6 dB or impulse bandwidth of the filter.

Generally, to resolve two signals of equal amplitude, the resolution bandwidth must be less than or equal to the frequency separation of the two signals. If the bandwidth is equal to the separation and the video bandwidth is less than the resolution bandwidth, a dip of approximately 3 dB is seen between the peaks of the two equal signals, and it is clear that more than one signal is present. See Figure 1-7.

In order to keep the analyzer measurement calibrated, sweep time is automatically set to a value that is inversely proportional to the square of the resolution bandwidth ($1/BW^2$ for resolution bandwidths $\geq$ 1 kHz). So, if the resolution bandwidth is reduced by a factor of 10, the sweep time is increased by a factor of 100 when sweep time and bandwidth settings are coupled. Sweep time is also a function of the type of detection selected (peak detection is faster than sample or average detection). For the shortest measurement times, use the widest resolution bandwidth that still permits discrimination of all desired signals. Sweeptime is also a function of which Detector is in use, Peak detector sweeps more quickly than Sample or Average detector. The analyzer allows you to select from 1 kHz to 3 MHz resolution bandwidths in a 1, 3, 10 sequence and select a 5 MHz resolution bandwidth. In addition you can select the CISPR bandwidths (9 kHz, and 120 kHz) for maximum measurement flexibility.

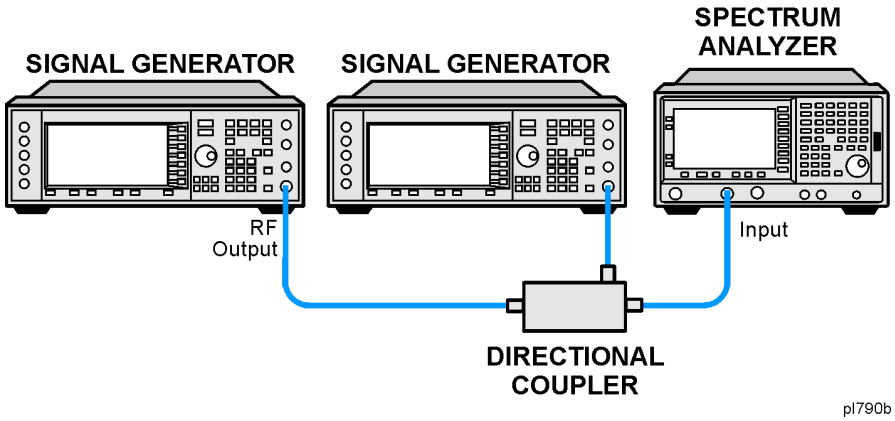
Option 1DR adds narrower resolution bandwidths, from 10 Hz to 300 Hz, in a 1-3-10 sequence and 200 Hz CISPR bandwidth. These bandwidths are digitally implemented and have a much narrower shape factor than the wider, analog resolution bandwidths. Also, the auto coupled sweep times when using the digital resolution bandwidths are much faster than analog bandwidths of the same width. For analyzers with Option 1DR, firmware revision A.08.00 and greater, and Option 1D5 which adds a high-stability frequency precision reference to the analyzer, resolution bandwidths of 1 Hz and 3 Hz are also available.

Resolving Signals Example:

Resolve two signals of equal amplitude with a frequency separation of 100 kHz.

1. Connect two sources to the analyzer input as shown in Figure 1-4.

Figure 1-4 Setup for Obtaining Two Signals



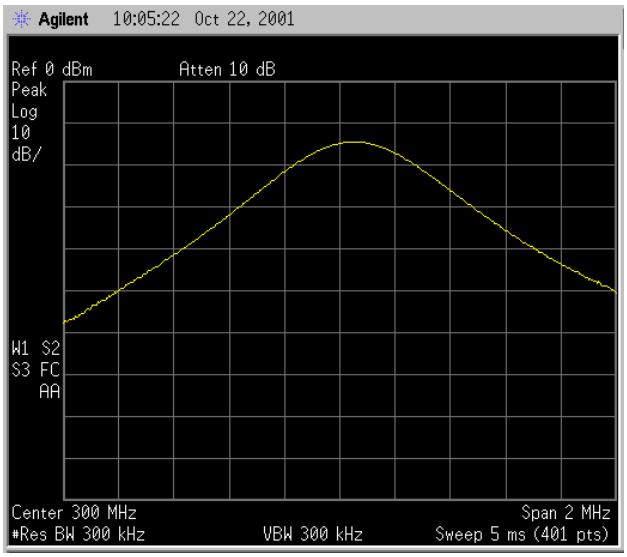
2. Set one source to 300 MHz. Set the frequency of the other source to 300.1 MHz. The amplitude of both signals should be approximately –20 dBm at the output of the bridge.
3. Set the analyzer as follows:
   a. Press **Preset**, **Factory Preset** (if present).
   b. Set the center frequency to 300 MHz by pressing **FREQUENCY, Center Freq**, 300, **MHz**.
   c. Set the span to 2 MHz by pressing **SPAN, Span**, 2, **MHz**.
   d. Set the resolution bandwidth to 300 kHz by pressing **BW/Avg**, **Res BW**, 300, **kHz**.

      A single signal peak is visible. See Figure 1-5.

NOTE If the signal peak is not present on the display, do the following:

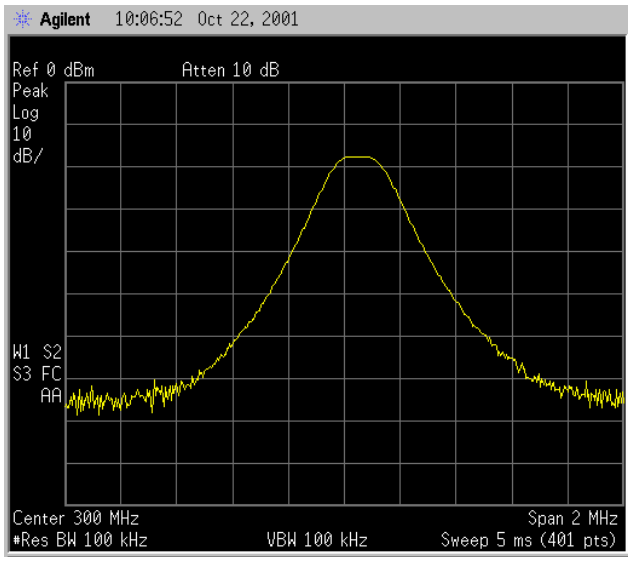
1. Increase the span to 20 MHz by pressing **SPAN, Span**, 20, **MHz**. The signal should be visible.
2. Press **Peak Search**, **FREQUENCY**, **Signal Track** (On)
3. Press **SPAN**, 2, **MHz** to bring the signal to center screen.
4. Press **FREQUENCY**, **Signal Track** (Off)

Figure 1-5 Unresolved Signals of Equal Amplitude



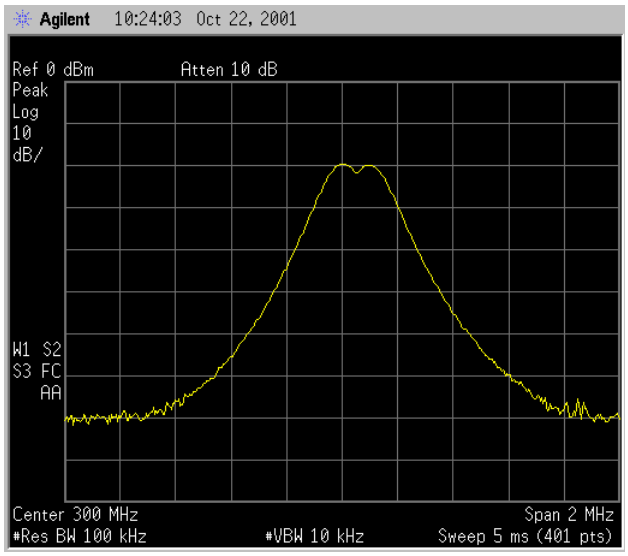
4. Since the resolution bandwidth must be less than or equal to the frequency separation of the two signals, a resolution bandwidth of 100 kHz must be used. Change the resolution bandwidth to 100 kHz by pressing **BW/Avg, Res BW**, 100, **kHz**. The peak of the signal has become flattened indicating that two signals may be present as shown in Figure 1-6. Use the knob or step keys to further reduce the resolution bandwidth and better resolve the signals.

Figure 1-6 Resolving Signals of Equal Amplitude Before Reducing the Video Bandwidth



Decrease the video bandwidth to 10 kHz, by pressing Video BW, 10, kHz. Two signals are now visible as shown in Figure 1-7. Use the front-panel knob or step keys to further reduce the resolution bandwidth and better resolve the signals.

Figure 1-7 Resolving Signals of Equal Amplitude After Reducing the Video Bandwidth



As the resolution bandwidth is decreased, resolution of the individual signals is improved and the sweep time is increased. For fastest measurement times, use the widest possible resolution bandwidth. Under factory preset conditions, the resolution bandwidth is "coupled" (or linked) to the span.

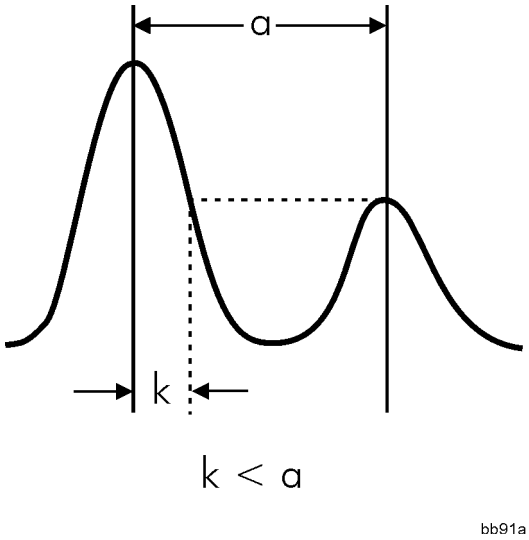
Since the resolution bandwidth has been changed from the coupled value, a # mark appears next to Res BW in the lower-left corner of the screen, indicating that the resolution bandwidth is uncoupled. (For more information on coupling, refer to the Auto Couple key description in the Agilent Technologies ESA Spectrum Analyzers User's Guide.)

NOTE To resolve two signals of equal amplitude with a frequency separation of 200 kHz, the resolution bandwidth must be less than the signal separation, and resolution of 100 kHz must be used. The next larger filter, 300 kHz, would exceed the 200 kHz separation and would not resolve the signals.

Resolving Small Signals Hidden by Large Signals

When dealing with the resolution of signals that are close together and not equal in amplitude, you must consider the shape of the IF filter of the analyzer, as well as its 3 dB bandwidth. (See "Resolving Signals of Equal Amplitude" on page 14 for more information.) The shape of a filter is defined by the selectivity, which is the ratio of the 60 dB bandwidth to the 3 dB bandwidth. (Generally, the IF filters in this analyzer have shape factors of 15:1 or less for resolution bandwidths ≥ 1 kHz and 5:1 or less for resolution bandwidths ≤ 300 Hz). If a small signal is too close to a larger signal, the smaller signal can be hidden by the skirt of the larger signal. To view the smaller signal, you must select a resolution bandwidth such that the separation between the two signals (*a*) is greater than half the filter width of the larger signal (k) measured at the amplitude level of the smaller signal. See Figure 1-8.

Figure 1-8 Resolution Bandwidth Requirements for Resolving Small Signals

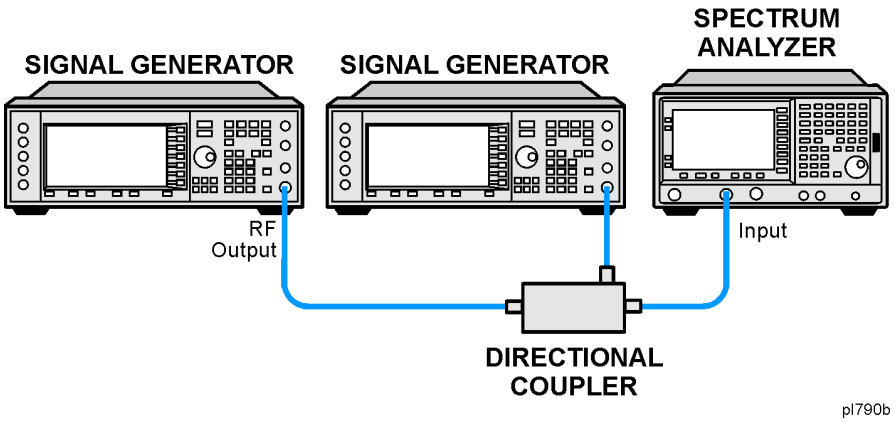


Resolving Signals Example:

Resolve two input signals with a frequency separation of 155 kHz and an amplitude separation of 60 dB.

1. Connect two sources to the analyzer input as shown in Figure 1-9.

Figure 1-9 Setup for Obtaining Two Signals



2. Set one source to 300 MHz at –10 dBm.
3. Set the second source to 300.155 MHz, so that the signal is 155 kHz higher than the first signal. Set the amplitude of the signal to –70 dBm (60 dB below the first signal).
4. Set the analyzer as follows:
   a. Press **Preset**, **Factory Preset** (if present).
   b. Set the center frequency to 300 MHz by pressing **FREQUENCY, Center Freq**, 300, **MHz**.
   c. Set the span to 2 MHz by pressing **SPAN, Span**, 2, **MHz**.

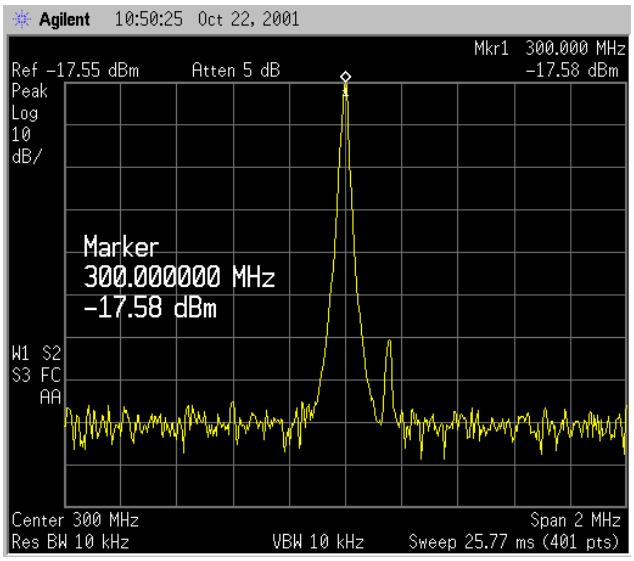
NOTE If the signal peak is not present on the display, do the following:

1. Increase the span to 20 MHz by pressing **SPAN, Span**, 20, **MHz**. The signal should now be visible.
2. Press **Peak Search**, **FREQUENCY**, **Signal Track** (On)
3. Press **SPAN**, 2, **MHz** to bring the signal to center screen.
4. Press **FREQUENCY**, **Signal Track** (Off)

5. Set the 300 MHz signal to the reference level by pressing **Mkr** → and then **Mkr** → **Ref Lvl**.

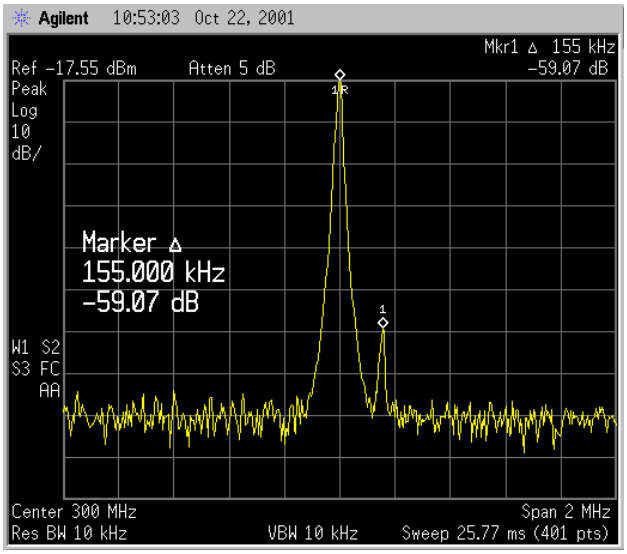
   If a 10 kHz filter with a typical shape factor of 15:1 is used, the filter will have a bandwidth of 150 kHz at the 60 dB point. The half-bandwidth (75 kHz) is narrower than the frequency separation, so the input signals will be resolved. See Figure 1-10.

Figure 1-10 Signal Resolution with a 10 kHz Resolution Bandwidth



6. Place a marker on the smaller signal by pressing **Marker**, **Delta**, **Peak Search**, **Next Pk Right**. Refer to Figure 1-11.

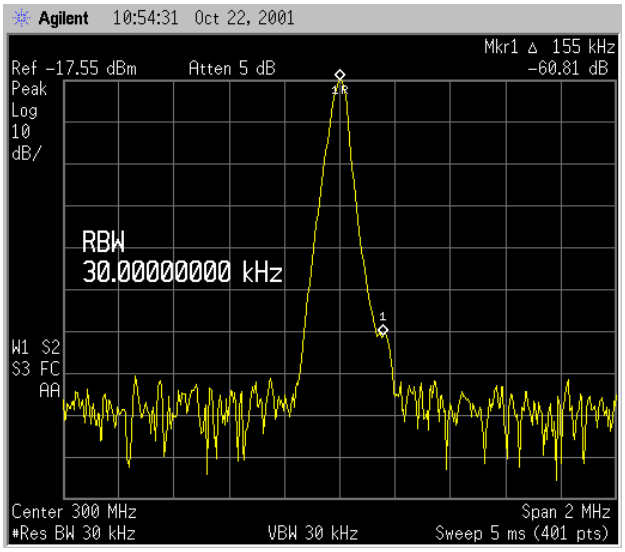
Figure 1-11 Signal Resolution with a 10 kHz Resolution Bandwidth



7. Set the resolution bandwidth to 30 kHz by pressing **BW/Avg**, **Res BW**, 30, **kHz**.

When a 30 kHz filter is used, the 60 dB bandwidth could be as wide as 450 kHz. Since the half-bandwidth (225 kHz) is wider than the frequency separation, the signals most likely will not be resolved. See Figure 1-12. (In this example, we used the 60 dB bandwidth value. To determine resolution capability for intermediate values of amplitude level differences, assume the filter skirts between the 3 dB and 60 dB points are approximately straight.)

Figure 1-12 Signal Resolution with a 30 kHz Resolution Bandwidth



Making Better Frequency Measurements

A built-in frequency counter increases the resolution and accuracy of the frequency readout. When using this function, if the ratio of the resolution bandwidth to the span is too small (less than 0.002), the `Marker Count: Widen Res BW` message appears on the display. It indicates that the resolution bandwidth is too narrow.

Better Frequency Measurement Example:

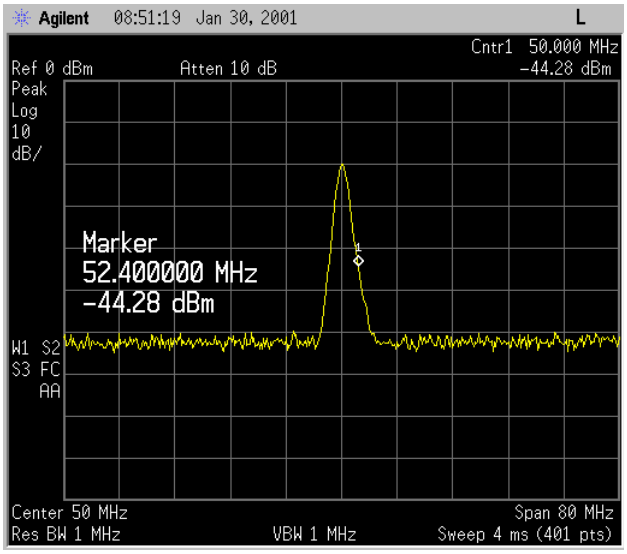
Increase the resolution and accuracy of the frequency readout on the signal of interest.

1. Perform a factory preset by pressing **Preset**, **Factory Preset** (if present).
2. Turn on the internal 50 MHz amplitude reference signal of the analyzer as follows:
   - For the E4401B and E4411B, use the internal 50 MHz amplitude reference signal of the analyzer as the signal being measured. Press **Input/Output**, **Amptd Ref** (On).
   - For all other models connect a cable between the front-panel AMPTD REF OUT to the analyzer INPUT, then press **Input/Output**, **Amptd Ref Out** (On).
3. Set the center frequency to 50 MHz by pressing **FREQUENCY**, **Center Freq**, 50, **MHz**.
4. Set the span to 80 MHz by pressing **SPAN**, **Span**, 80, **MHz**.
5. Press **Freq Count**. (Note that **Marker Count** has On underlined turning the frequency counter on.) The frequency and amplitude of the marker and the word `Marker` will appear in the active function area (this is not the counted result). The counted result appears in the upper-right corner of the display.
6. Move the marker, with the front-panel knob, half-way down the skirt of the signal response. Notice that the readout in the active frequency function changes while the counted frequency result (upper-right corner of display) does not. See Figure 1-13. To get an accurate count, you do not need to place the marker at the exact peak of the signal response.

NOTE Marker count properly functions only on CW signals or discrete spectral components. The marker must be > 26 dB above the noise.

7. Increase the counter resolution by pressing **Resolution** and then entering the desired resolution using the step keys or the numbers keypad. For example, press 10, **Hz**. The marker counter readout is in the upper-right corner of the screen. The resolution can be set from 1 Hz to 100 kHz.
8. The marker counter remains on until turned off. Turn off the marker counter by pressing **Freq Count**, then **Marker Count** (Off). **Marker**, **Off** also turns the marker counter off.

Figure 1-13 Using Marker Counter



Decreasing the Frequency Span Around the Signal

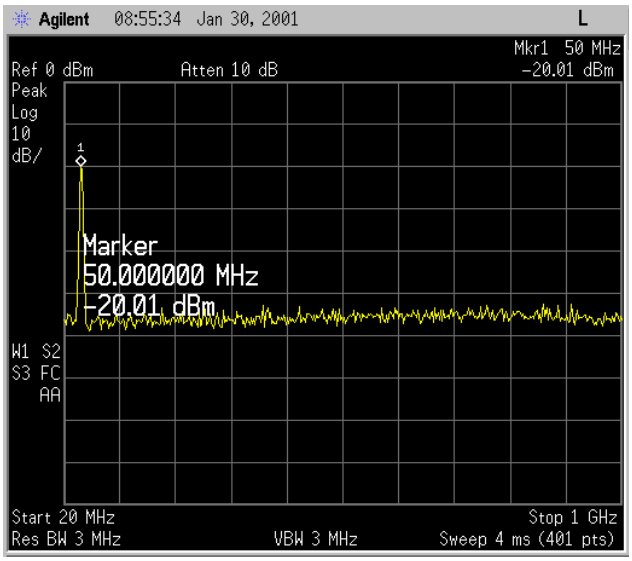
Using the analyzer signal track function, you can quickly decrease the span while keeping the signal at center frequency. This is a fast way to take a closer look at the area around the signal to identify signals that would otherwise not be resolved.

Decreasing the Frequency Span Example:

Examine a signal in a 200 kHz span.

1. Perform a factory preset by pressing **Preset**, **Factory Preset** (if present).
2. Turn on the internal 50 MHz amplitude reference signal of the analyzer as follows:
   - For the E4401B and E4411B, use the internal 50 MHz amplitude reference signal of the analyzer as the signal being measured. Press **Input/Output**, **Amptd Ref** (On).
   - For all other models connect a cable between the front-panel AMPTD REF OUT to the analyzer INPUT, then press **Input/Output**, **Amptd Ref Out** (On).
3. Set the start frequency to 20 MHz by pressing **FREQUENCY**, **Start Freq**, 20, **MHz**.
4. Set the stop frequency to 1 GHz by pressing **FREQUENCY**, **Stop Freq**, 1, **GHz**.
5. Press **Peak Search** to place a marker at the peak. See Figure 1-14.

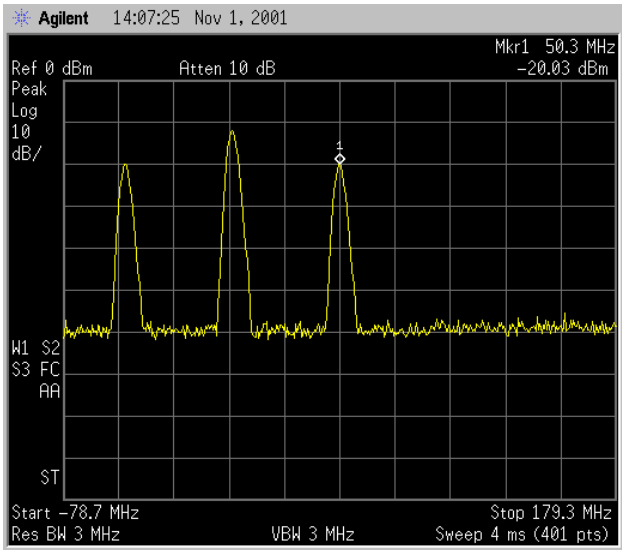
Figure 1-14 Detected Signal



6. Turn on the frequency tracking function by press **FREQUENCY** and **Signal Track** and the signal will move to the center of the screen, if it is not already positioned there. See figure Figure 1-15. (Note that the marker must be on the signal before turning signal track on.)

   Because the signal track function automatically maintains the signal at the center of the screen, you can reduce the span quickly for a closer look. If the signal drifts off of the screen as you decrease the span, use a wider frequency span. (You can also use **Span Zoom**, in the **SPAN** menu, as a quick way to perform the **Peak Search**, **FREQUENCY**, **Signal Track**, **SPAN** key sequence.)

Figure 1-15 Signal with Signal Tracking On

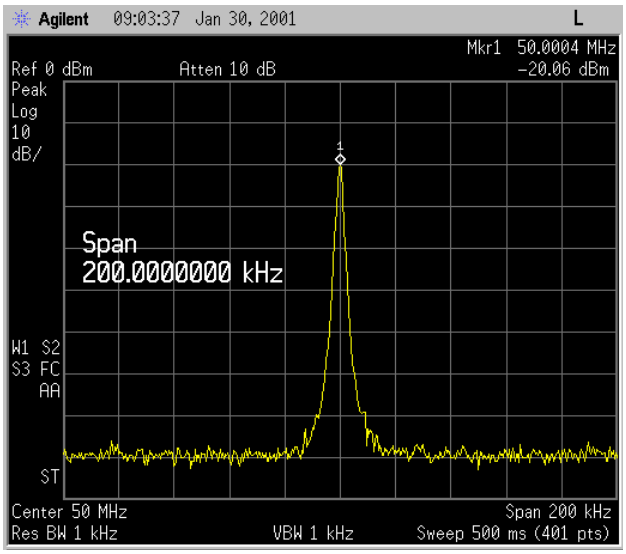


7. Reduce span and resolution bandwidth to zoom in on the marked signal by pressing **SPAN**, **Span**, 200, **kHz**.

If the span change is large enough, span will decrease in steps as automatic zoom is completed. See Figure 1-16. You can also use the front-panel knob or step keys to decrease the span and resolution bandwidth values.

8. Press **FREQUENCY**, **Signal Track** (so that Off is underlined) to turn off the signal track function.

Figure 1-16 After Zooming In on the Signal



Tracking Drifting Signals

The signal track function is useful for tracking drifting signals that drift relatively slowly. To place a marker on the signal you wish to track, use **Peak Search**. Pressing **FREQUENCY**, **Signal Track** (On) will bring that signal to the center frequency of the graticule and adjust the center frequency every sweep to bring the selected signal back to the center. A quick way to perform the **Peak Search**, **FREQUENCY**, **Signal Track**, **SPAN** key sequence is to use the **Span Zoom** key in the **SPAN** menu.

Note that the primary function of the signal track function is to track unstable signals, not to track a signal as the center frequency of the analyzer is changed. If you choose to use the signal track function when changing center frequency, check to ensure that the signal found by the tracking function is the correct signal.

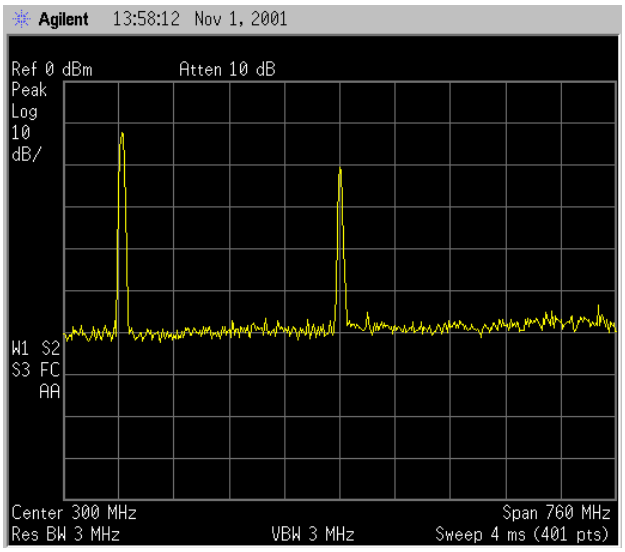
Tracking Signal Drift Example 1:

Use the signal track function to keep a drifting signal at the center of the display and monitor its change.

This example requires a signal generator. The frequency of the signal generator will be changed while you view the signal on the display of the analyzer.

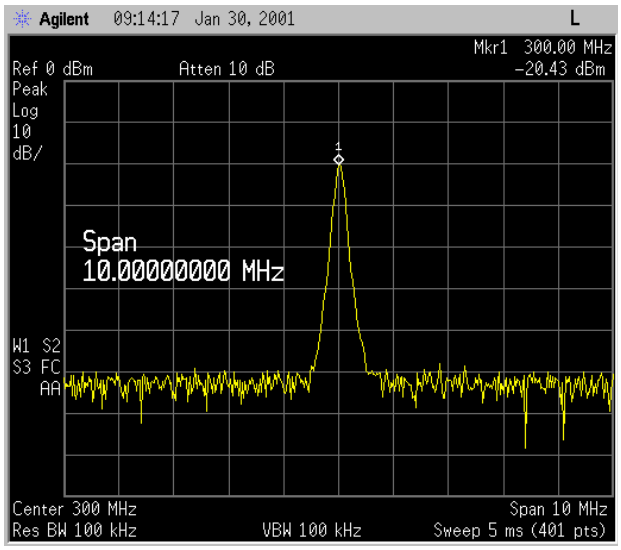
1. Connect a signal generator to the analyzer input.
2. Set the signal generator frequency to 300 MHz with an amplitude of –20 dBm.
3. Set the analyzer as follows:
   a. Press **Preset**, **Factory Preset** (if present).
   b. Set the center frequency to 300 MHz by pressing **FREQUENCY, Center Freq**, 300, **MHz**. See Figure 1-17.

Figure 1-17 Signal With Default Span



4. Press **Peak Search**.
5. Set the span to 10 MHz by pressing **SPAN**, **Span**, 10, **MHz**. See Figure 1-18.

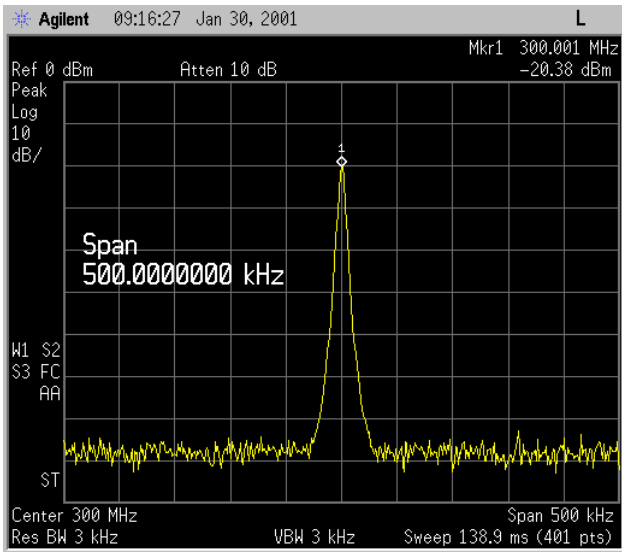
Figure 1-18 Signal With 10 MHz Span



6. Press **SPAN**, **Span Zoom**, 500, **kHz**.

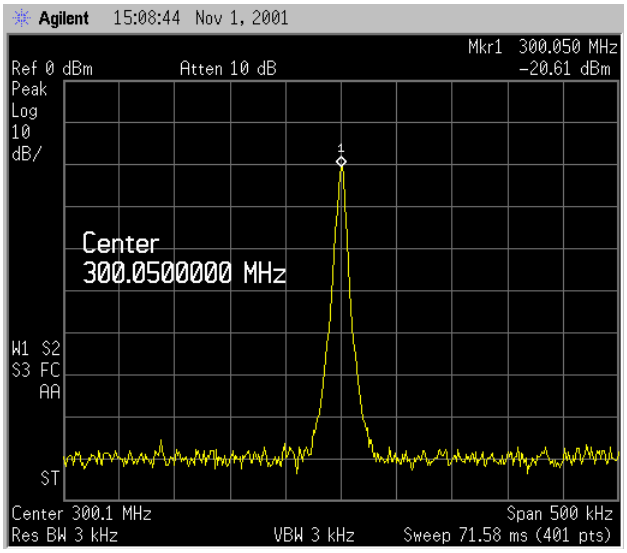
   Notice that the signal has been held in the center of the display. See Figure 1-19.

Figure 1-19 Signal With 500 kHz Span



7. Tune the frequency of the signal generator in 10 kHz increments. Notice that the center frequency of the analyzer also changes in 10 kHz increments, centering the signal with each increment. See Figure 1-20. Note that the center frequency has changed.

Figure 1-20 Using Span Zoom to Track a Drifting Signal

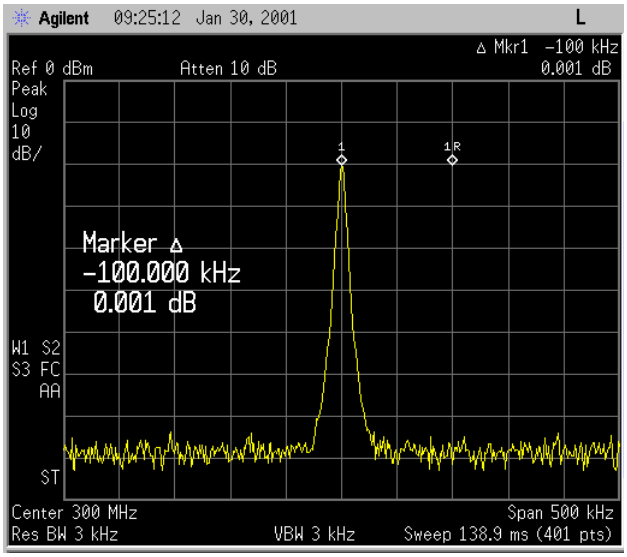


8. The signal frequency drift can be read from the screen if both the signal track and marker delta functions are active. Set the analyzer and signal generator as follows:

   a. Press **Marker, Delta**.

   b. Tune the frequency of the signal generator. The marker readout indicates the change in frequency and amplitude as the signal drifts. See Figure 1-21.

Figure 1-21 Using Signal Tracking to Track a Drifting Signal



Tracking Signal Drift Example 2:

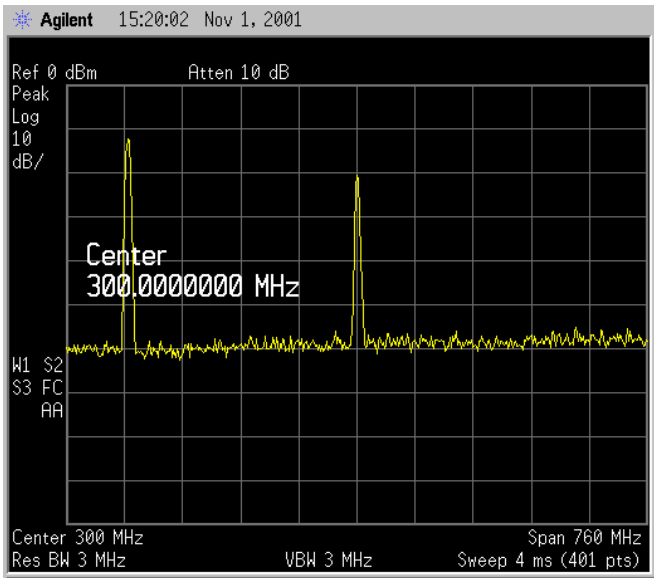
The analyzer can measure the short- and long-term stability of a source. The maximum amplitude level and the frequency drift of an input signal trace can be displayed and held by using the maximum-hold function. You can also use the maximum hold function if you want to determine how much of the frequency spectrum a signal occupies.

1. Connect a signal generator to the analyzer input.
2. Set the signal generator frequency to 300 MHz with an amplitude of –20 dBm.
3. Set the analyzer as follows:

   a. Press **Preset**, **Factory Preset** (if present).

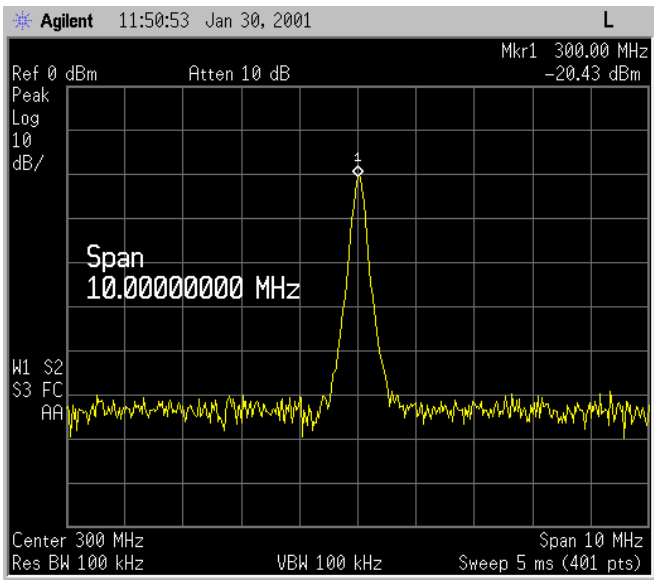
   b. Set the center frequency to 300 MHz by pressing **FREQUENCY, Center Freq**, 300, **MHz**. See Figure 1-22.

Figure 1-22 Signal With Default Span



4. Press **Peak Search**.
5. Set the span to 10 MHz by pressing **SPAN**, **Span**, 10, **MHz**. See Figure 1-23.

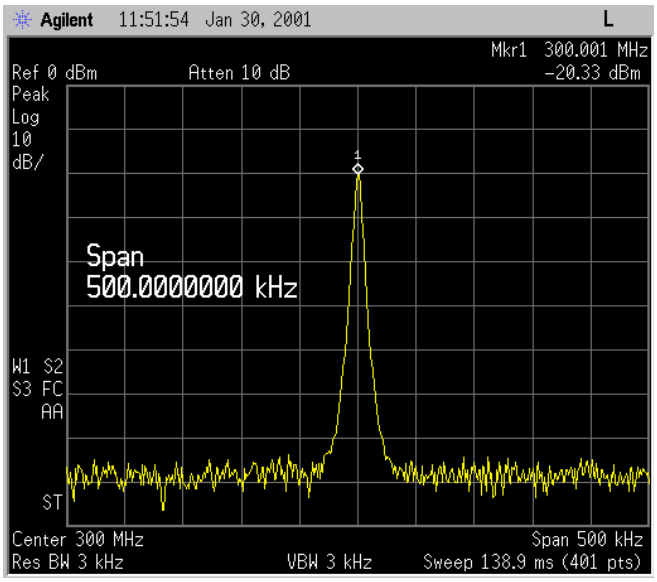
Figure 1-23 Signal With 10 MHz Span



6. Press **SPAN**, **Span Zoom**, 500, **kHz**.

   Notice that the signal has been held in the center of the display. See Figure 1-24.

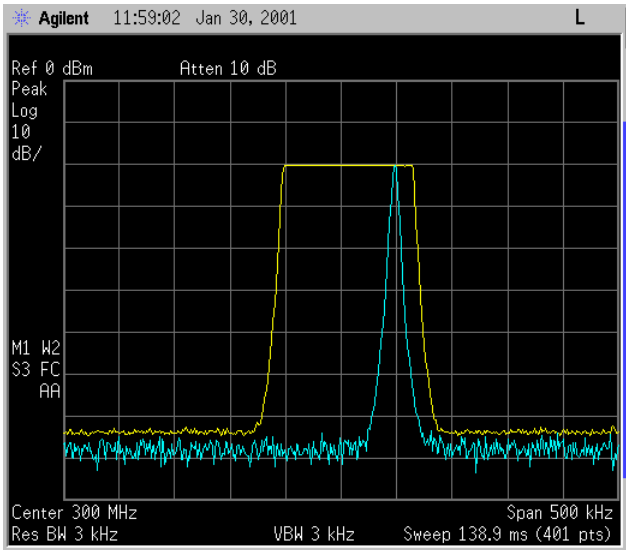
Figure 1-24 Signal With 500 KHz Span



7. Turn off the signal track function by pressing **FREQUENCY**, **Signal Track** (Off).
8. To measure the excursion of the signal, press **Trace/View**, **Max Hold**. As the signal varies, maximum hold maintains the maximum responses of the input signal.

   Annotation on the left side of the screen indicates the trace mode. For example, `M1 S2 S3` indicates trace 1 is in maximum-hold mode, trace 2 and trace 3 are in store-blank mode.
9. Press **Trace/View**, **Trace**, to select trace 2. (Trace 2 is selected when 2 is underlined.)
10. Press **Clear Write** to place trace 2 in clear-write mode, which displays the current measurement results as it sweeps. Trace 1 remains in maximum hold mode, showing the frequency shift of the signal.
11. Slowly change the frequency of the signal generator ± 50 kHz in 1 kHz increments. Your analyzer display should look similar to Figure 1-25.

Figure 1-25 Viewing a Drifting Signal With Max Hold and Clear Write



Measuring Low Level Signals

The ability of the analyzer to measure low level signals is limited by the noise generated inside the analyzer. A signal may be masked by the noise floor so that it is not visible. This sensitivity to low level signals is affected by the measurement setup.

The analyzer input attenuator and bandwidth settings affect the sensitivity by changing the signal-to-noise ratio. The attenuator affects the level of a signal passing through the instrument, whereas the bandwidth affects the level of internal noise without affecting the signal. In the first two examples in this section, the attenuator and bandwidth settings are adjusted to view low level signals.

If, after adjusting the attenuation and resolution bandwidth, a signal is still near the noise, visibility can be improved by using the video bandwidth and video averaging functions, as demonstrated in the third and fourth examples.

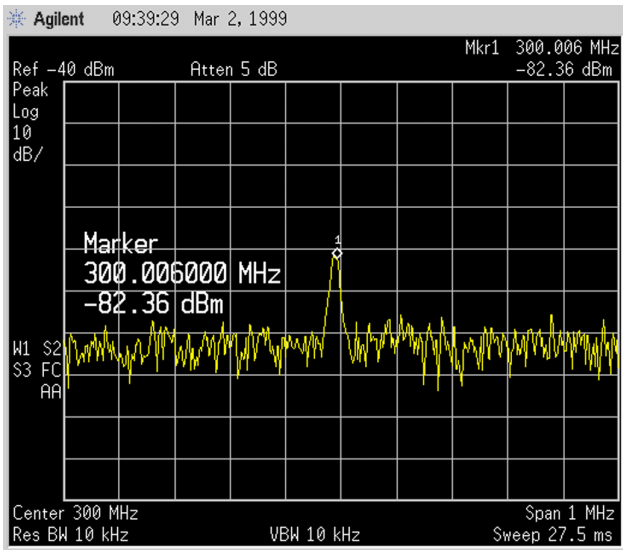
Measuring Low Level Signals Example 1:

If a signal is very close to the noise floor, reducing input attenuation brings the signal out of the noise. Reducing the attenuation to 0 dB maximizes signal power in the analyzer.

CAUTION The total power of all input signals at the analyzer input must not exceed the maximum power level for the analyzer.

1. Connect a signal generator to the analyzer input.
2. Set the signal generator frequency to 300 MHz with an amplitude of –80 dBm.
3. On the analyze, perform a factory preset by pressing **Preset**, **Factory Preset** (if present).
4. Set the center frequency of the analyzer to 300 MHz by pressing **FREQUENCY**, **Center Freq**, 300, **MHz**.
5. Set the span to 5 MHz by pressing **SPAN**, **Span**, 5, **MHz**.
6. Set the reference level to –40 dBm by pressing **AMPLITUDE**, **Ref Level**, –40, **dBm**.
7. Place the signal at center frequency by pressing **Peak Search**, **Marker→**, **Mkr→CF**.
8. Reduce the span to 1 MHz. Press **SPAN**, **Span**, and then use the step-down key (↓) until the span is set to 1 MHz. See Figure 1-26.

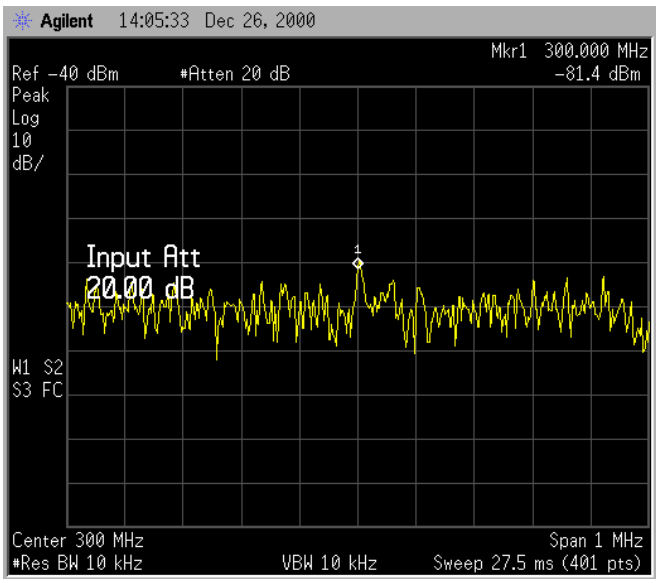
Figure 1-26 Low-Level Signal



9. Press **AMPLITUDE**, **Attenuation**. Press the step-up key (↑) to select 20 dB attenuation. Increasing the attenuation moves the noise floor closer to the signal.

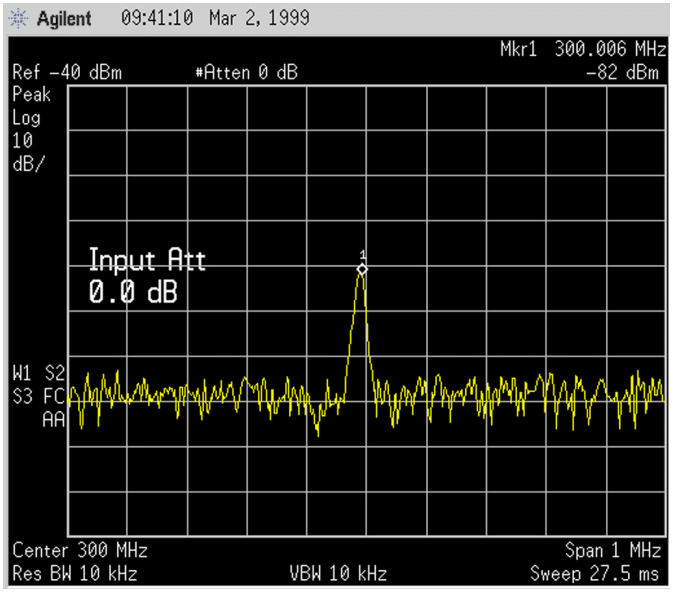
   A # mark appears next to the Atten annotation at the top of the display, indicating the attenuation is no longer coupled to other analyzer settings.

Figure 1-27 Using 20 dB Attenuation



10. To see the signal more clearly, enter 0 dB. Zero decibels of attenuation makes the signal more visible. See Figure 1-28.

Figure 1-28 Using 0 dB Attenuation



CAUTION Before connecting other signals to the analyzer input, increase the RF attenuation to protect the analyzer input: press **Attenuation** so that Auto is underlined or press **Auto Couple**.

Measuring Low Level Signals Example 2:

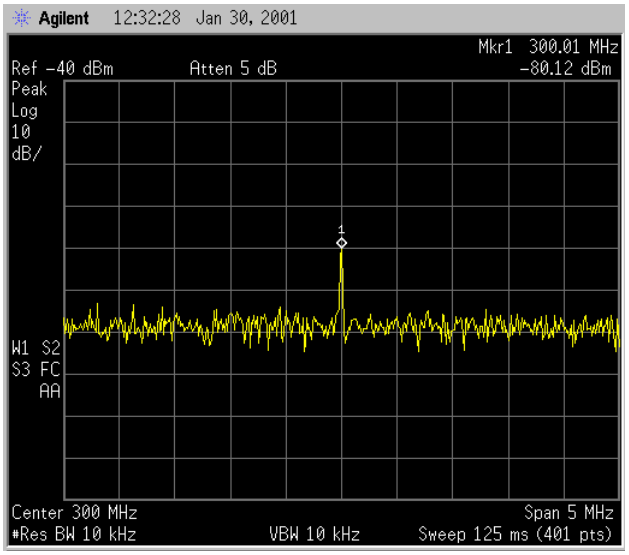
The resolution bandwidth can be decreased to view low level signals.

1. Connect a signal generator to the analyzer input.
2. Set the signal generator frequency to 300 MHz with an amplitude of –80 dBm.
3. On the analyzer, perform a factory preset by pressing **Preset**, **Factory Preset** (if present).
4. Set the center frequency of the analyzer to 300 MHz by pressing **FREQUENCY**, **Center Freq**, 300, **MHz**.
5. Set the span to 5 MHz by pressing **SPAN**, **Span**, 5, **MHz**.
6. Set the reference level to –40 dBm by pressing **AMPLITUDE**, **Ref Level**, –40, **dBm**.
7. Place the signal at center frequency by pressing **Peak Search**, **Marker→**, **Mkr→CF**.

8. Press **BW/Avg**, **Res BW**, and then ↓. The low level signal appears more clearly because the noise level is reduced. As shown in Figure 1-29.

   A # mark appears next to the **Res BW** annotation at the lower left corner of the screen, indicating that the resolution bandwidth is uncoupled. As the resolution bandwidth is reduced, the sweep time is increased to maintain calibrated data.

Figure 1-29 Decreasing Resolution Bandwidth



Measuring Low Level Signals Example 3:

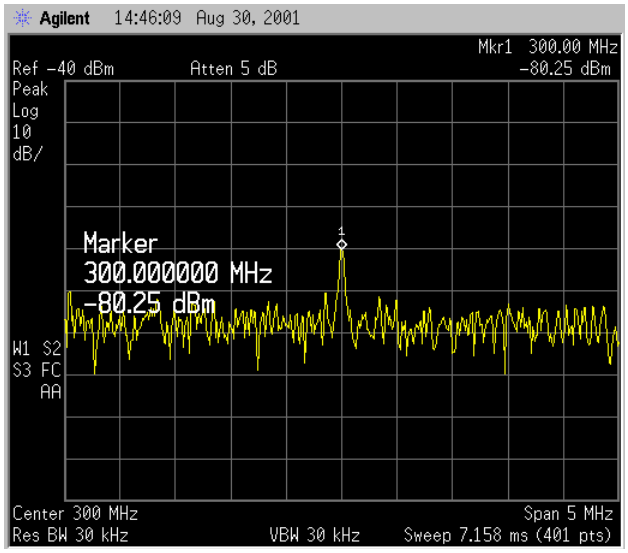
Narrowing the video filter can be useful for noise measurements and observation of low level signals close to the noise floor. The video filter is a post-detection low-pass filter that smooths the displayed trace. When signal responses near the noise level of the analyzer are visually masked by the noise, the video filter can be narrowed to smooth this noise and improve the visibility of the signal. (Reducing video bandwidths requires slower sweep times to keep the analyzer calibrated.)

Using the video bandwidth function, measure the amplitude of a low level signal.

1. Connect a signal generator to the analyzer input.
2. Set the signal generator frequency to 300 MHz with an amplitude of –80 dBm.
3. On the analyzer, perform a factory preset by pressing **Preset**, **Factory Preset** (if present).

4. Set the center frequency of the analyzer to 300 MHz by pressing **FREQUENCY**, **Center Freq**, 300, **MHz**.
5. Set the span to 5 MHz by pressing **SPAN**, **Span**, 5, **MHz**.
6. Set the reference level to –40 dBm by pressing **AMPLITUDE**, **Ref Level**, –40, **dBm**.
7. Place the signal at center frequency by pressing **Peak Search**, **Marker→**, **Mkr→CF**. See Figure 1-30.

Figure 1-30 30 kHz Video Bandwidth



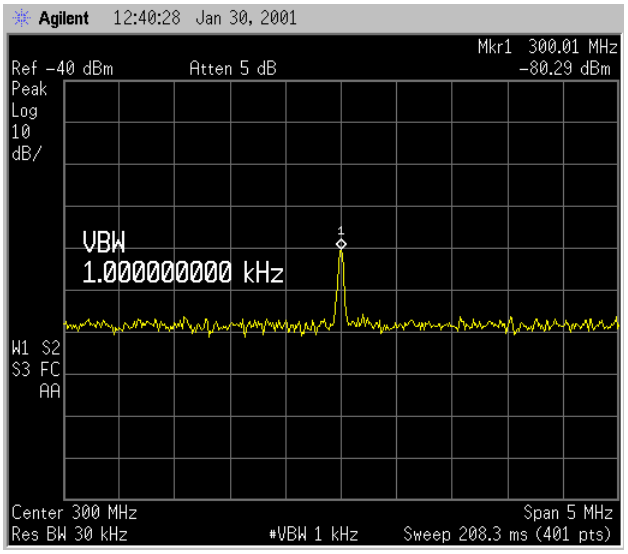
8. Narrow the video bandwidth by pressing **BW/Avg**, **Video BW**, and the step-down key (↓). This clarifies the signal by smoothing the noise, which allows better measurement of the signal amplitude.

   A # mark appears next to the VBW annotation at the bottom of the screen, indicating that the video bandwidth is not coupled to the resolution bandwidth. See Figure 1-31. As the video bandwidth is reduced, the sweep time is increased to maintain calibrated data.

   Instrument preset conditions couple the video bandwidth to the resolution bandwidth so that the video bandwidth is equal to the resolution bandwidth. If the bandwidths are uncoupled when video bandwidth is the active function, pressing **Video BW** (so that Auto is underlined) recouples the bandwidths.

NOTE The video bandwidth must be set wider than the resolution bandwidth when measuring impulse noise levels.

Figure 1-31 Decreasing Video Bandwidth



Measuring Low Level Signals Example 4:

If a signal level is very close to the noise floor, video averaging is another way to make the signal more visible.

NOTE The time required to construct a full trace that is averaged to the desired degree is approximately the same when using either the video bandwidth or the video averaging technique. The video bandwidth technique completes the averaging as a slow sweep is taken, whereas the video averaging technique takes many sweeps to complete the average. Characteristics of the signal being measured, such as drift and duty cycle, determine which technique is appropriate.

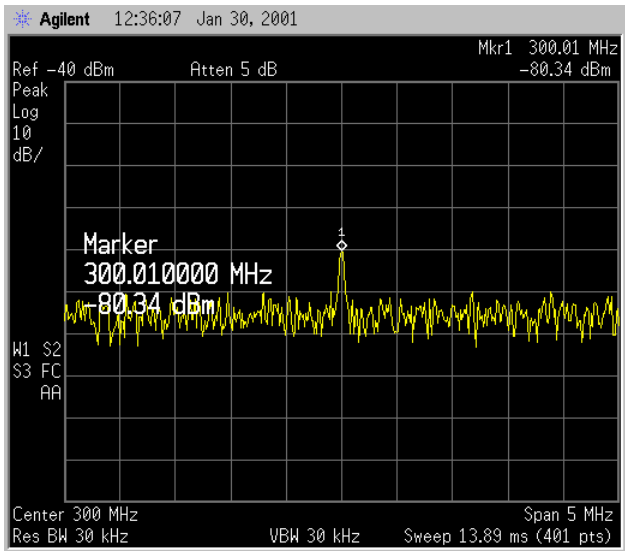
Video averaging is a digital process in which each trace point is averaged with the previous trace-point average. Selecting **Average Type**, **Video Avg** and **Average (On)** changes the detection mode from peak to sample. The result is a sudden drop in the displayed noise level. The sample mode displays the instantaneous value of the signal at the end of the time or frequency interval represented by each display point, rather than the value of the peak during the interval. Sample mode is not used to measure signal amplitudes accurately because it may not find the true peak of the signal.

Video averaging clarifies low-level signals in wide bandwidths by averaging the signal and the noise. As the analyzer takes sweeps, you can watch video averaging smooth the trace.

1. Connect a signal generator to the analyzer input.
2. Set the signal generator frequency to 300 MHz with an amplitude of –80 dBm.

3. On the analyzer, perform a factory preset by pressing **Preset**, **Factory Preset** (if present).
4. Set the center frequency of the analyzer to 300 MHz by pressing **FREQUENCY**, **Center Freq**, 300, **MHz**.
5. Set the span to 5 MHz by pressing **SPAN**, **Span**, 5, **MHz**.
6. Set the reference level to –40 dBm by pressing **AMPLITUDE**, **Ref Level**, –40, **dBm**.
7. Place the signal at center frequency by pressing **Peak Search**, **Marker→**, **Mkr→CF**. See Figure 1-32.

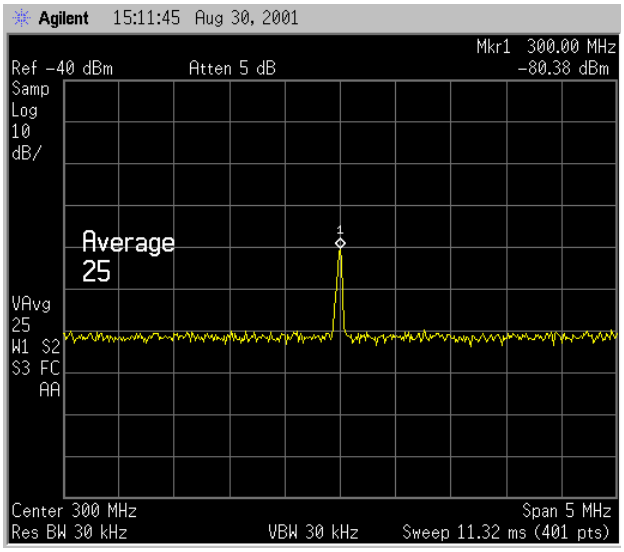
Figure 1-32 Without Video Averaging



8. Pressing **BW/Avg**, **Average Type**, **(Video Avg)**, **Average (On)**, initiates the video averaging routine. As the averaging routine smooths the trace, low level signals be come more visible. `Average 100` appears in the active function block. The number represents the number of samples (or sweeps) taken to complete the averaging routine. Once the set number of sweeps has been completed, the analyzer continues to provide a running average based on this set number.
9. To set the number of samples, use the numeric keypad. For example, press **Average (On)**, 25, **Enter**. As shown in Figure 1-33.

   During averaging, the current sample number appears at the left side of the graticule. The number of samples equals the number of sweeps in the averaging routine. Changes in active function settings, such as the center frequency or reference level, will restart the sampling. The sampling will also restart if video averaging is turned off and then on again. To see the sample number increment, turn video averaging off and on again by pressing **Average** (Off), **Average** (On).

Figure 1-33 Using the Video Averaging Function



Identifying Distortion Products

Distortion from the Analyzer

High level input signals may cause analyzer distortion products that could mask the real distortion measured on the input signal. Using trace 2 and the RF attenuator, you can determine which signals, if any, are internally generated distortion products.

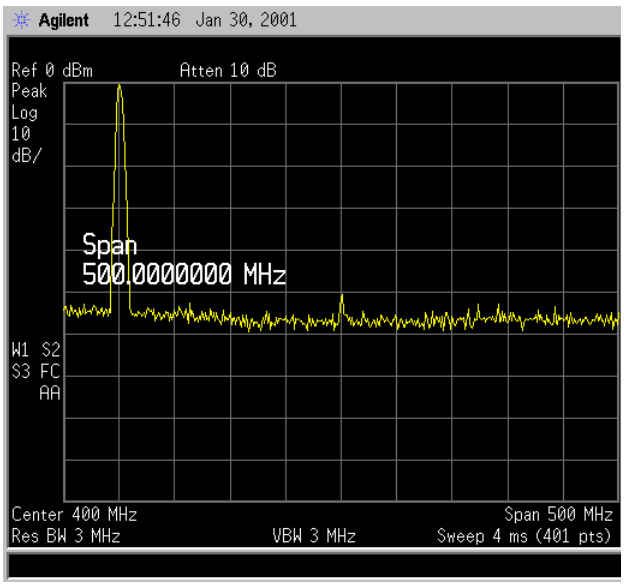
Identifying Analyzer Generated Distortion Example:

Using a signal from a signal generator, determine whether the harmonic distortion products are generated by the analyzer.

1. Connect a signal generator to the analyzer INPUT.
2. Set the signal generator frequency to 200 MHz and the amplitude to 0 dBm.
3. On the analyzer, perform a factory preset by pressing **Preset**, **Factory Preset** (if present).
4. Set the center frequency of the analyzer to 400 MHz by pressing **FREQUENCY**, **Center Freq**, 400, **MHz**.
5. Set the span to 500 MHz by pressing **SPAN**, **Span**, 500, **MHz**.

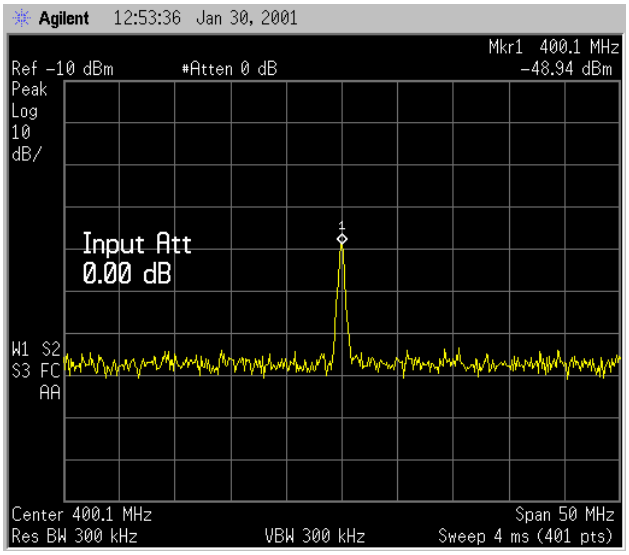
   The signal produces harmonic distortion products in the analyzer input mixer as shown in Figure 1-34.

Figure 1-34 Harmonic Distortion



6. Change the center frequency to the value of one of the observed harmonics by pressing Peak Search, Next Peak, Marker→, Mkr→CF.
7. Change the span to 50 MHz: press **SPAN**, **Span**, 50, **MHz**.
8. Ensure that the signal is still at the center frequency, if necessary press **Peak Search**, **Marker→**, **Mkr→CF**.
9. Change the attenuation to 0 dB: press **AMPLITUDE**, **Attenuation**, 0, **dBm**. Your display should be similar to Figure 1-35

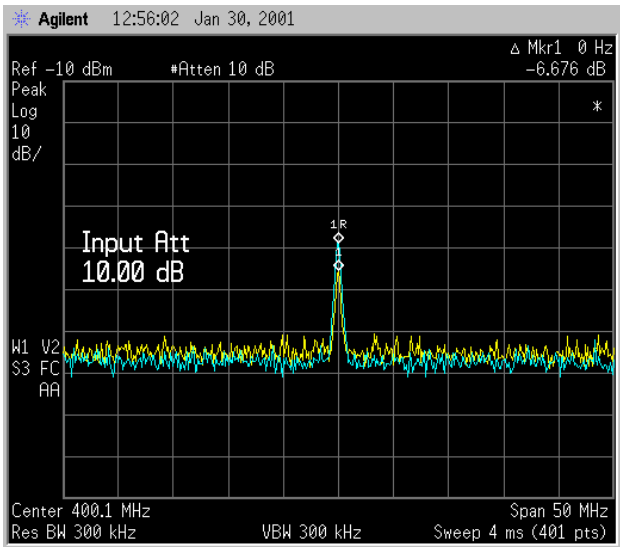
Figure 1-35 Harmonic Distortion with 0 dB Attenuation



10. To determine whether the harmonic distortion products are generated by the analyzer, first save the screen data in trace 2 as follows:
    a. Press **Trace/View**, **Trace (2)**, then **Clear Write**.
    b. Allow the trace to update (two sweeps) and press **Trace/View**, **View**, **Marker**, **Delta**. The analyzer display shows the stored data in trace 2 and the measured data in trace 1.
11. Next, increase the RF attenuation by 10 dB: press **AMPLITUDE**, **Attenuation**, and the step-up key (↑) twice. See Figure 1-36.

    Notice the ΔMkr1 amplitude reading. This is the difference in the distortion product amplitude readings between 0 dB and 10 dB input attenuation settings. If the ΔMkr1 amplitude absolute value is approximately ≥1 dB for an input attenuator change, the distortion is being generated, at least in part, by the analyzer. In this case more input attenuation is necessary.

Figure 1-36 RF Attenuation of 10 dB

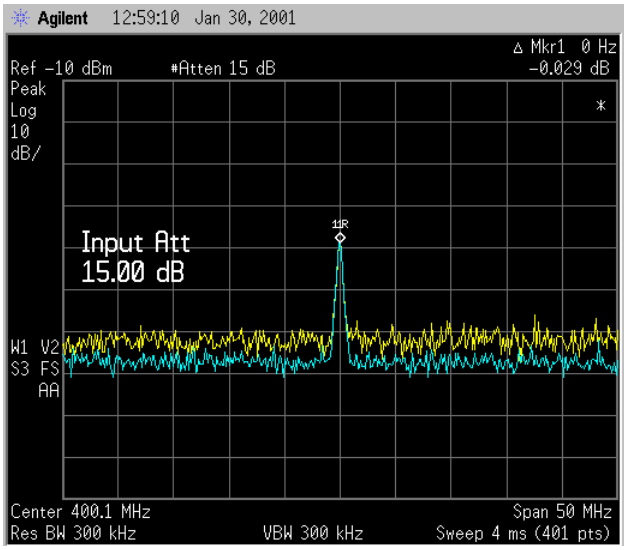


12. Press **Peak Search**, **Marker**, **Delta**

    Change the attenuation to 15 dB by pressing **Attenuation**, 15, **dB**.

    If the ΔMkr1 amplitude absolute value is approximately ≥1 dB as seen in Figure 1-36, then more input attenuation is required; some of the measured distortion is internally generated. If there is no change in the signal level, the distortion is not generated internally. For example, the signal that is causing the distortion shown in Figure 1-37 is not high enough in amplitude to cause internal distortion in the analyzer so any distortion that is displayed is present on the input signal.

Figure 1-37 No Harmonic Distortion



Third-Order Intermodulation Distortion

Two-tone, third-order intermodulation distortion is a common test in communication systems. When two signals are present in a non-linear system, they can interact and create third-order intermodulation distortion products that are located close to the original signals. These distortion products are generated by system components such as amplifiers and mixers.

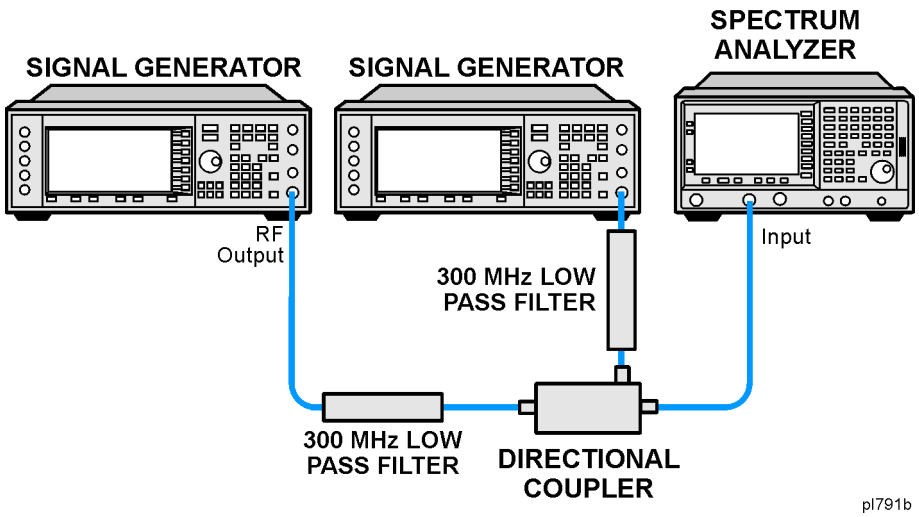
For the quick setup TOI measurement example, refer to "Measuring TOI Distortion with One Button Press" on page 48.

Identifying TOI Distortion Example:

Test a device for third-order intermodulation. This example uses two sources, one set to 300 MHz and the other to approximately 301 MHz. (Other source frequencies may be substituted, but try to maintain a frequency separation of approximately 1 MHz.)

1. Connect the equipment as shown in Figure 1-38. This combination of signal generators, low pass filters, and directional coupler (used as a combiner) results in a two-tone source with very low intermodulation distortion. Although the distortion from this setup may be better than the specified performance of the analyzer, it is useful for determining the TOI performance of the source/analyzer combination. After the performance of the source/analyzer combination has been verified, the device-under-test (DUT) (for example, an amplifier) would be inserted between the directional coupler output and the analyzer input and another measurement would be made.

Figure 1-38 Third-Order Intermodulation Equipment Setup



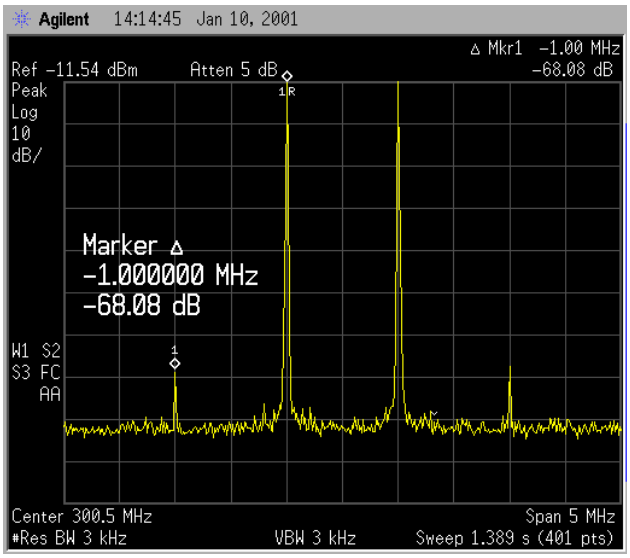
NOTE The combiner should have a high degree of isolation between the two input ports so the sources do not intermodulate.

2. Set one source (signal generator) to 300 MHz and the other source to 301 MHz, for a frequency separation of 1 MHz. Set the sources equal in amplitude as measured by the analyzer (in this example, they are set to –5 dBm).
3. On the analyzer, perform a factory preset by pressing **Preset**, **Factory Preset** (if present).
4. Set the span to 5 MHz by pressing **SPAN**, **Span**, 5, **MHz**. This is wide enough to include the distortion products on the screen.
5. Tune both test signals onto the screen by setting the center frequency 300.5 MHz, press **FREQUENCY**, **Center Freq**, 300.5, **MHz**.

   If necessary, use the front-panel knob to center the two test signals on the display.
6. To be sure the distortion products are resolved, reduce the resolution bandwidth until the distortion products are visible by pressing **BW/Avg**, **Res BW**, and then use the step-down key (↓) to reduce the resolution bandwidth until the distortion products are visible.
7. For best dynamic range, set the maximum mixer input level to –30 dBm and move the signal to the reference level: press **AMPLITUDE**, **More**, **Max Mixer Lvl**, –30, **dBm**.

   The analyzer automatically sets the attenuation so that a signal at the reference level will be a maximum of –30 dBm at the input mixer.
8. Press **BW/Avg**, **Res BW**, and then use the step-down key (↓) to reduce the resolution bandwidth until the distortion products are visible.
9. To measure a distortion product, press **Peak Search** to place a marker on a source signal.
10. Set the marked signal to the reference level by pressing **Mkr** → and then **Mkr** → **Ref Lvl**.
11. To activate the second marker, press **Marker**, **Delta**, **Peak Search**. Using the **Next Peak** key to place the second marker on the peak of the distortion product that is beside the test signal. The difference between the markers is displayed in the active function area. See Figure 1-39.

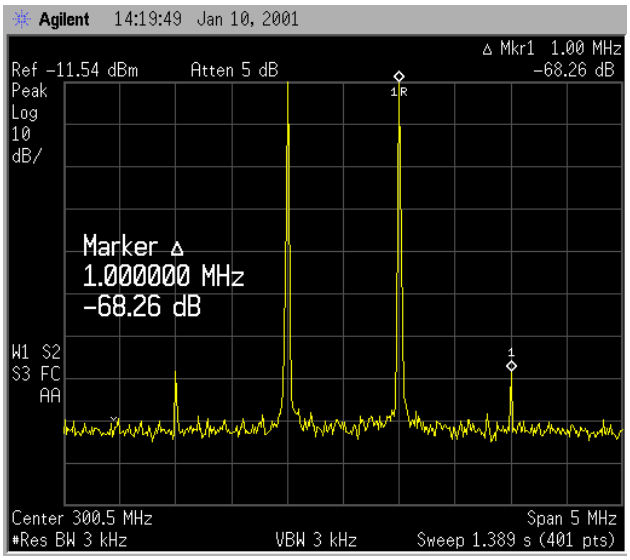
Figure 1-39 Measuring the Distortion Product



12. To measure the other distortion product, press **Marker**, **Normal**, **Peak Search**, **Next Peak**. This places a marker on the next highest peak, the other source signal.

13. To measure the difference between this test signal and the second distortion product, press **Delta**, **Peak Search**. Using the **Next Peak** key to place the second marker on the peak of the second distortion product. See Figure 1-40.

Figure 1-40 Measuring the Distortion Product



Measuring TOI Distortion with One Button Press

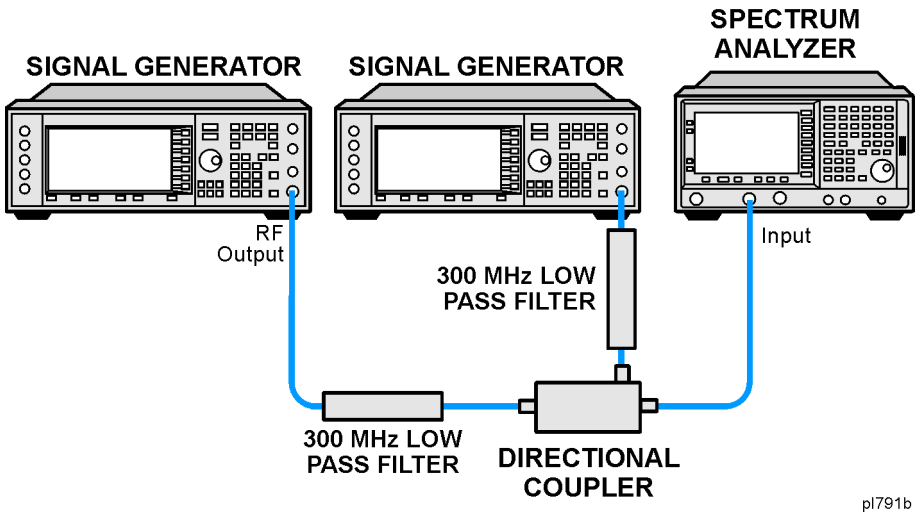
Test a device for third-order intermodulation. This example uses two sources, one set to 300 MHz and the other to approximately 301 MHz.

(Other source frequencies may be substituted, but try to maintain a frequency separation of approximately 1 MHz.)

NOTE Steps 1. through step 3 are identical to "Identifying TOI Distortion Example:" on page 46 and are repeated here for convenience.

1. Connect the equipment as shown in Figure 1-41.

Figure 1-41 Third-Order Intermodulation Equipment Setup

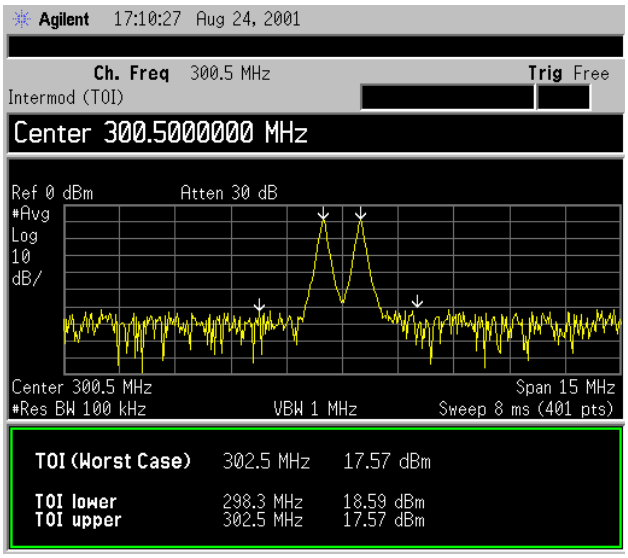


NOTE The combiner should have a high degree of isolation between the two input ports so the sources do not intermodulate.

2. Set one source to 300 MHz and the other source to 301 MHz, for a frequency separation of 1 MHz. Set the sources equal in amplitude, as measured by the analyzer (in this example, they are set to –5 dBm).
3. On the analyzer, perform a factory preset by pressing **Preset**, **Factory Preset** (if present).
4. Tune both test signals onto the screen by setting the center frequency 300.5 MHz, press **FREQUENCY**, **Center Freq**, 300.5, **MHz**.

   If necessary, use the front-panel knob to center the two test signals on the display.
5. Press **Measure**, **More**, **Intermod** (TOI) to display the distortion products as show below.

Figure 1-42 Measuring the Distortion Products



Measuring Signal-to-Noise

The signal-to-noise measurement procedure below may be adapted to measure any signal in a system if the signal (carrier) is a discrete tone. If the signal in your system is modulated, it will be necessary to modify the procedure to correctly measure the modulated signal level

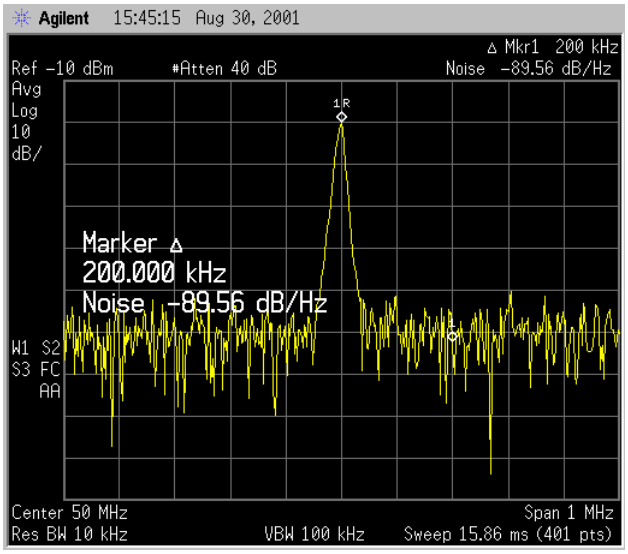
In this example the 50 MHz amplitude reference signal is used as the fundamental source. The amplitude reference signal is assumed to be the signal of interest and the internal noise of the analyzer is measured as the system noise. To do this, you will need to set the input attenuator such that both the signal and the noise are well within the calibrated region of the display.

Signal-to-Noise Measurement Example

Perform the steps below to measure the signal-to-noise.

1. Perform a factory preset by pressing **Preset**, **Factory Preset** (if present).
2. Turn on the internal 50 MHz amplitude reference signal of the analyzer as follows:
   - For the E4401B and E4411B, use the internal 50 MHz amplitude reference signal of the analyzer as the signal being measured. Press **Input/Output**, **Amptd Ref (On)**.
   - For all other models connect a cable between the front-panel AMPTD REF OUT to the analyzer INPUT, then press **Input/Output**, **Amptd Ref Out (On)**.
3. Set the center frequency to 50 MHz by pressing **FREQUENCY**, **Center Freq**, **50**, **MHz**.
4. Set the span to 1 MHz by pressing **SPAN**, **Span**, 1, **MHz**.
5. Set the reference level to –10 dBm by pressing **AMPLITUDE**, **Ref Level**, –10, **dBm**.
6. Set the attenuation to 40 dB by pressing **AMPLITUDE**, **Attenuation**, 40, **dB**.
7. Press **Peak Search** to place a marker on the peak of the signal.
8. Press **Marker**, **Delta**, 200, **kHz** to put the delta marker in the noise at the specified offset, in this case 200 kHz.
9. Press **More**, **Function**, **Marker Noise** to view the results of the signal to noise measurement. See Figure 1-43.

Figure 1-43 Measuring the Signal-to-Noise



Read the signal-to-noise in dB/Hz, that is with the noise value determined for a 1 Hz noise bandwidth. If you wish the noise value for a different bandwidth, decrease the ratio by $10 \times \log(BW)$. For example, if the analyzer reading is –70 dB/Hz but you have a channel bandwidth of 30 kHz:

$$\text{S/N} = -70 \text{ dB/Hz} + 10 \times \log(30 \text{ kHz}) = -25.23 \text{ dB}/(30 \text{ kHz})$$

Note that the display detection mode is now average. If the delta marker is within half a division of the response to a discrete signal, the amplitude reference signal in this case, there is a potential for error in the noise measurement. See “Making Noise Measurements” on page 53.

Making Noise Measurements

There are a variety of ways to measure noise power. The first decision you must make is whether you want to measure noise power at a specific frequency or the total power over a specified frequency range, for example over a channel bandwidth.

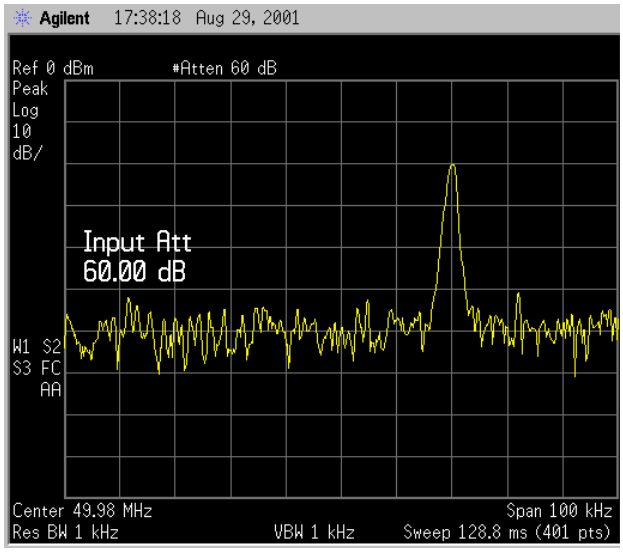
Noise Measurement Example 1:

Using the marker function, **Marker Noise**, is a simple method to make a measurement at a single frequency. In this example, attention must be made to the potential errors due to discrete signal (spectral components). This measurement will be made near the 50 MHz amplitude reference signal to illustrate the use of **Marker Noise.**

1. Perform a factory preset by pressing **Preset**, **Factory Preset** (if present).
2. Turn on the internal 50 MHz amplitude reference signal of the analyzer as follows:
   - For the E4401B and E4411B, use the internal 50 MHz amplitude reference signal of the analyzer as the signal being measured. Press **Input/Output**, **Amptd Ref** (On).
   - For all other models connect a cable between the front-panel AMPTD REF OUT to the analyzer INPUT, then press **Input/Output**, **Amptd Ref Out** (On).
3. Set the center frequency to 49.98 MHz by pressing **FREQUENCY**, **Center Freq**, 49.98, **MHz**.
4. Set the span to 100 kHz by pressing **SPAN**, **Span**, 100, **kHz**.
5. Set the attenuation to 60 dB by pressing **AMPLITUDE**, **Attenuation (Man)**, 60, **dB**. See Figure 1-44.

NOTE When making noise measurements and **AMPLITUDE**, **Scale Type** (Log) is selected (10 dB/division), position the trace between 3 and 6 graticule lines above the bottom by adjusting the reference level (**AMPLITUDE**, **Ref Level**). Measurement inaccuracies may occur if displayed trace is positioned outside this range.

Figure 1-44 Setting the Attenuation

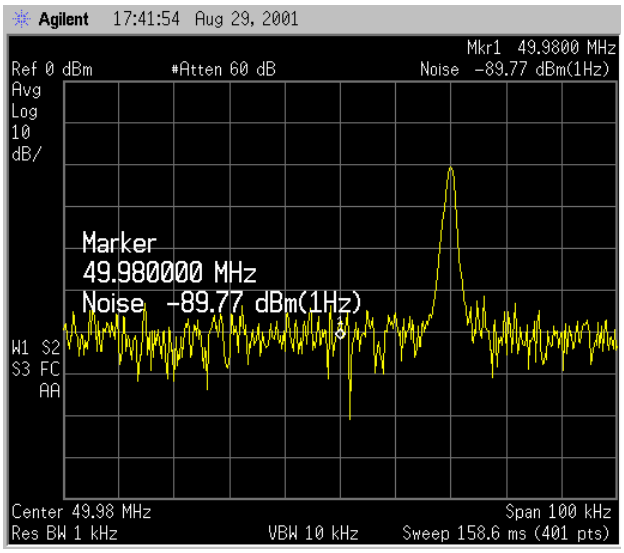


6. Activate the noise marker by pressing **Marker**, **More**, **Function**, **Marker Noise**.

   Note that the display detection automatically changed to “Avg” which can be manually set by pressing Det/Demod, **Average** (Video/RMS). The marker is floating between the maximum and the minimum of the noise. For firmware revisions earlier than A.08.00, the detection type when using **Marker Noise** changed to sample. If you wish to use sample detection, press **Det/Demod**, **Detector**, **Sample** and verify that “Average Type” is set to “Video average” by pressing **BW/Avg**, **Average Type**, **Video Avg.** This is not recommended as it is slower and does not increase accuracy.

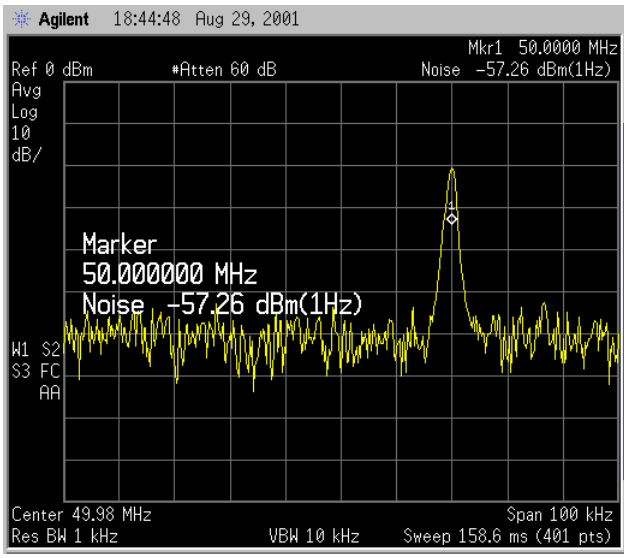
   The marker readout is in dBm(Hz) or dBm per unit bandwidth. See Figure 1-45. For noise power in a different bandwidth, add $10 \times \log(\mathrm{BW})$. For example, for noise power in a 1 kHz bandwidth, add $10 \times \log(1000)$ or 30 dB to the noise marker value.

Figure 1-45 Activating the Noise Marker



7. The noise marker value is based on the mean of 5% of the total number of sweep points centered at the marker. The points averaged span one-half of a division. To see the effect, move the marker to the 50 MHz signal by pressing **Marker**, 50, **MHz** (or use the front-panel knob to place marker at 50 MHz). See Figure 1-46.

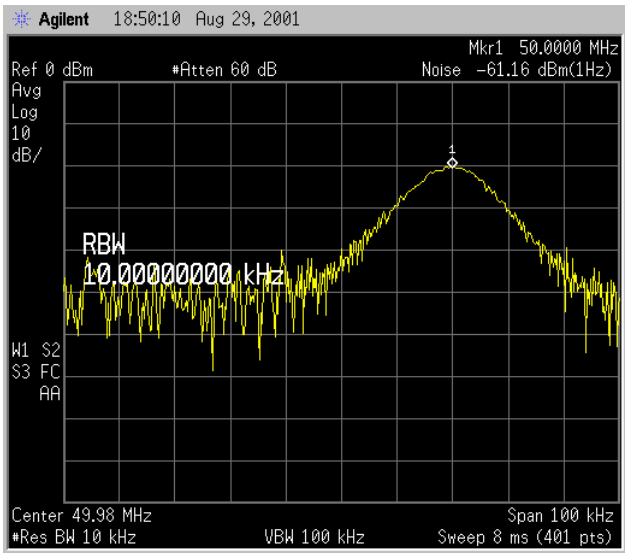
Figure 1-46 Noise Marker at 50 MHz



8. The marker does not go to the peak of the signal because not all averaged points are at the peak of the signal. Widen the resolution bandwidth by pressing **BW/Avg**, **Res BW**, 10, **kHz** (or up arrow) to see what happens. The marker is now much closer to the peak of the signal. See Figure 1-47.

NOTE Notice the video bandwidth changed to 100 kHz. The ratio between the video bandwidth (VBW) and the resolution bandwidth (RBW) must be ≥ 10/1 to maintain the accuracy of the measurement.

Figure 1-47 Increased Resolution Bandwidth

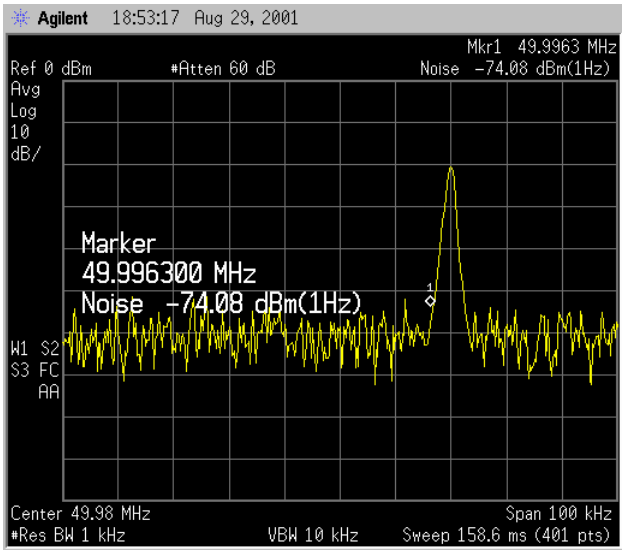


9. Return the resolution bandwidth to 1 kHz. Press **BW/Avg**, 1, **kHz**.

10. Measure the noise very close to the signal by pressing **Marker**, 50.0000, **MHz** (or use the front-panel knob to place the marker). See Figure 1-48.

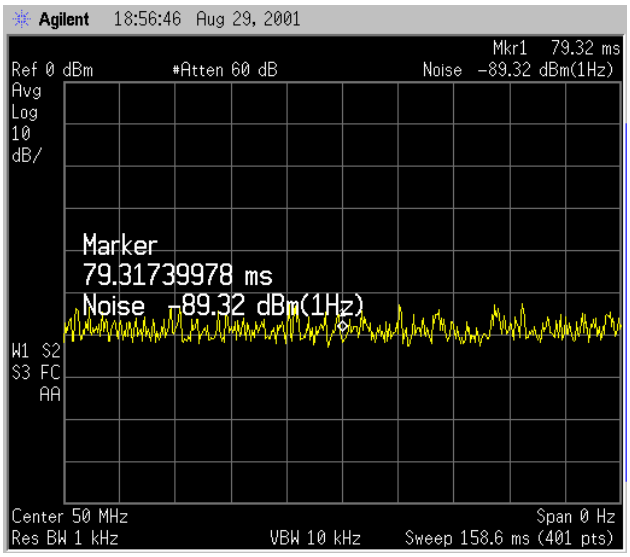
    Note that the marker reads a value that is too high because some of the averaged trace points are on the skirt of the signal response.

Figure 1-48 Noise Marker in Signal Skirt



11. Set the analyzer to zero span at the marker frequency by pressing **Mkr →**, **Mkr → CF**, **SPAN**, **Zero Span**, **Marker**. Note that the marker amplitude value is now correct since all points averaged are at the same frequency and not influenced by the shape of the bandwidth filter. See Figure 1-49.

Figure 1-49 Noise Marker with Zero Span



Noise Measurement Example 2:

The Normal marker can also be used to make a single frequency measurement as described in the previous example, again using video filtering or averaging to obtain a reasonably stable measurement. While video averaging automatically selects the sample display detection mode, video filtering does not. With sufficient filtering that results in a smooth trace, there is no difference between the sample and peak modes because the filtering takes place before the signal is digitized.

Be sure to account for the fact that the averaged noise is displayed approximately 2 dB too low for a noise bandwidth equal to the resolution bandwidth. Therefore, you must add 2 dB to the marker reading. For example, if the marker indicates –100 dBm, the actual noise level is –98 dBm.

Noise Measurement Example 3:

You may want to measure the total power of a noise-like signal that occupies some bandwidth. For example, you may want to determine the power in a communications channel. If the signal is noise and is flat across the band of interest, you can use the noise marker as described in example 1 and add $10 \times \log(\text{channel BW})$. However, if you are not certain of the characteristics of the signal, or if there are discrete spectral components in the band of interest, you can use the Channel Power measurement. This example uses the noise of the analyzer, adds a discrete tone, and assume a channel bandwidth of 50 kHz. If desired, a specific signal may be substituted.

1. Reset the analyzer by pressing **Preset**, **Factory Preset** (if present).
2. Tune the analyzer to the frequency of 50 MHz. In this example we are using the amplitude reference signal. Press **FREQUENCY**, 50, **MHz**.
3. Set the analyzer to setup the channel power measurement by pressing **MEASURE**, **Channel Power**, **Meas Setup**.

NOTE

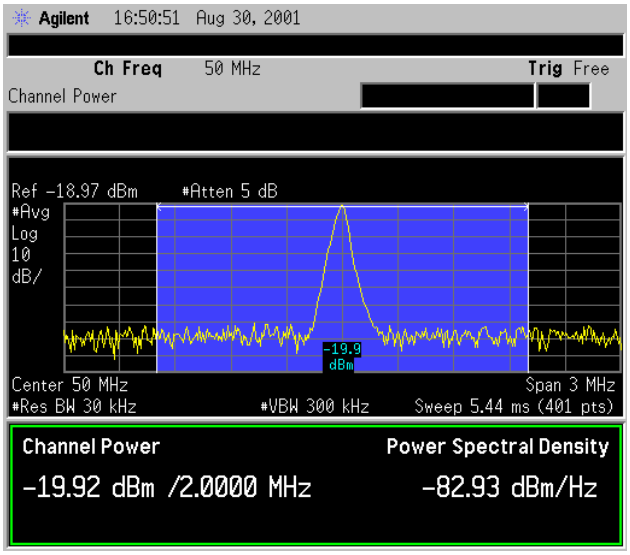
The display detection mode has been set to average (sample for firmware revisions before A.08.00) and the video bandwidth has been set to be ten times wider than the resolution bandwidth. This setting is important to prevent any averaging of the noise power on a logarithmic (decibel) scale by the video bandwidth filter, before the summing of the noise power on a power scale. You can reduce the sweep-to-sweep variation in the power reading by increasing the sweep time (or for revisions before A.08.00, by averaging over a number of sweeps).

4. Set the display view to combined by pressing **Trace/View** and **Combined**.

5. Turn average number on by pressing **Avg Number** (On). Ten is an acceptable number.
6. Add a discrete tone to see the effects on the reading. Turn on the 50 MHz amplitude reference signal of the analyzer (if you have not already done so).
   - For the E4401B and E4411B, use the internal 50 MHz amplitude reference signal of the analyzer as the signal being measured. Press **Input/Output**, **Amptd Ref** (On).
   - For all other models connect a cable between the front-panel AMPTD REF OUT to the analyzer INPUT, then press **Input/Output**, **Amptd Ref Out** (On).

   Your display should be similar to Figure 1-50.
7. Press **Meas Setup**, **Optimize Ref Level** to set the best reference level to measure this signal.

Figure 1-50 Measuring Channel Power



The power reading is essentially that of the tone; that is, the total noise power is far enough below that of the tone that the noise power contributes very little to the total.

The algorithm that computes the total power compensates for the fact that some of the trace points on the response to the continuous wave tone may be at or very close to the peak value of the tone and so yields the correct value whether the signal comprises just noise, a tone, or both.

Noise Measurement Example 4:

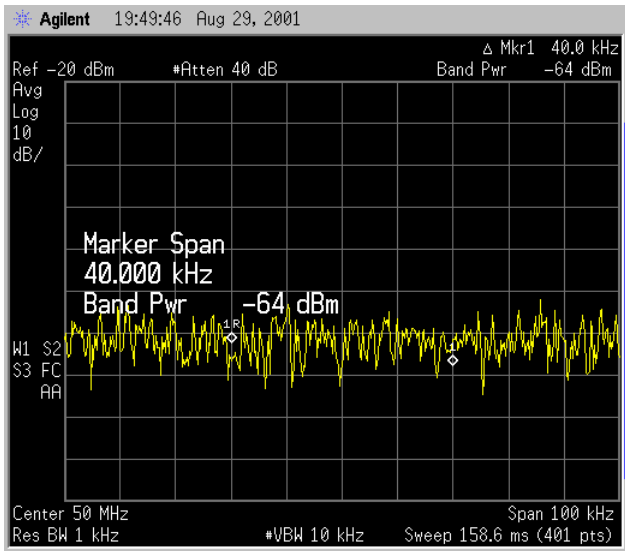
The functions described in example 3 also allow you to measure the total power of noise-like signals, tones, or both. A difference is that you use adjustable markers to set the frequency span over which power is measured. The markers allow you to easily and conveniently select any arbitrary portion of the displayed signal for measurement. However, while the analyzer does select the average (video/rms) display detection mode, you must set all of the other parameters.

1. Reset the analyzer by pressing **Preset**, **Factory Preset** (if present).
2. Tune the analyzer to the frequency of 50 MHz. In this example we are using the amplitude reference signal. Press **FREQUENCY**, 50, **MHz**.
3. Set the span to 100 kHz by pressing **SPAN**, 100, **kHz**.
4. Set the reference level to –20 dBm by pressing **AMPLITUDE**, **Ref Level**, –20, **dBm**.
5. Set the input attenuator to 40 dB by pressing **Attenuation**, 40, **dB**.
6. Set the marker span to 40 kHz by pressing **Marker**, **Span Pair** (Span), 40, **kHz**.

   The resolution bandwidth should be about 1 to 3% of the measurement (marker) span, 40 kHz in this example. The 1 kHz resolution bandwidth that the analyzer has chosen is fine. The video bandwidth should be ten times wider.
7. Set the video bandwidth to 10 kHz by pressing **BW/Avg**, **Video BW** (Man), 10, **kHz**.
8. Measure the power between markers by pressing **Marker**, **More, Function**, **Band Power**. The analyzer displays the total power between the markers. See Figure 1-51.
9. Add a discrete tone to see the effects of the reading. Turn on the internal 50 MHz amplitude reference signal of the analyzer (if you have not already done so) as follows:

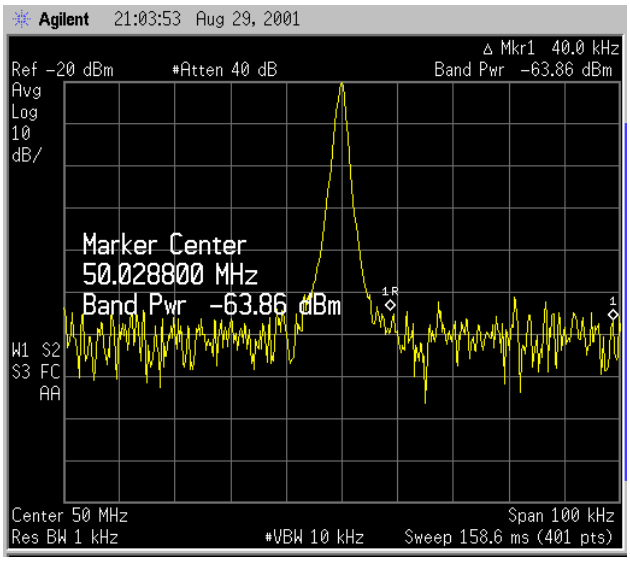
   - For the E4401B and E4411B, use the internal 50 MHz amplitude reference signal of the analyzer as the signal being measured. Press **Input/Output**, **Amptd Ref** (On).
   - For all other models connect a cable between the front-panel AMPTD REF OUT to the analyzer INPUT, then press **Input/Output**, **Amptd Ref Out** (On).

Figure 1-51 Viewing Power Between Markers



10. Move the measured span by pressing **Marker**, **Span Pair** (Center). Then use the knob to exclude the tone and note reading. You could have also used **Band Pair** or **Delta Pair** to set the measurement start and stop points independently. See Figure 1-52.

Figure 1-52 Measuring the Power in the Span



Demodulating AM Signals (Using the Analyzer As a Fixed Tuned Receiver)

The zero span mode can be used to recover amplitude modulation on a carrier signal. The analyzer operates as a fixed-tuned receiver in zero span to provide time domain measurements.

Center frequency in the swept-tuned mode becomes the tuned frequency in zero span. The horizontal axis of the screen becomes calibrated in time only, rather than both frequency and time. Markers display amplitude and time values.

The following functions establish a clear display of the waveform:

- Trigger stabilizes the waveform trace on the display by triggering on the modulation envelope. If the modulation of the signal is stable, video trigger synchronizes the sweep with the demodulated waveform.
- Linear mode should be used in amplitude modulation (AM) measurements to avoid distortion caused by the logarithmic amplifier when demodulating signals.
- Sweep time adjusts the full sweep time from 5 ms to 2000 s. (20 µs to 2000 s if Option AYX is installed). The sweep time readout refers to the full 10-division graticule. Divide this value by 10 to determine sweep time per division.
- Resolution and video bandwidth are selected according to the signal bandwidth.

Each of the coupled function values remains at its current value when zero span is activated. Video bandwidth is coupled to resolution bandwidth. Sweep time is not coupled to any other function.

NOTE Refer to "Demodulating and Listening to an AM Signal" on page 91 for more information on signal demodulation.

To obtain an AM signal, you can either connect a source to the analyzer input and set the source for amplitude modulation, or connect an antenna to the analyzer input and tune to a commercial AM broadcast station.

Demodulating an AM Signal Example 1:

View the modulation waveform of an AM signal in the time domain.

1. Connect an RF signal source to the analyzer INPUT. For this example, an Agilent E4433B Signal Generator was used with the following settings:

   a. RF Frequency 300 MHz

b. RF Output Power –10 dBm

c. AM On

d. AM Rate 1 kHz

e. AM Depth 80%

2. Set the analyzer as follows:

a. Press **Preset**, **Factory Preset** (if present).

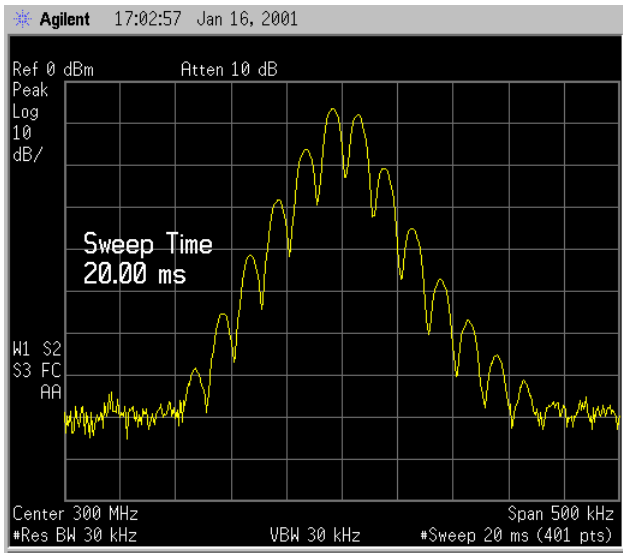
b. Set the center frequency to 300 MHz by pressing **FREQUENCY, Center Freq**, 300, **MHz**.

c. Set the span to 500 kHz by pressing **SPAN, Span**, 500, **kHz**.

d. Set the resolution bandwidth to 30 kHz by pressing **BW/Avg**, **Resolution BW**, 30, **kHz**.

e. Change the analyzer sweep to 20 msec by pressing **Sweep**, **Sweep Time**, 20, **ms**. See Figure 1-53.

Figure 1-53 Viewing an AM Signal



3. Set the Y-Axis Units to V by pressing **AMPLITUDE**, **More, Y-Axis Units, V**.

4. Position the signal peak near the reference level by pressing **AMPLITUDE** and rotating the front-panel knob.

5. Change the amplitude scale type to linear by pressing **AMPLITUDE**, **Scale Type** (Lin).

6. Select zero span by either pressing **SPAN**, 0, **Hz;** or pressing **SPAN**, **Zero Span**. See Figure 1-54.

7. Change the sweep time to 5 ms by pressing **Sweep**, **Sweep Time** (Man), 5, **ms**.
8. Since the modulation is a steady tone, you can use video trigger to trigger the analyzer sweep on the waveform and stabilize the trace, much like an oscilloscope by pressing **Trig**, **Video**, and adjusting the trigger level with the front-panel knob until the signal stabilizes. See Figure 1-55.

   If you are viewing an off-the-air signal you will not be able to stabilize the waveform.

NOTE If the Trigger Level is set too high or too low when this trigger mode is activated, the sweep will stop. You will need to adjust the trigger level up or down with the front-panel knob until the sweep begins again.

Figure 1-54 Measuring Modulation In Zero Span

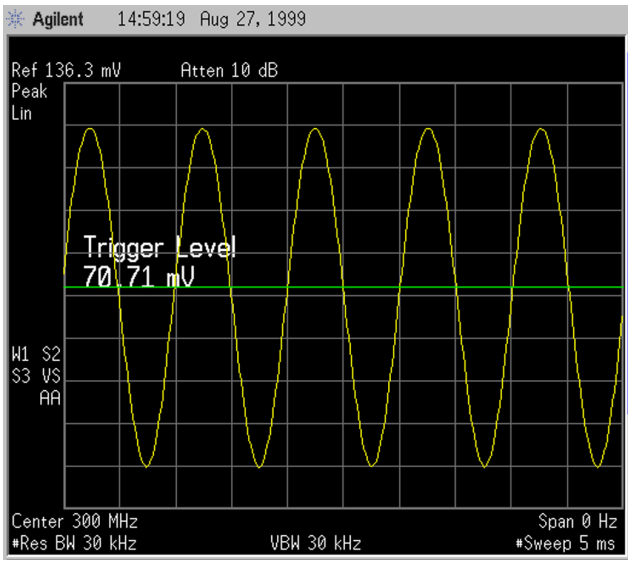


Figure 1-55 Measuring Modulation In Zero Span

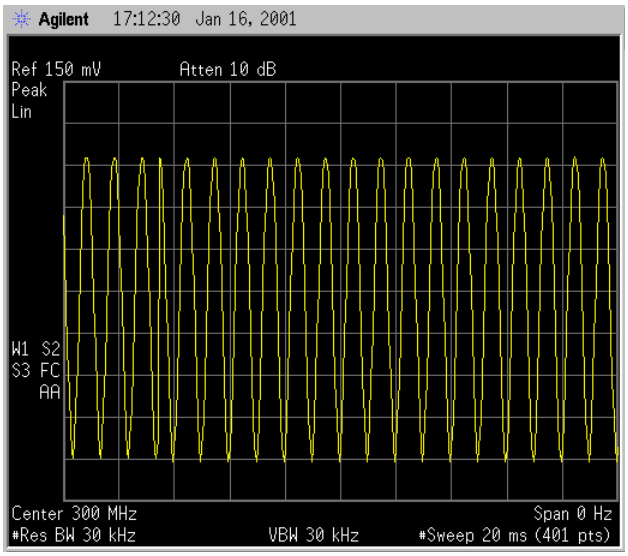
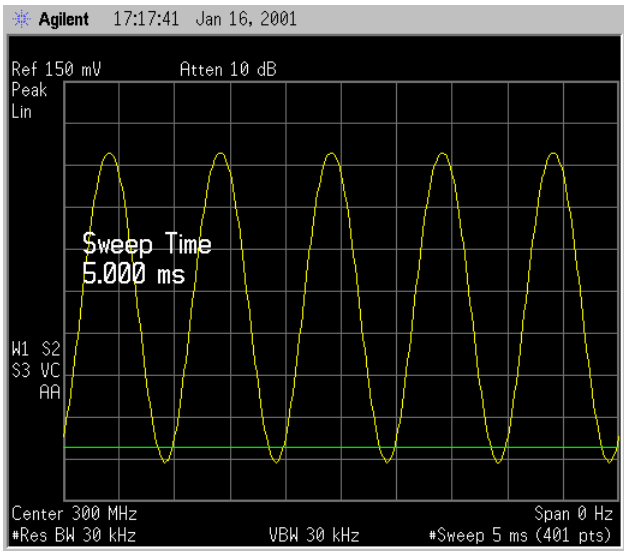


Figure 1-56 Measuring Modulation In Zero Span

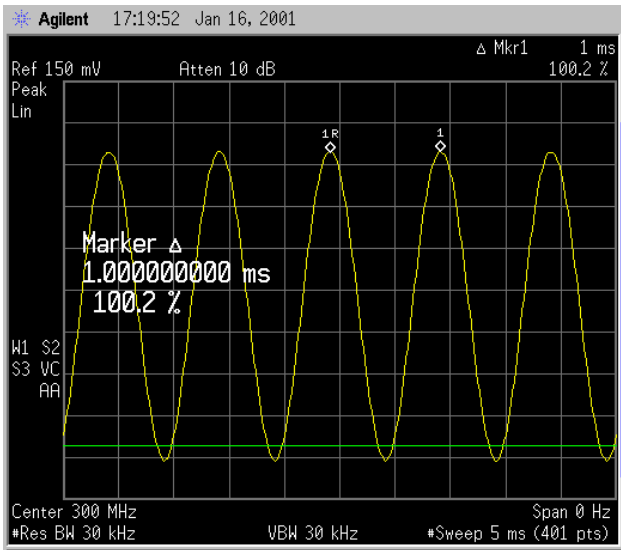


9. Use markers and delta markers to measure the time parameters of the waveform.

   a. Press **Marker** and center the marker on a peak using **Peak Search** or the front-panel knob.

   b. Press **Marker**, **Delta** and center the marker on the next peak using the front-panel knob or use **Peak Search** and **Next Pk Right** (or **Next Pk Left**). See Figure 1-57.

Figure 1-57 Measuring Time Parameters



10. You can turn your analyzer into a % AM indicator as follows:

    a. Set trigger to free run by pressing **Trig**, **Free Run**.

    b. Set the sweep time to 5 seconds by pressing **Sweep**, **Sweep Time**, 5, **s**.

    c. Set the video filter to 30 Hz by pressing **BW/Avg**, **Video BW**, 30, **Hz**.

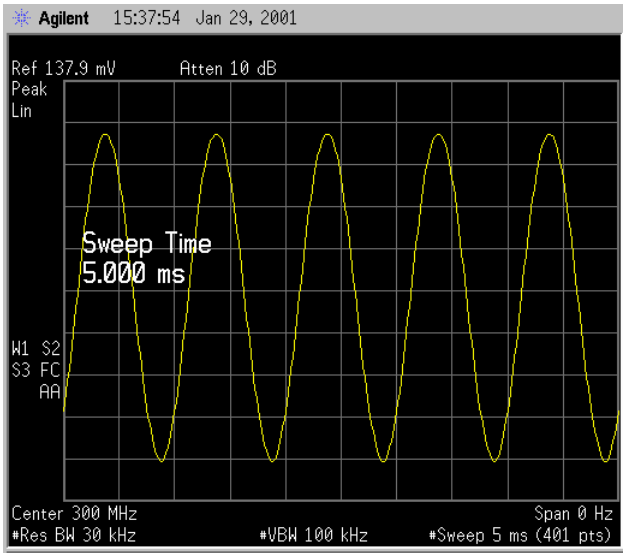
    d. Change the reference level to position the trace at midscreen by pressing **AMPLITUDE**, **Ref Level**, and adjacent the reference level using the front-panel knob.

    e. Reset the video filter to a high value. For example, press **BW/Avg**, **Video BW**, 100, **kHz**.

    f. Set the sweep time to 5 milliseconds by pressing **Sweep**, **Sweep Time**, 5, **ms**.

    The center horizontal line of the graticule now represents 0% AM; the top and bottom lines, 100% AM. See Figure 1-58.

Figure 1-58 Continuous Demodulation of an AM Signal



Demodulating FM Signals (Without Option BAA)

As with amplitude modulation (see page 62) you can utilize zero span to demodulate an FM signal. However, unlike the AM case, you cannot simply tune to the carrier frequency and widen the resolution bandwidth. The reason is that the envelope detector in the analyzer responds only to amplitude variations, and there is no change in amplitude if the frequency changes of the FM signal are limited to the flat part of the resolution bandwidth.

On the other hand, if you tune the analyzer slightly away from the carrier, you can utilize slope detection to demodulate the signal by performing the following steps.

1. Determine the correct resolution bandwidth.
2. Find the center of the linear portion of the filter skirt (either side).
3. Tune the analyzer to put the center point at mid screen of the display.
4. Select zero span.

The demodulated signal is now displayed; the frequency changes have been translated into amplitude changes., see Figure 1-61. To listen to the signal, turn on AM demodulation and the speaker.

In this example you will demodulate a broadcast FM signal that has a specified 75 kHz peak deviation.

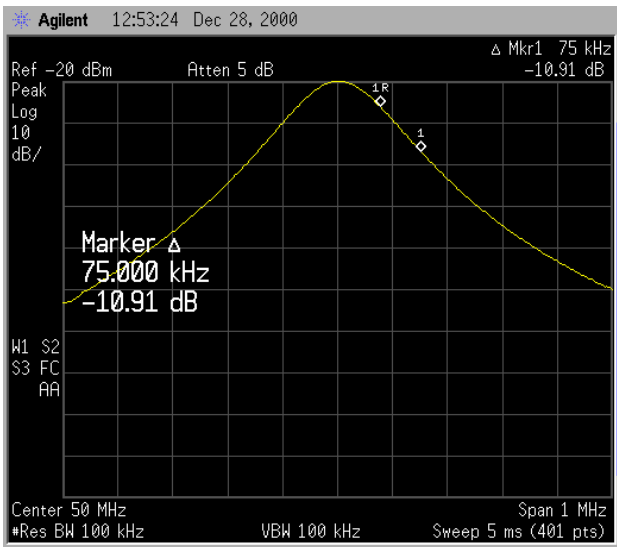
Demodulating a FM Signal Example:

Determine the correct resolution bandwidth. With a peak deviation of 75 kHz, your signal has a peak-to-peak excursion of 150 kHz. So we must find a resolution bandwidth filter with a skirt that is reasonably linear over that frequency range.

1. Perform a factory preset by pressing **Preset**, **Factory Preset** (if present).
2. Turn on the internal 50 MHz reference signal of the analyzer as follows:
   - For the E4401B and E4411B, use the internal 50 MHz amplitude reference signal of the analyzer as the signal being measured. Press **Input/Output**, **Amptd Ref** (On).
   - For all other models connect a cable between the front-panel AMPTD REF OUT to the analyzer INPUT, then press **Input/Output**, **Amptd Ref Out** (On).

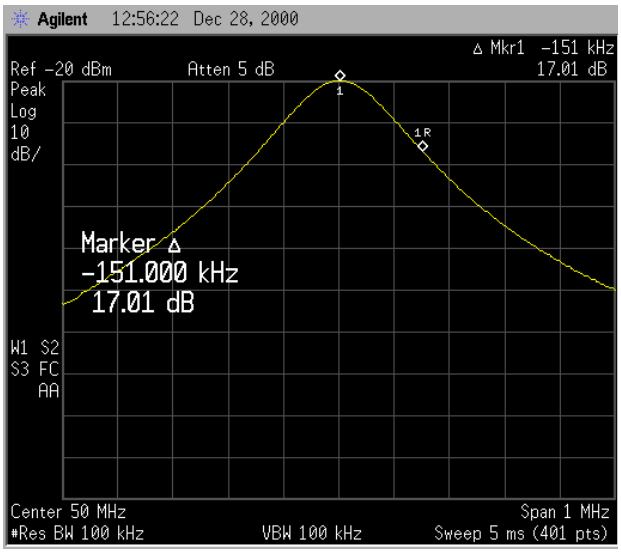
3. Set the center frequency to 50 MHz by pressing **FREQUENCY**, **Center Freq**, 50, **MHz**.
4. Set the span to 1 MHz by pressing **SPAN**, **Span**, 1, **MHz**.
5. Set the Y-Axis Units to dBm by pressing **AMPLITUDE**, **More**, **Y-Axis Units, dBm**.
6. Set the reference level to –20 dBm by pressing **AMPLITUDE**, **Ref Level**, –20, dB**m**.
7. Set the resolution bandwidth to 100 kHz by pressing **BW/Avg**, **Res BW**, 100, **kHz**. The skirt is reasonably linear starting approximately 5 dB below the peak.
8. Select a marker by pressing **Marker**, then move the marker approximately 1/2 division down the right of the peak (high frequency) using the front-panel knob.
9. Place a delta marker 150 kHz from the first marker by pressing **Delta**, 150, **kHz**. The skirt looks reasonably linear between markers.
10. Determine the offset from the signal peak to the desired point on the filter skirt by moving the delta marker to the midpoint. Press 75, **kHz** to move the delta marker to the midpoint. See Figure 1-59.

Figure 1-59 Establishing the Offset Point



11. Press **Delta** to make the active marker the reference marker.
12. Press **Peak Search** to move the delta marker to the peak. The delta value is the desired offset, for example 151 kHz. See Figure 1-60.

Figure 1-60 Determining the Offset



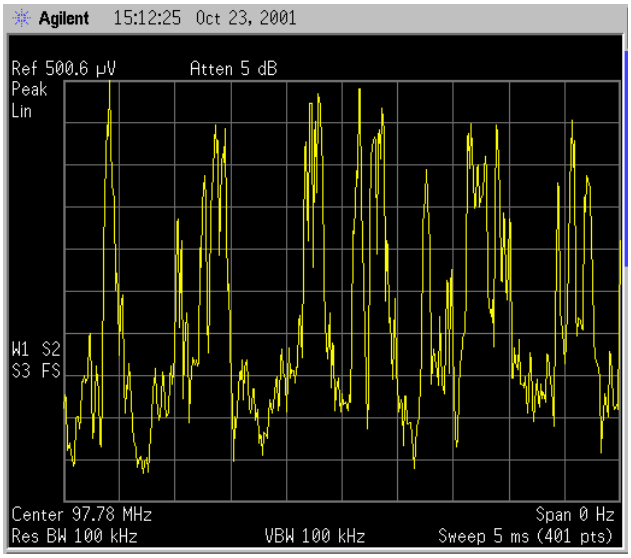
Demodulate the FM Signal

1. Connect an antenna to the analyzer INPUT.
2. Perform a factory preset by pressing **Preset**, **Factory Preset** (if present).
3. Tune the analyzer to a peak the peak of one of your local FM broadcast signals, for example 97.7 MHz by pressing **FREQUENCY**, **Center Freq**, 97.7, **MHz**.
4. Set the span to 1 MHz by pressing **SPAN**, **Span**, 1, **MHz**.
5. Press **AMPLITUDE**, **Ref Level**, and use the front-panel knob to bring the signal peak to the reference level.
6. Press **Scale Type** (Lin) to place the analyzer in linear scale mode.
7. Tune above or below the FM signal by the offset noted above in step 12, in this example 151 kHz. Press **FREQUENCY**, **CF Step,** 151, **kHz**, then press **Center Freq** and use the step-up key (↑) or step-down key (↓).
8. Set the resolution bandwidth to 100 kHz, by pressing **BW/Avg**, **Res BW**, 100, **kHz**.
9. Set the span to zero by pressing **SPAN**, **Zero Span**.
10. Turn off the automatic alignment by pressing **System**, **Alignments**, **Auto Align**, **Off**.

11. Listen to the demodulated signal through the speaker by pressing **Det/Demod**, **Demod**, **AM**, **Speaker** (On), then adjust the volume using the front-panel volume knob.

12. Activate single sweep by pressing **Single**. See Figure 1-61

Figure 1-61 Demodulating a Broadcast Signal



2 Making Complex Measurements

What's in This Chapter

This chapter provides information for making complex measurements. The procedures covered in this chapter are listed below.

To find descriptions of specific analyzer functions refer to the Agilent Technologies ESA Spectrum Analyzers User's Guide.

Required Test Equipment

Test Equipment	Specifications	Recommended Model
Adapters		
Type-N (m) to BNC (f) (2)		1250-0780
Cables		
(3) BNC, 122-cm (48-in)		10503A
Miscellaneous		
Bandpass Filter	Center Frequency: 200 MHz Bandwidth: 10 MHz	
Lowpass Filter	Cutoff Frequency: 10 MHz	

Making Stimulus Response Measurements

What Are Stimulus Response Measurements?

Stimulus response measurements require a source to stimulate a device under test (DUT), a receiver to analyze the frequency response characteristics of the DUT, and, for return loss measurements, a directional coupler or bridge. Characterization of a DUT can be made in terms of its transmission or reflection parameters. Examples of transmission measurements include flatness and rejection. Return loss is an example of a reflection measurement.

A spectrum analyzer combined with a tracking generator forms a stimulus response measurement system. With the tracking generator as the swept source and the analyzer as the receiver, operation is the same as a single channel scalar network analyzer. The tracking generator output frequency must be made to precisely track the analyzer input frequency for good narrow band operation. A narrow band system has a wide dynamic measurement range. This wide dynamic range will be illustrated in the following example.

Using An Analyzer With A Tracking Generator

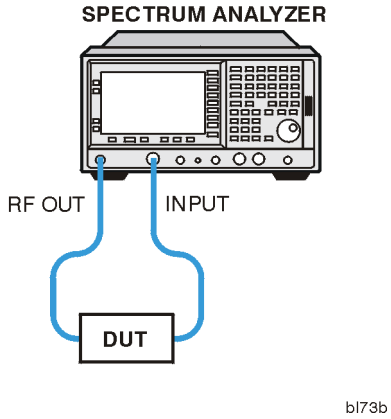
There are three basic steps in performing a stimulus response measurement, whether it is a transmission or a reflection measurement. The steps are to set all the analyzer settings, normalize, and measure.

The procedure below describes how to use a built in tracking generator system to measure the rejection of a band pass filter, a type of transmission measurement. Illustrated in this example are functions in the tracking generator menu such as adjusting the tracking generator output power. Normalization functions located in the trace menu are also used. Making a reflection measurement is similar and is covered in "Making a Reflection Calibration Measurement" on page 87.

Stepping Through a Transmission Measurement

1. To measure the rejection of a band pass filter, connect the equipment as shown in Figure 2-1. This example uses a 200 MHz bandpass filter as the DUT.

Figure 2-1 Transmission Measurement Test Setup

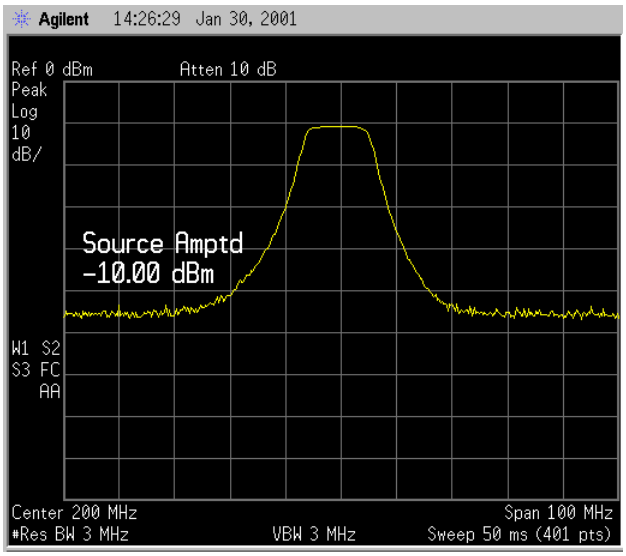


2. Perform a factory preset by pressing **Preset**, **Factory Preset** (if present).
3. Since we are only interested in the rejection of the bandpass filter, tune the analyzer center frequency and span to center the bandpass response and display the rejection ±50 MHz from the center of the bandpass.
   a. Set the span to 100 MHz by pressing **SPAN, Span**, 100, **MHz**.
   b. Set the center frequency to 200 MHz by pressing **FREQUENCY, Center Freq**, 200, **MHz**.
4. Set the resolution bandwidth to 3 MHz by pressing **BW/Avg**, **Res BW**, 3, **MHz**.
5. Turn on the tracking generator and if necessary, set the output power to –10 dBm by pressing **Source**, **Amplitude** (On), –10, **dBm**. See Figure 2-2.

CAUTION Excessive signal input may damage the DUT. Do not exceed the maximum power that the device under test can tolerate.

NOTE To reduce ripples caused by source return loss, use 10 dB (E4401B or E4411B) or 8 dB (all other models) or greater tracking generator output attenuation. Tracking generator output attenuation is normally a function of the source power selected. However, the output attenuation may be controlled in the **Source** menu. Refer to specifications and characteristics in your specifications guide for more information on the relationship between source power and source attenuation.

Figure 2-2 Tracking Generator Output Power Activated



6. Put the sweep time of the analyzer into stimulus response auto coupled mode by pressing **Sweep**, **Swp Coupling** (SR). Auto coupled sweep times are usually much faster for stimulus response measurements than they are for spectrum analyzer (SA) measurements. If necessary, adjust the reference level to place the signal on screen.

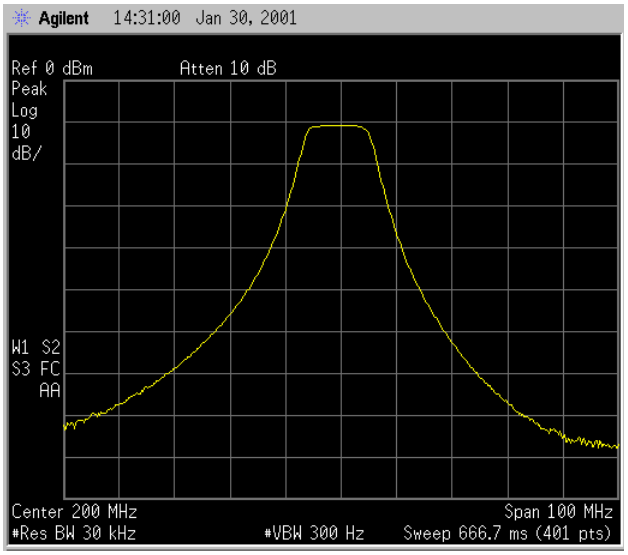
NOTE

In the stimulus response mode, the Q of the DUT can determine the fastest rate at which the analyzer can be swept. (Q is the quality factor, which is the center frequency of the DUT divided by the bandwidth of the DUT.) To determine whether the analyzer is sweeping too fast, slow the sweep and note whether there is a frequency or amplitude shift of the trace. Continue to slow the sweep until there is no longer a frequency or amplitude shift.

7. Decrease the resolution bandwidth to increase sensitivity by pressing **BW/Avg**, **Res BW**, and the step-down key (↓) until the sensitivity is increased. In Figure 2-3, the resolution bandwidth has been decreased to 30 kHz.

8. Narrow the video bandwidth to smooth the noise by pressing **BW/Avg**, **Video BW**, and the step-down key (↓) until the noise is reduced. In Figure 2-3, the video bandwidth has been decreased to 300 Hz.

Figure 2-3 Decrease the Resolution Bandwidth to Improve Sensitivity



9. You might notice a decrease in the displayed amplitude as the resolution bandwidth is decreased, (if the analyzer is an E4402B, E4403B, E4404B, E4405B, E4407B, or E4408B). This indicates the need for performing a tracking peak. Press **Source**, **Tracking Peak**. The amplitude should return to that which was displayed prior to the decrease in resolution bandwidth.

10. To make a transmission measurement accurately, the frequency response of the test system must be known. Normalization is used to eliminate this error from the measurement. To measure the frequency response of the test system, connect the cable (but not the DUT) from the tracking generator output to the analyzer input. Press **Trace/View**, **More**, **Normalize**, **Store Ref** (1→3), **Normalize** (On). The frequency response of the test system is automatically stored in trace 3 and a normalization is performed. This means that the active displayed trace is now the ratio of the input data to the data stored in trace 3. (The reference trace is Trace 3 with firmware revision A.04.00 and later)

    When normalization is on, trace math is being performed on the active trace. The trace math performed is (trace 1 – trace 3 + the normalized reference position), with the result placed into trace 1. Remember that trace 1 contains the measurement trace, trace 3 contains the stored reference trace of the system frequency response, and normalized reference position is indicated by arrowheads at the edges of the graticule.

NOTE Since the reference trace is stored in trace 3, changing trace 3 to Clear Write will invalidate the normalization.

11. Reconnect the DUT to the analyzer. Note that the units of the reference level have changed to dB, indicating that this is now a relative measurement.

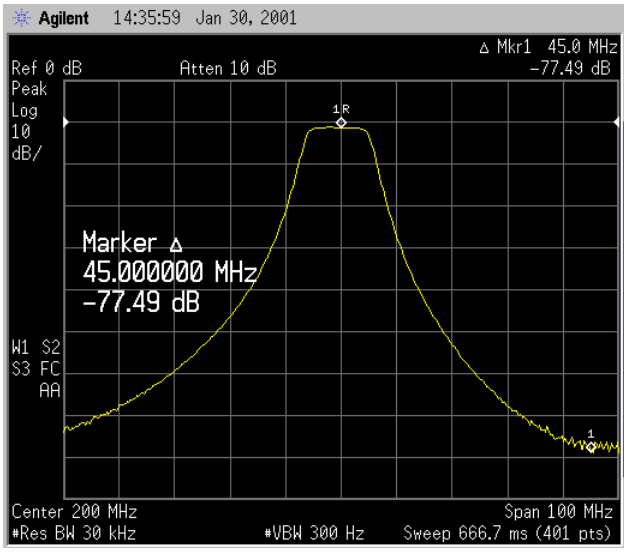
    Press **Trace/View, More, Normalize, Norm Ref Posn** to change the normalized reference position. Arrowheads at the left and right edges of the graticule mark the normalized reference position, or the position where 0 dB insertion loss (transmission measurements) or 0 dB return loss (reflection measurements) will normally reside. Using the knob results in a change in the position of the normalized trace, within the range of the graticule.

12. To measure the rejection of the filter 45 MHz above the center of the bandpass, press **Marker**, 200, **MHz** (to ensure that the marker is in the center of the signal), and then press **Delta**, 45, **MHz**. The marker readout displays the rejection of the filter at 45 MHz above the center of the bandpass. See Figure 2-4.

NOTE

Because the default trace is comprised of 401 discrete points, the indicated marker frequency may differ slightly from the frequency that you entered. Due to the horizontal resolution of the trace, the marker frequency value will be rounded to within 0.25% of the span of the value entered. If the analyzer is an ESA-E series with firmware revision A.04.00 or later, the number of sweep points may be set to any value between 101 and 8192.

Figure 2-4 Measure the Rejection Range



Tracking Generator Unleveled Condition

When using the tracking generator, the message **TG unleveled** may appear. The TG unleveled message indicates that the tracking generator source power (**Source**, **Amplitude**) could not be maintained at the selected level during some portion of the sweep. If the unleveled condition exists at the beginning of the sweep, the message will be displayed immediately. If the unleveled condition occurs after the sweep begins, the message will be displayed after the sweep is completed. A momentary unleveled condition may not be detected when the sweep time is short. The message will be cleared after a sweep is completed with no unleveled conditions.

The unleveled condition may be caused by any of the following:

- Start frequency is too low or the stop frequency is too high. The unleveled condition is likely to occur if the true frequency range exceeds the tracking generator frequency specification (especially the low frequency specification).
- Source attenuation may be set incorrectly (select **Attenuation** (Auto) for optimum setting).
- The source power may be set too high or too low, use **Amplitude** (Off) then **Amplitude** (On) to reset it.
- The source power sweep may be set too high, resulting in an unleveled condition at the end of the sweep. Use **Power Sweep** (Off) then **Power Sweep** (On) to decrease the amplitude.
- Reverse RF power from the device under test detected by the tracking generator ALC (automatic level control) system.

Measuring Device Bandwidth

It is often necessary to measure device bandwidth, such as when testing a bandpass filter. There is a key in the **Peak Search** menu that will perform this function. The device signal being measured must be displayed before activating the measurement. The span must include the full response.

Activate the measurement by toggling the **N dB Points** key to On. The analyzer places arrow markers at the –3 dB points on either side of the response and reads the bandwidth. For other bandwidth responses enter the number of dB down desired, from –1 dB to –80 dB.

No other signal can appear on the display within N dB of the highest signal. The measured signal cannot have more than one peak that is greater than or equal to N dB. A signal must have a peak greater than the currently defined peak excursion to be identified. The default value for the peak excursion is 6 dB.

Measurements are made continuously, updating at the end of each sweep. This allows you to make adjustments and see changes as they happen. The single sweep mode can also be used, providing time to study or record the data.

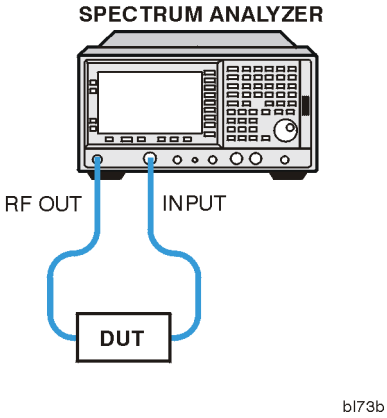
The N dB bandwidth measurement error is typically ±1% of the span.

Example:

Measure the 3 dB bandwidth of a 200 MHz bandpass filter.

1. To measure the rejection of a bandpass filter, connect the equipment as shown in Figure 2-5. This example uses a 200 MHz bandpass filter.

Figure 2-5 Transmission Measurement Test Setup



2. Perform a factory preset by pressing **Preset**, **Factory Preset** (if present).
3. Set the span to 100 MHz by pressing **SPAN, Span**, 100, **MHz**.
4. Set the center frequency to 200 MHz by pressing **FREQUENCY, Center Freq**, 200, **MHz**.
5. Set the resolution bandwidth to 10 kHz by pressing **BW/Avg**, **Res BW**, 10, **kHz**.
6. Turn on the tracking generator and if necessary, set the output power to –10 dBm by pressing **Source**, **Amplitude** (On), –10, **dBm**.

CAUTION Excessive signal input may damage the DUT. Do not exceed the maximum power that the device under test can tolerate.

NOTE To reduce ripples caused by source return loss, use 10 dB (E4401B or E4411B) or 8 dB (all other models) or greater tracking generator output attenuation. Tracking generator output attenuation is normally a function of the source power selected. However, the output attenuation may be controlled in the Source menu. Refer to specifications and characteristics in your specifications guide for more information on the relationship between source power and source attenuation.

7. Put the sweep time of the analyzer into stimulus response auto coupled mode by pressing **Sweep**, then **Swp Coupling** (SR). Auto coupled sweep times are usually much faster for stimulus response measurements than they are for spectrum analyzer (SA) measurements. Adjust the reference level if necessary to place the signal on screen.

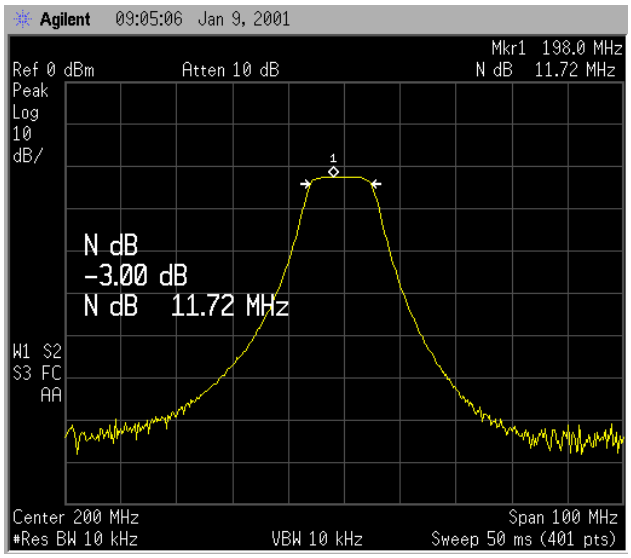
NOTE

In the stimulus response mode, the Q of the DUT can determine the fastest rate at which the analyzer can be swept. (Q is the quality factor, which is the center frequency of the DUT divided by the bandwidth of the DUT.) To determine whether the analyzer is sweeping too fast, slow the sweep and note whether there is a frequency or amplitude shift of the trace. Continue to slow the sweep until there is no longer a frequency or amplitude shift.

8. To activate the N dB bandwidth function press **Peak Search**, **More**, then **N dB Points** (On). See Figure 2-6.

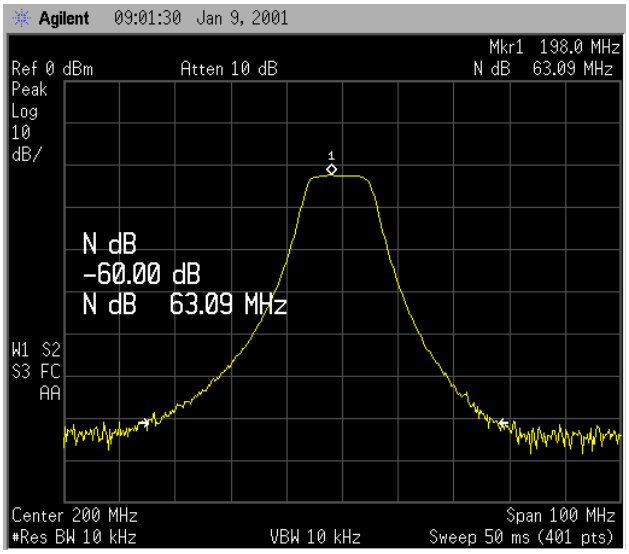
9. Read the measurement results displayed on the screen.

Figure 2-6 N dB Bandwidth Measurement at –3 dB



10. The knob or the data entry keys can be used to change the N dB value from –3 dB to –60 dB to measure the 60 dB bandwidth of the filter. See Figure 2-7.

Figure 2-7 N dB Bandwidth Measurement at –60 dB



11.Press **N dB Points** (Off) to turn the measurement off.

Measuring Stop Band Attenuation Using Log Sweep (E-Series Analyzers only)

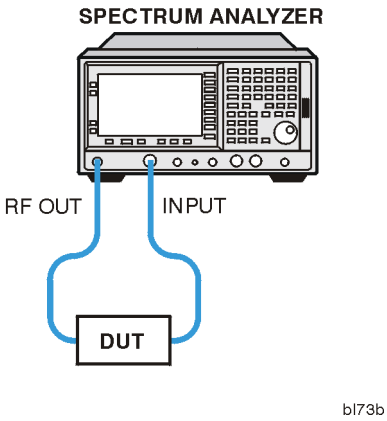
When measuring filter characteristics, it is useful to look at the stimulus response over a wide frequency range. Setting the analyzer x-axis (frequency) to display logarithmically provides this function.

Example:

Measure the stop band attenuation of a 10 MHz low pass filter.

1. To measure the response of a low pass filter, connect the equipment as shown in Figure 2-8. This example uses a 10 MHz low pass filter.

Figure 2-8 Transmission Measurement Test Setup

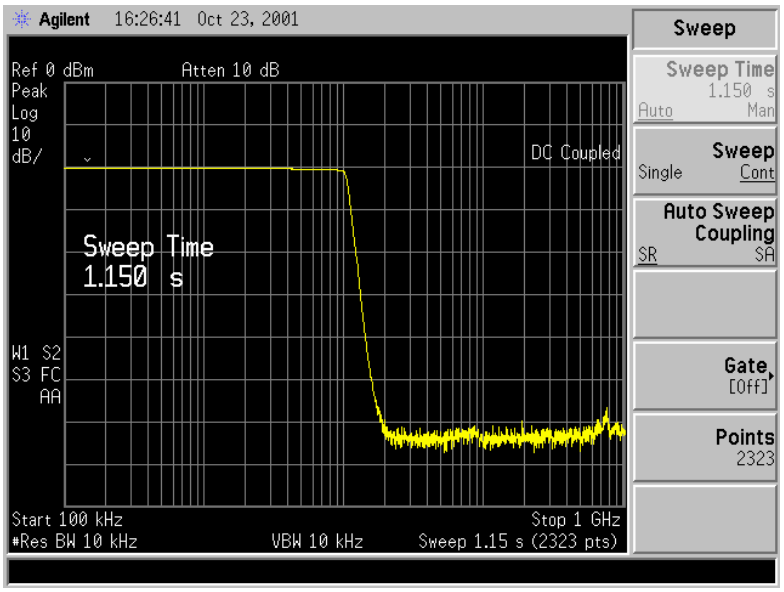


2. Perform a factory preset by pressing **Preset**, **Factory Preset** (if present).
3. Set the start frequency to 100 kHz by pressing **FREQUENCY, Start Freq**, 100, **kHz**.
4. Set the stop frequency to 1 GHz by pressing **Stop Freq,** 1, **GHz**.
5. Set the resolution bandwidth to 10 kHz by pressing **BW/Avg**, **Res BW**, 10, **kHz**.
6. Set the frequency scale to log by pressing **FREQUENCY**, **Scale Type** (Log).
7. For E4407B analyzers with Option UKB, set the input coupling to DC by pressing **Input**, **Coupling** (DC).
8. Turn on the tracking generator and if necessary, set the output power to –10 dBm by pressing **Source**, **Amplitude** (On), –10, **dBm**.

CAUTION Excessive signal input may damage the DUT. Do not exceed the maximum power that the device under test can tolerate.

NOTE To reduce ripples caused by source return loss, use 10 dB (E4401B) or 8 dB (all other models) or greater tracking generator output attenuation. Tracking generator output attenuation is normally a function of the source power selected. However, the output attenuation may be controlled in the **Source** menu. Refer to specifications and characteristics in your specifications guide for more information on the relationship between source power and source attenuation.

Figure 2-9 Tracking Generator Output Power Activated in Log Sweep



9. Put the sweep time of the analyzer into stimulus response auto coupled mode by pressing **Sweep**, then **Swp Coupling** (SR). See Figure 2-9. Auto coupled sweep times are usually much faster for stimulus response measurements than they are for spectrum analyzer (SA) measurements. Adjust the reference level if necessary to place the signal on screen.

10. To make a transmission measurement accurately, the frequency response of the test system must be known. Normalization is used to eliminate this error from the measurement. To measure the frequency response of the test system, connect the cable (but not the DUT) from the tracking generator output to the analyzer input. Press **Trace/View**, **More**, **Normalize**, **Store Ref** (1→3), **Normalize** (On).

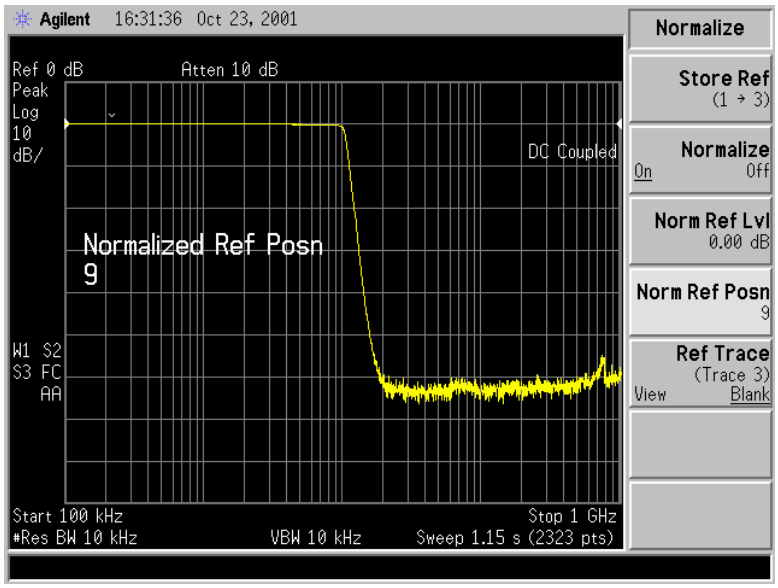
    This will activate the trace 1 minus trace 3 function and display the results in trace 1. The normalized trace or flat line represents 0 dB return loss. Normalization occurs each sweep.

NOTE Since the calibration trace is stored in trace 3, changing trace 3 to Clear Write, Max Hold, or Min Hold will invalidate the normalization.

11. Reconnect the DUT to the analyzer. Note that the units of the reference level have changed to dB, indicating that this is now a relative measurement. Refer to Figure 2-10.

12. Press **Trace/View, More, Normalize, Norm Ref Posn** to change the normalized reference position. Arrowheads at the left and right edges of the graticule mark the normalized reference position. Using the knob results in a change in the position of the normalized trace, within the range of the graticule. Refer to Figure 2-10.

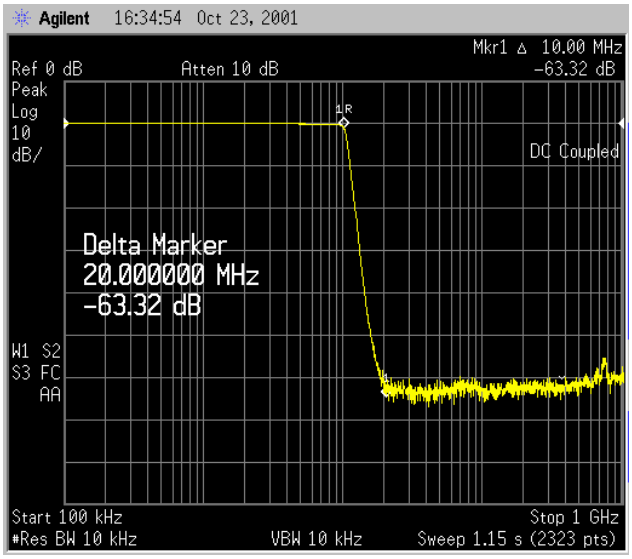
Figure 2-10 Normalized Trace After Reconnecting DUT



13. Press **Marker**, **Delta Pair** (Ref), 10, **MHz** to place the reference marker at the specified cutoff frequency.

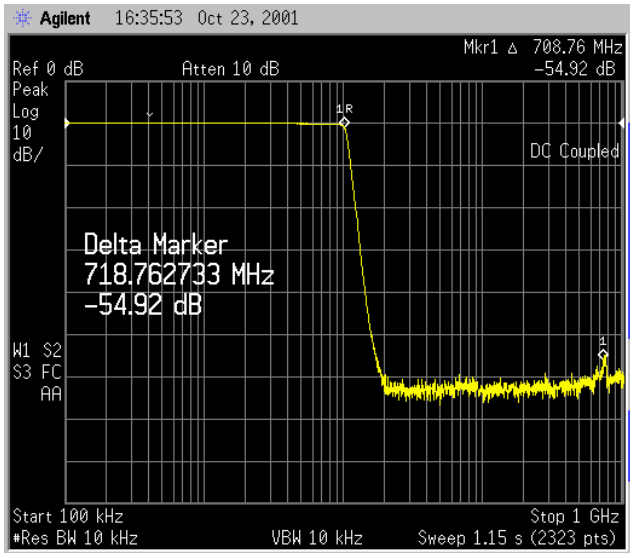
14. Press **Delta Pair** (Δ), 20, **MHz** to place the second marker at the 20 MHz point. In this example, the attenuation over this frequency range is 63.32 dB/octave (one octave above the cutoff frequency).

Figure 2-11 Determining Low Pass Filter Rolloff



15. Use the knob to place the marker at the highest peak in the stop band to determine the minimum stop band attenuation. In this example, the peak occurs at 708.76 MHz. The attenuation is 54.92 dB.

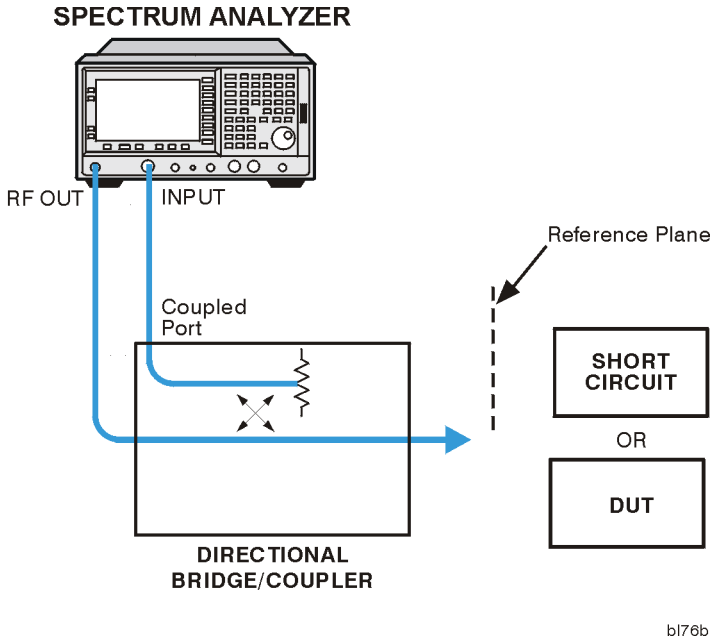
Figure 2-12 Minimum Stop Band Attenuation



Making a Reflection Calibration Measurement

The calibration standard for reflection measurements is usually a short circuit connected at the reference plane (the point at which the device under test (DUT) will be connected.) See Figure 2-13. A short circuit has a reflection coefficient of 1 (0 dB return loss). It reflects all incident power and provides a convenient 0 dB reference.

Figure 2-13 Reflection Measurement Short Calibration Test Setup



Example:

Measure the return loss of a filter. The following procedure makes a reflection measurement using a coupler or directional bridge. This example uses a 200 MHz bandpass filter as the DUT.

NOTE The analyzer center frequency and span for this measurement can easily be set up using the transmission measurement setup in "Making Stimulus Response Measurements" on page 75. Tune the analyzer so that the passband of the filter comprises a majority of the display, then proceed with the steps outlined below.

Reflection Calibration

1. Connect the DUT to the directional bridge or coupler as shown in Figure 2-13. Terminate the unconnected port of the DUT.

NOTE If possible, use a coupler or bridge with the correct test port connector for both calibrating and measuring. Any adapter between the test port and DUT degrades coupler/bridge directivity and system source match. Ideally, you should use the same adapter for the calibration and the measurement. Be sure to terminate the second port of a two port device.

2. Connect the tracking generator output of the analyzer to the directional bridge or coupler.
3. Connect the analyzer input to the *coupled* port of the directional bridge or coupler.
4. Perform a factory preset by pressing **Preset**, **Factory Preset** (if present).
5. Turn on the tracking generator and if necessary, set the output power to –10 dBm by pressing **Source**, **Amplitude** (On), –10, **dBm**.

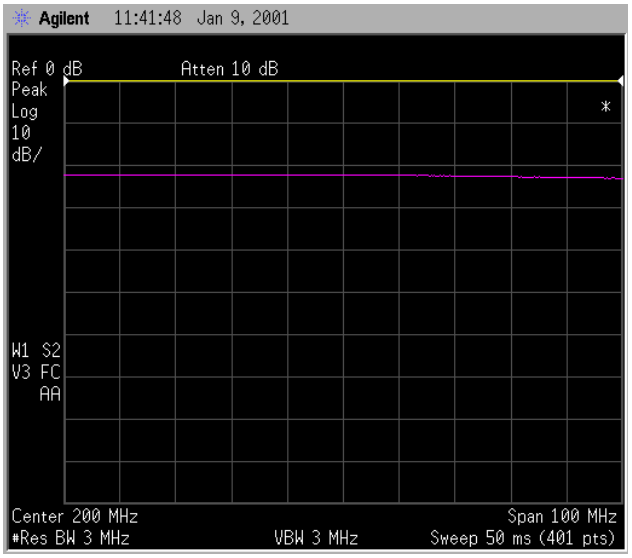
CAUTION Excessive signal input may damage the DUT. Do not exceed the maximum power that the device under test can tolerate.

6. Set the span to 100 MHz by pressing **SPAN, Span**, 100, **MHz**.
7. Set the center frequency to 200 MHz by pressing **FREQUENCY, Center Freq**, 200, **MHz**.
8. Set the resolution bandwidth to 3 MHz by pressing **BW/Avg**, **Res BW**, 3, **MHz**.
9. Replace the DUT with a short circuit.
10. Normalize the trace by pressing **Trace/View**, **More**, **Normalize**, **Store Ref** (1→3), **Normalize** (On). See Figure 2-14.

    This will activate the trace 1 minus trace 3 function and display the results in trace 1. The normalized trace or flat line represents 0 dB return loss. Normalization occurs each sweep. Replace the short circuit with the DUT.

NOTE Since the reference trace is stored in trace 3, changing trace 3 to Clear Write will invalidate the normalization.

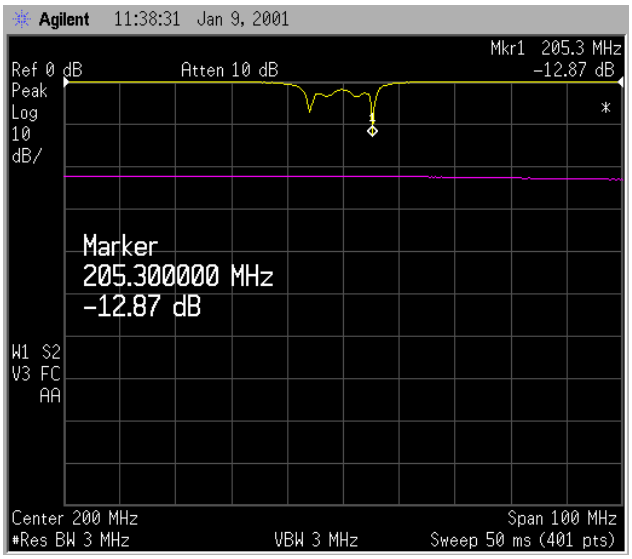
Figure 2-14 Short Circuit Normalized



Measuring the Return Loss

1. After calibrating the system with the above procedure, reconnect the filter in place of the short circuit without changing any analyzer settings.
2. Use the marker to read return loss. Press **Marker** and position the marker with the knob to read the return loss at that frequency. Or you can use the Min Search function to measure return loss by pressing **Peak Search**, **Min Search**, the analyzer will place a marker at the point where the return loss is maximized. See Figure 2-15.

Figure 2-15 Measuring the Return Loss of the Filter



Converting Return Loss to VSWR

Return loss can be expressed as a voltage standing wave ratio (VSWR) value using the following table or formula:

Table 2-1 Power to VSWR Conversion

Return Loss (dB)	VSWR	Return Loss (dB)	VSWR	Return Loss (dB)	VSWR	Return Loss (dB)	VSWR	Return Loss (dB)	VSWR
4.0	4.42	14.0	1.50	18.0	1.29	28.0	1.08	38.0	1.03
6.0	3.01	14.2	1.48	18.5	1.27	28.5	1.08	38.5	1.02
8.0	2.32	14.4	1.47	19.0	1.25	29.0	1.07	39.0	1.02
10.0	1.92	14.6	1.46	19.5	1.24	29.5	1.07	39.5	1.02
10.5	1.85	14.8	1.44	20.0	1.22	30.0	1.07	40.0	1.02
11.0	1.78	15.0	1.43	20.5	1.21	30.5	1.06	40.5	1.02
11.2	1.76	15.2	1.42	21.0	1.20	31.0	1.06	41.0	1.02
11.4	1.74	15.4	1.41	21.5	1.18	31.5	1.05	41.5	1.02
11.6	1.71	15.6	1.40	22.0	1.17	32.0	1.05	42.0	1.02
11.8	1.69	15.8	1.39	22.5	1.16	32.5	1.05	42.5	1.02
12.0	1.67	16.0	1.38	23.0	1.15	33.0	1.05	43.0	1.01
12.2	1.65	16.2	1.37	23.5	1.14	33.5	1.04	43.5	1.01
12.4	1.63	16.4	1.36	24.0	1.13	34.0	1.04	44.0	1.01
12.6	1.61	16.6	1.35	24.5	1.13	34.5	1.04	44.5	1.01
12.8	1.59	16.8	1.34	25.0	1.12	35.0	1.04	45.0	1.01
13.0	1.58	17.0	1.33	25.5	1.11	35.5	1.03	45.5	1.01
13.2	1.56	17.2	1.32	26.0	1.11	36.0	1.03	46.0	1.01
13.4	1.54	17.4	1.31	26.5	1.10	36.5	1.03	46.5	1.01
13.6	1.53	17.6	1.30	27.0	1.09	37.0	1.03	47.0	1.01
13.8	1.51	17.8	1.30	27.5	1.09	37.5	1.03	47.5	1.01

$$VSWR = \frac{1 + 10^{\frac{-RL}{20}}}{1 - 10^{\frac{-RL}{20}}}$$

Where: RL is the measured return loss value.

VSWR is sometimes stated as a ratio. For example: 1.2:1 “one point two to one” VSWR. The first number is the VSWR value taken from the table or calculated using the formula. The second number is always 1.

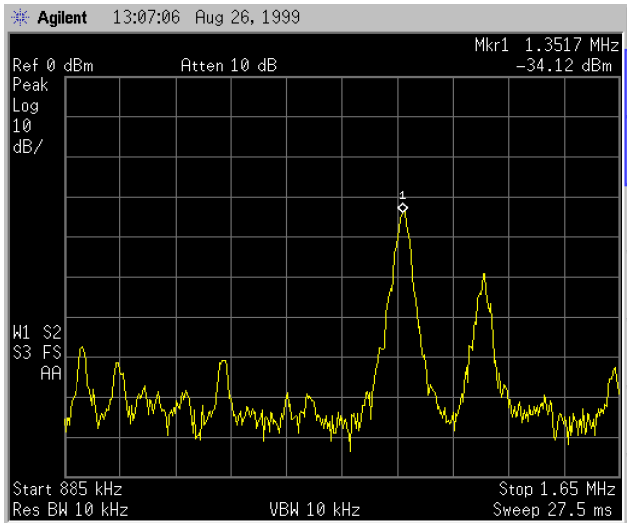
Demodulating and Listening to an AM Signal

The functions listed in the menu under **Det/Demod** allow you to demodulate and hear signal information displayed on the analyzer. Simply place a marker on a signal of interest, activate AM demodulation, turn the speaker on, and then listen.

Demodulating and Listening to an AM Signal Example 1:

1. Connect an antenna to the analyzer input.
2. Perform a factory preset by pressing **Preset**, **Factory Preset** (if present).
3. Select a frequency range on the analyzer, such as the range for AM radio broadcasts. For example, the frequency range for AM broadcasts in the United States is 550 kHz to 1650 kHz. Press **FREQUENCY**, **Start Freq**, 550, **kHz**, **Stop Freq**, 1650, **kHz**.
4. Place a marker on the signal of interest. Press **Peak Search** to place a marker on the highest amplitude signal, or press **Marker**, **Normal** and move the marker to a signal of interest. See Figure 2-16.
5. Press **Det/Demod**, **Demod**, **AM**. Use the front-panel volume knob to control the speaker volume.

Figure 2-16 Demodulation of an AM Signal



6. The signal is demodulated at the marker position only for the duration of the demod time. Use the step keys, knob, or numeric keypad to change the dwell time. For example, press the step up key (↑) to increase the dwell time to 2 seconds.
7. The marker search functions can be used to move the marker to

other signals of interest. Press **Peak Search** to access **Next Peak**, **Next Pk Right**, or **Next Pk Left**.

Demodulating and Listening to an AM Signal Example 2:

1. Connect an antenna to the analyzer input.
2. Perform a factory preset by pressing **Preset**, **Factory Preset** (if present).
3. Select a frequency range on the analyzer, such as the range for AM radio broadcasts. For example, the frequency range for AM broadcasts in the United States is 550 kHz to 1650 kHz. Press **FREQUENCY**, **Start Freq**, 550, **kHz**, **Stop Freq**, 1650, **kHz**.
4. Place a marker on the signal of interest. Press **Peak Search** to place a marker on the highest amplitude signal, or press **Marker**, **Normal** and move the marker to a signal of interest.
5. Set the frequency of the signal to center frequency by pressing **FREQUENCY** then **Signal Track** (On) if the signal of interest is the highest amplitude on screen signal. If it is not the highest amplitude signal on screen, move the signal to center screen by pressing **Peak Search**, Marker→, and **Mkr→CF**.
6. If the signal track function is on, press **SPAN**, 1, **MHz** to reduce the span to 1 MHz. If signal track is not used, use the step down key (↓) to reduce the span and use **Mkr→CF** to keep the signal of interest at center screen.
7. Set the span to zero by pressing **SPAN, Zero Span**. **Zero Span** turns off the signal track function.
8. Change the resolution bandwidth to 100 kHz by pressing **BW/Avg**, **Res BW**, then enter 100, **kHz**.
9. Set the signal in the top two divisions of the screen by changing the reference level. Press **AMPLITUDE**, and then the step down key (↓) until the signal is in the top two divisions. Set the amplitude scale to linear by pressing **Scale Type (**Lin). Keep the signal center displayed by pressing **AMPLITUDE, Ref Level** and using the (↑) (↓) step keys.
10. Press **Det/Demod**, **Detector**, **Sample** to set the detector mode of the analyzer to Sample.
11. Press **Det/Demod**, **Demod**, **AM**. Use the front panel volume knob to control the speaker volume.

12. You can turn your analyzer into a % AM indicator as follows:

    a. Set trigger to free run by pressing **Trig**, **Free Run**.

    b. Set the sweep time to 5 seconds by pressing **Sweep**, **Sweep Time**, 5, **s**.

    c. Set the video filter to 30 Hz by pressing **BW/Avg**, **Video BW**, 30, **Hz**.

    d. Change the reference level to position the trace at midscreen by pressing **AMPLITUDE**, **Ref Level**, and adjust the reference level using the front-panel knob.

    e. Reset the video filter to a high value. For example, press **BW/Avg**, **Video BW**, 30, **kHz**.

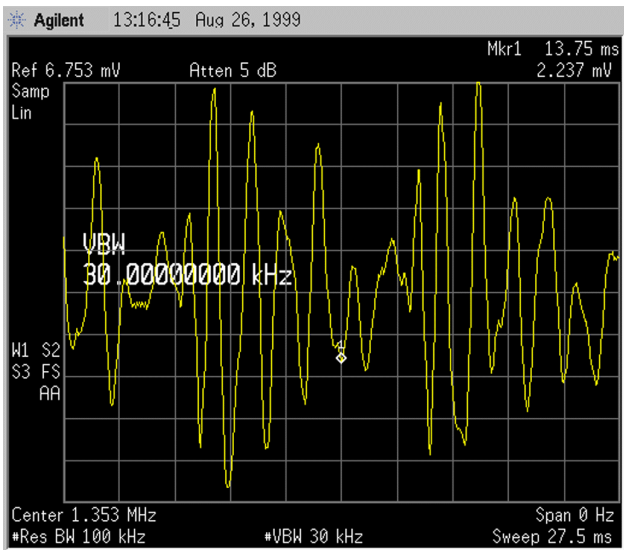
    f. Set the sweep time to 5 milliseconds by pressing **Sweep**, **Sweep Time**, 5, **ms**.

    The center horizontal line of the graticule now represents 0% AM; the top and bottom lines, 100% AM. See Figure 2-17

13. The signal to the speaker will be interrupted during retrace because the analyzer is performing automatic alignment routines. To eliminate the interruption and clicks between sweeps, turn the auto alignment function off by pressing **System**, **Alignments**, **Auto Align**, **Off**.

NOTE Refer to the specifications for information about operating the analyzer with the alignments turned off.

Figure 2-17 Continuous Demodulation of an AM Signal



Measuring Harmonics and Harmonic Distortion

The analyzer provides a measurement key that automates harmonic measurements (from the second to the tenth harmonic) and provides a calculation of the total harmonic distortion for continuous wave signals or complex digitally modulated carriers.

NOTE This measurement assumes that the peak signal found in the current span of the analyzer is the desired fundamental frequency.

When the harmonic distortion measurement is activated, the analyzer searches for the fundamental and determines the frequencies of the harmonics. The analyzer then changes to zero span, and measures the amplitude of each harmonic. The analyzer calculates the total harmonic distortion by dividing the root-sum-squares of the harmonic voltages by the fundamental signal voltage and then provides the result as a percentage.

$$\%THD = 100 \times \frac{\left(\sqrt{\sum_{h=2}^{H_{max}} E_h^{\,2}} \right)}{E_f}$$

Where:

%THD = Total Harmonic Distortion as a percentage
h = harmonic number
H_{max} = Maximum Harmonic Value listed
E_h = voltage of harmonic h
E_f = voltage of fundamental signal

Example of a THD calculation:

If the number of harmonics selected is 5 (H_{max} = 5) and the measured values are as follows:

E_f = 5 dBm = 3.162 mW = 397.6 mV

E_2 = –42 dBc = –37 dBm = 199.5 nW = 3.159 mV

E_3 = –26 dBc = –21 dBm = 7.943 µW = 19.93 mV

E_4 = –49 dBc = –44 dBm = 39.81 nW = 1.411 mV

E_5 = –36 dBc = –31 dBm = 794.3 nW = 6.302 mV

then,

$$\text{THD} = 100 \times \frac{\sqrt{3.159\ \text{mV}^2 + 19.93\ \text{mV}^2 + 1.411\text{mV}^2 + 6.301\ \text{mV}^2}}{397.6\ \text{mV}} = 5.33\%$$

Measuring Harmonics and Harmonic Distortion Example 1:

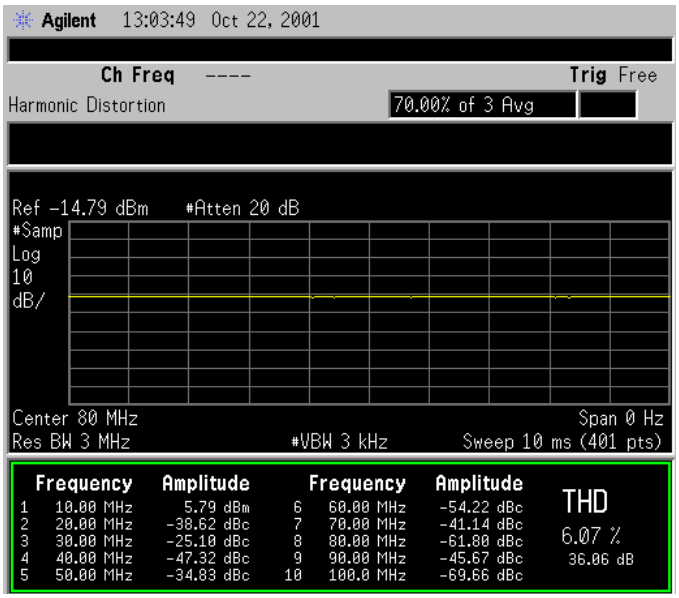
In this example, the 10 MHz Reference Output is used as the fundamental source. The harmonics and total harmonic distortion are measured.

1. Reset the analyzer by pressing **Preset**, **Factory Preset** (if present).
2. Connect the 10 MHz Reference Output from the rear of the analyzer to the analyzer INPUT with a BNC Cable.
3. Center the frequency on the incoming 10 MHz signal by pressing **Frequency**, 10, **MHz**.
4. Set the resolution bandwidth to 300 kHz by pressing **BW/Avg**, **Res BW**, 300, **kHz**.

NOTE Setting the resolution bandwidth to approximately 3% of the fundamental frequency before activating the harmonic distortion measurement increases measurement accuracy.

5. Press **Measure**, **More**, **Harmonic Distortion**.
6. Press the **Meas Setup**, **Avg Number** (On), 3, **Enter** to obtain the average of 3 harmonic distortion measurements.
7. Press **Avg Mode** (Exp) to continuously affect the result of the THD with subsequent sweeps. (Selecting **Avg Mode** (Repeat) clears the value calculated for THD after the total specified number of averages has been taken.)
8. Set the number of measured harmonics to 10 by pressing **Harmonics**, 10, **Enter**.
9. Press **ST/Harmonic** (Auto) to allow the analyzer to determine the optimum sweep time.
10. Press **Optimize Ref Level** to ensure all the harmonics are within the range of measurement.

Figure 2-18 Measuring the Harmonic Distortion



11. The amplitudes of the harmonics are listed relative to the fundamental frequency. To display the total harmonic distortion, press **Trace/View**, **Harmonics & THD**.

Measuring Harmonics and Harmonic Distortion on an IS-95 signal at 870.03 MHz Example 2:

In this example, an IS-95 signal is used as the fundamental source. The harmonics and total harmonic distortion are measured.

1. Reset the analyzer by pressing **Preset**, **Factory Preset** (if present).
2. Connect an IS-95 signal at 870.03 Mhz to the analyzer.
3. Set the center frequency to 870.03 MHz by pressing **FREQUENCY**, **Center Freq**, 870.03, **MHz**.
4. Set the resolution bandwidth to 300 kHz by pressing **BW/Avg**, **Res BW**, 300, **kHz**.

NOTE Setting the resolution bandwidth to approximately 3% of the fundamental frequency before activating the harmonic distortion measurement increases measurement accuracy.

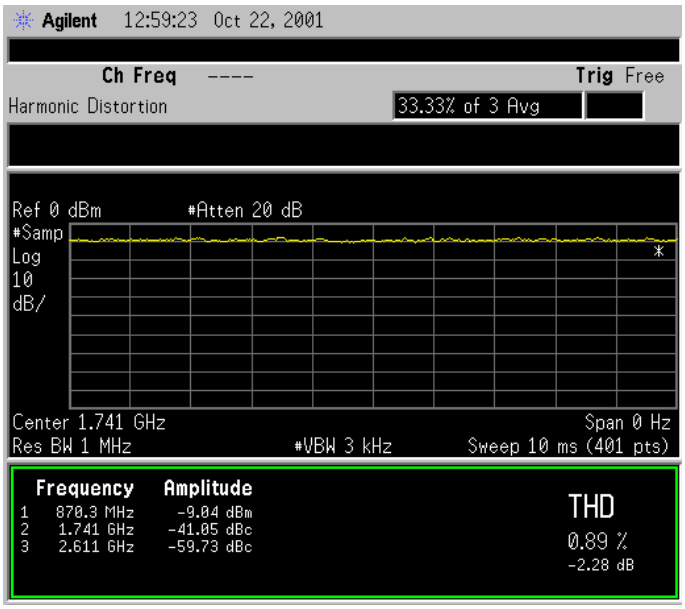
5. Press **Measure**, **More**, **Harmonic Distortion**.

NOTE

At this frequency (870.03 MHz), you may see the message: `One or more harmonics past freq limit: number decreased` on the bottom of the screen. This indicates that one or more harmonics are at frequencies beyond the measurement capabilities of the analyzer. The analyzer will measure all harmonics that fall within its measurement capabilities.

6. Press the **Meas Setup**, **Avg Number** (On), 3, **Enter** to obtain the average of 3 harmonic distortion measurements.
7. Press **Avg Mode** (Exp) to continuously affect the result of the THD with subsequent sweeps. (Selecting **Avg Mode** - clears the value calculated for THD after the total specified number of averages has been taken.)
8. Set the number of measured harmonics to 3 by pressing **Harmonics**, 3, **Enter**.
9. Press **Optimize Ref Level** to ensure all the harmonics are within the range of measurement.

Figure 2-19 Measuring the Harmonic Distortion of an IS-95 Signal



10. The amplitudes of the harmonics are listed relative to the fundamental frequency. Press **Trace/View**, **Harmonics & THD**, to display the total harmonic distortion (%THD).

NOTE An asterisk (*) appearing next to the Total Harmonic Distortion value indicates that the ideal resolution bandwidths for one or more harmonics could not be set. These harmonics for which the resolution bandwidths could not be set are flagged with an asterisk beside their amplitude value. The measurement will still be accurate as long as the signal has little or no modulation.

End of Measurement

The measurement is complete. To exit the measurement, press **Measure**, **Meas Off**.

Making Measurements Using Segmented Sweep (ESA E-Series Analyzers only)

Monitoring Multiple Frequency Bands Concurrently

Segmented sweep allows you to define many bands of interest and display them as a single trace. This function has many valuable applications. Although the following examples only involve a maximum of 3 segments, you can set up the trace to display 32 segments simultaneously.

The first example involves monitoring the U.S. IS-95A cellular band. Use any communications band of interest in your area. Segments one and two will be set up to display the reverse and forward links, respectively. If a signal of interest appears, the normal marker will be used to determine the frequency of the signal, and a third segment will be created to zoom in on that signal. In Figure 2-21, two traces are displayed. Trace 1 is set to maximum hold, to see if there are any potentially interfering signals, and trace 2 will be set to min hold to catch any dropouts.

The second example shows how limit lines can be used in conjunction with segmented sweep. Three segments will be created as in example 1. The first segment will measure the fundamental and determine if the phase noise is within a specified limit. The remaining two segments will also have limit lines to test the acceptable levels of the second and third harmonics.

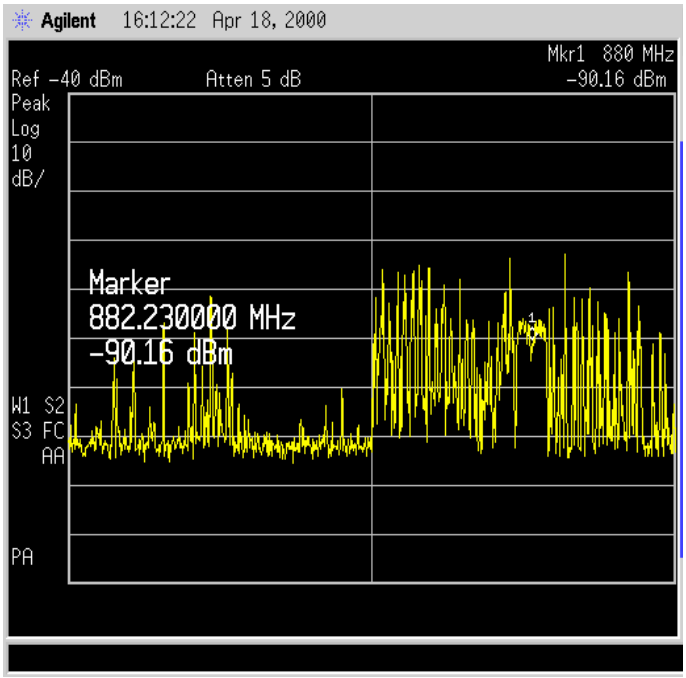
The third example demonstrates the increased measurement speed gained when using segmented sweep to display widely spaced signals occurring in the frequency domain. First, the trace will be set up to view the 50 MHz amplitude reference signal and its second and third harmonics without using segmented sweep. This will show the long sweep time involved to view the signals in this manner. Then, using the segmented sweep function of the analyzer, three segments will be set up to show the reduced sweep time necessary to display these same three signals.

Making Measurements Using Segmented Sweep Example 1:

1. Connect an antenna to the analyzer input. Press **Preset**, **Factory Preset** (if present).
2. Turn on the internal preamp (if installed) by pressing **AMPLITUDE**, **More**, **Int Preamp** (On).
3. Open the segmented sweep editor. Press **Sweep, Segmented, Modify, Edit**.

4. To monitor the cellular activity in the cdmaOne band, set the center frequency of the first segment to 836.5 MHz by pressing **Center Freq**, 836.5, **MHz**.
5. Set the span to 25 MHz by pressing **Span**, 25, **MHz**.
6. Set the resolution bandwidth to 10 kHz. Press **Res BW**, 10, **kHz**. The remaining parameters will automatically be set to the default values for Video BW, points, and sweep time.
7. Adjust the reference level to place the signal in the middle of the display. In this example, set the reference level to –40 dBm by pressing **AMPLITUDE**, **Ref Level**, 40, –**dBm**. (See Figure 2-20, "Monitoring the Reverse and Forward Links,".)
8. To view the activity of the base station in the cdmaOne cellular band, set the center frequency of the second segment to 881.5 MHz by pressing **Sweep**, **Segmented**, **Modify**, **Edit**, **Segment**, 2, **Enter**, **Center Freq**, 881.5, **MHz**.
9. Set the span to 25 MHz by pressing **Span**, 25, **MHz**.
10. Set the resolution bandwidth to 10 kHz. Press **Res BW**, 10, **kHz**. The remaining parameters will automatically be set to the default values for Video BW, points, and sweep time.

Figure 2-20 Monitoring the Reverse and Forward Links

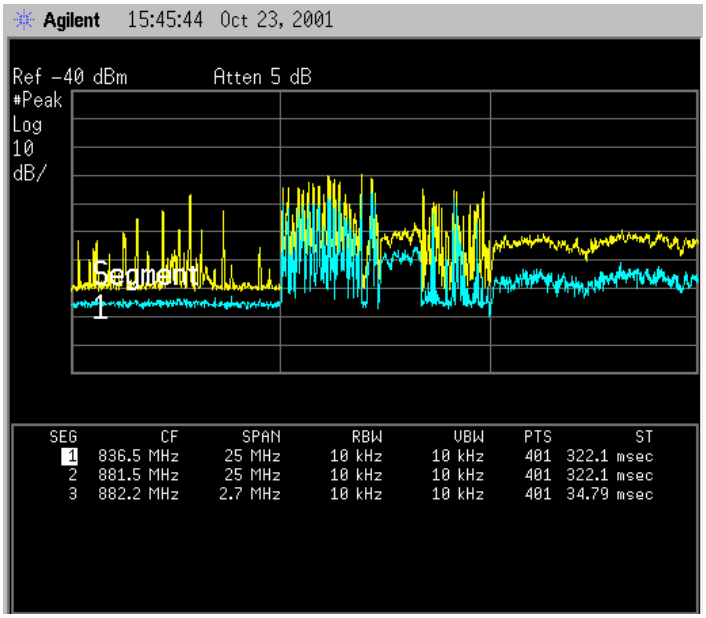


11. Press **Marker**, then rotate the knob to locate the frequency of the cdmaOne signal.

12. In this example, two cdmaOne carriers were found to be located at 882.23 MHz. as shown in Figure 2-20. Set the center frequency of segment 3 to 882.23 MHz by pressing Sweep, Segmented, Modify, Edit, **Segment**, 3, **Enter**, **Center Freq**, 882.23, **MHz**.

13. Since there are two signals, the span is set to greater than twice the cdmaOne bandwidth (2 times 1.2288 MHz). Set your span accordingly. For this example, the span is set to 2.7 MHz by pressing **Span**, 2.7, **MHz**.

Figure 2-21 Monitoring Cellular Bands in Segmented Sweep



14. Set the resolution bandwidth to 10 kHz. Press **Res BW**, 10, **kHz**. The remaining parameters will automatically be set to the default values for Video BW, points, and sweep time.

15. Set trace 1 to maximum hold by pressing **Trace/View**, **Trace 1**, **Max Hold**.

16. Set trace 2 to minimum hold by pressing **Trace/View**, **Trace 2**, **Min Hold**.

Figure 2-21 displays the results of this setup over an extended period of time. As expected, the minimum hold trace of segment 1 shows no continuously present carriers. This is indicative of cellular activity. If a persistent signal exists, it may represent the presence of an interfering carrier. In this example, segment 3 shows continuous transmission of the carrier without interference.

Making Measurements Using Segmented Sweep Example 2:

1. Connect the 10 MHz REF OUT from the rear panel to the front-panel INPUT. Press **Preset**, **Factory Preset** (if present).
2. Adjust the reference level to place the peak of the fundamental signal at the top of the graticule. In this example, the reference level is set at +8 dBm by pressing **AMPLITUDE**, **Ref Level**, 8, **dBm** (see Figure 2-22).
3. Open the segmented sweep editor. Press **Sweep, Segmented, Modify, Edit**.
4. To display the fundamental signal, set the center frequency of the first segment to 10 MHz by pressing **Segment**, 1, **Enter**, **Center Freq**, 10, **MHz**.
5. Set the span to 500 kHz by pressing **Span**, 500, **kHz**.
6. Set the resolution bandwidth to 3 kHz. Press **Res BW**, 3, **kHz**.
7. Set the video bandwidth to 30 Hz by pressing **Video BW**, 30, **Hz**.
8. Set the number of points to 1000. Press **Points**, 1000, **Enter**. The sweep time defaults to the minimum.
9. To display the second harmonic, set the center frequency of the second segment to 20 MHz by pressing **Segment**, 2, **Enter**, **Center Freq**, 20, **MHz**.
10. Set the span to 100 kHz by pressing **Span**, 100, **kHz**. Allow the remaining parameters to take their default values as shown in Figure 2-22.
11. Press **Tab** ⇒| to begin a new segment.
12. To display the third harmonic, set the center frequency of the third segment to 30 MHz by pressing **Center Freq**, 30, **MHz**.
13. Set the span to 100 kHz by pressing **Span**, 100, **kHz**. Allow the remaining parameters to take their default values as shown in Figure 2-22.
14. Open the limit line editor by pressing **Display**, **Limits**, **Limit 1**, **Type** (Upper) **Edit**.
15. The limit line data is displayed in Table below. Enter the data for the first point by pressing **Point**, 1, **Enter**, **Frequency**, 9.75, **MHz**, **Amplitude**, –80, **dBm**, **Connected** (Yes). Enter the subsequent points in the same manner.

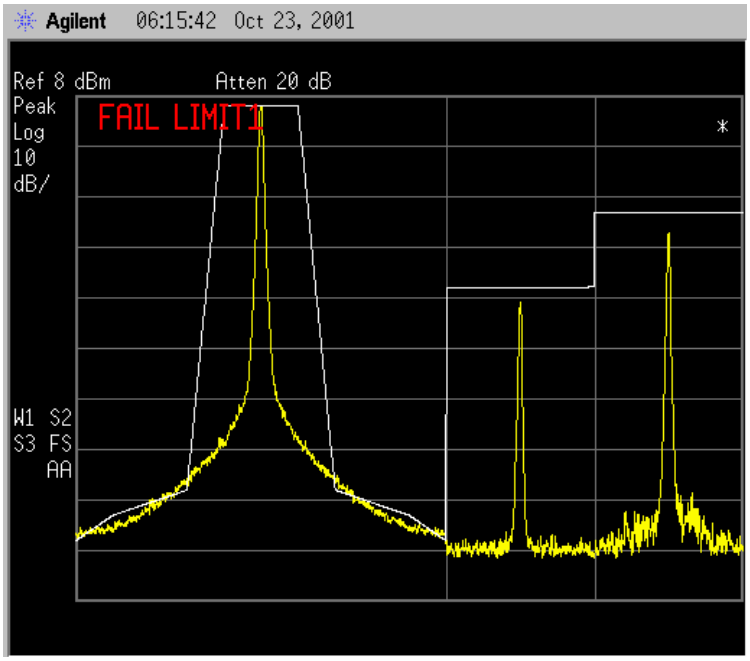
Table 2-2 Limit Line Data

Point	Frequency	Amplitude	Connected
1	9.75 MHz	– 80 dBm	Yes
2	9.8 MHz	– 75 dBm	Yes
3	9.9 MHz	– 70 dBm	Yes
4	9.95 MHz	6 dBm	Yes
5	10.05 MHz	6 dBm	Yes
6	10.1 MHz	– 70 dBm	Yes
7	10.2 MHz	– 75 dBm	Yes
8	10.25 MHz	– 80 dBm	Yes
9	19.95 MHz	– 30 dBm	Yes
10	29.95 MHz	– 15 dBm	Yes
11	30.05 MHz	– 15 dBm	Yes

16. Turn the limits on by pressing Return, **Limit** (On).

17. Press **Test** (On) to determine if the fundamental and harmonics pass the specified limits test.

Figure 2-22 Phase Noise and Harmonic Measurements



This procedure verifies compliance with phase noise and harmonics

specifications. You can save this instrument state and limit lines in a "setup" type file for future applications (**File**, **Save**, **Type**, **Setup**, **Save Now**). Replace the 10 MHz reference signal of the analyzer with the device under test (DUT) to create a similar measurement.

Making Measurements Using Segmented Sweep Example 3:

This example will demonstrate the differences in sweep time required for some measurements using the standard sweep mode verses the segmented sweep mode. First you will measure a signal the standard sweep mode then you will measure the same signal using segmented sweep.

Signal Measured Using Standard Sweep Mode

1. Turn on the 50 MHz amplitude reference signal of the analyzer.

   For the E4401B and E4411B, use the 50 MHz amplitude reference signal of the analyzer as the signal being measured. Press **Preset**, Factory Preset (if present), **Input/Output**, **Amptd Ref** (On).

   For all other models connect a cable between the front-panel AMPTD REF OUT to the analyzer INPUT, then press **Preset**, Factory Preset (if present), **Input/Output**, **Amptd Ref Out** (On).

2. Set the Center frequency to 50 MHz. Press **FREQUENCY**, 50, **MHz**.

3. Set the span from 20 MHz to 170 MHz by pressing **Start Freq**, 20, **MHz**, **Stop Freq**, 170, **MHz**.

4. Set the resolution bandwidth to 1 kHz. Press **BW/Avg**, **Res BW**, 1, **kHz**.

5. Set the video bandwidth to 1 kHz by pressing **Video BW**, 1, **kHz**.

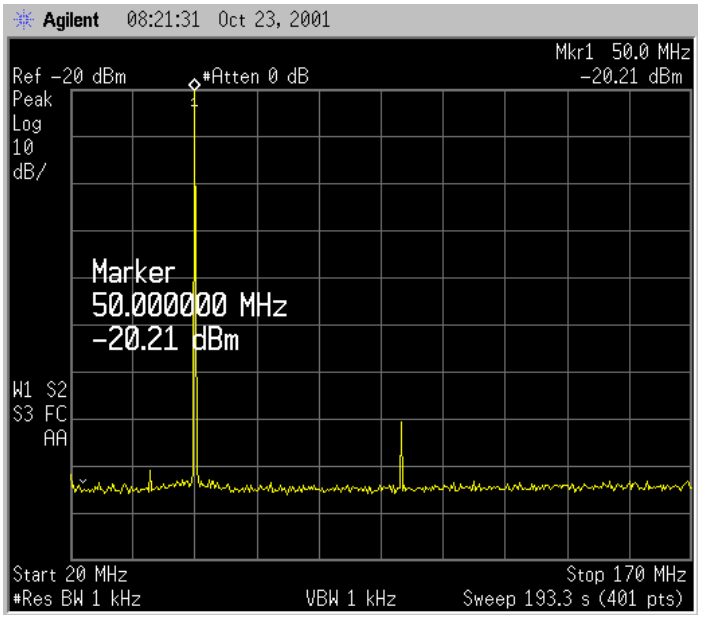
6. Set the number of points to 401. Press **Sweep, Points**, 401, **Enter**.

7. Set the Reference level to -20.00 dBm by pressing **AMPLITUDE**, **Ref Level**, –20, **dBm**.

8. The attenuation should be set at 0 dBm by pressing **Attenuation**, 0, **dBm**.

9. Place a marker at the 50 MHz signal by pressing **Peak Search**. Refer to Figure 2-23 to see the result of this first measurement in this example. Notice the minimum sweep time is set at 193.3 seconds in order to provide accurate data. If you attempt to enter a faster sweep time, the `Meas Uncal` message is shown in the upper right corner of the display.

Figure 2-23 Longer Sweep Times are Required Without Segmented Sweep



Signal Measured Using Segmented Sweep Mode

The same signals will now be monitored using segmented sweep.

1. Ensure that the 50 MHz amplitude reference signal is turned on as you did in step 1, above.
2. The reference level for this example is set to –20 dBm by pressing **Amplitude**, **Ref Level**, –20 **dBm**. Set the attenuation to 5 dB by pressing **Attenuation**, 5, **dB**.
3. Open the segmented sweep editor. Press **Sweep, Segmented, Modify, Edit**.
4. Set up the three segments shown in Figure 2-24 to view the fundamental, second harmonic, and third harmonic.
5. Set the center frequency of the first segment to 50 MHz by pressing **Segment**, 1, **Enter**, **Center Freq**, 50, **MHz**.
6. Set the span to 100 kHz by pressing **Span**, 100, **kHz**.
7. Set the video bandwidth to 30 Hz by pressing **Video BW**, 30, **Hz**.

   The remaining parameters will automatically be set to the default values for points and sweep time.
8. Set the center frequency of the second segment to 100 MHz by pressing **Segment**, 2, **Enter**, **Center Freq**, 100, **MHz**.
9. Set the span to 100 kHz by pressing **Span**, 100, **kHz**. Allow the remaining parameters to take their default values as shown in Figure 2-22.

10. Set the video bandwidth to 30 Hz by pressing **Video BW**, 30, **Hz**.

    The remaining parameters will automatically be set to the default values for points and sweep time.

11. Set the center frequency of the third segment to 150 MHz by pressing **Segment**, 3, **Enter**, **Center Freq**, 150, **MHz**.

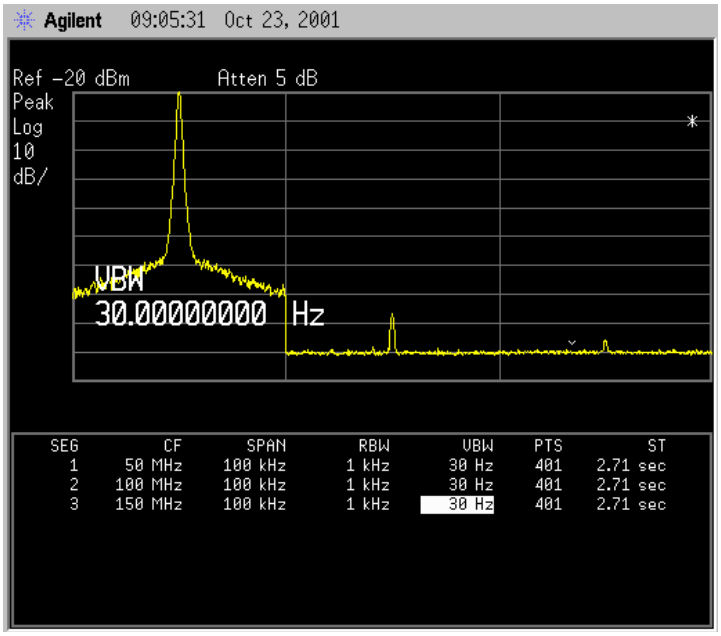
12. Set the span to 100 kHz by pressing **Span**, 100, **kHz**.

13. Set the video bandwidth to 30 Hz by pressing **Video BW**, 30, **Hz**.

    The remaining parameters will automatically be set to the default values for points and sweep time.

Notice the total sweep time for the 3 segments is only 5.42 seconds. This is considerably shorter than the 193.3 seconds required to view the same signals without segmented sweep.

Figure 2-24 Reducing Sweep Time with Segmented Sweep



Making Power Measurements on Burst Signals

The Burst Power measurement is a very accurate method of determining the average power for the specified burst. The analyzer is set into zero-span mode, with a sweep time that captures at least one burst. The default is just more than a single burst, but the user may change this using the 'Sweep Time' softkey in the 'Sweep' menu.

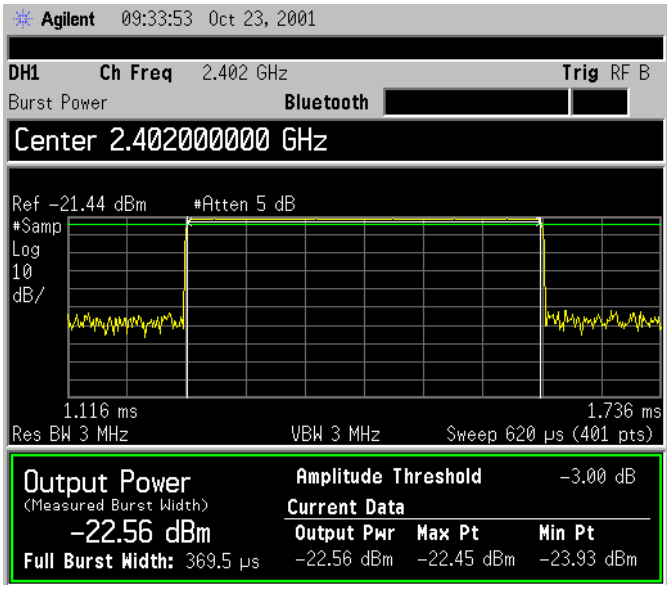
The following example demonstrates how to make a burst power measurement on a Bluetooth™ signal broadcasting at 2.402 GHz.

Burst Power Measurement of a Bluetooth™ Signal Example 1:

1. Connect a DH1 Bluetooth™ signal to the analyzer input.
2. Press **Preset, Factory Preset** (if present), **FREQUENCY**, **Center Freq**, 2.402, **GHz** to set the frequency of the signal to be measured.
3. Set the attenuation to auto by pressing **AMPLITUDE**, **Attenuation** (Auto) to prevent IF Overload.
4. Press **Mode Setup**, **Radio Std**, **More**, **Bluetooth** to set the radio standard to Bluetooth™.
5. Press **Mode Setup**, **Std Setup**, **Packet Type** (DH1) to set the DH1 Bluetooth™ standard.
6. Press **MEASURE**, **More**, **Burst Power** to select the burst power measurement.
7. Optimize the reference level signal by pressing **Meas Setup, Optimize Ref Level**. This automatically sets the best reference level for this measurement on this signal.
8. If you have firmware revision A.07.xx, go to step 9 AND SKIPPING step 10. If you have firmware A.08.xx or later and Option B7E with board part number E4401-60224 or higher, go to step 10. (To determine firmware revision level, press **System**, **More**, **Show System**. To determine board part number, press **System**, **More**, **Show Hdwr**.)
9. Press **Trig**, **Video** to steady the display and synchronize the measurement with the signal. This triggers the measurement at the point at which the rising signal crosses the lower horizontal green line on the display. If an external trigger is available, connect this to Gate Trig/Ext Trig on the rear of the instrument and press **Trig**, **External**. It might be necessary to adjust the trigger level (as indicated by the lower horizontal green line) by rotating the front panel knob or by entering a numeric value on the keypad. In this example, set the trigger level to -30 dBm as shown in Figure 2-25 below.
10. To steady the display and synchronize the measurement with the

signal, press **Trig**, **RF Burst**. If you do not get a display similar to Figure 2-25, it might be necessary to adjust the trigger level by pressing **More**, **RF Burst Setup**, **Trigger Level**, and then rotating the front panel knob or by entering a numeric value on the keypad until the signal stabilizes on the display. You may also need to use the other settings in this menu to adjust the display.

Figure 2-25 Burst Power Measurement on a Bluetooth™ Signal

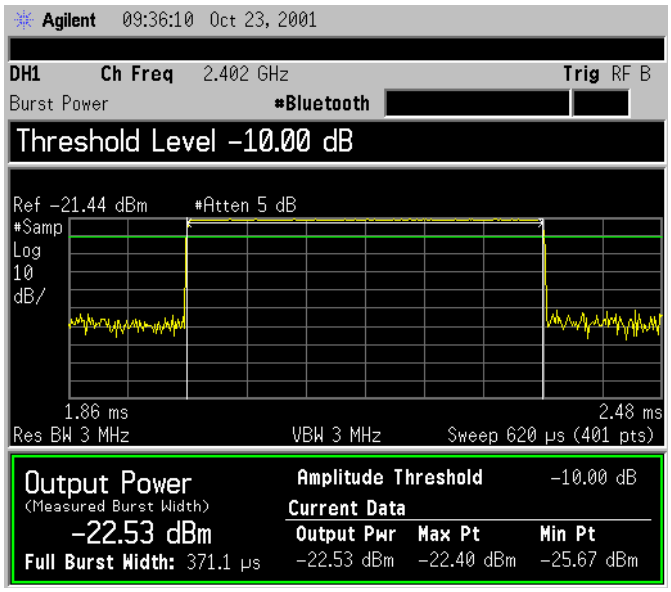


NOTE Although the Trigger Level allows the analyzer to detect the presence of a burst, the Burst Power measurement is determined by the threshold level, as described next.

11. Press **Meas Setup**, **Threshold Lvl** (Rel) to set the relative level above which the burst power measurement will be calculated. Press –10, **dB** to set the threshold level for this example.

    The burst power measurement is started from the position where the rising signal level rises above this threshold, and finishes when the signal passes below it. The threshold level is indicated on the display by the upper horizontal line. If you have a color display using the default display colors, it is shown as a green line. In this example, the threshold level has been set to be 10 dB below the reference value. The mean power of the burst is measured from all data above the threshold level.

Figure 2-26 Manually Setting the Threshold Level

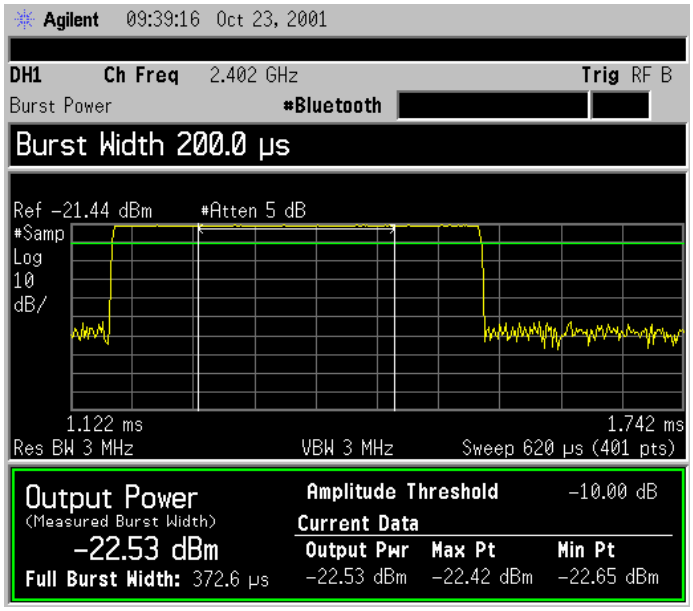


12. To measure just the central 200 µs of the burst: Press **Meas Setup**, **Meas Method**, **Measured Burst Width**, **Burst Width** (Man), 200, **µs** to specify the burst width for which the measurement will be taken.

    The burst width is indicated on the screen by two vertical white lines. These two white lines moved in towards the center of the graphical display when you set the burst width to 200 µs as shown in Figure 2-27 below. This specifies the time period (burst width) used to calculate the measurement.

    Manually setting the burst width allows you to make it a long time interval (to include the rising and falling edges of the burst) or to make it a short time interval, thus measuring only a small central section of the burst.

Figure 2-27 Manually Setting the Burst Width



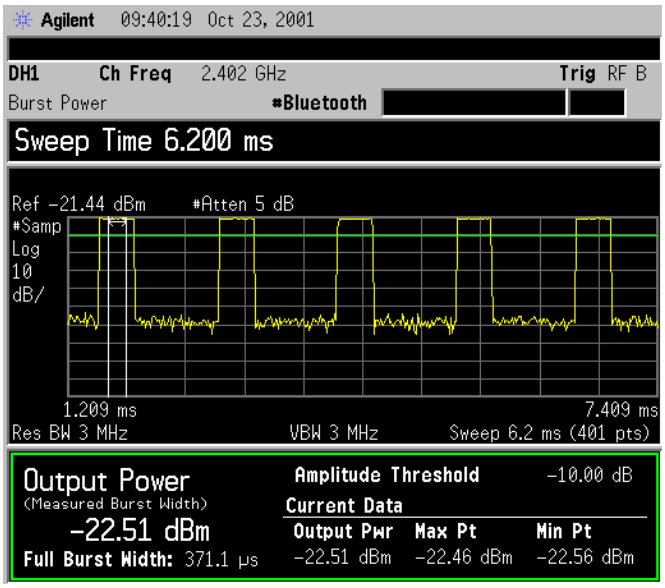
NOTE The Bluetooth™ standard states that power measurements should be taken from the central 60% of the burst only. Other radio standards use different figures.

NOTE If you set the burst width manually to be wider than the screen's display, the vertical white lines will move off the edges of the screen. This could give misleading results as only the data on the screen can be measured.

13. Change the sweep time to display more than one burst at a time:
    Press **Sweep**, **Sweep Time**, 6200, **µs** (or 6.2, **ms**) using the numeric keypad.

    The screen display will now show several bursts in a single sweep as shown in Figure 2-28 below. The burst power measurement will measure the mean power of the first burst, indicated by the vertical white lines either around it or, as in this example, within it.

Figure 2-28 Displaying Multiple Bursts



NOTE Although the burst power measurement still runs correctly when several bursts are displayed simultaneously, the timing accuracy of the measurement is degraded. For the best results (including the best trade-off between measurement variations and averaging time), it is recommended that the measurement be performed on a single burst.

Making Statistical Power Measurements (CCDF) (Option AYX or B7D only)

What is a CCDF (Complementary Cumulative Distribution Function) Measurement?

All CDMA signals, and W-CDMA signals in particular, are characterized by high power peaks that only occur infrequently. It is important that these peaks are preserved otherwise separate data channels will not be received properly. Too many peak signals can also cause spectral regrowth. If a CDMA system works well for most of the time and only fails occasionally, this can often be caused by compression of the higher peak signals.

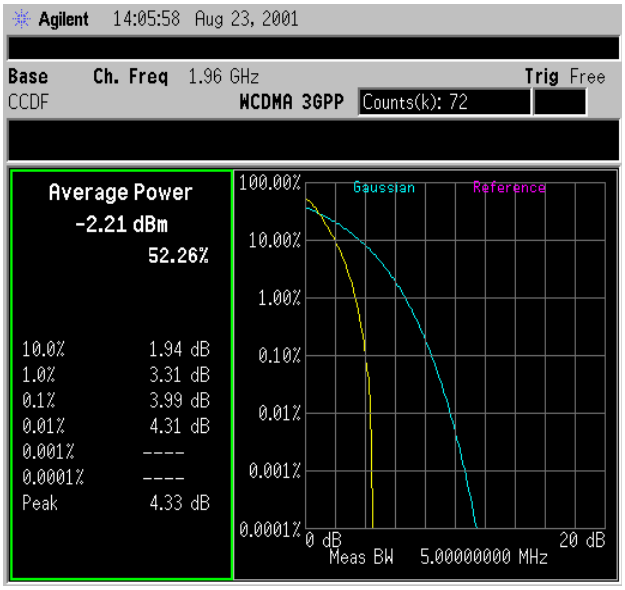
The CCDF measurement is a statistical measurement of a signal's high-level or peak power. It shows in both graphical and tabular form for what percentage of the time a signal exceeds its average power, and by how much this average is exceeded.

The following example shows how to make a CCDF measurement on a W-CDMA signal broadcasting at 1.96 GHz.

Power Stat CCDF Measurement of a W-CDMA Signal Example 1:

1. Connect a W-CDMA signal to the analyzer input.
2. Set the frequency of the signal to be measured by pressing **Preset, Factory Preset** (if present), **FREQUENCY**, **Center Freq**, 1.96, **GHz**.
3. Press **Mode Setup**, **Radio Std**, **W-CDMA** to set the radio standard to Wideband CDMA
4. Press **Std Setup**, **Device** (BTS) to set up the Base Station W-CDMA radio standard.
5. Press **MEASURE, Power Stat CCDF** to select the Power Stat CCDF measurement
6. Finally, press **Meas Setup**, **Optimize Ref Level** to optimize the reference level. This automatically sets the best attenuation and reference level for this measurement on this signal.

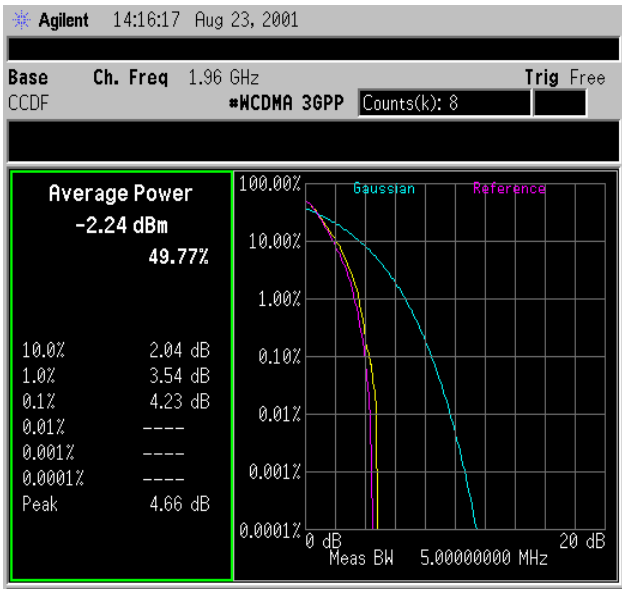
Figure 2-29 Power Stat CCDF Measurement on a W-CDMA Signal



7. Press **Display**, **Store Ref Trace** to store your current measurement trace for future reference. When the Power Stat CCDF measurement is first made, the graphical display should show a signal typical of pure noise. This is labelled 'Gaussian', and is shown in aqua. Your measurement will show as a yellow plot. You have stored this measurement's plot to make for easy comparison with subsequent measurements.

8. Press **Ref Trace** (On) to display the stored trace. The stored trace from your last measurement is displayed as a magenta plot (as shown in Figure 2-30), and allows direct comparison with your current measurement.

Figure 2-30 Storing and Displaying a Power Stat CCDF Measurement

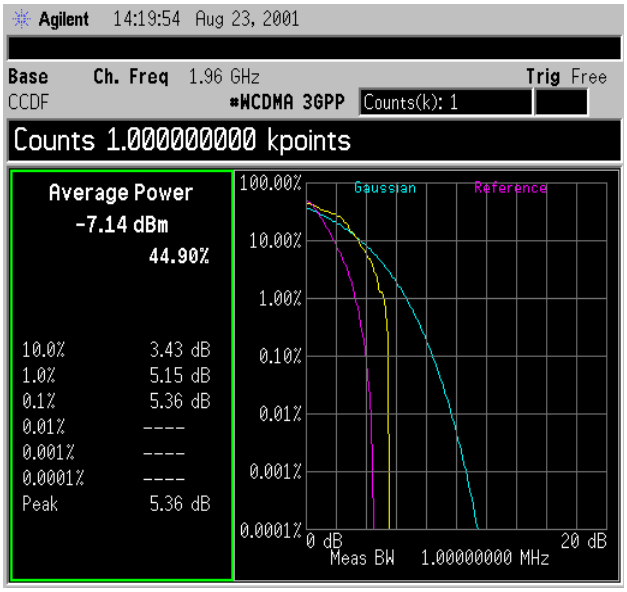


9. Press **Meas Setup**, **Meas BW**, 1, **MHz** to change the measurement bandwidth to 1 MHz.

NOTE The measurement bandwidth range is dependent on the options installed. There is only a discrete number of settings available which run in a sequence of 1, 3, 10, 30... as well as the 5 MHz default setting. If you choose a measurement bandwidth setting that the machine cannot display, it will automatically set itself to the closest available bandwidth setting.

10. Press **Counts**, 1, **kpoints** to change the number of measured points from 100,000 (100 k) to 1,000 (1 k). Reducing the number of points decreases the measurement time, however the number of points is a factor in determining measurement uncertainty and repeatability. Notice how the displayed plot loses a lot of its smoothness. You are gaining speed but reducing repeatability and increasing measurement uncertainty.

Figure 2-31 Reducing the Number of Measurement Points to 1,000



NOTE The number of plots collected per sweep is dependent on the sampling rate and the measurement interval. The number of samples that have been processed will be indicated at the top of the screen. The graphical plot will also be updated so you will be able to see it getting smoother as measurement uncertainty is reduced and repeatability improves.

11. Press **SPAN**, **X-Axis**, 1, **dB** to change the scale of the X-axis. The X-axis scale can be changed to suit the measurement you are making.

Making Measurements of Adjacent Channel Power (ACP)

The Adjacent Channel Power (ACP) measurement is sometimes referred to as the Adjacent Channel Power Ratio (ACPR) in CDMA measurement literature. We will use the term ACP to refer to this measurement.

ACP measures the total RMS power in the specified channel and in up to six pairs of offset frequencies. The measurement result reports the ratios of the offset powers to the main channel power. The offset frequencies (the adjacent channels being measured) can be modified at any time, but the default values are those specified by the relevant international standards. The results are displayed by default both as power relative to the carrier (in dB or dBc) and as absolute power (dBm).

NOTE Measurement results are reported in dB when **Mode Setup**, **Radio Std**, **PDC** or **Mode Setup, Radio Std, NADC, Std Setup, MS** is selected.

The following example shows how to make an ACP measurement on a W-CDMA Base Station signal broadcasting at 1.96 GHz.

Example - ACP Measurement on a Wideband CDMA Base Station Signal:

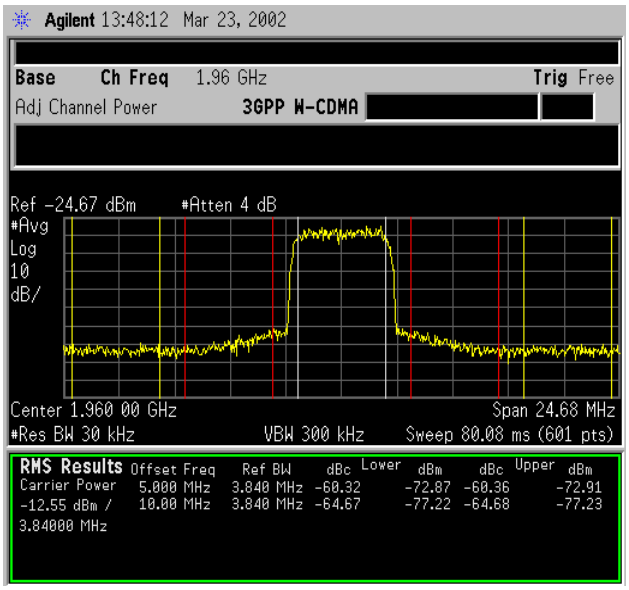
1. Connect a Wideband CDMA signal at 1.96 GHz to the analyzer's input.
2. Set the frequency to be measured by pressing **Preset, Factory Preset** (if present), **FREQUENCY**, **Center Freq**, 1.96, **GHz**.
3. Set the W-CDMA radio standard by pressing **Mode Setup, Radio Std, W-CDMA.**
4. Select base station mode by pressing **Mode setup**, **Std setup**, **Device** (BTS).

NOTE Although this example refers to a Base Station measurement throughout, the process for Mobile Station measurements is the same.

5. Select the Adjacent Channel Power measurement by pressing **MEASURE, ACP**.
6. Optimize the signal reference level by pressing **Meas Setup**, **Optimize Ref Level**. Results are displayed as shown in Figure 2-32.

NOTE This optimization protects against input signal overloads, but does not necessarily set the input attenuation for optimum measurement dynamic range.

Figure 2-32 ACP Measurement on a Base Station W-CDMA Signal



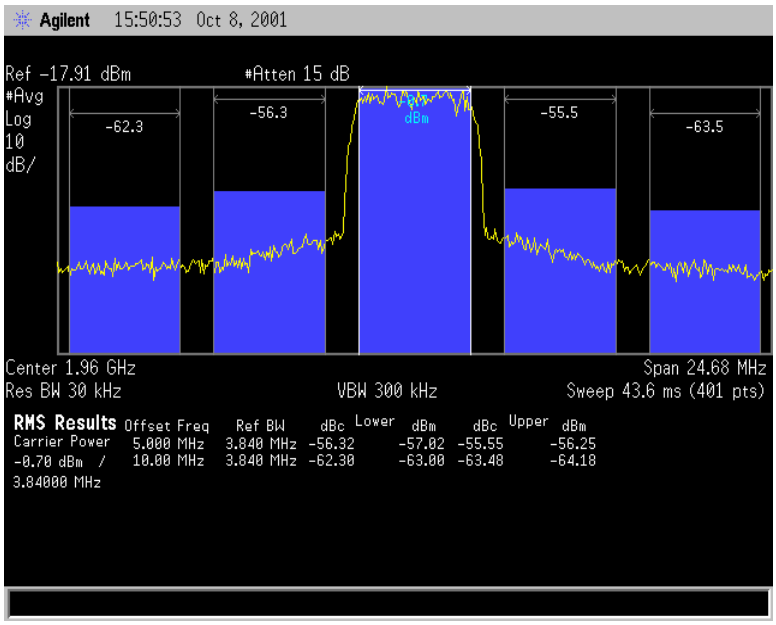
The Frequency Offsets, Channel Integration Bandwidths, and Span settings can all be modified. They default to the relevant settings for the radio standard you have currently selected.

Two vertical white lines indicate the bandwidth limits of the central channel being measured.

Offsets A and B are designated by the adjacent pairs of red and yellow lines, in this case: 5 MHz and 10 MHz from the center frequency respectively.

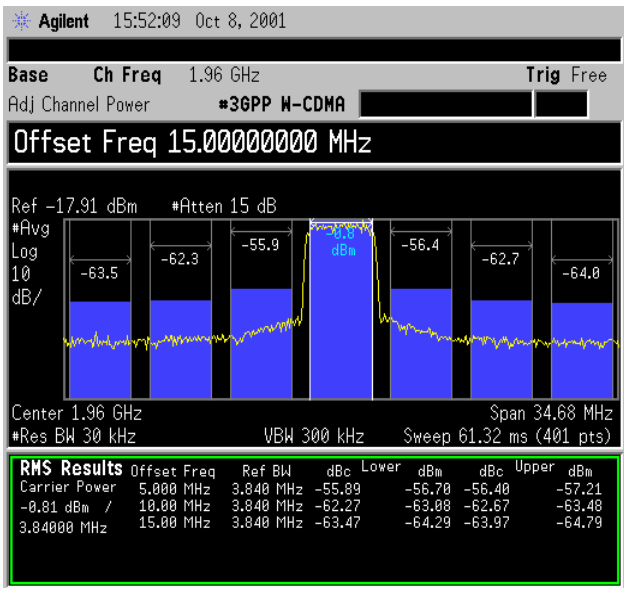
7. Press **Trace/View**, **Combined** to display both the spectrum and bar graph views of the results.
8. Press **Display**, **Full Screen** to display a larger view of the trace as shown in Figure 2-33.

Figure 2-33 ACP Measurement in Full Screen Display



9. Press **Meas Setup**, **Offset/Limits**, **Offset**, **C**, **Offset Freq** (On), 15, **MHz** to set a third pair of offset frequencies. This third pair of offset frequencies will be offset by 15.0 MHz from the center frequency and is shown on the screen by two more pairs of blue bars. (The blue bars are not displayed if you have a monochrome display, but the numeric readout in each channel is displayed.) An example screen with this extra pair of frequencies is shown in Figure 2-34. Three further pairs of offset frequencies (D, E and F) are available and are displayed similarly.

Figure 2-34 Measuring a Third Adjacent Channel

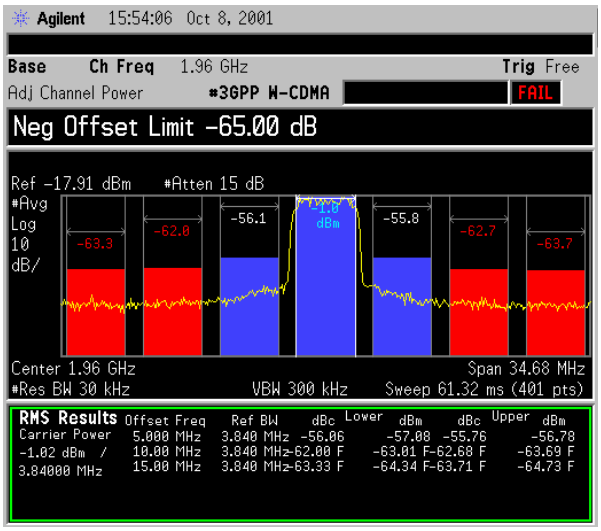


10. Press **Offset** (A), **Neg Offset Limit**, –55, **dB**, **Pos Offset Limit**, –55, **dB**,

Offset (B), **Neg Offset Limit**, –65, **dB**, **Pos Offset Limit**, –65, **dB**, **Offset** (C), **Neg Offset Limit**, –65, **dB**, **Pos Offset Limit**, –65, **dB** to set pass/fail limits for each offset.

11. Press **Meas Setup**, **More**, **Limit Test** (On) to show the results as shown in Figure 2-35. Offset A has passed, however Offsets B and C have failed. Failures are identified by the red letter “F” next to the levels (dBc and dBm) listed in the lower portion of the window called, “RMS Results”. The offset bar graph is also shaded red to identify a failure.

Figure 2-35 Setting Offset Limits



NOTE You may increase the repeatability by increasing the sweep time.

Making Measurements of Multi-Carrier Power (MCP)

MCP measures the total power in two or more transmit channels and their adjacent channels for up to three pairs of offset frequencies. The offset frequencies can be modified at any time, but the default values are those specified by the relevant international standard that you select. This measurement is available with no radio standard selected or when you select any of the following radio standards: None, IS-95, cdma2000 SR1, or W-CDMA. Results for carriers without power present are displayed relative to the reference carrier. Results for adjacent channels are displayed in absolute power (dBm).

The following example shows how to make an MCP measurement on W-CDMA Base Station broadcasting 10 carriers. Eight carriers have power present at the following frequencies: 1.0225 GHz, 1.0175 GHz, 1.0125 GHz, 1.0075 GHz, 992.5 MHz, 987.5 MHz, 982.5 MHz, and 992.5 MHz.

NOTE When **Radio Std**, **None** is selected you must manually set most parameters required to perform this measurement. When selecting **Radio Std**, **W-CDMA 3GPP**, these parameters are already set by the analyzer.

Example - MCP Measurement on a Wideband CDMA Base Station Signal:

1. Set the Mode to Spectrum Analysis.

   Press **Mode**, **Spectrum Analysis**.

2. Connect a W-CDMA signal with multiple carriers broadcasting at the frequencies stated above.

3. Set the center frequency of the analyzer to the midpoint of all the carriers to be measured.between the main channel center frequency and the second carrier center frequency. In this example, that would be 1.0 GHz.

   Press **Preset, Factory Preset** (if present), **FREQUENCY**, **Center Freq**, 1, **GHz**.

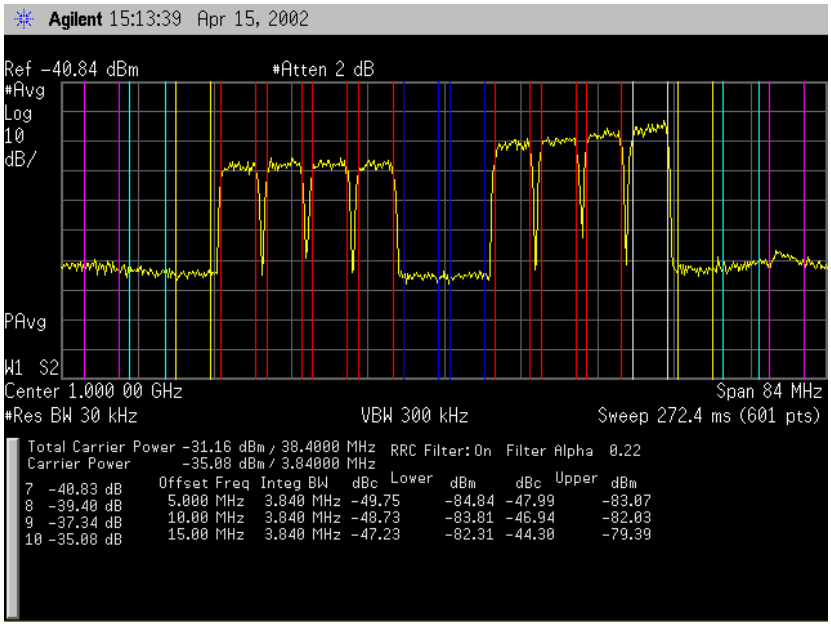
4. Set the W-CDMA radio standard by pressing **Mode Setup, Radio Std, W-CDMA.**

5. Select base station mode by pressing **Mode setup**, **Std setup**, **Device** (BTS).

NOTE Although this example refers to a Base Station measurement throughout, the process for Mobile Station measurements is the same.

6. Select the Multi Carrier Power measurement by pressing **MEASURE, Multi-Carrier Power**.
7. Optimize the signal reference level by pressing **Meas Setup**, **Optimize Ref Level**.
8. Set the carrier number to 10. Press **Carrier Setup**, **Carriers**, 10, **Enter**.
9. Configure carrier 5 to have no power present. Press **Configure Carriers**, 5, **Enter**, **Carrier Pwr Present** (No).
10. Repeat step 5, configuring carrier 6 to have no power present.
11. Display the results in full screen view. Press **Display**, **Full Screen**. Refer to Figure 2-36.

Figure 2-36 MCP Measurement on a Base Station W-CDMA Signal

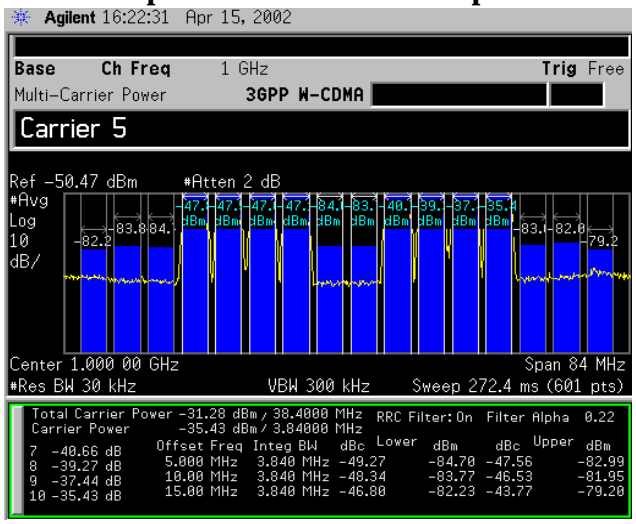


In this example, the intermodulation falls outside the transmit channels which are marked by the colored vertical lines. The white set indicates the reference carrier. The red sets contain the carriers with power present and the blue lines mark the carriers without power present. Limits for the upper and lower offsets can also be set as shown in the example: "Making Measurements of Adjacent Channel Power (ACP)" on page 115.

12. View the results table of carriers 7-10. Press **Meas Setup**, **Carrier Result**, 7, **Enter**.
13. View the results in a combined spectrum and bar graph by pressing

Trace/View, **Combined**.

Figure 2-37 Combined Spectrum and Bar Graph View



14. Save the results file to a disk. Press **File**, **Save**, **Type**, **Measurement Results**, **Save Now**.

    The results are stored in a comma separated values format to be viewed by any personal computer spreadsheet application. All data shown on the display is included in this file.

Demodulating and Viewing Television Signals (Option B7B)

Option B7B (TV Trigger and Picture on Screen) allows you to trigger the sweep of the analyzer on a specific television line of a demodulated TV waveform. Option B7B also allows you to view the television picture represented by the TV waveform on the color LCD display of the analyzer.

The following examples describe how to set the analyzer for viewing the demodulated TV waveform, viewing the TV picture, and measuring the depth of modulation of the RF carrier.

Demodulating and Viewing Television Signals Example 1:

To set up the TV video waveform for triggering and picture viewing on the analyzer, perform the following:

1. Connect a source which contains suitable TV carrier signals (for example, terrestrial broadcast or CATV signals).
2. Press **System**, **Alignments**, **Auto Align**, **Off** to disable the background alignment process while viewing TV waveforms and TV pictures. This is necessary to prevent the background alignment process from interrupting the signal paths of the analyzer during the sweep retrace period so as to maintain a constant, uninterrupted video waveform.

NOTE After viewing the TV waveform or picture, re-enable the background alignment process by pressing **System**, **Alignments**, **Auto Align**, **All**. If the background alignment has been disabled for more than 60 minutes or the ambient temperature has changed more than 3 degrees centigrade, press **System**, **Alignments**, **Align Now**, **All** to ensure measurement accuracy. See the Specifications and Characteristics Chapter for your analyzer model in the specifications guide, for information about alignment requirements.

3. Set the center frequency of the analyzer to match the TV video carrier frequency by pressing **FREQUENCY**, then entering the desired value and units.
4. Set the span to 20 MHz by pressing **SPAN**, 20, **MHz**.
5. Press **Auto Couple**.
6. Adjust the reference level of the analyzer to the peak video carrier level by pressing **AMPLITUDE** and using the knob or step keys.

NOTE If the signal is weak and accompanied by excessive noise, you may choose to enable the internal preamp, if installed, to improve the signal-to-noise ratio. Press **AMPLITUDE**, **More**, **Int Preamp** (On).

7. View the spectrum. There should be a strong, “noise-like” video carrier at the center frequency, a weaker, “noise-like” chrominance sub-carrier located 3.58 MHz (NTSC standard) or 4.3 to 4.4 MHz (PAL or SECAM standards) above the video carrier, and a tightly-grouped sound carrier located 4.5 to 6.5 MHz above the video carrier. If you are viewing broadcast or cable TV signals, the lower adjacent channel sound carrier may be very close to the video carrier at the center frequency.
8. Press **BW/Avg**, **Res BW**, 3, **MHz** to change the resolution bandwidth of the analyzer to 3 MHz. If the test signal does not have adjacent channels present, change the resolution bandwidth to 5 MHz. However, if strong adjacent channel signals are present (primarily the sound carrier of the lower adjacent channel), set the resolution bandwidth and video bandwidth to 1 MHz.
9. Press **AMPLITUDE**, **Scale Type** (Lin) to set the amplitude scale type of the analyzer to Linear.
10. Press **Det/Demod**, **Detector**, **Sample** to set the detector mode of the analyzer to Sample.
11. Set the span to 0 Hz by pressing **SPAN**, **Zero Span**.
12. Press **AMPLITUDE** and using the knob or arrow keys, adjust the reference level so that the signal peaks are within half of a division of the top of the display.
13. Press **Trig**, **TV Trig Setup** and set the following in the TV Trig Setup menu: Press **Field**, **Entire Frame**. Press **Sync** (Pos) (**Sync** (Neg) if viewing a SECAM broadcast). Press **Standard** and then select the appropriate standard for the video signal. Press **TV Source**, **SA**.
14. Press **Trig**, **TV** to enable the TV trigger. The default line number for triggering can be changed to any value from 1 to 525 or from 1 to 625, depending on the selected video standard.
15. If you have Option AYX (Fast Digitized Time Domain Sweeps) or Option B7D (DSP and Fast ADC), press **Sweep**, **Sweep Time**, 500, **μs**. This allows you to view several TV lines.

A time domain display of the demodulated TV waveform will now be visible. The signals used for Figure 2-38 and Figure 2-39 were produced by a Phillips PM 5518-TX Color TV Pattern Generator.

Figure 2-38 Demodulated RF Waveform (NTSC; PAL is Similar)

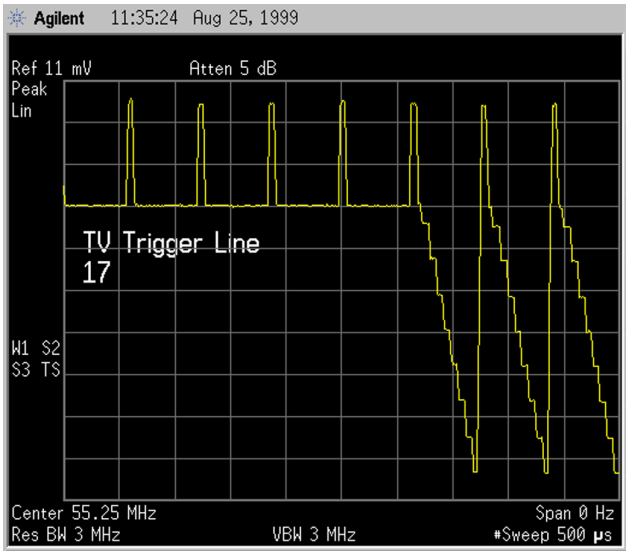
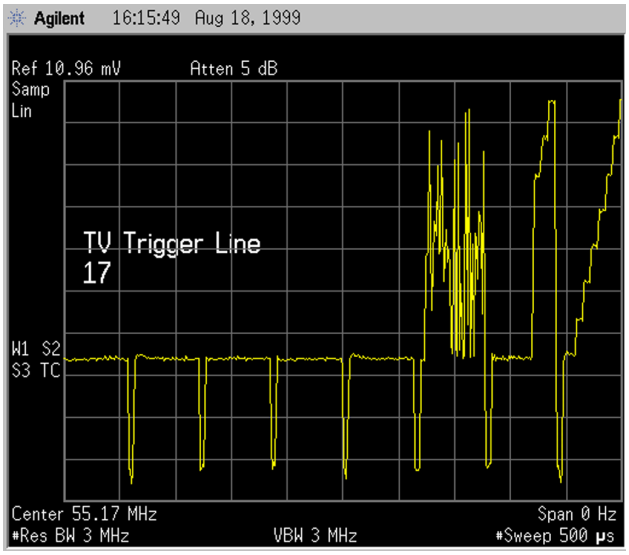


Figure 2-39 Demodulated RF Waveform (SECAM)



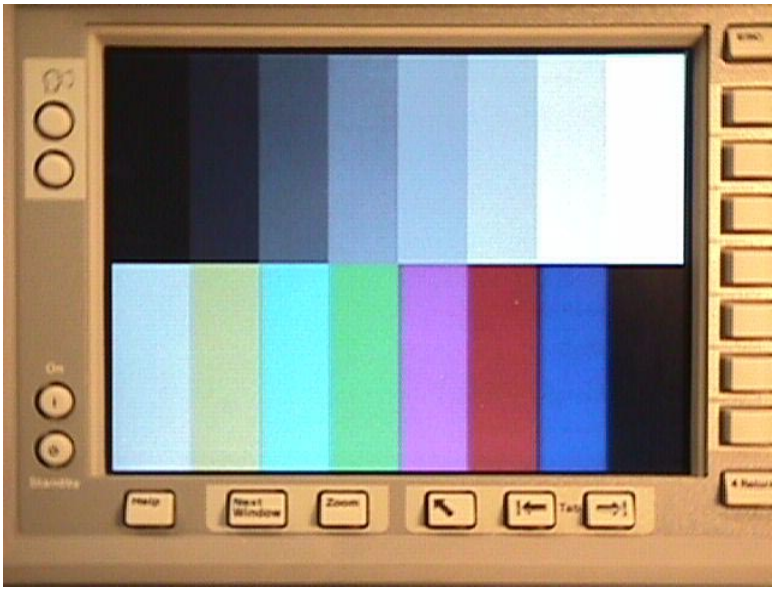
If **Trig**, **TV** is the active function, the line number used to trigger the analyzer sweep can be changed to examine the different parts of the video waveform. The line numbering scheme varies with TV standard, but TV test patterns will often be inserted in or near line 17.

Viewing the TV Picture

To view the TV picture, perform the following:

1. Press **Sweep**, **Sweep Time**, 100, **s** to set the sweep time to 100 seconds. The long sweep time is necessary to minimize disruption of the analog signal path during instrument retrace, which optimizes picture quality.
2. Press **Trig**, **TV Trig Setup**, **TV Monitor**.

Figure 2-40 TV Picture Display



When the picture is active, you can adjust the value of the function that was active prior to enabling the picture. For example, if center frequency was the active function and the frequency step size was set to the TV channel frequency spacing, you can increment or decrement through the TV channels by pressing the step keys (⇓ ⇑) of the analyzer. If resolution bandwidth was the active function, you can increase or decrease the amount of filtering to deal with strong adjacent channel signals.

NOTE When using the knob to vary the value of the active function, be aware that the instrument settings will not be updated until you stop turning the knob. Make small movements of the knob with frequent pauses.

Demodulating and Viewing Television Signals Example 2:

Measure Depth of Modulation

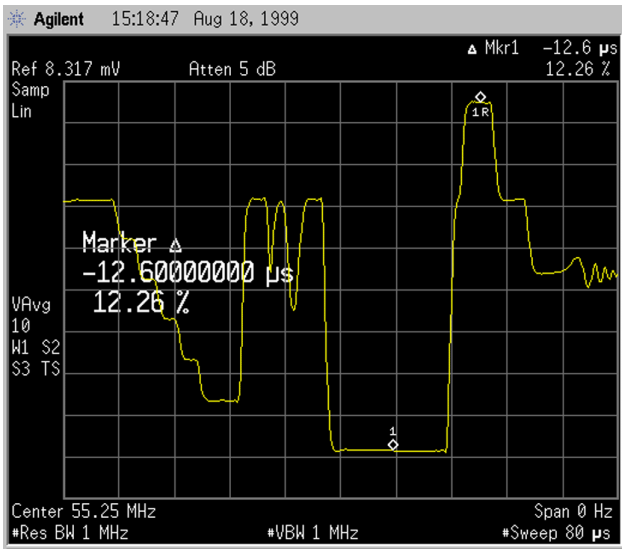
The depth of modulation provides a measure of the percentage of amplitude modulation (AM) on the visual carrier. With Option B7B, the analyzer can be used to measure the horizontal synchronization pulse level and vertical interval test signal (VITS) white level on an individual TV line from which a calculation of percent AM can be made. Note that this measurement method will not be valid for some types of scrambled video signals.

To perform a measurement of depth of modulation on an individual TV line, perform the following steps:

NOTE Before continuing, be sure you have performed the steps in Example 1 at the beginning of this section, "Demodulating and Viewing Television Signals (Option B7B)" on page 122 to display a TV waveform on the analyzer display with linear scaling.

1. Press **Trig**, **TV**, and enter a line number which has a TV test signal containing the reference white level within the waveform (100 IRE) such as the FCC, NTC-7 or ITU Composite Test Signal (typically found at or near line 17 within field 1 or field 2).
2. Set the sweep time to 80 µs (Option AYX or B7D required) by pressing **Sweep**, **Sweep Time**, 80, **µs**. This allows viewing of a complete TV line.
3. Set the resolution bandwidth and the video bandwidth to 1 MHz by pressing **BW/Avg**, **Res BW**, 1, **MHz** and **Video BW**, 1, **MHz**.
4. Turn on video averaging by pressing **BW/Avg**, **Average** (On), and entering a value of 10. This minimizes the waveform variations caused by the presence of additional RF signals near the picture carrier, as well as waveform noise or jitter.
5. Enable the marker by pressing **Marker**, **Normal**. Using the knob, locate the marker within the sync tip (NTSC or PAL waveforms) or the white level (SECAM waveforms) at the top of the waveform. Alternatively, you may press **Peak Search** to locate the marker on the peak excursion.
6. Enable delta marker mode by pressing **Marker**, **Delta**. Using the knob, locate the delta marker within the white level (NTSC or PAL waveforms) or the sync tip (SECAM waveforms) at the bottom of the waveform. Alternatively, you may press **Peak Search**, **Min Search** to locate the delta marker on the minimum excursion of the waveform. The delta marker readout will be in percent and may have a value near 12.5%.

Figure 2-41 Measuring Marker Delta (%) for Depth of Modulation (NTSC)



The depth of modulation (in percent) can now be determined by subtracting the marker readout (in percent) from 100. A typical value would be 87.5%.

TV Trig Setup Menu Functions

- **TV Source**

  When **TV Source** is set to **SA**, the analyzer serves to demodulate the TV signal, thus using the analyzer as a fixed tuned receiver. This allows stable, zero span sweeps of the baseband video waveform (although somewhat band limited by the RBW and VBW filters).

  When **TV Source** is set to **EXT VIDEO IN**, an external baseband video signal may be used to produce the TV line trigger. In this case, one may use an external TV tuner to obtain the baseband waveform of the given RF carrier for triggering the analyzer sweep, allowing the analyzer to be used in a swept mode for measurements of the RF spectrum, allowing the sweep to be synchronized to the video modulation. The **EXT VIDEO IN** connector is located on the rear panel of the analyzer.

- **TV Standard**

  Selection of a TV standard establishes the number of TV lines and the kind of color encoding method that is used. The number of TV lines establishes the defaults for the TV line counting circuits of the analyzer and the color encoding method is used to properly set up the TV picture display circuits. Option B7B supports both 525 line and 625 line systems and can provide a color TV picture for NTSC and PAL color encoding methods. A black and white picture is provided for the SECAM method.

  The ability to display a color picture is limited by the bandwidth

settings of the analyzer (resolution bandwidth and video bandwidth). However, baseband video signals input to the EXT VIDEO IN connector on the rear panel of the analyzer are minimally filtered, allowing a full color display of NTSC or PAL TV signals.

Table 2-3

TV Standard	Number of Lines per Frame	Approximate Field Rate	Color Encoding Method	Color Subcarrier Frequency
NTSC-M	525	60 Hz	NTSC	3.58 MHz
NTSC-Japan (no pedestal)	525	60 Hz	NTSC	3.58 MHz
PAL-M	525	60 Hz	PAL	4.43 MHz
PAL-B,D,G,H,I	625	50 Hz	PAL	4.43 MHz
PAL-N	625	50 Hz	PAL	4.43 MHz
PAL-N Combination	625	50 Hz	PAL	3.58 MHz
SECAM	625	50 Hz	SECAM	4.406 MHz, 4.250 MHz

- **Field**

  A television image or frame is composed of 525 (or 625 lines) delivered in two successive fields of 262.5 (or 312.5 lines) interlaced together on a CRT when displayed.

  When **Field** is set to **Entire Frame**, the line count starts at line one in field one (often referred to as the “odd field”) and ends at 525 (or 625) in field two (often referred to as the “even field”).

  When **Field** is set to **Field One** or **Field Two**, the line count begins at “1” with the first full line in the selected field and ends at count 263 (or 313) for Field One, and 262 (or 312) for Field Two.

- **Sync**

  Analog broadcast or cable television signals are usually amplitude modulated on an RF carrier. For NTSC and PAL broadcasts, typically the RF carrier amplitude is maximized at the sync tips of the baseband video waveform and minimized at the “white” level. This results in a demodulated waveform on the analyzer where the sync pulses are on top, or positive (**Sync** (Pos)).

With SECAM broadcasts, typically the RF carrier amplitude is minimized at the sync tips of the video waveform and maximized at the “white” level. This results in a waveform on the analyzer where the sync pulses are at the bottom, or negative (**Sync** (Neg)).

A normal baseband video waveform for all TV standards will have the sync tips on the bottom. When **TV Source** is set to **Ext Video In**, **Sync** should be set to **Neg**.

- **TV Monitor**

  When **TV Monitor** is pressed, the picture represented by the video waveform selected with **TV Source** is presented on the LCD display of the analyzer. The picture can only be viewed, not printed or saved. Pressing a key that normally brings up a menu restores the original graphical display with the selected menu enabled.

Fast Time Domain Sweeps

Trigger delay can be used to move the sweep trigger point arbitrarily across a given TV line or lines to allow closer examination of waveform patterns (Press **Trig**, **Trig Delay**, and enter a delay time).

In fast sweeps (20 µs to less than 5 ms), there may be up to one trace point of variation in the start time of the waveform digitalization process with respect to the actual TV trigger pulse. This randomness leads to the appearance of visual jitter on the LCD display of the analyzer. In this situation, video averaging may be used (N = 5, for example) to improve the “visual stability” of the displayed waveform. This type of jitter does not occur when sweep times are set greater than or equal to 5 ms where digitalization begins less than 100 ns after the trigger pulse in that mode (much less than 1 trace point of jitter).

Using External Millimeter Mixers (Option AYZ)

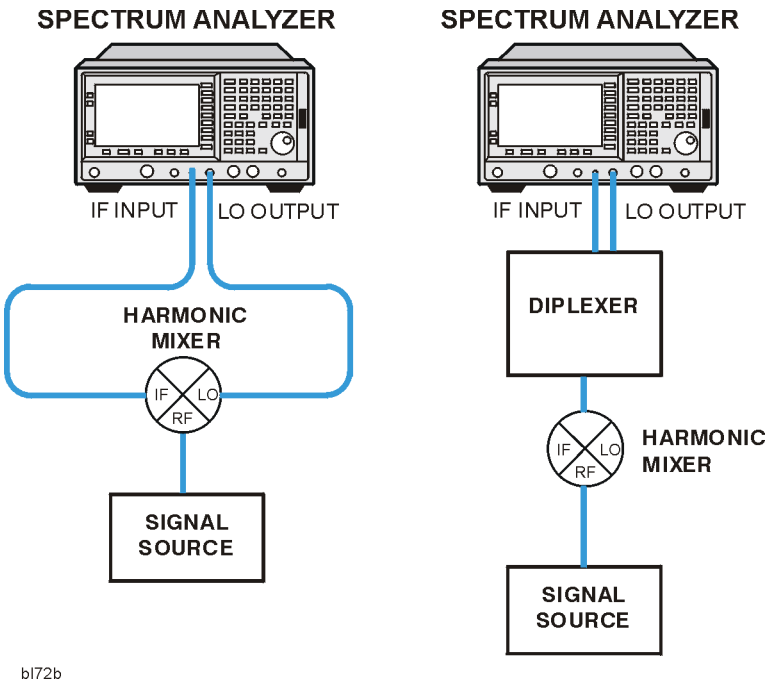
External millimeter mixers can be used to extend the frequency coverage of the E4407B spectrum analyzer. Agilent Technologies manufactures external mixers that do not require biasing and cover frequency ranges from 18 GHz to 110 GHz. Other manufacturers sell mixers that extend the range to 325 GHz, but may require biasing. The Agilent Technologies E4407B spectrum analyzer will support both types of mixers.

The Agilent Technologies E4407B spectrum analyzer contains an extensive menu of functions that help with millimeter measurements. The following examples explain how to connect external mixers to the spectrum analyzer, how to choose the band of interest, how to store and activate conversion-loss factors, and how to use the signal-identification functions.

Example 1: Making Measurements With Unpreselected Millimeter-wave Mixers

1. Connect the signal source and harmonic mixer to the analyzer as shown in Figure 2-42.

Figure 2-42



CAUTION The analyzer local oscillator output power is approximately +16 dBm. Be sure that your external harmonic mixer can accommodate the power level before connecting it to the analyzer.

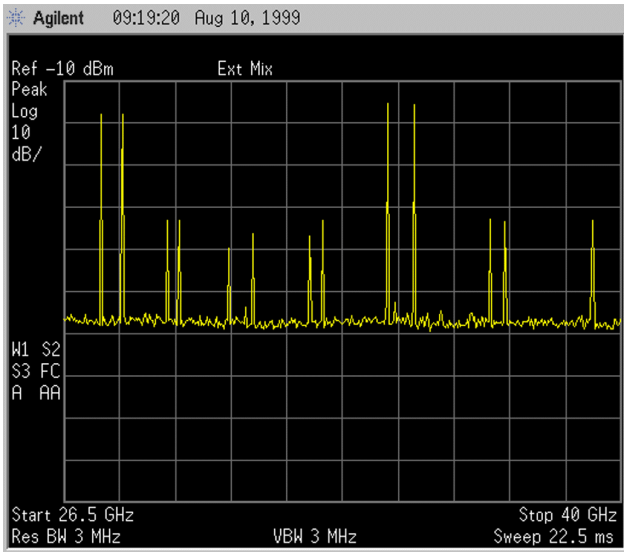
NOTE Agilent 5061-5458 SMA type cables should be used to connect the mixer IF and LO ports to the analyzer. Do not over-tighten the cables. The maximum torque should not exceed 112 N-cm (10 in-lb.)

2. On the analyzer, press **Preset**, **Factory Preset**, if present. Select external mixing by pressing **Input/Output**, **Input Mixer**, **Input Mixer** (Ext). The analyzer frequency band will be set to **26.5 – 40 GHz** (A). To choose a different band, press **Ext Mix Band** and then press the desired band frequency range/letter key. For this example, we will use band A, which ranges from 26.5 GHz to 40 GHz.

3. To correct for the conversion-loss of the harmonic mixer in use, the analyzer amplitude correction feature is used. Access this feature by pressing, **AMPLITUDE Y Scale**, **More**, **Corrections**. Select a correction set for use with external mixing. The recommended set to use is **Other** although any available set could be used. Press **Edit** to enter the appropriate conversion loss data for the mixer in use. On the Agilent 11970 harmonic mixers, these values are listed on the mixer. A single average value could be entered for the entire band by using only a single correction value. More correction points entered across the band in use will improve frequency response accuracy. Up to 200 points may be defined for each set. Once the desired correction points are entered, press **Return**, **Correction** (On) to activate correction set **Other**. This will also turn corrections on resulting in a calibrated display. It is recommended that the correction set entered be saved on the internal memory or the floppy drive for future reference.

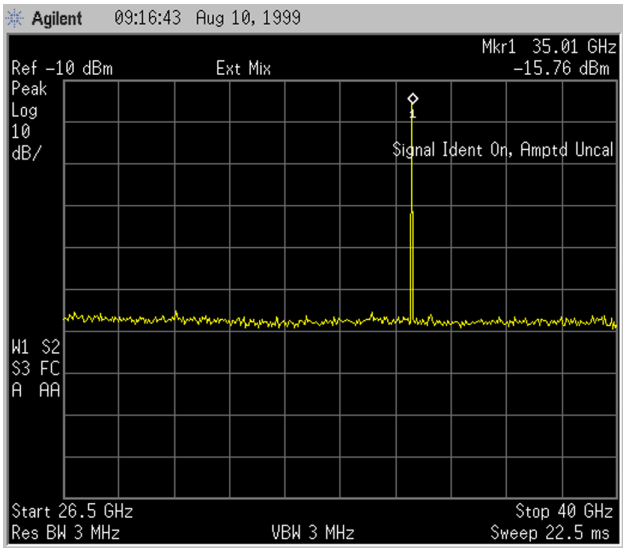
4. The IF output of a harmonic mixer will contain a signal at the intermediate frequency of the analyzer whenever the harmonic frequency of the LO and the frequency of the RF differ by the intermediate frequency. As a result, within a single harmonic band, a single input signal can produce multiple responses on the analyzer display, only one of which is valid (see Figure 2-43). These responses come in pairs, where members of the valid response pair are separated by 642.8 MHz and either the right-most (for negative harmonics) or left-most (for positive harmonics) member of the pair is the correct response.

Figure 2-43



5. Identification of valid responses is achieved by simply turning on the signal-identification feature. (**Preset** selects the **Image Suppress** signal identification mode.) Press **Input/Output**, **Input Mixer**, **Signal Ident** (On) and note that now only the valid response (35 GHz) remains. Refer to Figure 2-44. Press **Peak Search** to place a marker on the remaining response. The signal-identification routine can introduce slight amplitude errors which is indicated by the message `Signal Ident On, Amptd Uncal`. After identifying a signal of interest, press **Signal Ident** (Off) before making final amplitude measurements.

Figure 2-44



6. The Agilent 11970 Series harmonic mixers do not require bias. Mixers requiring bias can also be used with the E4407B. Generally,

the conversion loss calibration data for mixers requiring bias, will be most accurate when the correct bias conditions are present. Set the bias as follows:

a. To measure a signal, access external mixing and set the band as described previously.

b. To activate bias press **Input/Output**, **Input Mixer**, **Mixer Config**, **Mixer Bias** (On). A `+I` or `–I` will appear in the display annotation indicating bias is on.

c. Enter the desired bias current in mA with the data control keys.

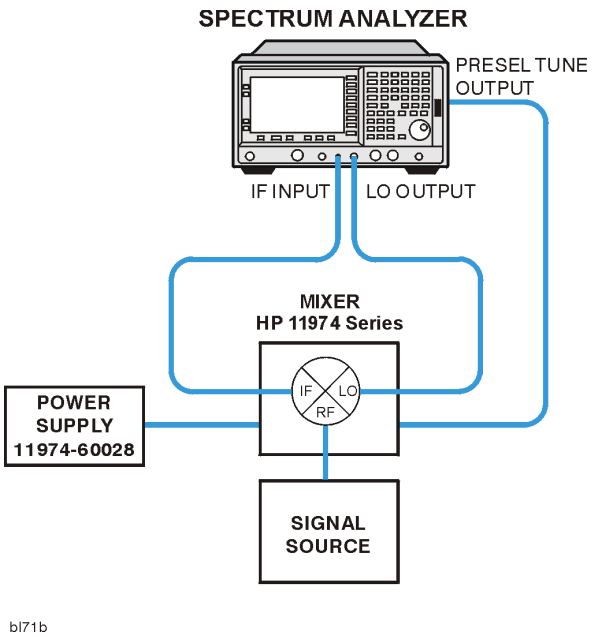
WARNING **The open-circuit bias voltage can be as great as ±3.5V through a source resistance of 500 ohms. Such voltage levels may appear when recalling an instrument state in which a bias setting has been stored.**

NOTE The bias value that appears on the analyzer display is expressed in terms of short-circuit current (that is, the current that would flow if the **IF INPUT** were shorted to ground). The actual current flowing into the mixer will be less.

Example 2: Making Measurements with Agilent 11974 Series Preselected Millimeter-Wave Mixers

1. Connect the signal source and preselected mixer to the analyzer as shown in Figure 2-45.

Figure 2-45



Frequency Tracking Calibration

This procedure is used to align the frequency of the preselector filter of the Agilent 11974 to the tuned frequency of the analyzer. This procedure should be followed any time that the Agilent 11974 is connected to a different analyzer. The calibration should be periodically checked.

1. Set the Agilent 11974 rear-panel switches “Agilent 70907B” and “LEDS” to the ON position, and the other two switches to the OFF position, in order for the Agilent 11974 to properly scale to the tune signal of the analyzer.
2. Configure the analyzer for a preselected external mixer by pressing the following keys:

   Preset, **Factory Preset** (if present), **Input/Output**, **Input Mixer** (Ext), **Mixer Config**, **Mixer Type** (Presel)
3. Set the desired band of operation. Press **Ext Mix Band**, **A**, **Q**, **U**, or **V**. (Note that only A,Q,U, and V bands are available.)
4. Set the Presel Adjust to 0 MHz by pressing **AMPLITUDE**, **Presel Adjust**, 0, **MHz**.
5. Set the analyzer to zero span by pressing **SPAN**, **Zero Span**.
6. Set the analyzer center frequency by pressing **FREQUENCY**, **Center Freq**, and enter the corresponding value for the appropriate mixer. (Refer to Table 2-4) On the rear panel of the Agilent 11974, adjust the corresponding potentiometer until one or both of the green LEDs are lit.

Table 2-4

Mixer Agilent P/N	Analyzer Center Frequency	Potentiometer
11974A	26.5 GHz	“26.5 GHz Adjust”
11974Q	33.0 GHz	“33.0 GHz Adjust”
11974U	40.0 GHz	“40.0 GHz Adjust”
11974V	50.0 GHz	“50.0 GHz Adjust”

7. Change the analyzer center frequency to the value indicated in Table 2-5 and again adjust the corresponding potentiometer on the rear panel of the HP 11974 until one or both of the green LEDs are lit.

Table 2-5

Mixer Agilent P/N	Analyzer Center Frequency	Potentiometer
11974A	40.0 GHz	“40.0 GHz Adjust”
11974Q	50.0 GHz	“50.0 GHz Adjust”
11974U	60.0 GHz	“60.0 GHz Adjust”
11974V	75.0 GHz	“75.0 GHz Adjust”

8. Repeat steps 6 and 7 until the green LEDs are lit at both frequencies without additional adjustments.

Band Selection

1. If necessary, configure the analyzer for preselected external mixing by pressing the following keys:

   Input/Output, **Input Mixer** (Ext), **Mixer Config**, **Mixer Type** (Presel)

2. If necessary, adjust the tracking of the Agilent 11974 to the analyzer being used for preselected external mixing, by using the Frequency Tracking Calibration procedure above.

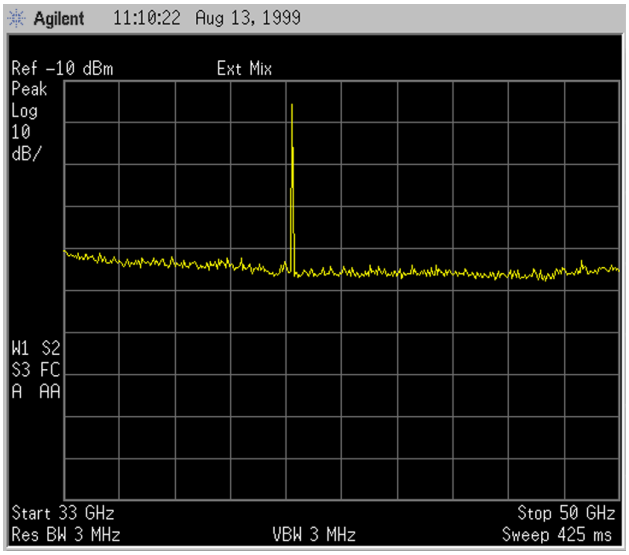
3. Press the following keys to select the desired mixing band: (In this example, we will use an Agilent 11974Q (33.0 to 50.0 GHz) to view a 40 GHz, -15 dBm signal.)

   Input/Output, **Input Mixer**, **Ext Mix Band**, **33-50 GHz** (Q)

Amplitude Calibration

1. Enter the conversion-loss versus frequency data from either the calibration label on the bottom of the Agilent 11974, or the supplied calibration sheet by using the procedure in step 3 of Example 1. The full Q-band is displayed in Figure 2-46.

Figure 2-46



2. To complete the amplitude calibration process, the preselector must be adjusted at each frequency of interest. Before making final amplitude measurements with the analyzer, perform the following:

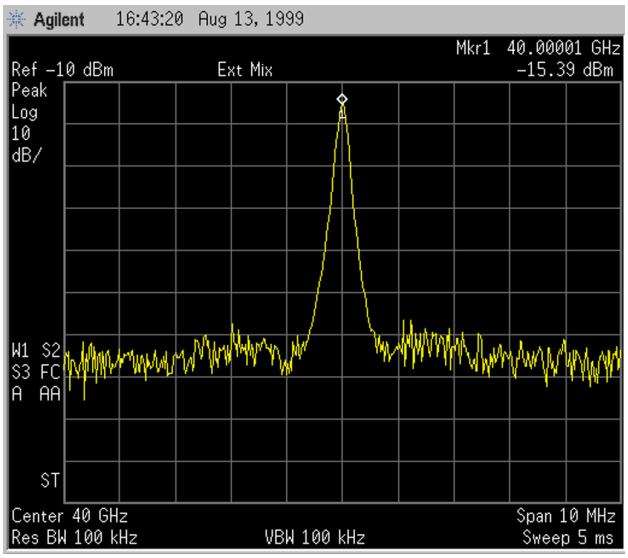
   a. Place a marker on the signal of interest.

   b. Press **SPAN**, **Span Zoom**, 10, **MHz** to zoom in on the signal.

   c. Press **AMPLITUDE**, **Presel Center**.

   The final amplitude measurement can now be read out with the marker. See Figure 2-47.

Figure 2-47


Agilent 16:43:20 Aug 13, 1999
Mkr1 40.00001 GHz
Ref -10 dBm
Ext Mix
-15.39 dBm
Peak
Log
10
dB/
W1 S2
S3 FC
A AA
ST
Center 40 GHz
Span 10 MHz
Res BW 100 kHz
VBW 100 kHz
Sweep 5 ms

3 Programming Examples

This chapter includes examples of how to program the analyzer using the analyzer SCPI programming commands. Twelve examples are written for an analyzer with an GPIB interface (Option A4H). Three examples are written for an analyzer with an RS-232 interface (Option 1AX). These examples do not apply to analyzers having Option 290 (8590 Series Programming Code Compatibility).

List of Programming Examples

The programming examples included in this chapter are:

Programming Examples System Requirements

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

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

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

C Programming Examples using VTL

The programming examples that are provided in this guide are written using the C programming language and the VTL (VISA transition library). This section includes some basic information about programming in the C language. Refer to your C programming language documentation for more details. (This information is taken from the manual “HP VISA Transition Library”, HP part number E2090-90026.) If you are using the National Instruments VISA library, most of this information will still apply, but the include and library files will be in different subdirectories. Also, this information assumes a computer running a Windows 95 operating system with an HP/Agilent 82341C GPIB interface card is being used. The following topics are included:

Typical Example Program Contents

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

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

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

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

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

`viPrintf`
`viScanf` These are the VTL formatted I/O functions that are patterned after those used in the C programming language. The **`viPrintf`** call sends the SCPI commands to the analyzer. The

viPrintf call can also be used to query the analyzer. The **viScanf** call is then used to read the results.

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

Linking to VTL Libraries

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

32-bit Version (assumes Windows 95 operating system):

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

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

16-bit Version:

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

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

See the following section for information on how to use the VTL run-time libraries.

Compiling and Linking a VTL Program

32-bit Applications (assumes Windows 95 operating system)

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

For Microsoft Visual C++ version 2.0 compilers:

- Select `Project | Update All Dependencies` from the menu.
- Select `Project | Settings` from the menu. Click on the `C/C++` button. Select `Code Generation` from the `Use Run-Time Libraries` list box. VTL requires these definitions for WIN32. Click on `OK` to close the dialog boxes.

- Select `Project | Settings` from the menu. Click on the `Link` button and add `visa32.lib` to the `Object / Library Modules` list box. Optionally, you may add the library directly to your project file. Click on `OK` to close the dialog boxes.
- You may wish to add the include file and library file search paths. They are set by doing the following:

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

For Borland C++ version 4.0 compilers:

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

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

16-bit Applications

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

For Microsoft Visual C++ version 1.5:

- To set the memory model, do the following:

  1. Select `Options | Project`.
  2. Click on the `Compiler` button, then select `Memory Model` from the `Category` list.
  3. Click on the `Model` list arrow to display the model options, and select `Large`.
  4. Click on `OK` to close the `Compiler` dialog box.
- You may wish to add the include file and library file search paths. They are set under the `Options | Directories` menu selection:

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

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

Example Program

This example program queries a GPIB device for an identification string and prints the results. Note that you must change the address if something other than the ESA default value of 18 is required.

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

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

void main ()
{
     /*Open session to 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 the sessions */
     viClose(vi);
     viClose(defaultRM);
}
```

Including the VISA Declarations File

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

```
#include “visa.h”
```

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

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

Opening a Session

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

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

NOTE

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

Device Sessions

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

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

viOpenDefaultRM (*sesn*);

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

The session returned from **viOpenDefaultRM** must be used in the *sesn* parameter of the **viOpen** function. The **viOpen** function then uses that session and the

device address specified in the *(resource name)* parameter to open a device session. The *vi* parameter in **viOpen** returns a session identifier that can be used with other VTL functions.

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

The following summarizes the parameters in the previous function calls:

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

The following is an example of opening sessions with 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.

VXI 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:

VXI0::24::INSTR Device at VXI logical address 24 that is of VISA type INSTR.

VXI2::128 Device at VXI logical address 128, in the third VXI system

(VXI2).

GPIB-VXI0::24	A VXI device at logical address 24. This VXI device is connected via an GPIB/VXI command module.
GPIB0::7::0	An 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 address 23.

```
ViSession defaultRM, vi;

.
.

viOpenDefaultRM(&defaultRM);

viOpen(defaultRM, “GPIB0::23::INSTR”, VI_NULL,VI_NULL,&vi);

.
.

viClose(vi);

viClose(defaultRM);
```

Closing a Session

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

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

Using Marker Peak Search and Peak Excursion

Example:

```
/*************************************************************/
/* Using Marker Peak Search and Peak Excursion               */
/*                                                           */
/* This example is for the E44xxB ESA Spectrum Analyzers     */
/* and E740xA EMC Analyzers.                                 */
/*                                                           */
/* This C programming example does the following.            */
/* The SCPI instrument commands used are given as            */
/* reference.                                                */
/*                                                           */
/* - Opens a GPIB session at address 18                      */
/* - Clears the Analyzer                                     */
/*      *CLS                                                 */
/* - Resets the Analyzer                                     */
/*      *RST                                                 */
/* - Sets the analyzer center frequency, span and units      */
/*      SENS:FREQ:CENT freq                                  */
/*      SENS:FREQ:SPAN freq                                  */
/*      UNIT:POW DBM                                         */
/* - Set the input port to the 50 MHz amplitude reference    */
/*      CAL:SOUR:STAT ON                                     */
/* - Set the analyzer to single sweep mode                   */
/*      INIT:CONT 0                                          */
/* - Prompt the user for peak excursion and set them         */
/*      CALC:MARK:PEAK:EXC dB                                */
/* - Set the peak threshold to -90 dBm                       */
/*      TRAC:MATH:PEAK:THR:STAT ON                           */
/*      TRAC:MATH:PEAK:THR -90                               */
/* - Trigger a sweep and wait for sweep to complete          */
/*      INIT:IMM;*WAI                                        */
/* - Set the marker to the maximum peak                      */
/*      CALC:MARK:MAX                                        */
/* - Query and read the marker frequency and amplitude       */
/*      CALC:MARK:X?                                         */
/*      CALC:MARK:Y?                                         */
/* - Close the session                                       */
/*************************************************************/

#include <stdio.h>
```

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

#define hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#definehpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"

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

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

  viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
  iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
  if( iResult == 0 )
  {
       /*Set the input port to the 50MHz amplitude reference for the models*/
       /*E4401B, E4411B and E7401A*/
       viPrintf(viESA,"CAL:SOUR:STAT ON \n");
  }
  else
  {
       /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
       /* to connect the amplitude reference output to the input*/
       printf ("Connect AMPTD REF OUT to the INPUT \n");
       printf ("......Press Return to continue \n");
       scanf( "%c",&cEnter);

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

void main()
{
```

```
/*Program Variables*/
ViStatus viStatus  = 0;
double dMarkerFreq = 0;
double dMarkerAmpl = 0;
float fPeakExcursion =0;
long lOpc = 0L;

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

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

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

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

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

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

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

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

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

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

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

  /*Trigger a sweep and wait for completion*/
  viPrintf(viESA,"INIT:IMM;*WAI\n");

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

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

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

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

Using Marker Delta Mode and Marker Minimum Search

```
/*********************************************************/
/* Using Marker Delta Mode and Marker Minimum Search     */
/*                                                       */
/* This example is for the E44xxB ESA Spectrum Analyzers */
/* and E740xA EMC Analyzers.                             */
/*                                                       */
/* This C programming example does the following.        */
/* The SCPI instrument commands used are given as        */
/* reference.                                            */
/*                                                       */
/* - Opens a GPIB session at address 18                  */
/* - Clears the Analyzer                                 */
/* - Resets the Analyzer                                 */
/*      *RST                                             */
/* - Set the input port to the 50 MHz amplitude reference*/
/*      CAL:SOUR:STAT ON                                 */
/* - Set the analyzer to single sweep mode               */
/*      INIT:CONT 0                                      */
/* - Prompts the user for the start and stop frequencies */
/* - Sets the start and stop frequencies                 */
/*      SENS:FREQ:START freq                             */
/*      SENS:FREQ:STOP freq                              */
/* - Trigger a sweep and wait for sweep completion       */
/*      INIT:IMM;*WAI                                    */
/* - Set the marker to the maximum peak                  */
/*      CALC:MARK:MAX                                    */
/* - Set the analyzer to activate the delta marker       */
/*      CALC:MARK:MODE DELT                              */
/* - Trigger a sweep and wait for sweep completion       */
/*      INIT:IMM;*WAI                                    */
/* - Set the marker to the minimum amplitude mode        */
/*      CALC:MARK:MIN                                    */
/* - Query and read the marker amplitude                 */
/*      CALC:MARK:Y?                                     */
/* - Close the session                                   */
/*********************************************************/

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

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

#define hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#define hpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"

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

/*Set the input port to the 50MHz amplitude reference*/
void Route50MHzSignal()
{
  viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
  iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
  if( iResult == 0 )
  {
       /*Set the input port to the 50MHz amplitude reference for the models*/
       /* E4401B, E4411B and E7401A*/
       viPrintf(viESA,"CAL:SOUR:STAT ON \n");
  }
  else
  {
       /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
       /* to connect the amplitude reference output to the input*/
       printf ("Connect AMPTD REF OUT to the INPUT \n");
       printf ("......Press Return to continue \n");
       scanf( "%c",&cEnter);

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

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

```
double dMarkerAmplitude = 0.0;
long   lOpc =0L;

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

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

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

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

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

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

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

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

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

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

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

/*Set the marker to the maximum peak*/
```

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

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

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

  /*Set the marker to minimum amplitude*/
  viPrintf(viESA,"CALC:MARK:MIN\n");

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

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

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

Performing Internal Self-alignment

```
/*********************************************************/
/* Performing Internal Self-alignment                    */
/*                                                       */
/* This example is for the E44xxB ESA Spectrum Analyzers */
/* and E740xA EMC Analyzers.                             */
/*                                                       */
/* This example shows two ways of executing an internal  */
/* self-alignment. The first demonstrates using the *OPC? */
/* query to determine when the alignment has completed. The */
/* second demonstrates using the query form of the CAL:ALL */
/* command to not only determine when the alignment has  */
/* been completed, but the pass/fail status of the align- */
/* ment process.                                         */
/*                                                       */
/* This C programming example does the following.        */
/* The SCPI instrument commands used  are given as       */
/* reference.                                            */
/*                                                       */
/* - Opens a GPIB session at address 18                  */
/* - Clears the Analyzer                                 */
/*      *CLS                                             */
/* - Resets the Analyzer                                 */
/*      *RST                                             */
/* - VISA function sets the time out to infinite         */
/* - Initiate self-alignment                             */
/*      CAL:ALL                                          */
/* - Query for operation complete                        */
/*      *OPC?                                            */
/* - Query for results of self-alignment                 */
/*      CAL:ALL?                                         */
/* - Report the results of the self-alignment            */
/* - Close the session                                   */
/*********************************************************/

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

#define hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
```

```
#define hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#definehpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"

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

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

  viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
  iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
  if( iResult == 0 )
  {
       /*Set the input port to the 50MHz amplitude reference for the models*/
       /*E4401B, E4411B, and E7401A*/
       viPrintf(viESA,"CAL:SOUR:STAT ON \n");
  }
  else
  {
       /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
       /* to connect the amplitude reference output to the input*/
       printf ("Connect AMPTD REF OUT to the INPUT \n");
       printf ("......Press Return to continue \n");
       scanf( "%c",&cEnter);

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

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

  /* Open a GPIB session at address 18*/
  viStatus=viOpenDefaultRM(&defaultRM);
  viStatus=viOpen(defaultRM,"GPIB0::18",VI_NULL,VI_NULL,&viESA);
```

```
if(viStatus)
{
     printf("Could not open a session to GPIB device at address 18!\n");
     exit(0);
}
/*Clear the instrument*/
viClear(viESA);

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

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

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

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

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

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

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

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

```
  {
       printf("Program Abort! error ocurred: last command was not completed!\n");
       exit(0);
  }
  /*Close the session*/
  viClose(viESA);
  viClose(defaultRM);
}
```

Reading Trace Data using ASCII Format (GPIB)

```
/*************************************************************/
/* Reading Trace Data using ASCII Format (GPIB)              */
/*                                                           */
/* This example is for the E44xxB ESA Spectrum Analyzers     */
/* and E740xA EMC Analyzers.                                 */
/*                                                           */
/* This C programming example does the following.            */
/* The required SCPI instrument commands are given as        */
/* reference.                                                */
/*                                                           */
/* - Opens a GPIB session at address 18                      */
/* - Clears the Analyzer                                     */
/* - Resets the Analyzer                                     */
/*      *RST                                                 */
/* - Set the input port to the 50 MHz amplitude reference    */
/*   E4411B or E4401B                                        */
/*      CAL:SOUR:STAT ON                                     */
/*   E4402, E4403B, E4404BE, 4405B, E4407B or E4408B         */
/*      Prompt to connect AMPTD REF OUT to INPUT             */
/*      CAL:SOUR STAT ON                                     */
/* - Query for the number of sweep points (only applies to   */
/*   firmware revisions A.04.00 and later); default is 401   */
/*      SENS:SWE:POIN?                                       */
/* - Sets the analyzer center frequency to 50 MHz            */
/*      SENS:FREQ:CENT 50 MHZ                                */
/* - Sets the analyzer span to 50 MHz                        */
/*      SENS:FREQ:SPAN 50 MHZ                                */
/* - Set the analyzer to single sweep mode                   */
/*      INIT:CONT 0                                          */
/* - Trigger a sweep and wait for sweep to complete          */
/*      INIT:IMM;*WAI                                        */
/* - Specify units in dBm                                    */
/*      UNIT:POW DBM                                         */
/* - Set the analyzer trace data to ASCII                    */
/*      FORM:DATA: ASC                                       */
/* - Trigger a sweep and wait for sweep to complete          */
/*      INIT:IMM;*WAI                                        */
/* - Query the trace data                                    */
/*      TRAC:DATA? TRACE1                                    */
/* - Remove the "," from the ACSII data                      */
/* - Save the trace data to an ASCII file                    */
/* - Close the session                                       */
```

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

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

#define hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#definehpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"

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

/*Set the input port to 50MHz amplitude reference*/
void Route50MHzSignal()
{
  viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
  iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
  if( iResult == 0 )
  {
       /*Set the input port to the 50MHz amplitude reference for the models*/
       /*E4401B, E4411B and E7401A*/
       viPrintf(viESA,"CAL:SOUR:STAT ON \n");
  }
  else
  {
       /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
       /* to connect the amplitude reference output to the input*/
       printf ("Connect AMPTD REF OUT to the INPUT \n");
       printf ("......Press Return to continue \n");
       scanf( "%c",&cEnter);

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

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

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

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

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

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

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

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

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

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

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

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

  /*Trigger a sweep and wait for sweep to complete */
  viPrintf(viESA,"INIT:IMM;*WAI\n");

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

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

  /*Trigger a sweep and wait for sweep to complete */
  viPrintf(viESA,"INIT:IMM;*WAI\n");

  /*Query the Trace Data using ASCII Format */
  viQueryf(viESA,"%s\n", "%#t","TRAC:DATA? TRACE1" , &iNum , cResult);

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

  /*Save trace data to an ASCII to a file, by removing the "," token*/
  fTraceFile=fopen("C:\\temp\\ReadAscGpib.txt","w");
  fprintf(fTraceFile,"ReadAscGpib.exe Output\nAgilent Technologies 2000\n\n");
  fprintf(fTraceFile,"\tAmplitude of point[%d] = %s dBm\n",lCount+1,cToken);
    while (cToken != NULL)
       {
       lCount++;
       cToken =strtok(NULL,",");
             if (lCount != iSwpPnts)
             fprintf(fTraceFile,"\tAmplitude of point[%d] = %s
dBm\n",lCount+1,cToken);
       }
  fprintf(fTraceFile,"\nThe Total trace data points of the spectrum are:[%d]
\n\n",lCount);
  fclose(fTraceFile);

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

Reading Trace Data Using 32-bit Real Format (GPIB)

```
/*********************************************************/
/* Reading Trace Data using 32-bit Real Format (GPIB)    */
/*                                                       */
/* This example is for the E44xxB ESA Spectrum Analyzers */
/* and E740xA EMC Analyzers.                             */
/*                                                       */
/* This C programming example does the following.        */
/* The SCPI instrument commands used are given as        */
/* reference.                                            */
/*                                                       */
/* - Opens a GPIB session at address 18                  */
/* - Clears the Analyzer                                 */
/* - Resets the Analyzer                                 */
/*      *RST                                             */
/* - Set the input port to the 50 MHz amplitude reference*/
/*      CAL:SOUR:STAT ON                                 */
/* - Query for the number of sweep points (for firmware  */
/*   revisions A.04.00 and later). Default is 401.       */
/*       SENS:SWE:POIN?                                  */
/* - Calculate the number of bytes in the header         */
/* - Set the analyzer to single sweep mode               */
/*      INIT:CONT 0                                      */
/* - Sets the analyzer center frequency and span to 50 MHz*/
/*      SENS:FREQ:CENT 50 MHZ                            */
/*      SENS:FREQ:SPAN 50 MHZ                            */
/* - Specify 10 dB per division for the amplitude scale in*/
/*   and dBm Units                                       */
/*      DISP:WIND:TRAC:Y:SCAL:PDIV 10 dB                 */
/*      UNIT:POW DBM                                     */
/* - Set the analyzer trace data to 32-bit Real          */
/*      FORM:DATA: REAL,32                               */
/* - Set the binary order to swap                        */
/*      FORM:BORD SWAP                                   */
/* - Trigger a sweep and wait for sweep to complete      */
/*      INIT:IMM;*WAI                                    */
/* - Calculate the number of bytes in the trace record   */
/* - Query the trace data                                */
/*      TRAC:DATA? TRACE1                                */
/* - Remove the "," from the ACSII data                  */
/* - Save the trace data to an ASCII file                 */
/* - Close the session                                   */
```

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

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

#define  hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define  hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#define hpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"

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

void Route50MHzSignal()
{
viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
if( iResult == 0 )
{
     /*Set the input port to the 50MHz internal reference source for the models*/
     /*E4401B, E4411B and E7401A*/
     viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
else
{
     /* For the analyzers having frequency limits >= 3GHz, prompt the user to*/
     /* connect the amplitude reference output to the input*/
     printf ("Connect AMPTD REF OUT to the INPUT \n");
     printf ("......Press Return to continue \n");
     scanf( "%c",&cEnter);

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

void main()
{
```

```
/*Program Variables*/
ViStatus viStatus= 0;
    ViChar _VI_FAR cResult[5000] = {0};
ViReal32 dTraceArray[401] = {0};
char cBufferInfo[6]= {0};
long lNumberBytes =0L;
long lOpc =0L;
unsigned long lRetCount = 0L;
int iSize = 0;
/*BytesPerPoint is 4 for Real32 or Int32 formats, 8 for Real64, and 2 for Uint16*/
int iBytesPerPnt = 4;
int iSwpPnts = 401;
int iDataBytes=1604;
int iHeaderBytes=6;
FILE *fTraceFile;

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

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

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

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

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

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

```
iDataBytes = (iSwpPnts*iBytesPerPnt);
lNumberBytes = iDataBytes;
while ((iDataBytes = (iDataBytes / 10 )) > 0 )
{
     iHeaderBytes++;
}

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

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

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

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

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

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

/*Trigger a sweep and wait for sweep to complete*/
viPrintf(viESA,"INIT:IMM;*WAI\n");

/*Calculate size of trace record. This will be sum of HeaderBytes, NumberBytes*/
/*(the actual data bytes) and the "/n" terminator*/
iSize = lNumberBytes +iHeaderBytes+1;

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

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

/*Save trace data to an ASCII file*/
fTraceFile=fopen("C:\\temp\\ReadTrace32Gpib.txt","w");
fprintf(fTraceFile,"ReadTrace32Gpib.exe Output\nAgilent Technologies 2000\n\n");
fprintf(fTraceFile,"The %d trace data points of the
spectrum:\n\n",(lNumberBytes/4));
for ( long i=0;i<lNumberBytes/4;i++)
```

```
    fprintf(fTraceFile,"\tAmplitude of point[%d] = %.2lf
dBm\n",i+1,dTraceArray[i]);
fclose(fTraceFile);

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

Reading Trace Data Using ASCII Format (RS-232)

```
/*********************************************************/
/* Reading Trace Data using ASCII Format (RS-232)        */
/*                                                       */
/* This example is for the E44xxB ESA Spectrum Analyzers */
/* and E740xA EMC Analyzers.                             */
/*                                                       */
/* This C programming example does the following.        */
/* The SCPI instrument commands used are given as        */
/* reference.                                            */
/*                                                       */
/* - Opens an RS-232 session at COM1/COM2                */
/* - Clears the Analyzer                                 */
/* - Resets the Analyzer                                 */
/*      *RST                                             */
/* - Set the input port to the 50 MHz amplitude reference*/
/*      CAL:SOUR:STAT ON                                 */
/* - Query for the number of sweep points (for firmware  */
/*   revisions A.04.00 and later). Default is 401.       */
/*      SENS:SWE:POIN?                                   */
/* - Set the analyzer to single sweep mode               */
/*      INIT:CONT 0                                      */
/* - Sets the analyzer center frequency and span to 50 MHz*/
/*      SENS:FREQ:CENT 50 MHZ                            */
/*      SENS:FREQ:SPAN 50 MHZ                            */
/* - Trigger a sweep                                     */
/*      INIT:IMM                                         */
/* - Check for operation complete                        */
/*      *OPC?                                            */
/* - Specify dBm Unit                                    */
/*      UNIT:POW DBM                                     */
/* - Set the analyzer trace data ASCII                   */
/*      FORM:DATA: ASC                                   */
/* - Trigger a sweep                                     */
/*      INIT:IMM                                         */
/* - Check for operation complete                        */
/*      *OPC?                                            */
/* - Query the trace data                                */
/*      TRAC:DATA? TRACE1                                */
/* - Remove the "," from the ACSII data                  */
/* - Save the trace data to an ASCII file                */
/* - Close the session                                   */
/*********************************************************/
```

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

#define  hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define  hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#define  hpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"

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

/*Set the input port to 50MHz amplitude reference*/
void Route50MHzSignal()
{
viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
if( iResult == 0 )
{
      /*Set the input port to the 50MHz amplitude reference for the models*/
      /*E4411B and E4401B*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
else
{
      /* For the analyzers having frequency limits >= 3GHz, prompt the user to/*
      /* connect the amplitude reference output to the input*/
      printf ("Connect AMPTD REF OUT to the INPUT \n");
      printf ("......Press Return to continue \n");
      scanf( "%c",&cEnter);

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

void main()
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
/*********************************************************/
/* Reading Trace Data using 32-bit Real Format (RS-232)  */
/*                                                       */
/* This example is for the E44xxB ESA Spectrum Analyzers */
/* and E740xA EMC Analyzers.                             */
/*                                                       */
/* This C programming example does the following.        */
/* The SCPI instrument commands used are given as        */
/* reference.                                            */
/*                                                       */
/* - Opens an RS-232 session at COM1/COM2                */
/* - Clears the Analyzer                                 */
/* - Resets the Analyzer                                 */
/*      *RST                                             */
/* - Set the input port to the 50 MHz amplitude reference*/
/*      CAL:SOUR:STAT ON                                 */
/* - Query for the number of sweep points (for firmware  */
/*   revision A.04.00 and later). Default is 401.        */
/*      SENS:SWE:POIN?                                   */
/* - Calculate the number of bytes in the header         */
/* - Set the analyzer to single sweep mode               */
/*      INIT:CONT 0                                      */
/* - Sets the analyzer center frequency and span to 50 MHz*/
/*      SENS:FREQ:CENT 50 MHZ                            */
/*      SENS:FREQ:SPAN 50 MHZ                            */
/* - Specify 10 dB per division for the amplitude scale in*/
/*   and dBm Units                                       */
/*      DISP:WIND:TRAC:Y:SCAL:PDIV 10 dB                 */
/*      UNIT:POW DBM                                     */
/* - Set the analyzer trace data to 32-bit Real          */
/*      FORM:DATA: REAL,32                               */
/* - Set the binary order to swap                        */
/*      FORM:BORD SWAP                                   */
/* - Trigger a sweep                                     */
/*      INIT:IMM                                         */
/* - Check for operation complete                        */
/*      *OPC?                                            */
/* - Calculate the number of bytes in the trace record   */
/* - Set VISA timeout to 60 seconds, to allow for slower */
/*   transfer times caused by higher number of sweep points*/
/*   at low baud rates.                                  */
/* - Set VISA to terminate read after buffer is empty    */
```

```
/* - Query the trace data                                    */
/*      TRAC:DATA? TRACE1                                    */
/* - Reset VISA timeout to 3 seconds                         */
/* - Remove the "," from the ACSII data                      */
/* - Save the trace data to an ASCII file                    */
/* - Close the session                                       */
/*************************************************************/

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

#define hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#definehpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"

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

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

viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
if( iResult == 0 )
{
     /*Set the input port to the 50MHz amplitude reference for the models*/
     /*E4411B and E4401B*/
     viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
else
{
     /* For the analyzers having frequency limits >= 3GHz, prompt the user to*/
     /* connect the amplitude reference output to the input*/
     printf ("Connect AMPTD REF OUT to the INPUT \n");
     printf ("......Press Return to continue \n");
```

```
     scanf( "%c",&cEnter);

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
/*Calculate size of trace record. This will be the sum of HeaderBytes, Number*/
/*Bytes (the actual data bytes) and the "\n" terminator*/
iSize = lNumberBytes + iHeaderBytes + 1;

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

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

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

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

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

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

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

Using Limit Lines

```
/*********************************************************/
/* Using Limit Lines                                     */
/*                                                       */
/* This example is for the E44xxB ESA Spectrum Analyzers */
/* and E740xA EMC Analyzers.                             */
/*                                                       */
/* This C programming example does the following.        */
/* The SCPI instrument commands used are given as        */
/* reference.                                            */
/*                                                       */
/*                                                       */
/* - Open a GPIB session at address 18.                  */
/* - Clear the analyzer.                                 */
/*       *CLR                                            */
/* - Reset the analyzer.                                 */
/*       *RST                                            */
/* - Set Y-Axis Units to dBm                             */
/*       UNIT:POW DBM                                    */
/* - Define the upper limit line to have frequency/      */
/*   amplitude pairs.                                    */
/*       CALC:LLINE1:CONT:DOM FREQ                       */
/*       CALC:LLINE1:TYPE UPP                            */
/*       CALC:LLINE1:DISP ON                             */
/*       CALC:LLINE1:DATA freq1,amp1,1,freq2,amp2,1...   */
/* - Define the lower limit line to have frequency/amplitude*/
/*   pairs.                                              */
/*       CALC:LLINE2:CONT:DOM FREQ                       */
/*       CALC:LLINE2:TYPE LOW                            */
/*       CALC:LLINE2:DISP ON                             */
/*       CALC:LLINE2:DATA freq1,amp1,1,freq2,amp2,1...   */
/* - Turn the limit line test function on.               */
/*       CALC:LLINE2:STAT ON                             */
/* - Set the analyzer to a center frequency of 50 MHz, span */
/*   to 20 MHz, and resolution bandwidth to 1 MHz.       */
/*       SENS:FREQ:SPAN 20 MHZ                           */
/*       SENS:FREQ:CENT 50 MHZ                           */
/*       SENS:BWID:RES 1 MHZ                             */
/* - Turn the limit line test function on.               */
/* - Set the analyzer reference level to 0 dBm.          */
/*       DISP:WIND:TRAC:Y:SCAL:RLEV 0                    */
/* - Set the input port to the 50 MHz amplitude reference.  */
/*       CAL:SOUR:STAT ON                                */
```

```
/* - Check to see if limit line passes or fails.  It should */
/*   pass.                                                  */
/*       CALC:LLINE:FAIL?                                   */
/* - Pause for 5 seconds.                                   */
/* - Deactivate the 50 MHz alignment signal.                */
/*       CAL:SOUR:STAT OFF                                  */
/* - The limit line test should fail.                       */
/* - Close the session.                                     */
/************************************************************/

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

#define  hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define  hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#define hpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"

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

/*Set the input port to 50MHz amplitude reference*/
void Route50MHzSignal()
{
  viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
  iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
  if( iResult == 0 )
  {
       /*Set the input port to the 50MHz amplitude reference for the models*/
       /*E4401B, E4411B, and E7401A*/
       viPrintf(viESA,"CAL:SOUR:STAT ON \n");
  }
  else
  {
```

```
      /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
      /* to connect the amplitude reference output to the input*/
      printf ("Connect AMPTD REF OUT to the INPUT \n");
      printf ("......Press Return to continue \n");
      scanf( "%c",&cEnter);

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

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

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

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

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

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

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

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

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

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

  /*Set the Y-Axis Units to dBm */
  viPrintf(viESA, "UNIT:POW DBM\n");

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

/*Pause for 5 seconds*/
YIELD;

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

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

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

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

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

Measuring Noise

```
/*********************************************************/
/* Measuring Noise                                       */
/*                                                       */
/* This example is for the E44xxB ESA Spectrum Analyzers */
/* and E740xA EMC Analyzers.                             */
/*                                                       */
/* This C programming example does the following.        */
/* The SCPI instrument commands used are given as        */
/* reference.                                            */
/*                                                       */
/* - Opens a GPIB session at address 18                  */
/* - Clears the Analyzer                                 */
/* - Resets the Analyzer                                 */
/*      *RST                                             */
/* - Sets the center frequency and span                  */
/*      SENS:FREQ:CENT 50 MHZ                            */
/*      SENS:FREQ:SPAN 10 MHZ                            */
/* - Set the input port to the 50 MHz amplitude reference*/
/*      CAL:SOUR:STAT ON                                 */
/* - Set the analyzer to single sweep mode               */
/*      INIT:CONT 0                                      */
/* - Trigger a sweep and wait for sweep completion       */
/*      INIT:IMM;*WAI                                    */
/* - Set the marker to the maximum peak                  */
/*      CALC:MARK:MAX                                    */
/* - Set the analyzer to active delta marker             */
/*      CALC:MARK:MODE DELT                              */
/* - Set the delta marker to 2 MHZ                       */
/*      CALC:MARK:X 2E+6                                 */
/* - Activate the noise marker function                  */
/*      CALC:MARK:FUNC NOIS                              */
/* - Trigger a sweep and wait for sweep completion       */
/*      INIT:IMM;*WAI                                    */
/* - Query the marker delta amplitude from the analyzer  */
/*      CALC:MARK:Y?                                     */
/* - Report the marker delta amplitude as the carrier to */
/*   noise ratio in dBc/Hz                               */
/* - Close the session                                   */
/*********************************************************/

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

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

#define hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#definehpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"

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

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

viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
if( iResult == 0 )
{
      /*Set the input port to the 50MHz amplitude reference for the models*/
      /*E4401B, E4411B amd E7401A*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
else
{
      /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
      /* to connect the amplitude reference output to the input*/
      printf ("Connect AMPTD REF OUT to the INPUT \n");
      printf ("......Press Return to continue \n");
      scanf( "%c",&cEnter);

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

void main()
{
```

```
/*Program Variables*/
ViStatus viStatus  = 0;
double dMarkAmp =0.0;
long lOpc=0L;

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

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

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

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

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

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

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

/*Trigger a sweep and wait for sweep completion*/
viPrintf(viESA,"INIT:IMM;*WAI \n");

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

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

```
}

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

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

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

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

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

/*Trigger a sweep and wait for sweep completion*/
viPrintf(viESA,"INIT:IMM;*WAI \n");

/*Query and read the marker delta amplitude from the analyzer*/
viQueryf(viESA,"CALC:MARK:Y? \n","%lf",&dMarkAmp);

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

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

Entering Amplitude Correction Data

```
/*********************************************************/
/* Entering Amplitude Correction Data                    */
/*                                                       */
/* This example is for the E44xxB ESA Spectrum Analyzers */
/* and E740xA EMC Analyzers.                             */
/*                                                       */
/* This C programming example does the following.        */
/* The SCPI instrument commands used are given as        */
/* reference.                                            */
/*                                                       */
/* - Opens a GPIB session at address 18                  */
/* - Clears the Analyzer                                 */
/* - Resets the Analyzer                                 */
/*      *RST                                             */
/* - Sets the stop frequency to 1.5 GHz                  */
/*      SENS:FREQ:STOP 1.5 GHZ                           */
/* - Set the input port to the 50 MHz amplitude reference*/
/*      CAL:SOUR:STAT ON                                 */
/* - Enter amplitude correction frequency/amplitude pairs:*/
/*   0 Hz/ 0 dB, 100 MHz/5 dB, 1 GHz/-5 dB, 1.5 GHz/ 10 dB*/
/*      SENS:CORR:CSET1:DATA 0,0,100E6,5.0,1.0E9,-5.0,...*/
/* - Activate amplitude correction                       */
/*      SENS:CORR:CSET1:DATA                             */
/*      SENS:CORR:CSET1:ALL:STAT ON                      */
/* - Query the analyzer for the amplitude corection factors */
/*      SENS:CORR:CSET1:DATA?                            */
/* - Store them in an array                              */
/* - Display the array                                   */
/* - Close the session                                   */
/*********************************************************/

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

#define hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#define hpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"
```

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

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

viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
if( iResult == 0 )
{
      /*Set the input port to the 50MHz amplitude reference for the models*/
      /*E4401B, E4411B, and E7401A*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
else
{
      /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
      /* to connect the amplitude reference output to the input*/
      printf ("Connect AMPTD REF OUT to the INPUT \n");
      printf ("......Press Return to continue \n");
      scanf( "%c",&cEnter);

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

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

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

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

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

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

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

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

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

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

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

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

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

/*Store the array (frequency) value into a two-dimensional real array*/
aRealArray[lFreq=0][lCount=0] = atof( cToken);

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

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

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

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

Status Register–Determine When a Measurement is Done

```
/*********************************************************/
/* Status Register - Determine when a measurement is done   */
/*                                                          */
/* This example is for the E44xxB ESA Spectrum Analyzers    */
/* and E740xA EMC Analyzers.                                */
/*                                                          */
/* This C programming example does the following.           */
/* The SCPI instrument commands used are given as           */
/* reference.                                               */
/*                                                          */
/* - Opens a GPIB session at address 18                     */
/* - Resets the Analyzer                                    */
/*      *RST                                                */
/* - Clears the analyzer status byte                        */
/*      *CLS                                                */
/* - Sets the analyzer to single sweep mode                 */
/*      INIT:CONT 0                                         */
/* - Route the amplitude reference to the analyzer input    */
/*      CAL:SOUR:STAT ON                                    */
/* - Set the analyzer center frequency, span and Res BW     */
/*      SENS:FREQ:CENT 50 MHz                               */
/*      SENS:FREQ:SPAN 10 MHz                               */
/*      SENS:BAND:RES 300 kHz                               */
/* - Trigger a sweep and wait for completion of sweep       */
/*      INIT:IMM                                            */
/*      *OPC?                                               */
/* - Sets the service request mask to assert SRQ when       */
/*   either a measurement is uncalibrated or an error       */
/*   message has occurred.                                  */
/*      *SRE 96                                             */
/*      *ESE 35                                             */
/* - Set the computer to response to an interrupt           */
/* - Send an undefined command to the ESA                   */
/*       IDN (illegal command)                              */
/* - Wait for the SRQ                                       */
/* - When an interrupt occurs, poll all instruments         */
/* - Report the nature of the interrupt on the ESA analyzer */
/* - Pause 5 seconds to observe the analyzer                */
/* - Set the ESA to perform 80 video averages               */
/*       SENS:AVER:TYPE LPOW                                */
/*       SENS:AVER:COUN 80                                  */
```

```
/*       SENS:AVER:STAT ON                                        */
/* - Trigger a measurement, and set OPC bit when done            */
/*       INIT:IMM                                                 */
/*       *OPC                                                     */
/* - Wait for the SRQ                                            */
/* - When an interrupt occurs, poll all instruments              */
/* - Report the nature of the interrupt on the ESA analyzer      */
/* - Clear the status register enable                            */
/*      *SRE 0                                                   */
/* - Clear the status byte of the ESA                            */
/*      *CLS                                                     */
/* - Close the session                                           */
/*****************************************************************/

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

#define  hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define  hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#define hpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"
#define  YIELD Sleep(10)

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


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

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

}

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

viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
if( iResult == 0 )
{
       /*Set the input port to the 50MHz amplitude reference for the models*/
       /*E4401B, E4411B and E7401A*/
       viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
else
{
       /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
       /* to connect the amplitude reference output to the input*/
       printf ("Connect AMPTD REF OUT to the INPUT \n");
       printf ("......Press Return to continue \n");
       scanf( "%c",&cEnter);

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

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

```
/* Make sure it is an SRQ event, ignore if stray event*/
if (eventType!=VI_EVENT_SERVICE_REQ)
{
      printf ("\n Stray event type0x%1x\n",eventType);

      /*Return successfully*/
      return VI_SUCCESS;
}
/* When an interrupt occurs,determine which device generated the interrupt
(if an instrument other than the ESA generates the interrupt, simply report
"Instrument at GPIB Address xxx Has Generated an Interrupt").*/
printf ("\n\n SRQ event occurred!\n");
printf ("\n ... Original Device Session = %t\n",viESA);

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

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

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

/* Open a GPIB session at address 18*/
viStatus=viOpenDefaultRM(&defaultRM);
int address =18;
viStatus=viOpen(defaultRM,"GPIB0::18",VI_NULL,VI_NULL,&viESA);
if(viStatus)
{
```

```
      printf("Could not open a session to GPIB device at address 18!\n");
      exit(0);
}
/*Clear the instrument*/
viClear(viESA);

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

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

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

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

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

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

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

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

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

/*Make sure the previous command has been completed*/
viQueryf(viESA, "*OPC?\n", "%d", &lOpc);
if (!lOpc)
{
      printf("Program Abort! error ocurred: last command was not completed!\n");
      exit(0);
}
/* Set the service request mask to assert SRQ when either a measurement
   is completed or an error message has occurred.*/
viPrintf(viESA,"*SRE 96\n");
viPrintf(viESA,"*ESE 35\n");

/* Configure the computer to respond to an interrupt*/
```

```
/*install the handler and enable it */
viInstallHandler(viESA, VI_EVENT_SERVICE_REQ, mySrqHdlr,iAddress);
viEnableEvent(viESA, VI_EVENT_SERVICE_REQ,VI_HNDLR,VI_NULL);

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

/*Wait for SRQ */
WaitForSRQ();

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

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

/* Set the service request mask to assert SRQ when either a measurement
   is completed or an error message has occurred.*/
viPrintf(viESA,"*SRE 96\n");
viPrintf(viESA,"*ESE 35\n");

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

/*Wait for SRQ */
WaitForSRQ();

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


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

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

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

Determine if an Error has Occurred

```
/*********************************************************/
/* Determine if an error has occurred                    */
/*                                                       */
/* This example is for the E44xxB ESA Spectrum Analyzers */
/* and E740xA EMC Analyzers.                             */
/*                                                       */
/* This C programming example does the following.        */
/* The SCPI instrument commands used are given as        */
/* reference.                                            */
/*                                                       */
/* - Opens a GPIB session at address 18                  */
/* - Clears the Analyzer                                 */
/*      *CLS                                             */
/* - Resets the Analyzer                                 */
/*      *RST                                             */
/* - Sets the service request mask to assert SRQ when    */
/*   either a measurement is uncalibrated or an error    */
/*   message has occurred.                               */
/*      STAT:QUES:ENAB 512                               */
/*      STAT:QUES:INT:ENAB 8                             */
/*      *ESE 35                                          */
/*      *SRE 104                                         */
/* - Set the center frequency to 500MHz and span to 100MHz */
/*      SENS:FREQ:CENT 500 MHZ                           */
/*      SENS:FREQ:SPAN 100 MHZ                           */
/* - Set the analyzer to an uncalibrated state           */
/* - When an interrupt occurs, poll all instruments      */
/* - Report the nature of the interrupt on the ESA analyzer */
/* - Pause 5 seconds to observe the analyzer             */
/* - Sets the service request mask to assert SRQ when    */
/*   either a measurement is uncalibrated or an error    */
/*   message has occurred.                               */
/*      *ESE 35                                          */
/*      *SRE 96                                          */
/* - Send an illegal command to the ESA                  */
/*       IDN  (illegal command)                          */
/* - When an interrupt occurs, poll all instruments      */
/* - Report the nature of the interrupt on the ESA analyzer */
/* - Clear the analyzer status registers                 */
/*      *SRE 0                                           */
/*      *ESE 0                                           */
/*      STAT:QUES:ENAB 0                                 */
```

```
/*      STAT:QUES:INT:ENAB 0                                    */
/*      *CLS                                                    */
/* - Continue monitoring for an interrupt                       */
/* - Close the session                                          */
/*************************************************************/

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

#define   hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define   hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#define  hpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"
#define   YIELD Sleep(10)

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


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

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

```
}
printf("\n");

}

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

viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
if( iResult == 0 )
{
      /*Set the input port to the 50MHz amplitude reference for the models*/
      /*E4401B, E4411B and E7401A*/
      viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
else
{
      /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
      /* to connect the amplitude reference output to the input*/
      printf ("Connect AMPTD REF OUT to the INPUT \n");
      printf ("......Press Return to continue \n");
      scanf( "%c",&cEnter);

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

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

/* Make sure it is an SRQ event, ignore if stray event*/
if (eventType!=VI_EVENT_SERVICE_REQ)
{
      printf ("\n Stray event type0x%1x\n",eventType);
      /*Return successfully*/
      return VI_SUCCESS;
}
```

```
/* When an interrupt occurs, determine which device generated the interrupt
   (if an instrument other than the ESA generates the interrupt, simply report
   "Instrument at GPIB Address xxx Has Generated an Interrupt").*/
printf ("\n SRQ Event Occurred!\n");
printf ("\n ... Original Device Session = %1d\n",viESA);

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

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

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

/*Return successfully*/
iSrqOccurred   =1;
viReadSTB(viESA, &iStatusByte);
return VI_SUCCESS;
}

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

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

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

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

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

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

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

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

/*Set the service request mask to assert SRQ when either a measurement
is uncalibrated (i.e. "Meas Uncal" displayed on screen) or an error
message has occurred.*/
viPrintf(viESA,"STAT:QUES:ENAB 512\n");
 viPrintf(viESA,"STAT:QUES:INT:ENAB 8\n");
viPrintf(viESA,"*ESE 35\n");
viPrintf(viESA,"*SRE 104\n");

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

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

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

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

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

/*Allow analyzer to sweep several times.*/
```

```
viPrintf(viESA,"INIT:CONT 1 \n");

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

/*Wait for SRQ*/
WaitForSRQ();

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

/* Set the service request mask to assert SRQ when either a measurement
   is completed or an error message has occurred.*/
viPrintf(viESA,"*SRE 96\n");
viPrintf(viESA,"*ESE 35\n");

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

/*Wait for SRQ*/
WaitForSRQ();

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

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

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

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

Measuring Harmonic Distortion (GPIB)

```
/*************************************************************/
/* Measuring Harmonic Distortion (GPIB)                      */
/*                                                           */
/* This example is for the E44xxB ESA Spectrum Analyzers     */
/* and E740xA EMC Analyzers.                                 */
/*                                                           */
/* This C programming example does the following.            */
/* The SCPI instrument commands used are given as            */
/* reference.                                                */
/*                                                           */
/* - Opens a GPIB session at address 18                      */
/* - Clears the Analyzer                                     */
/*      *CLS                                                 */
/* - Resets the Analyzer                                     */
/*      *RST                                                 */
/* - Set the input port to the 50 MHz reference              */
/*      CAL:SOUR:STAT ON                                     */
/* - Set the analyzer center frequency to the fundamental    */
/*      SENS:FREQ:CENT freq                                  */
/* - Set the analyzer to 10 MHz span                         */
/*      SENS:FREQ:SPAN 10 MHZ                                */
/* - Set the analyzer to single sweep mode                   */
/*      INIT:CONT 0                                          */
/* - Take a sweep and wait for sweep completion              */
/*      INIT:IMM;*WAI                                        */
/* - Perform the peak search                                 */
/*      CALC:MARK:MAX                                        */
/* - Set the marker to reference level                       */
/*      CALC:MARK:SET:RLEV                                   */
/* - Take a sweep and wait for sweep completion              */
/*      INIT:IMM;*WAI                                        */
/* - Perform the peak search                                 */
/*      CALC:MARK:MAX                                        */
/* - Change VISA timeout to 60 seconds                       */
/* - Activate signal track                                   */
/*      CALC:MARK:TRCK:STAT ON                               */
/* - Perform narrow span and wait                            */
/*      SENS:FREQ:SPAN 10e4                                  */
/* - Check for operation complete                            */
/*      *OPC?                                                */
/* - De-activate signal track                                */
/*      CALC:MARK:TRCK:STAT OFF                              */
```

```
/* - Reset VISA timeout to 3 seconds                          */
/* - Set units to dBm                                         */
/*      UNIT:POW DBM                                          */
/* - Take a sweep and wait for sweep completion               */
/*      INIT:IMM;                                             */
/*      *OPC?                                                 */
/* - Perform the peak search                                  */
/*      CALC:MARK:MAX                                         */
/* - Read the marker amplitude,this is the fundamental Level*/
/*      CALC:MARK:Y?                                          */
/* - Change the amplitude units to volts                      */
/*      UNIT:POW V                                            */
/* - Take a sweep                                             */
/*      INIT:IMM                                              */
/* - Check for operation complete                             */
/*      *OPC?                                                 */
/* - Read the marker amplitude in volts, this is the          */
/*   fundamental amplitude in volts.                          */
/*      CALC:MARK:Y?                                          */
/* - Read the marker frequency                                */
/*      CALC:MARK:X?                                          */
/* - Measure each harmonic amplitude as follows:             */
/*     Set the span to 20 MHz                                 */
/*        SENS:FREQ:SPAN 20 MHZ                               */
/*     Set the center frequency to the desired harmonic       */
/*        SENS:FREQ:CENT <freq>                               */
/*     Take a sweep and wait for operation complete           */
/*        INIT:IMM                                            */
/*        *OPC?                                               */
/*     Perform peak search                                    */
/*        CALC:MARK:MAX                                       */
/*     Set VISA timeout to 60 seconds                         */
/*     Activate signal track                                  */
/*        CALC:MARK:TRCK:STAT ON                              */
/*     Zoom down to a 100 kHz span                            */
/*        SENS:FREQ:SPAN 10E4                                 */
/*     Take a sweep and wait for operation complete           */
/*        INIT:IMM                                            */
/*        *OPC?                                               */
/*     Signal track off                                       */
/*        CALC:MARK:TRCK:STAT OFF                             */
/*     Reset VISA timeout to 3 seconds                        */
/*     Perform Peak Search                                    */
/*        CALC:MARK:MAX                                       */
/*     Set marker amplitude in volts                          */
/*        UNIT:POW V                                          */
```

```
/*     Query, read the marker amplitude in volts           */
/*        CALC:MARK:Y?                                     */
/*     Change the amplitude units to dBm and read the      */
/*     marker amplitude.                                   */
/*        UNIT:POW DBM                                     */
/* - Calculate the relative amplitude of each harmonic     */
/*   reletive to the fundamental                           */
/* - Calculate the total harmonic distortion               */
/* - Display the fundamental amplitude in dBm, fundamental */
/*   frequency in MHz, relative amplitude of each harmonic */
/*   in dBc and total harmonic distortion in percent       */
/* - Close the session                                     */
/***********************************************************/

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

#define hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#definehpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"

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

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

viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
if( iResult == 0 )
{
     /*Set the input port to the 50MHz amplitude reference for the models*/
     /*E4401B, E4411B, and E7401A*/
     viPrintf(viESA,"CAL:SOUR:STAT ON \n");
```

```
}
else
{
     /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
     /* to connect the amplitude reference output to the input*/
     printf ("Connect AMPTD REF OUT to the INPUT \n");
     printf ("......Press Return to continue \n");
     scanf( "%c",&cEnter);
     /*Externally route the 50MHz Signal*/
     viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
}

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

/*Trigger a sweep, wait for sweep completion*/
```

```
viPrintf(viESA,"INIT:IMM;*WAI\n");

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

     /*Query and read the Marker Amplitude in Volts*/
     /*Store the result in the array.*/
```

```
     viQueryf(viESA,"CALC:MARK:Y?\n","%1f", &fHarmV[lNum]);

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

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

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

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

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

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

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

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

Measuring Harmonic Distortion (RS-232)

```
/*********************************************************/
/* Measuring Harmonic Distortion (RS-232)                */
/*                                                       */
/* This example is for the E44xxB ESA Spectrum Analyzers */
/* and E740xA EMC Analyzers.                             */
/*                                                       */
/* This C programming example does the following.        */
/* The SCPI instrument commands used are given as        */
/* reference.                                            */
/*                                                       */
/* - Opens an RS-232 session to the COM1 serial port     */
/* - Clears the Analyzer                                 */
/*      *CLS                                             */
/* - Resets the Analyzer                                 */
/*      *RST                                             */
/* - Set the input port to the 50 MHz reference          */
/*      CAL:SOUR:STAT ON                                 */
/* - Set the analyzer center frequency to the fundamental*/
/*      SENS:FREQ:CENT freq                              */
/* - Set the analyzer to 10 MHz span                     */
/*      SENS:FREQ:SPAN 10 MHZ                            */
/* - Set the analyzer to single sweep mode               */
/*      INIT:CONT 0                                      */
/* - Trigger a sweep and wait for sweep completion       */
/*      INIT:IMM;*WAI                                    */
/* - Perform the peak search                             */
/*      CALC:MARK:MAX                                    */
/* - Set the marker to reference level                   */
/*      CALC:MARK:SET:RLEV                               */
/* - Trigger a sweep and wait for sweep completion       */
/*      INIT:IMM;*WAI                                    */
/* - Perform the peak search                             */
/*      CALC:MARK:MAX                                    */
/* - Change VISA timeout to 60 seconds                   */
/* - Activate signal track                               */
/*      CALC:MARK:TRCK:STAT ON                           */
/* - Perform narrow span and wait                        */
/*      SENS:FREQ:SPAN 10e4                              */
/* - Check for operation complete                        */
/*      *OPC?                                            */
/* - De-activate signal track                            */
/*      CALC:MARK:TRCK:STAT OFF                          */
```

```
/* - Reset VISA timeout to 3 seconds                          */
/* - Set units to dBm                                         */
/*      UNIT:POW DBM                                          */
/* - Take a sweep and wait for sweep completion               */
/*      INIT:IMM;                                             */
/*      *OPC?                                                 */
/* - Perform the peak search                                  */
/*      CALC:MARK:MAX                                         */
/* - Read the marker amplitude,this is the fundamental Level*/
/*      CALC:MARK:Y?                                          */
/* - Change the amplitude units to volts                      */
/*      UNIT:POW V                                            */
/* - Take a sweep                                             */
/*      INIT:IMM                                              */
/* - Check for operation complete                             */
/*      *OPC?                                                 */
/* - Read the marker amplitude in volts, this is the          */
/*   fundamental amplitude in volts.                          */
/*      CALC:MARK:Y?                                          */
/* - Read the marker frequency                                */
/*      CALC:MARK:X?                                          */
/* - Measure each harmonic amplitude as follows:             */
/*     Set the span to 20 MHz                                 */
/*        SENS:FREQ:SPAN 20 MHZ                               */
/*     Set the center frequency to the desired harmonic       */
/*        SENS:FREQ:CENT <freq>                               */
/*     Take a sweep and wait for operation complete           */
/*        INIT:IMM                                            */
/*        *OPC?                                               */
/*     Perform peak search                                    */
/*        CALC:MARK:MAX                                       */
/*     Set VISA timeout to 60 seconds                         */
/*     Activate signal track                                  */
/*        CALC:MARK:TRCK:STAT ON                              */
/*     Zoom down to a 100 kHz span                            */
/*        SENS:FREQ:SPAN 10E4                                 */
/*     Take a sweep and wait for operation complete           */
/*        INIT:IMM                                            */
/*        *OPC?                                               */
/*     Signal track off                                       */
/*        CALC:MARK:TRCK:STAT OFF                             */
/*     Reset VISA timeout to 3 seconds                        */
/*     Perform Peak Search                                    */
/*        CALC:MARK:MAX                                       */
/*     Set marker amplitude in volts                          */
/*        UNIT:POW V                                          */
```

```
/*     Query, read the marker amplitude in volts           */
/*        CALC:MARK:Y?                                     */
/*     Change the amplitude units to dBm and read the      */
/*     marker amplitude.                                   */
/*        UNIT:POW DBM                                     */
/* - Calculate the relative amplitude of each harmonic     */
/*   reletive to the fundamental                           */
/* - Calculate the total harmonic distortion               */
/* - Display the fundamental amplitude in dBm, fundamental */
/*   frequency in MHz, relative amplitude of each harmonic */
/*   in dBc and total harmonic distortion in percent       */
/* - Close the session                                     */
/*********************************************************/

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

#define hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#definehpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"

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

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

viQueryf(viESA, "*IDN?\n", "%t", &cIdBuff);
iResult = (strncmp( cIdBuff, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cIdBuff, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cIdBuff,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
if( iResult == 0 )
{
      /*Set the input port to the 50MHz amplitude reference for the models*/
      /*E4411B, E4401B*/
      viPrintf(viESA,"CAL:SOUR:STAT ON\n");
```

```
}
else
{
      /* For the analyzers having frequency limits >= 3GHz, prompt the user*/
      /* to connect the amplitude reference output to the input*/
      printf ("Connect AMPTD REF OUT to the INPUT \n");
      printf ("......Press Return to continue \n");
      scanf( "%c",&cEnter);

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

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

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


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

```
dFundamental =50.0;


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

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

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

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

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

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

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

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

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

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

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

/* Place the signal at the reference level using the
   marker-to-reference level command and take sweep */
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Performing an IS-95A ACPR Base Station Measurement

```
/*********************************************************/
/*  ACP.C   Agilent Technologies 2000                    */
/*                                                       */
/* This C programming example does the following:        */
/* Performs a IS95A ACPR Base Station Measurement        */
/* and Writes the Results to the display                 */
/*                                                       */
/* The required SCPI instrument commands are given as    */
/* reference.                                            */
/*                                                       */
/* - Opens a GPIB device at address 18                   */
/* - Clears and Resets the Analyzer to a known state     */
/*      SYST:PRES:TYPE FACT                              */
/*      *RST                                             */
/* - Identify the Instrument model                       */
/*      *IDN?                                            */
/* - Sets the analyzer center frequency and span         */
/*      SENS:FREQ:CENT freq                              */
/*      SENS:FREQ:SPAN freq                              */
/* - Sets the analyzer resolution bandwidth              */
/*      SENS:BAND rbw                                    */
/* - Select single sweep mode                            */
/*      INIT:CONT OFF                                    */
/* - Disable local display                               */
/*      DISP:ENAB OFF                                    */
/* - Select internal machine ASCII  data format          */
/*      FORM:DATA ASCII                                  */
/* - Select IS95A Standard                              */
/*SENS:RAD:STAN IS95                                    */
/* - Select Device = Base Station                       */
/*SENS:RAD:STAN:DEV BTS\n'                              */
/* - Start the ACP Measurement                          */
/*CONF:ACP                                              */
/* - Select appropriate byte order (Intel)              */
/*      FORM:BORD SWAP                                   */
/*   - Trigger a measurement and wait for completion     */
/*      INIT:*OPC?                                       */
/* - Perform the ACPR Measurement                       */
/*READ:ACP?                                             */
/* - Display measurement results                        */
/* - Close session and Return instrument to local control */
/*********************************************************/
```

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


#define CENTER 870          /* center frequency in MHz, an integer
*/
#define SPAN 2             /* span frequency in MHz, an integer                */
#define RBW 30             /* resolution BW in kHz, an integer                 */

#define DISPLAY  1          /* ESA display enable, disable for speed
*/

ViChar _VI_FAR cResult[4096] = {0};

int iNum = 408;/* Number of Characters in measurement result */
static ViChar *cToken;
int iCount=1;

int iRbw = RBW,             /* resolution bandwidth                             */
    iSpan = SPAN,           /* Analyzer Frequency Span in MHz                   */
    iCenter = CENTER;       /* Analyzer Center frequency in MHz
*/


char cCommand[100];
char cBuffer[100];



ViSession defaultRM, viESA;


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

        int iModelNumber;
        char * cpModel;

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

        iModelNumber = 0;
```

```
        if( !(strstr(cBuffer,"E44") == NULL))  {
                cpModel = strstr(cBuffer,"44");
                cpModel[4] = 0;
                iModelNumber = atoi(cpModel);
        }
        else    {
                printf("\nNo E44xx instrument found, program is exiting\n");
                exit(1);
        }
/* Set the Center Frequency */
        viPrintf(viESA, ":SENS:FREQ:CENT %i MHz\n", iCenter);

/* Set the Span */
        viPrintf(viESA,":SENS:FREQ:SPAN %i MHZ\n", iSpan);

/* Set the Resolution Bandwidth */
viPrintf(viESA, ":SENS:BAND %i KHZ\n", iRbw);

/* Input  a valid cdmaOne signal to the ESA */
        printf ("\nConnect a valid cdmaOne signal at a carrier frequency of %d MHz
\n", iCenter);
        printf (".....Press <Enter> to continue \n");
        scanf( "%c", &cBuffer);

        viPrintf(viESA, ":CONF:ACP\n");

viPrintf(viESA, ":SENS:RAD:STAN IS95\n");

viPrintf(viESA, ":SENS:RAD:STAN:DEV BTS\n");



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

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

        /* Transfer data in ASCII format             */

        viPrintf(viESA, ":FORM:DATA ASCII\n" );

        /* select the byte order; low-byte first for Intel platforms      */
        /* To further increase measurement rate,:FORM:BORD NORM could     */
```

```
        /* be used instead. The byte ordering would then need to be              */
         /* done within this program.
*/
          viPrintf(viESA, ":FORM:BORD SWAP\n");

         return;
}



/*******************************       Main      ******************************/
void  main(void)  {

    ViStatus viStatus;
long lOpc=0L;

    /* Open a GPIB session at address 18                                        */
    viStatus = viOpenDefaultRM(&defaultRM);
    viStatus = viOpen(defaultRM,"GPIB0::18",VI_NULL,VI_NULL,&viESA);

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

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

/* Setup the IS95A ACPR measurement */

    setup_acp();

/* Execute a Single Sweep          */

viPrintf(viESA,"INIT:IMM \n");
viQueryf(viESA,"*OPC?\n","%d",&lOpc);

/* Execute the ACPR Measurement a read the results */


viQueryf(viESA, "%s\n","%#t",":READ:ACP?",&iNum, cResult);

cToken = strtok(cResult,",");
```

```
/* Print the results of the ACPR Measurement       */

printf("This is element %d value is %s \n",iCount++,cToken);

while (cToken != NULL)
{
cToken = strtok(NULL,",");
printf("This is element %d value is %s \n",iCount++,cToken);
}


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

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

Performing an ACPR Adjacent Channel Power Measurement

```
/*************************************************************************/
/*  ACPR.C   Agilent Technologies 2001                                   */
/*                                                                       */
/* This C programming example does the following:                        */
/* Performs an adjacent channel power measurement                        */
/*                                                                       */
/*     Instrument Requirements:                                          */
/*        ESA with firmware version >= A.08.00                           */
/*     Note: You can select which ACPR radio standard you would like by  */
/*           changing the standard for the RADIO:STANDARD command.       */
/*           This example sets the radio standard to IS95.               */
/*                                                                       */
*************************************************************************/
/                                                                        /
/                                                                        /
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "visa.h"

void main ()
{
/*program variable*/
ViSession defaultRM, viPSA;
ViStatus viStatus    = 0;
ViChar _VI_FAR cResult[2000] = {0};
int iNum  =0;
int iSwpPnts = 401;
double freq,value;
static ViChar *cToken ;
long lCount=0L;
char sTraceInfo  [1024]= {0};
FILE *fDataFile;
```

```
 unsigned long lBytesRetrieved;
char *psaSetup =// PSA setup initialization
"*RST;*CLS;"// Reset the device and clear status
":INIT:CONT 0;"// Set analyzer to single sweep mode
":RADIO:STANDARD IS95";// Set the Radio Standard to IS95

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

/*check opening session sucess*/
if(viStatus)
{
printf("Could not open a session to GPIB device at address 18!\n");
exit(0);
}
/*Increase timeout to 20 sec*/
viSetAttribute(viPSA,VI_ATTR_TMO_VALUE,20000);

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

/*Get the center freq from user*/
printf("What is the center carrier frequency in MHz?\n");
scanf( "%lf",&freq);

/*Set the center freq*/
viPrintf(viPSA,"freq:center %lf MHZ\n",freq);

/*Perform an ACPR measurement*/
viQueryf(viPSA,"%s\n", "%#t","READ:ACP?;*wai" , &iNum , cResult);

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

/*Save data to an ASCII to a file, by removing the "," token*/
```

```
fDataFile=fopen("C:\\ACPR.txt","w");
fprintf(fDataFile,"ACPR.exe Output\nAgilent Technologies 2001\n\n");
fprintf(fDataFile,"Please read Programer's Reference for an\n");
fprintf(fDataFile,"explanation of returned results.\n\n");
while (cToken != NULL)
{
lCount++;
value = atof(cToken);
fprintf(fDataFile,"\tReturn value[%d] = %lf\n",lCount,value);
cToken =strtok(NULL,",");
  }
fprintf(fDataFile,"\nTotal number of return points of ACPR measurement :[%d]
\n\n",lCount);
fclose(fDataFile);

/*print message to the standard output*/
printf("The The ACPR Measurement Result was saved to C:\\ACPR.txt file\n\n");

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

Making Faster Measurements (multiple measurements)

```
/*********************************************************/
/*  Average.c   Agilent Technologies 1999                */
/*                                                       */
/* This C programming example does the following:        */
/* Performs Power Averaging of Multiple ESA Measurements */
/* and Writes the Result back to a Trace for display     */
/*                                                       */
/* The required SCPI instrument commands are given as    */
/* reference.                                            */
/*                                                       */
/* - Opens a GPIB device at address 18                   */
/* - Clears and Resets the Analyzer to a known state     */
/*      SYST:PRES:TYPE FACT                              */
/*      *RST                                             */
/* - Identify the Instrument model                       */
/*      *IDN?                                            */
/* - Sets the analyzer center frequency and span         */
/*      SENS:FREQ:CENT freq                              */
/*      SENS:FREQ:SPAN freq                              */
/* - Sets the analyzer resolution bandwidth              */
/*      SENS:BAND rbw                                    */
/* - Selects sampled as the detector mode                */
/*      SENS:DET SAMP                                    */
/* - Disable optional Input/Output functions             */
/*     :SYST:PORT:IFVS:ENAB OFF                         */
/* - Turn off auto-alignment                             */
/*      CAL:AUTO OFF                                     */
/* - Select the desired number of sweep points           */
/*      SWE:POINTS points                                */
/* - Select the appropriate display reference level and  */
/*     amplitude reference routing                       */
/*   E4402B/03B/04B/05B/07B/08B or E7402A/03A/04A/05A    */
/*      DISP:WIND:TRAC:Y:RLEV -20 DBM                    */
/*      CAL:SOUR:STAT ON                                 */
/*   E4401B, E4411B, or E7401A                           */
/*      DISP:WIND:TRAC:Y:RLEV -25 DBM    */
/*      CAL:SOUR:STAT ON;                                */
/* - Select single sweep mode                            */
/*      INIT:CONT OFF                                    */
/* - Disable local display                               */
/*      DISP:ENAB OFF                                    */
/* - Select internal machine binary data format (milli-dBm) */
```

```
/*      FORM:DAT INT,32                                          */
/* - Select appropriate byte order (Intel)                       */
/*      FORM:BORD SWAP                                           */
/* - Repeat the following the requested number of times:         */
/*   - Trigger a measurement and wait for completion             */
/*      INIT:*OPC?                                               */
/*   - Read the resulting measurement trace                      */
/*      TRAC:DATA? TRACE1                                        */
/* - Compute running averaged power at all trace points          */
/* - Display measurement statistics                              */
/* - Write averaged data to second trace display                 */
/*      TRAC:DATA TRACE2 <definite length block of data>         */
/* - Enable viewing of second trace                              */
/*      TRACE2:MODE VIEW                                         */
/* - Enable local display for viewing                            */
/*      DISP:ENAB ON                                             */
/* - Select continuous sweep mode                                */
/*      INIT:CONT ON                                             */
/* - Close session and Return instrument to local control        */
/*****************************************************************/

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

#define   hpESA_IDN_E4401B  "Hewlett-Packard, E4401B"
#define   hpESA_IDN_E4411B  "Hewlett-Packard, E4411B"
#define  hpEMC_IDN_E7401A  "Hewlett-Packard, E7401A"

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

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

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

```
*/

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

int iResult =0;

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

double dDelta, dTimePer, dPower;

struct timeb start_time, stop_time, elapsed_time;

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

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

ViSession defaultRM, viESA;

                          /* reserve space for the header, data and terminator    */
ViChar cInBuffer[sizeof("#nyyyyl") + (MAX_POINTS * DATA_LENGTH) ];
ViChar cOutBuffer[sizeof("TRAC:DATA TRACE2,#nyyyyl") + (MAX_POINTS * DATA_LENGTH
) ];

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

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

        return(iHeaderLength);
}

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

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

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

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

        /* Turn auto align off to maximize measurement rate                     */
        viPrintf(viESA, ":CAL:AUTO OFF\n");

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

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

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

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

       iResult = (strncmp( cBuffer, hpESA_IDN_E4401B, strlen(hpESA_IDN_E4401B)) &&
strncmp( cBuffer, hpESA_IDN_E4411B, strlen(hpESA_IDN_E4411B)) && strncmp( cBuffer,
hpEMC_IDN_E7401A, strlen(hpEMC_IDN_E7401A)));
if( iResult == 0 )
{
/*Set the input port to the 50MHz amplitude reference for the models*/
/*E4401B, E4411B and E7401A*/
```

```
viPrintf(viESA,":DISP:WIND:TRAC:Y:RLEV -25 DBM\n");
viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}
else
{
/* For the analyzers having frequency limits >= 3GHz, prompt the user*/
/* to connect the amplitude reference output to the input*/
printf ("Connect AMPTD REF OUT to the INPUT \n");
printf ("......Press Return to continue \n");
scanf( "%c",&cEnter);

/*Externally route the 50MHz Signal*/
viPrintf(viESA, ":DISP:WIND:TRAC:Y:RLEV -20 DBM\n");
viPrintf(viESA,"CAL:SOUR:STAT ON \n");
}


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

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

        /* transfer data in definite length,32 bit integer blocks. Select */
        /* machine units (milli-dBm) to maximize measurement rate         */
        viPrintf(viESA, ":FORM:DATA INT,32\n" );

        /* select the byte order; low-byte first for Intel platforms      */
        /* To further increase measurement rate,:FORM:BORD NORM could     */
        /* be used instead. The byte ordering would then need to be       */
        /* done within this program.                                      */
        viPrintf(viESA, ":FORM:BORD SWAP\n");

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

}

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

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

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

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

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

    setup();

    iTotalRetCount = lRetCount = 0;

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

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

/* trigger a new measurement and wait for complete              */
        viPrintf(viESA, ":INIT:IMM;*WAI\n");

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

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

        /* calculate a running dPower-average                                  */
        for(iLoop = 0; iLoop < NUM_POINTS; iLoop++) {
            /* running average of dPower, in milliwatts                        */
```

```
            dPower = exp( dLogTen * (iArray[iLoop]/10000.0));
            if(i > 0)   {
                dPwrAvgArray[iLoop] += ((dPower - dPwrAvgArray[iLoop])/(i+1));
            }
            else    {
                dPwrAvgArray[iLoop] = dPower;
            }
        }
    }   /* end of event loop                                                    */

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

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

        /* This is measurement time in milliseconds                             */
        dDelta = (1000.0 * elapsed_time.time) + (elapsed_time.millitm);

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

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

    /* Open a GPIB session at address 18                                        */
    viStatus = viOpenDefaultRM(&defaultRM);
    viStatus = viOpen(defaultRM,"GPIB0::18",VI_NULL,VI_NULL,&viESA);
    if(viStatus)
    {
```

```
        printf("Could not open a session to GPIB device at address 18!\n");
                exit(0);
    }

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

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

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

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

    /* build 'TRAC:DATA TRACE2,#nyyy' header and write the result to Trace 2  */
    sprintf(cCommand,":TRAC:DATA TRACE2,#%i%i", HeaderLength(iArrayLength)-2,
iArrayLength);
    write_binary_trace(cCommand, iAvgArray);

    /* enable the trace, local display and return to continuous sweep         */
    viPrintf(viESA,":TRACE2:MODE VIEW;:DISP:ENAB ON;:INIT:CONT ON\n");

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

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