

#### **NI RF Vector Signal Analyzers Help**

#### February 2007, 372058B-01

This help file contains hardware and software information for NI RF vector signal analyzers. This file contains an introduction to fundamental RF vector signal analyzer concepts, <u>guidelines</u> for accurate measurements, and programming reference information for LabVIEW, C, and LabWindows<sup>™</sup>/CVI<sup>™</sup>.

To navigate this help file, use the **Contents**, **Index**, and **Search** tabs to the left of this window.

For more information about this help file, refer to the following topics:

<u>Conventions</u>—formatting and typographical conventions in this help file

**Related Documentation** 

**Important Information** 

Technical Support and Professional Services

To comment on National Instruments documentation, refer to the <u>National</u> <u>Instruments Web site</u>.

© 2006—2007 National Instruments Corporation. All rights reserved.

### Conventions

This help file uses the following conventions:

»	The » symbol leads you through nested menu items and dialog box options to a final action. The sequence <b>File Page Setup Options</b> directs you to pull down the <b>File</b> menu, select the <b>Page Setup</b> item, and select <b>Options</b> from the last dialog box.
8	This icon denotes a tip, which alerts you to advisory information.
	This icon denotes a note, which alerts you to important information.
bold	Bold text denotes items that you must select or click on in the software, such as menu items and dialog box options. Bold text also denotes parameter names, emphasis, or an introduction to a key concept.
digitizer	This term refers to the NI 5142 digitizer hardware module.
downconverter	This term refers to the NI 5600 3-slot RF downconverter hardware module.
<u>green</u>	Underlined text in this color denotes a link to a help topic, help file, or Web address.
italic	Italic text denotes variables or cross references. This font also denotes text that is a placeholder for a word or value that you must supply.
monospace	Text in this font denotes text or characters that you should enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames, and extensions.

### **Related Documentation**

Some NI RF signal analyzer manuals also are available as PDFs. You must have Adobe Reader with Search and Accessibility 5.0.5 or later installed to view the PDFs. Refer to the <u>Adobe Systems Incorporated</u> <u>Web site</u> at www.adobe.com to download Adobe Reader. Refer to the <u>National Instruments Product Manuals Library</u> at ni.com/manuals for updated documentation resources.

The following documents contain information that you may find helpful as you use this help file:

Refer to this document for basic information on setup, configuration, and operation of the RF Signal Analyzer hardware and software.

- <u>NI RF Vector Signal Analyzers Getting Started Guide</u>, printed and included in your NI RF vector signal analyzer kit. This document is also available in PDF format at **Start»Programs»National Instruments»NI-RFSA»Documentation**.
- NI RF Signal Analyzers Readme, accessible from Start»Programs»National Instruments»NI-RFSA»Documentation. Refer to this document for system requirement information, operating system support information, installation locations, new features, and known issues.
- Device specifications, printed and included in your NI RF vector signal analyzer kit. This document is also accessible in PDF format online by searching <u>www.ni.com/manuals</u>.
- Spectral Measurements Toolkit Help, accessible from Start»Programs»National Instruments»Spectral Measurements»Documentation. Refer to this help file for information about measurements and examples available for the RF vector signal analyzer.
- Spectral Measurements Toolkit User Guide, accessible in PDF format from Start»Programs»National Instruments»Spectral Measurements»Documentation.
- Modulation Toolkit Help, accessible from Start»Programs»National Instruments»Modulation»LabVIEW Support. The Modulation Toolkit integrates with SMT and NI-RFSA for modulation/demodulation measurements and analysis.
- NI High-Speed Digitizers Help, accessible from

#### Start»Programs»National Instruments»NI-

**SCOPE**»**Documentation**. Refer to this help file for information about implementing trigging for the RF Signal Analyzer.

For the latest NI-RFSA development information and the latest NI-RFSA examples, search the <u>NI Developer Zone</u>. This link requires an internet connection. Visit <u>ni.com/manuals</u> for the most current revisions of documentation and for newly released documentation.

#### **Fundamentals**

Expand this book for more information about measurement concepts and terminology used in this help file.

#### **Guidelines for Making Accurate Measurements**

This section provides guidelines for making accurate measurements with the RF Signal Analyzer hardware and software. Click one of the following measurements for more information:

- General Amplitude/Spectrum
- Harmonic Distortion
- Two-Tone Third-Order Intermodulation Distortion
- 1 dB Gain Compression
- Noise Figure

### **General Amplitude/Spectrum Measurement**

Amplitude dynamic range is the difference between the maximum input level of a device and its minimum detectable signal level.

Dynamic range estimates the ability of the RF Signal Analyzer to distinguish and measure the amplitude difference of two signals. The RF Signal Analyzer can make signal measurements over a frequency range from 9 kHz to 2.7 GHz, and over an amplitude dynamic range of greater than 100 dB.

Signals of large amplitude can saturate the system and cause spurious effects. These spurs may be large enough to be mistaken for real signals. Avoid this effect by properly adjusting the amplitude of the incoming signal. Achieving proper signal levels may involve attenuating the signal before it gets to the first mixer, either by programming the internal attenuators or by using external attenuation.

The RF Signal Analyzer must be properly configured before making a measurement. A small signal can be buried in noise if the resolution bandwidth setting is too large. To measure a small signal, make sure that the input attenuators are switched off and lower the resolution bandwidth setting to reduce the noise content.

For signals below the noise floor of the RF Signal Analyzer, use an external low-noise amplifier (LNA) in front of the RF Signal Analyzer to raise the signal level. If the update speed is not fast enough to resolve a signal due to the processing demands imposed by a narrow resolution bandwidth, an LNA helps provided it does not significantly affect system linearity. For example, with a signal level of 100 dBm, set the resolution bandwidth to 1 kHz or less.

#### **Harmonic Distortion**

Harmonic distortion is a measure of the amount of power contained in the harmonics of a fundamental signal. Harmonic distortion is inherent to devices and systems that possess nonlinear characteristicsthe more nonlinear the device, the greater its harmonic distortion.

Harmonic distortion can be expressed as a power ratio or as a percentage ratio. Use the following formula to express it as a power ratio:

$$P_{HD} = P_{fund} - P_{harm} (dBc)$$

where  $P_{HD}$  is the power of the harmonic distortion in dBc,  $P_{fund}$  is the fundamental signal power in dB or dBm, and  $P_{harm}$  is the power of the harmonic of interest in dB or dBm.

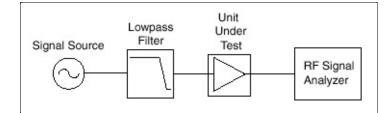
Convert the powers to voltages to express harmonic distortion as a percentage ratio:

Percentage of Distortion = 
$$\frac{V_{harm}}{V_{fund}} \times 100 \%$$

In some applications, the harmonic distortion is measured as a total percentage harmonic distortion (*THD*). This measurement involves the power summation of all the harmonics in the spectrum band, defined in the following equation:

$$THD = \frac{\sqrt{V_{h2}^2 + V_{h3}^2 + V_{h4}^2 + \dots + V_{hN}^2}}{V_{fund}} \times 100\%$$

A typical setup to perform a harmonic distortion measurement is shown in the figure below. A lowpass or bandpass filter passes the fundamental signal while suppressing its harmonics. This setup injects a very clean sinusoidal signal into the unit under test (UUT). Any harmonic content at the UUT output is assumed to be generated by the UUT instead of the source.



**Typical Harmonic Distortion Measurement Setup** 

# Understanding the RF Signal Analyzer Harmonic Distortion Limits

As with all analyzers, residual distortions are inherent in the RF Signal Analyzer. It is important that these distortions do not corrupt your measurement.

The level of internal distortion is a function of the linearity of the system, which is primarily determined at the input mixer. Increasing input power at the mixer increases distortion, so if the input signal is too high, the internally generated harmonics overwhelm the harmonics of the original signal.

The specifications for the second- and third-order harmonic intercept points provide sufficient information about the linearity of the system. For example, to measure a second-order harmonic at –70 dBc, the fundamental power at the mixer input must satisfy the following condition:

 $P_{mixer} \le IIP_2 + Distortion - 3dB = IIP_2 - 70dBc - 3dB$ 

where  $IIP_2$  is the second-order intercept point.

If the input signal power is greater than this value, the signal must be attenuated before the first mixer. There is an upper limit on the amount of attenuation you can switch in because the noise floor rises by the same amount as the attenuation. To lower the noise level decrease the resolution bandwidth, but keep in mind that there is also a practical lower limit on the resolution bandwidth. Decreasing the resolution bandwidth increases measurement time.

The harmonic distortion dynamic range (HDDR) indicates the minimum distortion an instrument can measure, which is about 96 dBc/Hz for the RF Signal Analyzer.

#### Choosing an Optimal Setting for the RF Signal Analyzer

Because the level of harmonic distortion is often unknown, the optimal attenuation level can be difficult to determine. Complete the following steps to find the proper attenuation setting for the RF Signal Analyzer:

1. Set the attenuation so that the input power at the mixer is about 30 dBm.

*mixer level = reference level – attenuation.* 

- 2. Tune to the harmonic frequency of interest and then decrease the resolution bandwidth until the harmonic spur appears.
- 3. Increase the attenuation level. If the harmonic spur decreases, attenuate more.
- 4. Repeat step 3 until the harmonic level does not decrease any further. Attenuation does not lower the harmonics of the original signal; it only lowers the internally generated harmonics.
- 5. Decrease the resolution bandwidth to lower the noise floor.

The setting you obtain is the optimal attenuation setting.

#### **Two-Tone Third-Order Intermodulation Distortion Measurement**

Two-tone third-order intermodulation distortion  $(IMD_3)$  is the measure of the third-order distortion products produced by a nonlinear device when two tones closely spaced in frequency are fed into its input. This distortion product is usually so close to the carrier that it is almost impossible to filter out and can cause interference in multichannel communications equipment.

If  $F_1$  and  $F_2$  are the frequencies of the two tones, then the third-order distortion products occur on both sides of these tones at  $2F_2 - F_1$  and  $2F_1 - F_2$ . Assuming that the power levels of the two tones are equal,  $IMD_3$  is the difference between the power of the fundamental signals and the third-order products, as defined in the following equation:

 $IMD_3 = P_o - P_{o3}$ 

where  $_{o}$  refers to the output of the UUT,  $P_{o3}$  is the power level of one of the output third-order products, and  $P_{o}$  is the power level of one of the fundamental tones.

The math becomes more involved if the powers of the two tones are different. After you measure the  $IMD_3$ , calculate the UUT output third-order intercept point ( $OIP_3$ ) using the following equation:

$$OIP_{3} = \frac{IMD_{3}}{2} + P_{o} = \frac{1}{2} (3P_{o} - P_{o3}) \text{ (dB)}$$

The input third-order intercept point  $(IIP_3)$  is defined as:

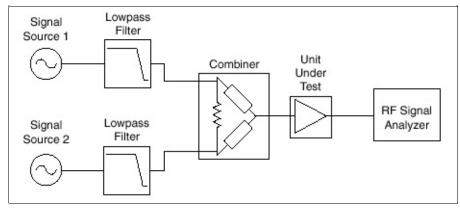
$$IIP_3 = (OIP_3 - G)$$

where *G* is the gain of the device. The  $IIP_3$  number quantifies the thirdorder linearity of a device. Use the  $IIP_3$  specification of the RF Signal Analyzer as a guide to optimize its settings when measuring the  $IMD_3$  of an external device.

The two tones injected into the UUT must be free from any third-order products. These two tones are combined, or summed, at or before the UUT input. If the two tones are not well isolated, they intermodulate with each other and cause distortion. A signal combiner with good input-to-input isolation is recommended to minimize distortion of the input tones.

#### **Measurement Setup**

A typical  $IMD_3$  measurement setup is shown in the figure below. Lowpass filters are employed at the source outputs to suppress harmonics.



Typical IMD<sub>3</sub> Measurement Setup

#### Understanding Two-Tone Third-Order Intermodulation Distortion Limits of the RF Signal Analyzer

The RF Signal Analyzer generates its own distortion spurs, which are capable of swamping the  $P_{03}$  of the UUT and giving rise to an erroneous measurement. Too much power at the signal input of the RF Signal Analyzer may drive the system into a nonlinear region of operation and produce very large distortion products. Choosing an appropriate attenuation setting for the RF Signal Analyzer minimizes its  $IMD_3$  contribution to the measurement. The  $IMD_3$  improves by 2 dB for every 1 dB of input power decrement.

To measure the  $IMD_3$  of a UUT, input power to the RF Signal Analyzer mixer must satisfy the following condition:

$$P_{mix} \leq IIP_{3rfsa} - \frac{IMD_3 + 3dB}{2}$$
 (dBm)

where  $IIP_{3rfsa}$  is the input third-order intercept point of the RF Signal Analyzer (about 10 dBm). For example, to accurately measure an  $IMD_3$  of 80 dBc the input power to the mixer must be less than 31.5 dBm.

If the powers of two-tone signals are larger than this optimal level, they must be attenuated, either with the attenuators internal to the RF Signal Analyzer or with external attenuators. However, as attenuation raises the noise floor of the RF Signal Analyzer, there is a limit to how much attenuation can be used before noise overwhelms the distortion spurs. Its spurious-free dynamic range (SFDR) specification indicates the largest  $IMD_3$  value the RF Signal Analyzer can accurately measure, assuming 0 dB attenuation and input signals whose powers satisfy the equation above.

#### Choosing an Optimal Setting for the RF Signal Analyzer

Complete the following steps to set optimal attenuation levels for an  $IMD_3$  measurement when the level of the third-order distortion spur ( $P_{03}$ ) is unknown:

 Set the attenuation so that the input power at the mixer is about 30 dBm. When using the RF Signal Analyzer Demo Panel,

*mixer level = reference level – attenuation.* 

- 2. Tune to the third-order distortion product frequency of interest, either  $2F_2 F_1$  or  $2F_1 F_2$ . Then decrease the resolution bandwidth until a distortion spur appears.
- 3. Increase the attenuation level.
- 4. If the harmonic spur decreases, repeat step 3.
- 5. Repeat step 4 until the harmonic level does not decrease any further. Attenuation does not lower the distortion products of the signal; it only lowers the distortion products generated internally to the RF Signal Analyzer. Decrease the resolution bandwidth to lower the noise floor.

The setting you obtain is the optimal attenuation setting.

#### **Noise Figure Measurement**

All devices have inherent noise. When noise is quantified, it is usually referred to the device input. In other words, all noise power a UUT inherits is assumed to come from its input. The noise figure of a UUT is the ratio in dB of its noise power to the noise power that a matched resistive load would deliver at room temperature. If you terminate a UUT input with a matched resistive load (typically 50  $\Omega$ ) and measure the noise power density at its output ( $N_0$ ), the noise figure (*NF*) is given by the following equation:

$$NF = 10\log\left(\frac{N_o/G}{kT_o}\right)$$

where G is the power gain of the UUT,  $k \approx 1.38 \times 10^{-23}$  is Boltzmanns constant, and  $T_0 \approx 290^\circ$  K is the room temperature.

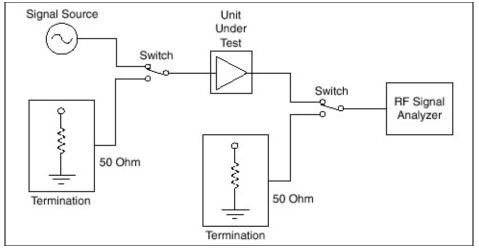
If you use the RF Signal Analyzer to measure the output noise of a UUT, the result of the measurement contains not only UUT noise but also noise intrinsic to the RF Signal Analyzer. If the UUT gain (*G*) is known, compute the noise figure of the UUT with the following equation:

$$NF = 10\log\left(\frac{N_m - N_{rfsa}}{kT_o} + 1\right) - 10\log G$$

where  $N_{rfsa}$  is the noise measured by the RF Signal Analyzer when its input is terminated with a matched resistive load and  $N_m$  is the measured noise with UUT attached. Both  $N_{rfsa}$  and  $N_m$  are given in Watts; *G* is a linear power gain.

#### **Measurement Setup**

A typical noise figure measurement setup is shown in the following figure.



**Typical Noise Figure Measurement Setup** 

#### Measuring Noise Figure with the RF Signal Analyzer

To measure the noise figure, complete the following steps:

- 1. Turn on the RF Signal Analyzer and let it warm up for 20 minutes.
- 2. Turn on the UUT if it is active.
- 3. Set the RF Signal Analyzer to the frequency of interest, and decrease the resolution bandwidth to about 1 kHz.
- 4. Terminate the RF Signal Analyzer input with a broadband resistive load.
- 5. Obtain an average reading of the noise level. Make sure to take enough readings to obtain a good average.
- 6. Convert a reading taken in dBm to watts and normalize it to 1 Hz by dividing by the resolution bandwidth. This value is the noise floor of the RF Signal Analyzer at that frequency, which is  $N_{rfsa}$  in this document.
- 7. Remove the load termination from the RF Signal Analyzer input.
- 8. Attach the output of the UUT to the RF Signal Analyzer input.
- 9. Input a known small signal into the UUT input. This signal level should be less then 10 dB below the 1 dB compression point of the UUT.
- 10. Measure the output signal level on the RF Signal Analyzer to determine the gain (G) of the UUT.
- 11. Remove the signal source and terminate the UUT input with a broadband resistive load.
- 12. Make another averaged reading of the noise with the UUT attached by repeating steps 5 and 6. This average is the noise value for the UUT and the RF Signal Analyzer ( $N_{\rm m}$ ).
- 13. Substitute your values into the equation:

$$NF = 10\log\left(\frac{N_m - N_{rfsa}}{kT_o} + 1\right) - 10\log G$$

as follows to determine the UUT noise figure:

- The value from step 6 is  $N_{rfsa}$
- The value from step 10 is G
- The value from step 12 is  $N_{\rm m}$

### NI 5661 RF Vector Signal Analyzer

This section includes information useful about the NI 5661 hardware, including module <u>front panels</u>, <u>theory of operation</u>, <u>signal paths</u>, <u>block</u> <u>diagrams</u>, and <u>calibration</u> information. The NI 5661 is a modular RF vector signal analyzer consisting of two PXI hardware modules:

- NI 5142 14 bit, 100 megasample-per-second (MS/s) IF digitizer module with onboard signal processing (OSP)
- NI 5600 wideband RF downconverter module with input frequencies between 9 kHz and 2.7 GHz.
- Note There is no physical device labeled the "NI PXI-5661." The NI 5661 RF Vector Signal Analyzer is the instrument comprised of the two hardware modules (NI 5600 and NI 5142) and the software included in the kit. Refer to the <u>NI RF Vector Signal Analyzers</u> <u>Getting Started Guide</u> for more information about installing and configuring your hardware.

The NI 5661 has the following characteristics and features:

- 9 kHz to 2.7 GHz frequency range
- 20 MHz real-time bandwidth
- 10 MHz oven-controlled crystal oscillator (OCXO) timebase
  - ±20 ppb frequency stability
  - ±50 ppb frequency accuracy
- >80 dB spurious-free dynamic range
- +30 dBm full signal input range
- Up to 64 MS of onboard waveform memory
- Software for performing frequency-domain and IQ measurements
- Four slots wide PXI/3U Compact PCI form factor

The NI 5661 follows industry-standard Plug and Play specifications for the PXI bus and seamlessly integrates with compliant systems.



**Note** Refer to the *NI PXI-5661 Specifications* document included in the RF Signal Analyzer kit for complete hardware specifications.

### NI 5661 Theory of Operation

The NI 5600 downconverter module performs two primary functions: frequency shifting, or downconversion, and input signal conditioning. Frequency shifting is performed using a tunable oscillator in the superheterodyne signal chain.

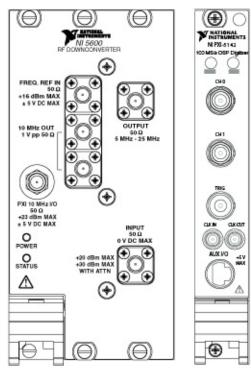
Input signal conditioning is accomplished using three stages of mixer conversion and two sets of gain <u>attenuators</u> whose levels are programmable. The first set of attenuators can be set to minimize distortion and other spurious signals when input levels are high and to minimize noise when input levels are low. Attenuator levels are set using the <u>Attenuation</u> property or the <u>NIRFSA\_ATTR\_ATTENUATION</u> attribute.

The second set of <u>attenuators</u> is in the second intermediate frequency (IF) path before the third mixer and ensures an appropriate output signal level even if the first mixer is intentionally driven into compression. These attenuators are also set when performing linearity measurements. For more information on proper attenuation levels, refer to <u>Guidelines for Making Accurate Measurements</u>.

#### Hardware Front Panel Connectors and Indicators

These sections describe the connectors and LED indicators on the front panels of the NI 5661 hardware modules. All inputs and outputs are AC-coupled.

**Click a Module Front Panel for Description** 



# NI 5600 Front Panel

This section describes the connectors and LED indicators on the hardware front panel of the NI 5600 RF downconverter module. All inputs and outputs are AC-coupled.

Connector	Use
FREQ REF IN	Routes an external frequency reference signal to which the NI 5600 can lock. This signal can be propagated to the PXI backplane when the NI 5600 is installed in PXI Slot 2.
10 MHz OUT 10 MHz OUT	Connect the lower 10 MHz OUT connector to the CLK IN connector on the NI 5122 module front panel. Both connectors output replications of the downconverter 10 MHz frequency reference signal, useful for driving other devices. Each replication is 180 degrees out-of- phase with the other. The signal output at these connectors is always on and cannot be disabled.
PXI 10 MHz I/O	Bidirectional connection to the PXI 10 MHz backplane clock. This connector can be used to drive the PXI 10 MHz backplane clock <i>only</i> when the NI 5600 downconverter module is installed in PXI Slot 2. To drive the PXI backplane with the NI 5600 onboard frequency reference, connect the PXI 10 MHz I/O connector to the 10 MHz OUT connector on the NI 5600 RF downconverter module front panel as shown in <i>NI RF</i> <i>Vector Signal Analyzers Getting Started Guide</i> . Refer to the <u>niRFSA Configure Ref Clock</u> VI or the <u>niRFSA_ConfigureRefClock</u> function for more information. This connector can be used to export the PXI 10 MHz backplane clock when the NI 5600 downconverter is installed in any PXI slot.
OUTPUT	Connect to the INPUT connector on the NI 5142 digitizer module front panel.

Outputs the frequency-translated IF signal for digitization.
Connect to the analog RF input signal to be measured by the RF Vector Signal Analyzer.

The following table provides LED and indications information for the NI 5600 RF downconverter module front panel LEDs:

LED	Indications
POWER	Indicates the basic hardware power status of the NI 5600 downconverter module. This LED functions identically to the ACCESS LED on the digitizer module front panel.
	<ul> <li>OFFThe module is not yet functional, or has detected a problem with a PXI power rail.</li> </ul>
	<ul> <li>GREENThe module is functional and receiving power.</li> </ul>
STATUS	Indicates the status of the NI 5600 downconverter module PLLs.
	<ul> <li>OFFThe module is in an uninitialized state, or the module PLLs are attempting to lock.</li> </ul>
	<ul> <li>GREENThe module is in a ready state; applicable PLLs are locked.</li> </ul>

Note Refer to the NI 5661 RF Vector Signal Analyzer Specifications document included in the NI 5661 RF Signal Analyzer kit for more information on NI 5600 connectors.

# NI 5142 Front Panel

This section describes the connectors on the hardware front panel of the NI 5142 IF digitizer module. All inputs and outputs are AC-coupled.

Connector	Function	
CH 0, CH 1	Analog input connection; digitizes data and triggers acquisitions	
	Note For NI 5661 operation, NI recommends that you use CH 0 to connect the NI 5142 to the NI 5600 downconverter.	
TRIG	External analog trigger connection.	
CLK IN	Imports an external reference or sample clock to the digitizer	
CLK OUT	Exports the digitizer reference or sample clock	
AUX I/O	Provides access to the external digital trigger lines, PFI 0 and PFI 1 (with optional cable)	

Note Refer to the *NI PXI/PCI-5142 Specifications* for more information on NI 5142 connectors.

### NI 5661 Signal Paths

A signal takes the following path from the RF Signal Analyzer <u>front panel</u> to the PXI controller:

- 1. A signal enters the RF Signal Analyzer through the INPUT front panel connector of the NI 5600 RF downconverter module.
- The NI 5600 RF downconverter module "zooms in" on a 20 MHz block of spectrum and frequency-translates it to center around 15 MHz. The translated IF signal is sent to the NI 5600 downconverter module OUTPUT connector.
- 3. The IF signal is passed from the NI 5600 RF downconverter module front panel OUTPUT connector to the NI 5142 IF digitizer module front panel INPUT connector.
- 4. The NI 5142 IF digitizer module filters and conditions the signal and applies gain and dither.
- 5. The A/D Converter (ADC) converts the signal from analog to digital.
- 6. The data is sent to onboard memory (the buffer).
- 7. The data is transferred to the host computer.

### **RF Attenuation and Signal Levels**

The RF downconverter module has five programmable attenuators: three RF attenuators at the beginning of its signal chain and two IF attenuators near the end of the signal chain. They are set up in the following sequence.

Attenuator Sequence		
Attenuator	Asserted Value	
RF Attenuator 1	20 dB	
RF Attenuator 2	20 dB	
RF Attenuator 3	10 dB	
IF Attenuator 1	20 dB	
IF Attenuator 2	10 dB	

Attenuators are either set/asserted or not set/asserted. This architecture allows a dynamic range of RF attenuation from 0–50 dB and a dynamic range of IF attenuation from 0–30 dB. Overall attenuation within the signal chain is the sum of all the attenuators set, for a range of 0–80 dB.

#### **Reference Level and Mixer Level**

Understanding the parts played by the reference level and the mixer level is key to setting the individual attenuators. NI-RFSA sets the RF attenuators first.

Let *D* be the difference between the reference level and the mixer level, such that

D = referenceLevel - mixerLevel

This formula correlates directly with the range of possible RF attenuator settings. Recall that the RF attenuators have a range of 0 - 50 dB. The coercions described above ensure that

mixerLevel < referenceLevel

and that

*referenceLevel* - *mixerLevel* ≤ 50 dBm.

*D* is then directly proportional to the total RF attenuation, a value between 0-50 dB.

The attenuators are set as follows (refer to the <u>Attenuator Sequence</u> <u>table</u> above).

	<b>RF</b> Attenuator Settings		
D	Attenuators		
0	None are set		
10	RF Attenuator 3		
20	RF Attenuator 1		
30	RF Attenuators 1 and 3		
40	RF Attenuators 1 and 2		
50	RF Attenuators 1, 2, and 3		

#### Attenuation in NI-RFSA

You modify attenuation using the <u>Attenuation</u> property or the <u>NIRFSA\_ATTR\_ATTENUATION</u> attribute.

#### Hardware Block Diagrams

This section provides hardware block diagrams for the <u>NI 5142 digitizer</u> and the <u>NI 5600 RF downconverter</u>. These two modules are interconnected to comprise the NI 5661 RF signal analyzer.

#### NI 5600 RF Downconverter Module Block Diagram

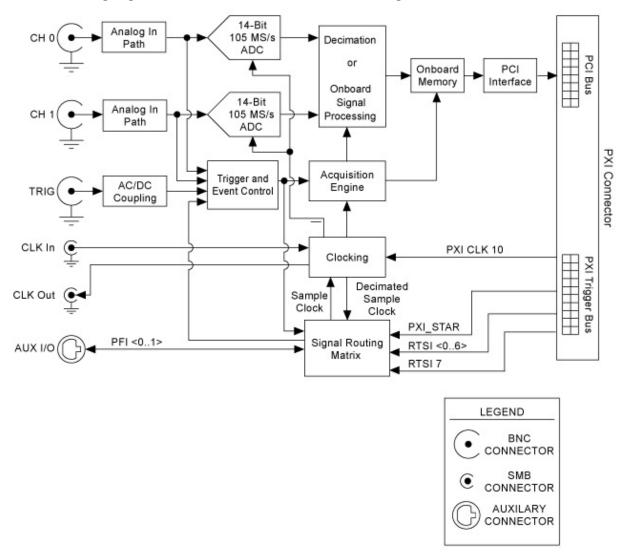
The NI 5600 RF downconverter module translates any 20 MHz-wide band of incoming signal to center at 15 MHz. Thus the downconverter module converts any block of spectrum, up to 20 MHz wide and centered anywhere between 9 kHz and 2.7 GHz, to an IF band between 5– 25 MHz. The NI 5600 hardware always downconverts a 20 MHz band. This IF band is then passed to the NI 5142 digitizer module for further processing.

BPF 20 MHz BW BPF Mixer Atten Mixer I PF Mixer Amp Atten IF Out BE In R) R) R) т Phase Phase Phase Locked LO2 Locked Locked LO1 LO3 External 10 MHz Out 10 MHz In Buffe to Digitizer Internal Clock Reference Calibration Buffe ►10 MHz Out DAC PXI Clock IN/OUT Power, Control, and Clock Routing PXI

The following figure shows the NI 5600 block diagram.

#### NI PXI-5142 IF Digitizer Block Diagram

The following figure shows a detailed block diagram of the NI PXI-5142.



# Calibration

Every NI 5661 RF Vector Signal Analyzer is individually calibrated for accurate frequency response at the factory and ships with a calibration certificate verifying NIST-traceable accuracy levels.

During frequency-response calibration, the RF Signal Analyzer is used to measure a NIST-certified high-precision signal. Any error in the returned data is quantified as a set of calibration constants. These calibration constants are used by the software to calculate and apply correction to your analysis based upon the spectrum of interest. For more information on applying calibration correction, refer to the example programs installed with the RF Signal Analyzer.

To preserve specified accuracy and NIST traceability, NI recommends returning both modules of the RF Signal Analyzer to the factory for annual recalibration. The RF downconverter module and the IF digitizer module are calibrated independently of one another, not as a combined system. For more information on calibration, contact NI or visit <u>ni.com/calibration</u>.

### **Programming Reference**

This section provides reference and programming information regarding the NI-RFSA API and its supported ADEs.

Refer to <u>Fundamentals</u> for an introduction to basic concepts of the NI-RFSA VIs. Refer to the <u>LabVIEW Reference</u> for more information about using specific LabVIEW VIs. Refer to <u>Related Documentation</u> for more information about advanced use of the NI-RFSA VIs, and setup of the RF vector signal analyzer hardware and software.

### **Getting Started with NI-RFSA**

To successfully build your application, install NI-RFSA. You also must install one of the following ADEs:

- LabVIEW
- LabWindows/CVI
- Any C compiler capable of calling a 32-bit DLL

# Using NI-RFSA in LabVIEW

This topic assumes that you are using the National Instruments LabVIEW ADE to manage your code development and that you are familiar with the ADE.

To develop an NI-RFSA application in LabVIEW, follow these general steps:

- 1. Open an existing or new LabVIEW VI.
- 2. From the Function Palette, locate the NI-RFSA VIs at **Instrument I/O»NI-RFSA** .
- 3. Select the VIs that you want to use and drop them on the block diagram to build your application.

#### **Example Programs for LabVIEW**

You can use the NI Example Finder to search or browse examples. NI-RFSA examples are classified by keyword, so you can search for a particular device or measurement function.

To browse the NI-RFSA examples available in LabVIEW, launch LabVIEW, click **Open»Examples**, and navigate to **Hardware Input and Output»Modular Instruments»NI-RFSA**.

The <u>NI RF Signal Analyzers Readme</u> includes the default installation location of the NI-RFSA LabVIEW examples.

## Using NI-RFSA in LabWindows/CVI

This topic assumes that you are using the LabWindows™/CVI™ ADE to manage your code development and that you are familiar with the ADE.

To develop an NI-RFSA application in LabWindows/CVI, follow these general steps:

- 1. Open an existing or new project file.
- 2. Load the NI-RFSA function panel at IVI\Drivers\niRFSA\nirfsa.fp.
- 3. Use the function panel to navigate the function hierarchy and generate function calls with the proper syntax and variable values.

#### Example Programs for LabWindows/CVI

You can use the NI Example Finder to search or browse examples. NI-RFSA examples are classified by keyword, so you can search for a particular device or measurement function.

To browse the NI-RFSA examples available in LabWindows/CVI, launch LabWindows/CVI, select **Help»NI Example Finder**, and navigate to **Hardware Input and Output»Modular Instruments»NI-RFSA**.

You can find example programs installed at the location specified in the <u>NI RF Signal Analyzer Readme</u>.

# Available Add-On Software for Measurement and Analysis

The ni5660 VIs perform data acquisition using the RF Signal Analyzer. NI provides add-on software toolkits, such as the Spectral Measurements Toolkit (included) and the Modulation Toolkit, which extend the capability of the RF Signal Analyzer to include frequency- and modulation-domain measurements and analysis of analog- and digitally-modulated IF signals.



**Note** To enable proper operation of the RF Signal Analyzer, you must install the *Spectral Measurements Toolkit* CD after installing the *NI-RFSA* CD.

#### **Spectral Measurements Toolkit**

Use the Spectral Measurements Toolkit (SMT) VIs and functions for frequency-domain analysis, measurement, and display of data acquired using the RF Signal Analyzer. The SMT can perform several operations, including the following:

- Zoom FFT processing and spectrum averaging
- Spectral measurements such as band power, adjacent channel power, and peak frequency and magnitude Spectrogram display and analysis
- RF Vector Signal Analyzer hardware configuration for frequencydomain measurements

Refer to the Spectral Measurements Toolkit documentation, accessible from **Start»Programs»National Instruments»Spectral Measurements**, for complete information about SMT VIs and functions.

#### **Modulation Toolkit**

The Modulation Toolkit integrates with SMT and NI-RFSA for modulation/demodulation measurements and analysis. The Modulation Toolkit VIs and functions generate and analyze analog and digital modulated IF signals in FSK, MSK, PSK, QAM, AM, FM, and PM formats. The Modulation Toolkit is capable of measuring several aspects of signals generated by a unit under test, including the following:

- Modulation quality and modulation index
- Signal impairments, bit error rate, and phase noise
- Carrier frequency drift and complementary cumulative distribution functions (CCDF) values

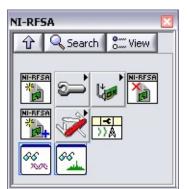
Refer to the Modulation Toolkit documentation, accessible from **Start»Programs»National Instruments»Modulation**, for complete information about Modulation Toolkit VIs and functions.

## LabVIEW Reference

This section describes the VIs and properties included with NI-RFSA that you can use to configure and operate your NI RF vector signal analyzer.

## **VI Reference**

Use the VIs on the NI-RFSA palette to build the block diagram. **Click the icons for VI and function descriptions.** 

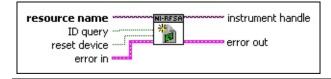


## niRFSA Initialize

Creates a new session for the device. This VI performs the following initialization actions:

- Creates a new instrument driver session to the RF signal analyzer, using the downconverter resource name you specify.
- Sends initialization commands to reset both hardware modules to a known state necessary for NI-RFSA operation.

Note Before initializing the NI 5661, an NI 5142 IF digitizer module must be associated with the NI 5600 downconverter module in MAX. After association, pass the NI 5600 device name to this VI to initialize both modules. To change the digitizer association, modify the NI 5600 Properties page in MAX, or use the Initialize With Options VI to override the association in MAX. Refer to the NI RF Vector Signal Analyzers Getting Started Guide, installed at Start»Programs»National Instruments»NI-RFSA»Documentation for information on MAX association.



**resource name** specifies the resource name of the device to initialize.

Example #	Device Type	Syntax
1	5	NI-DAQmx device, device name = "myDAQmxDevice"
2		IVI logical name or virtual instrument, name = "myLogicalName"

For NI-DAQmx devices, the syntax is the device name specified in MAX, as shown in Example 1. Typical default names for NI-DAQmx devices in MAX are Dev1 or PXI1Slot1. You can rename an NI-DAQmx device by right-clicking on the name in MAX and entering a new name. You can also pass in the name of an IVI logical name configured with the IVI Configuration utility. For additional information, refer to the IVI topic in the *Measurement & Automation Explorer Help*.



**Caution** NI-DAQmx device names are not case-sensitive. However, all IVI names, such as logical names, are casesensitive. If you use an IVI logical name, make sure the

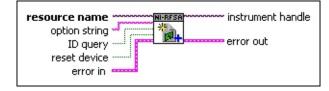
## niRFSA Initialize With Options

Creates a new session for the device. This VI performs the following initialization actions:

- Creates a new instrument driver session to the RF signal analyzer, using the downconverter resource name you specify.
- Sends initialization commands to reset both hardware modules to a known state necessary for NI-RFSA operation.

Note Before initializing the NI 5661, an NI 5142 IF digitizer module must be associated with the NI 5600 downconverter module in MAX. After association, pass the NI 5600 device name to this VI to initialize both modules. To change the digitizer association, modify the NI 5600 Properties page in MAX, or use this VI to override the association in MAX. Refer to the *NI RF Vector Signal Analyzers Getting Started Guide*, installed at Start»Programs»National Instruments»NI-





## **resource name** specifies the resource name of the device to initialize.

Example #	Device Type	Syntax
1		NI-DAQmx device, device name = "myDAQmxDevice"
2	myLogicalName	IVI logical name or virtual instrument, name = "myLogicalName"

For NI-DAQmx devices, the syntax is the device name specified in MAX, as shown in Example 1. Typical default names for NI-DAQmx devices in MAX are Dev1 or PXI1Slot1. You can rename an NI-DAQmx device by right-clicking on the name in MAX and entering a new name. You can also pass in the name of an IVI logical name configured with the IVI Configuration utility. For additional information, refer to the IVI topic in the *Measurement & Automation Explorer Help*.

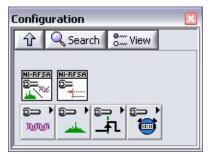


**Caution** NI-DAQmx device names are not case-sensitive. However, all IVI names, such as logical names, are case-

## **Configuration Subpalette**

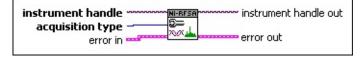
Use the VIs located on the **NI-RFSA**»**Configuration** palette to configure operations with your RF vector signal analyzer.

#### Click the icons for VI and function descriptions.



## niRFSA Configure Acquisition Type

Configures whether the session acquires IQ data or computes a power spectrum over the specified frequency range.



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- **acquisition type** configures the type of acquisition.

Configures the driver for IQ acquisitions.

Spectrum Configures the driver for spectrum acquisitions.

- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **instrument handle out** passes a reference to your instrument session to the next VI. **instrument handle** is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- error out contains error information. If error in indicates that an error occurred before this VI ran, error out contains the same error information. Otherwise, it describes the error status that this VI produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.



IQ

a sum a share will be the eviction of the sum of a supervision and is in

#### niRFSA Configure Reference Level

Configures the reference level. The reference level represents the maximum expected power of an input RF signal.



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- **reference level** specifies the expected total integrated power of the RF input signal in dBm.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- error out contains error information. If error in indicates that an error occurred before this VI ran, error out contains the same error information. Otherwise, it describes the error status that this VI produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - code the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or

## IQ Subpalette

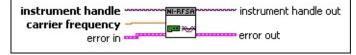
Use the VIs located on the **NI-RFSA**»**Configuration**»**IQ** palette to configure the RF vector signal analyzer for an IQ acquisition.

Click the icons for VI and function descriptions.

IQ	
🔒 🔍 Search	S View

#### niRFSA Configure IQ Carrier Frequency

Configures the IQ carrier frequency of the RF input signal.



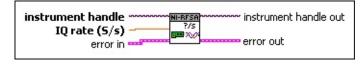
- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- **Carrier frequency** specifies the carrier frequency of the acquired RF signal. NI-RFSA sets the <u>IQ Carrier Frequency</u> property to this value. Refer to the specifications document that shipped with your device for allowable frequency settings.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **instrument handle out** passes a reference to your instrument session to the next VI. **instrument handle** is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- error out contains error information. If error in indicates that an error occurred before this VI ran, error out contains the same error information. Otherwise, it describes the error status that this VI produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in

## niRFSA Configure IQ Rate

Configures the rate at which the device samples IQ values. Bandwidth is equal to the coerced **IQ rate** times 0.8.



**Note** You should not need to configure an IQ rate higher than 25 MHz, since the NI PXI-5600 downconverter bandwidth is 20 MHz. If you choose to configure a higher IQ rate, you may see aliasing effects at negative frequencies because the IF frequency of the downconverter is at 15 MHz.

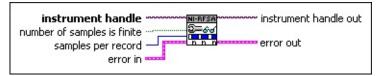


- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- **IQ rate** specifies the IQ rate for the acquisition. The value is expressed in S/s.
- error in (no error) describes error conditions that occur before this VI runs.
  - **status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **instrument handle out** passes a reference to your instrument session to the next VI. **instrument handle** is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize</u> With Options VIs and identifies a particular instrument session.
- error out contains error information. If error in indicates that an error occurred before this VI ran, error out contains the same error information. Otherwise, it describes the error status that this VI produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.

## niRFSA Configure Number of Samples

Configures the number of samples in a finite acquisition or configures the device to continuously acquire samples. If you configure the device for finite acquisition, it acquires the specified number of samples and stops the acquisition. You can configure the device to acquire multiple records using the <u>niRFSA Configure Number of Records</u> VI, each record containing the number of samples specified in this VI. The default number of records to acquire is 1.

If the device is configured to continuously acquire samples, it continues acquiring data until you call <u>niRFSA Abort</u> to abort the acquisition. The device stores data in onboard memory in a circular fashion. Once the device fills the memory, it starts overwriting previously acquired data from the beginning of the memory buffer. Retrieve the samples using the <u>niRFSA Fetch IQ</u> VI as they are being acquired to avoid data being overwritten before you can retrieve it.

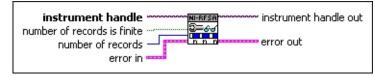


- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- **number of samples is finite** specifies whether to configure the device to acquire a finite number of samples or to acquire samples continuously.
- **samples per record** specifies the number of samples per record if number of samples is finite is set to TRUE.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.

## niRFSA Configure Number of Records

Configures the number of records in a finite acquisition or configures the device to continuously acquire records. Notice that you can only configure the device to acquire multiple records if **number of samples is finite** is set to TRUE.

If you configure the device to acquire records continuously, it continues acquiring records until you call <u>niRFSA Abort</u> to abort the acquisition. The device stores records in onboard memory in a circular fashion. Once the device fills the memory, it starts overwriting previously acquired records from the beginning of the memory buffer. Fetch the records as they are being acquired to avoid data being overwritten before you can retrieve it.



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- **number of records is finite** set to TRUE to configure the device to stop after acquiring the specified number of records. Set to FALSE to acquire records continuously until you <u>abort</u> the acquisition.
- **number of records** specifies the number of records to acquire if **number of records is finite** is set to TRUE.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **instrument handle out** passes a reference to your instrument session to the next VI. **instrument handle** is obtained from the

## **Spectrum Subpalette**

Use the VIs located on the **NI-RFSA**»**Configuration**»**Spectrum** palette to configure the RF vector signal analyzer for a spectrum acquisition.

Click the icons for VI and function descriptions.

Spectrum	$\mathbf{X}$
🕆 🔍 Search	S View

#### niRFSA Configure Spectrum Frequency

Configures the hardware for a spectrum frequency acquisition.



Note If you configure the spectrum span ( **stop frequency** - **start frequency**) to a value larger than 20 MHz, RFSA performs multiple acquisitions and combines them into a spectrum of the size you requested.

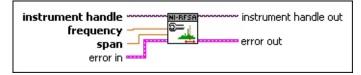
•

Use the pull-down menu to select an instance of this VI.

Select an instance

#### niRFSA Configure Spectrum Frequency Center Span

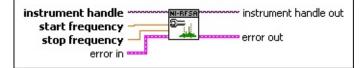
Configures the span and center frequency of a spectrum acquisition. An acquisition consists of a span of data surrounding the center frequency.



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- **frequency** specifies the center frequency in a spectrum acquisition. The value is expressed in Hertz.
- **span** specifies the span of a spectrum acquisition. The value is expressed in Hertz.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- error out contains error information. If error in indicates that an error occurred before this VI ran, error out contains the same error information. Otherwise, it describes the error status that this VI produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code

#### niRFSA Configure Spectrum Frequency Start Stop

Configures the start and stop frequency of a spectrum acquisition.

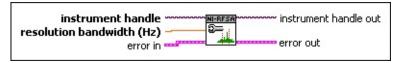


- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- **start frequency** specifies the lower band of a span of frequencies.
- **stop frequency** specifies the upper band of a span of frequencies.
- error in (no error) describes error conditions that occur before this VI runs.
  - **status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **instrument handle out** passes a reference to your instrument session to the next VI. **instrument handle** is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- error out contains error information. If error in indicates that an error occurred before this VI ran, error out contains the same error information. Otherwise, it describes the error status that this VI produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in

## niRFSA Configure Resolution Bandwidth

Configures the resolution bandwidth of a spectrum acquisition. The resolution bandwidth controls the width of the frequency bins in the power spectrum computed by NI-RFSA. A larger value for resolution bandwidth means the frequency bins are wider, and hence you get fewer bins or spectral lines.

By default, the resolution bandwidth value corresponds to the 3 dB bandwidth of the window type NI-RFSA uses to compute the spectrum. To specify the frequency bin width directly, change the resolution bandwidth type attribute to bin width. Refer to the <u>Resolution Bandwidth</u> <u>Type</u> property for more information.



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- resolution bandwidth (Hz) specifies the resolution bandwidth of a spectrum acquisition. The value is expressed in hertz. Configure the type of resolution bandwidth with the Resolution Bandwidth Type property.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **instrument handle out** passes a reference to your instrument session to the next VI. **instrument handle** is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- error out contains error information. If error in indicates that an

## **Trigger Subpalette**

Use the VIs located on the **NI-RFSA**»**Configuration**»**Trigger** palette to configure the triggers for an RF vector signal analyzer acquisition.

Click the icons for VI and function descriptions.

Trigger	×
🗘 🔍 Search ا ី 🗰 View	

## niRFSA Configure Trigger

Configures the Start, Reference, and Advance triggers. Use the pull-down menu to select an instance of this VI.

•

Select an instance

#### niRFSA Disable Ref Trigger

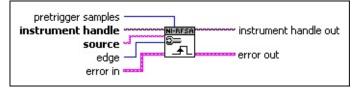
Configures the device to not wait for a Reference trigger to mark a reference point within a record.



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **instrument handle out** passes a reference to your instrument session to the next VI. **instrument handle** is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- error out contains error information. If error in indicates that an error occurred before this VI ran, error out contains the same error information. Otherwise, it describes the error status that this VI produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.

#### niRFSA Configure Digital Edge Ref Trigger

Configures the device to wait for a digital edge Reference trigger to mark a reference point within the record.



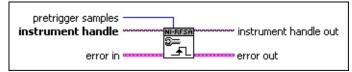
- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- **source** specifies the source of the digital edge for the Reference trigger.

<u>unggen</u>	
PFI0	The trigger is received on PFI 0.
PFI1	The trigger is received on PFI 1.
PXI_Trig0	The trigger is received on PXI trigger line 0.
PXI_Trig1	The trigger is received on PXI trigger line 1.
PXI_Trig2	The trigger is received on PXI trigger line 2.
PXI_Trig3	The trigger is received on PXI trigger line 3.
PXI_Trig4	The trigger is received on PXI trigger line 4.
PXI_Trig5	The trigger is received on PXI trigger line 5.
PXI_Trig6	The trigger is received on PXI trigger line 6.
PXI_Trig7	The trigger is received on PXI trigger line 7.
PXI_STAR	The trigger is received on the PXI star trigger line.

- edge specifies the edge to detect. You can choose **Rising Edge** or **Falling Edge**.
- **pretrigger samples** specifies the number of samples to store for each record that were acquired in the time period immediately before the trigger occurred.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE,

#### niRFSA Configure Software Edge Ref Trigger

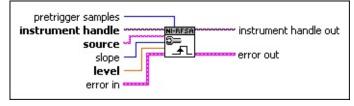
Configures the device to wait for a software Reference trigger to mark a reference point within the record. The device will wait until you call the <u>niRFSA Send Software Edge Trigger</u> VI to assert the trigger.



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- **pretrigger samples** specifies the number of samples to store for each record that were acquired in the time period immediately before the trigger occurred.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **instrument handle out** passes a reference to your instrument session to the next VI. **instrument handle** is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- error out contains error information. If error in indicates that an error occurred before this VI ran, error out contains the same error information. Otherwise, it describes the error status that this VI produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.

#### niRFSA Configure IQ Power Edge Ref Trigger

Configures the device to wait for the complex power of the IQ data to cross the specified threshold to mark a reference point within the record.



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- **source** specifies the source of the RF signal for the power edge Reference trigger. The only supported value is "0."
- **slope** specifies whether the device will detect a rising or falling edge on the trigger signal.
- **Ievel** specifies the threshold above or below which the device will trigger.
- **pretrigger samples** specifies the number of samples to store for each record that were acquired in the time period immediately before the trigger occurred.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **instrument handle out** passes a reference to your instrument session to the next VI. **instrument handle** is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- error out contains error information. If error in indicates that an error occurred before this VI ran, error out contains the same error

#### niRFSA Disable Start Trigger

Configures the device to not wait for a Start trigger at the beginning of the acquisition.



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **instrument handle out** passes a reference to your instrument session to the next VI. **instrument handle** is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- error out contains error information. If error in indicates that an error occurred before this VI ran, error out contains the same error information. Otherwise, it describes the error status that this VI produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.

#### niRFSA Configure Digital Edge Start Trigger

Configures the device to wait for a digital edge Start trigger at the beginning of the acquisition.

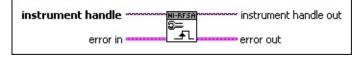


- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- **source** specifies the source of the digital edge for the Start trigger.

- edge specifies the edge to detect. You can choose **Rising Edge** or **Falling Edge**.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.

#### niRFSA Configure Software Edge Start Trigger

Configures the device to wait for a software Start trigger at the beginning of the acquisition. The device will wait until you call the <u>niRFSA Send Software Edge Trigger</u> VI to assert the trigger.



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **instrument handle out** passes a reference to your instrument session to the next VI. **instrument handle** is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- error out contains error information. If error in indicates that an error occurred before this VI ran, error out contains the same error information. Otherwise, it describes the error status that this VI produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.

#### niRFSA Configure Digital Edge Advance Trigger

Configures the device to wait for a digital edge Advance trigger between records.



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- **source** specifies the source of the digital edge for the Advance trigger.

PFI0	The trigger is received on PFI 0.
PFI1	The trigger is received on PFI 1.
PXI_Trig0	The trigger is received on PXI trigger line 0.
PXI_Trig1	The trigger is received on PXI trigger line 1.
PXI_Trig2	The trigger is received on PXI trigger line 2.
PXI_Trig3	The trigger is received on PXI trigger line 3.
PXI_Trig4	The trigger is received on PXI trigger line 4.
PXI_Trig5	The trigger is received on PXI trigger line 5.
PXI_Trig6	The trigger is received on PXI trigger line 6.
PXI_Trig7	The trigger is received on PXI trigger line 7.
PXI_STAR	The trigger is received on the PXI star trigger line.

- edge specifies the edge to detect. You can choose **Rising Edge** or **Falling Edge**.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.

#### niRFSA Configure Software Edge Advance Trigger

Configures the device to wait for a software Advance trigger between records. The device waits until you call the <u>niRFSA Send Software Edge</u> <u>Trigger</u> VI to assert the trigger.



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **instrument handle out** passes a reference to your instrument session to the next VI. **instrument handle** is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- error out contains error information. If error in indicates that an error occurred before this VI ran, error out contains the same error information. Otherwise, it describes the error status that this VI produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.

#### niRFSA Disable Advance Trigger

Configures the device to not wait for an Advance trigger between records of a multirecord acquisition.



- error in (no error) describes error conditions that occur before this VI runs.
  - **status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- **instrument handle out** passes a reference to your instrument session to the next VI. **instrument handle** is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- error out contains error information. If error in indicates that an error occurred before this VI ran, error out contains the same error information. Otherwise, it describes the error status that this VI produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.

#### niRFSA Send Software Edge Trigger

Sends a trigger to the device when you use the <u>niRFSA Configure</u> <u>Trigger</u> VI to choose a software version of the trigger and the device is waiting for the trigger to be sent. This VI also can be used to override a hardware trigger.

This VI returns an error in the following situations:

- You configure an invalid trigger
- You are in spectrum mode
- You have not previously called the <u>niRFSA Initiate</u> VI.

NI-Scope handles other errors.

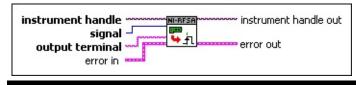
instrument handle " NI-RESA instrument handle out trigger 🚽 Ē₽ error out error in 🚥

- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- **trigger** specifies the software signal to send. You can send a Start, Reference, Advance, or Arm Reference trigger.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- error out contains error information. If error in indicates that an error occurred before this VI ran, error out contains the same error information. Otherwise, it describes the error status that this VI produces

## niRFSA Export Signal

Routes signals (triggers, clocks, and events) to the specified output terminal.

**Details** 



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- **signal** specifies the type of signal to route. You can choose to export the Start, Reference, and Advance triggers and the Ready for Start, Ready for Advance, Ready for Ref, End of Record, and Done events.
- **output terminal** specifies the terminal where the signal will be exported. You can choose not to export any signal.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **instrument handle out** passes a reference to your instrument session to the next VI. **instrument handle** is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- error out contains error information. If error in indicates that an error occurred before this VI ran, error out contains the same error information. Otherwise, it describes the error status that this VI produces.
  - **status** is TRUE (X) if an error occurred or FALSE

#### Details

If you export a signal with this VI and <u>commit</u> the session, the signal is routed to the output terminal you specify. If you then reconfigure the signal to have a different output terminal, the previous output terminal is tristated when the session is next committed. If you change the **output terminal** to **Do Not Export** and <u>commit</u>, the previous output terminal is tristated.

Any signals, except for PXI trigger lines, that are exported within a session persist after the session closes to prevent signal glitches between sessions. PXI trigger lines are always set to tristate when a session is closed. If you wish to have the terminal that the signal was exported to tristated when the session closes, first change the **output terminal** for the exported signal to **Do Not Export** and <u>commit</u> the session again before closing it.

You can also tristate all PFI lines by setting the **reset device** parameter in the <u>niRFSA Initialize</u> VI or by using the <u>niRFSA Reset</u> VI.

## **Clock Subpalette**

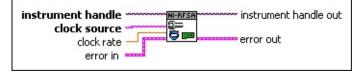
Use the VIs located on the **NI-RFSA**»**Configuration**»**Clock** palette to configure the clock signals for an RF vector signal analyzer acquisition.

#### Click the icons for VI and function descriptions.

Clock	×
🔒 🔍 Search	S View
NI-RFSA S= S= S= S= S= S= S= S= S= S= S= S= S=	

## niRFSA Configure Ref Clock

Configures the NI-RFSA device reference clock.



instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.

**source** specifies the reference clock source.

OnboardClock	Lock the NI-RFSA device to the NI PXI-5600 onboard clock.
RefIn	Lock the NI-RFSA device to the external REF IN connector on the NI PXI-5600.
PXI_Clk10	Lock the NI-RFSA device to the PXI backplane clock using the NI PXI-5600. You must connect the PXI 10 MHz connector to the REF IN connector on the NI PXI-5600 front panel to use this option.

- **Clock rate** specifies the reference clock rate, expressed in Hertz. The default value is 10 MHz, which is the only currently supported value.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **instrument handle out** passes a reference to your instrument session to the next VI. **instrument handle** is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.

### niRFSA Configure PXI Chassis Clk10

Specifies the signal to drive the 10 MHz reference clock on the PXI backplane. This option can only be configured when the NI PXI-5600 is in Slot 2 of the PXI chassis.



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- **PXI Clk10 source** specifies the signal to drive the 10 MHz reference clock on the PXI backplane. This option can only be configured when the NI PXI-5600 is in Slot 2 of the PXI chassis.

None	The device does not drive the PXI 10 MHz backplane reference clock.
OnboardClock	The device drives the PXI 10 MHz backplane reference clock with the NI PXI-5600 onboard clock. You must connect the 10 MHz OUT connector to the PXI 10 MHz I/O on the NI PXI 5600 front panel to use this option.
Refin	The device drives the PXI 10 MHz backplane reference clock with the reference source attached to the NI PXI-5600 REF IN connector. You must connect the 10 MHz OUT connector to the PXI 10 MHz I/O on the NI PXI 5600 front panel to use this option.

- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.

# **Acquisition Subpalette**

Use the VIs located on the **NI-RFSA**»Acquisition palette to control acquisition operations with your RF vector signal analyzer.

Click the icons for VI and function descriptions.

Acquis	sition	×
Ŷ	Search	
NI-RFS		

## niRFSA Read Power Spectrum (Cluster)

Initiates a spectrum acquisition and returns power spectrum data.



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- **timeout** specifies in seconds the time allotted for the function to complete before returning a timeout error. A value of -1 specifies the VI waits until all data is available.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **instrument handle out** passes a reference to your instrument session to the next VI. **instrument handle** is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- **power spectrum** returns power spectrum data.
  - **f0** returns the start frequency of the spectrum, expressed in Hertz.
  - **df** returns the frequency interval between data points in the spectrum, expressed in Hertz
  - **basis** data returns the acquired data as a cluster.
    - **t0** returns the trigger (start) time of the acquired signal.
    - **dt** returns the time interval between data points in the acquired signal. The IQ data sampling rate is the reciprocal of this value.

## niRFSA Read IQ

Initiates an acquisition and fetches a single IQ data record. Do not use this VI if you have configured the device to continuously acquire data samples or to acquire multiple records.

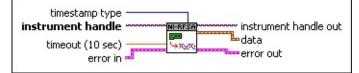
Use the pull-down menu to select an instance of this VI.

Select an instance

**\_** 

#### niRFSA Read IQ (Complex WDT 1Rec 1Chan)

Returns the IQ data as a complex waveform data type (WDT).



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- **timeout** specifies in seconds the time allotted for the function to complete before returning a timeout error. A value of -1 specifies the VI waits until all data is available.
- **timestamp type** specifies the time format of the **data**.

	When converted to a DBL value, the timestamp corresponds to the difference in seconds between the first sample returned and the Reference trigger location.
Absolute	The timestamp corresponds to the date and time of the

Absolute The timestamp corresponds to the date and time of the acquisition of the first sample returned.

- error in (no error) describes error conditions that occur before this VI runs.
  - **status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **instrument handle out** passes a reference to your instrument session to the next VI. **instrument handle** is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- data returns the baseband (downconverted) time-domain data for demodulation.
- error out contains error information. If error in indicates that an

#### niRFSA Read IQ (Complex Cluster 1Rec 1Chan)

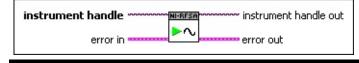
Returns the IQ data as a complex cluster.



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- **timeout** specifies in seconds the time allotted for the function to complete before returning a timeout error. A value of -1 specifies the VI waits until all data is available.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **instrument handle out** passes a reference to your instrument session to the next VI. **instrument handle** is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- **basis** data returns the acquired waveform.
  - **t0** returns the trigger (start) time of the acquired signal. The timestamp corresponds to the difference in seconds between the first sample returned and the Reference trigger location.
  - **dt** returns the time interval between data points in the acquired signal. The IQ data sampling rate is the reciprocal of this value.
  - Y returns the complex-valued time domain data array. The real and imaginary parts of this complex data array correspond to the in-phase (I) and quadrature-phase (Q)

### niRFSA Initiate

Starts an IQ acquisition. You may use this function in conjunction with the <u>niRFSA Fetch IQ</u> VI to retrieve acquired IQ data, or use the <u>niRFSA</u> <u>Read IQ</u> VI to both initiate the acquisition and retrieve IQ data at one time.



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- error out contains error information. If error in indicates that an error occurred before this VI ran, error out contains the same error information. Otherwise, it describes the error status that this VI produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning

## niRFSA Fetch IQ

Transfers acquired waveform data from device memory to PC memory. The data was acquired to onboard memory previously by the hardware after it was initiated. If the number of samples specified in **samples to read** is not available after the time duration specified in **timeout**, this VI returns no data with a timeout error.

This VI is not necessary if you use the <u>niRFSA Read IQ</u> VI, as the fetch is performed as part of that function.

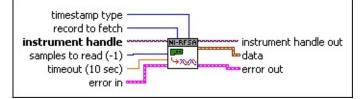
•

Use the pull-down menu to select an instance of this VI.

Select an instance

#### niRFSA Fetch IQ (Complex WDT 1Rec 1Chan)

Fetches IQ data from a single record in an acquisition.



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- samples to read specifies the number of samples to fetch. A value of -1 specifies that NI-RFSA fetch all samples.
- timeout specifies in seconds the time allotted for the function to complete before returning a timeout error. A value of -1 specifies the VI waits until all data is available. A value of 0 specifies the VI returns available data immediately.

**timestamp type** specifies the time format of the **data**.

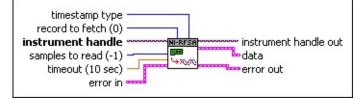
	When converted to a DBL value, the timestamp corresponds to the difference in seconds between the first sample returned and the Reference trigger location.
Absolute	The timestamp corresponds to the date and time of the acquisition of the first sample returned.

- **record to fetch** specifies the record to retrieve. Record numbers are zero-indexed.
- error in (no error) describes error conditions that occur before this VI runs.
  - **status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.

**instrument handle out** passes a reference to your instrument

#### niRFSA Fetch IQ (Complex Cluster 1Rec 1Chan)

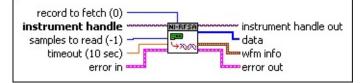
Fetches IQ data from a single record in an acquisition.



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- samples to read specifies the number of samples to fetch. A value of -1 specifies that NI-RFSA fetch all samples.
- **timeout** specifies in seconds the time allotted for the function to complete before returning a timeout error. A value of -1 specifies the VI waits until all data is available. A value of 0 specifies the VI returns available data immediately.
- **record to fetch** specifies the record to retrieve. Record numbers are zero-indexed.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **instrument handle out** passes a reference to your instrument session to the next VI. **instrument handle** is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- **beam seturns** the acquired data as a cluster.
  - **t0** returns the trigger (start) time of the acquired signal. The timestamp corresponds to the difference in seconds between the first sample returned and the Reference trigger location.

#### niRFSA Fetch IQ (1D I16)

Fetches binary IQ data from a single record in an acquisition.

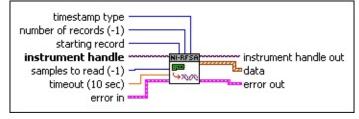


- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- **samples to read** specifies the number of samples to fetch. A value of -1 specifies that NI-RFSA fetch all samples.
- timeout specifies in seconds the time allotted for the function to complete before returning a timeout error. A value of -1 specifies the VI waits until all data is available. A value of 0 specifies the VI returns available data immediately.
- **record to fetch** specifies the record to retrieve. Record numbers are zero-indexed.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- **data** returns the acquired complex waveform. The array is composed of interleaved I and Q samples, where the order of the array is as shown:

Array[0] = 10

#### niRFSA Fetch IQ (1D Complex WDT NRec 1Chan)

Fetches IQ data from a single record in an acquisition.



- instrument handle identifies your instrument session. instrument handle is obtained from the niRFSA Initialize or the niRFSA Initialize With Options VIs and identifies a particular instrument session.
- **samples to read** specifies the number of samples to fetch. A value of -1 specifies that NI-RFSA fetch all samples.
- **timeout** specifies in seconds the time allotted for the function to complete before returning a timeout error. A value of -1 specifies the VI waits until all data is available. A value of 0 specifies the VI returns available data immediately.
- timestamp type specifies the time format of the data.

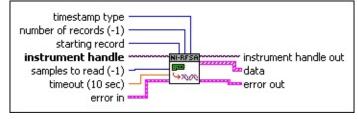
When converted to a DBL value, the timestamp corresponds to the difference in seconds between the first sample returned and the Reference trigger location.
The timestamp corresponds to the date and time of the acquisition of the first sample returned.

- **number of records** specifies the number of records to fetch. A value of -1 specifies that NI-RFSA fetches all records in an acquisition starting with the record specified by **starting record**. Record numbers are zero-indexed.
- **starting record** specifies the first record to retrieve.
- error in (no error) describes error conditions that occur before this VI runs.
  - **status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.

able source describes the origin of the error or warning and is in

#### niRFSA Fetch IQ (1D Complex Cluster NRec 1Chan)

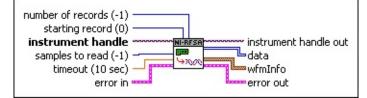
Fetches IQ data from multiple records in an acquisition.



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- **samples to read** specifies the number of samples to fetch. A value of -1 specifies that NI-RFSA fetch all samples.
- timeout specifies in seconds the time allotted for the function to complete before returning a timeout error. A value of -1 specifies the VI waits until all data is available. A value of 0 specifies the VI returns available data immediately.
- **number of records** specifies the number of records to fetch. A value of -1 specifies that NI-RFSA fetches all records in an acquisition starting with the record specified by **starting record**. Record numbers are zero-indexed.
- **starting record** specifies the first record to retrieve.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **instrument handle out** passes a reference to your instrument session to the next VI. **instrument handle** is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- ter lister with the acquired data as a cluster

#### niRFSA Fetch IQ (2D I16)

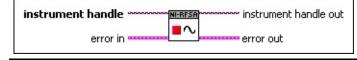
Fetches binary IQ data from multiple records in an acquisition.



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- samples to read specifies the number of samples to fetch. A value of -1 specifies that NI-RFSA fetch all samples.
- **timeout** specifies in seconds the time allotted for the function to complete before returning a timeout error. A value of -1 specifies the VI waits until all data is available. A value of 0 specifies the VI returns available data immediately.
- **number of records** specifies the number of records to fetch. A value of -1 specifies that NI-RFSA fetches all records in an acquisition starting with the record specified by **starting record**. Record numbers are zero-indexed.
- **starting record** specifies the first record to retrieve.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **instrument handle out** passes a reference to your instrument session to the next VI. **instrument handle** is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- **data** returns the acquired complex waveform per record. Each

### niRFSA Abort

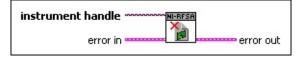
Stops an acquisition previously started with the <u>niRFSA Initiate</u> VI. Unless you want to stop an acquisition before it is complete or you are continuously acquiring data, calling this VI is optional.



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- error out contains error information. If error in indicates that an error occurred before this VI ran, error out contains the same error information. Otherwise, it describes the error status that this VI produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.

### niRFSA Close

Closes the session to the device.

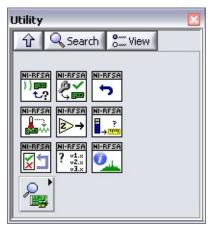


- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- error out contains error information. If error in indicates that an error occurred before this VI ran, error out contains the same error information. Otherwise, it describes the error status that this VI produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.

## **Utility Subpalette**

Use the VIs located on the **NI-RFSA**»**Utility** palette to access utility features of NI-RFSA.

Click the icons for VI and function descriptions.



#### niRFSA Check Acquisition Status

Checks the status of the acquisition. Use this VI to check for any errors that may occur during signal acquisition or to check whether the device has completed the acquisition operation.



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **instrument handle out** passes a reference to your instrument session to the next VI. **instrument handle** is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- **Description IDE** when signal acquisition is complete.
- error out contains error information. If error in indicates that an error occurred before this VI ran, error out contains the same error information. Otherwise, it describes the error status that this VI produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - code the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or

### niRFSA Commit

Commits settings to hardware. Calling this VI is optional. Settings are automatically committed to hardware when you call the <u>niRFSA Initiate</u>, <u>niRFSA Read IQ</u>, or <u>niRFSA Read Power Spectrum (Cluster)</u> VI.



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- instrument handle out passes a reference to your instrument session to the next VI. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- error out contains error information. If error in indicates that an error occurred before this VI ran, error out contains the same error information. Otherwise, it describes the error status that this VI produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.

### niRFSA Reset

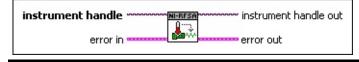
Resets the device to a known initialization state.



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- error in (no error) describes error conditions that occur before this VI runs.
  - **status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **instrument handle out** passes a reference to your instrument session to the next VI. **instrument handle** is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- error out contains error information. If error in indicates that an error occurred before this VI ran, error out contains the same error information. Otherwise, it describes the error status that this VI produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.

### niRFSA Perform Thermal Correction

Measurements are affected by changes in temperature. NI-RFSA internally acquires the temperature every time you initiate an acquisition. If you are performing a very long continuous acquisition, National Instruments recommends calling this VI once every 10 minutes in a stable temperature environment to periodically update temperature calibration.



- **instrument handle** identifies your instrument session. **instrument** handle is obtained from the niRFSA Initialize or the niRFSA Initialize With Options VIs and identifies a particular instrument session.
- error in (no error) describes error conditions that occur before this VI runs.
  - **status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **instrument handle out** passes a reference to your instrument session to the next VI. instrument handle is obtained from the niRFSA Initialize or the niRFSA Initialize With Options VIs and identifies a particular instrument session.
- **error out** contains error information. If **error in** indicates that an error occurred before this VI ran, error out contains the same error information. Otherwise, it describes the error status that this VI produces.
  - **Status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **EXAMPLE** STATE IN THE INTERIOR INTERIOR IN THE INTERIOR INTERIORI INTERIOR non-zero error code. If status is FALSE, code is 0 or a warning code.

  - **We course** describes the origin of the error or werping and is in

#### niRFSA Get IQ Components

Separates a complex IQ array into an I array and a corresponding Q array. This utility can be used to conveniently graph I or Q or to perform operations that apply to one or the other component.

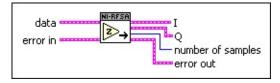
Use the pull-down menu to select an instance of this VI.

Select an instance

**\_** 

#### niRFSA Get IQ Components (Complex Cluster)

Returns the I and Q data as a complex cluster.

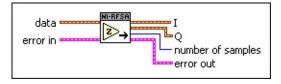


**data** specifies the complex waveform to split into I and Q components.

- **t0** specifies the trigger (start) time of the acquired signal.
- **dt** specifies the time interval between data points in the acquired signal. The IQ data sampling rate is the reciprocal of this value.
- Y specifies the complex-valued time domain data array. The real and imaginary parts of this complex data array correspond to the in-phase (I) and quadrature-phase (Q) data, respectively. To calculate the instantaneous power of a sampled IQ point, use the equation (I <sup>2</sup> + Q <sup>2</sup>)/2R, where R is the input impedance in ohms. For NI RF signal analyzers, R = 50 ohms.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **I** returns the I component of the **data**.
- **Q** returns the Q component of the **data**.
- **number of samples** returns the number of samples in the input waveform.
- error out contains error information. If error in indicates that an error occurred before this VI ran, error out contains the same error information. Otherwise, it describes the error status that this VI produces.

#### niRFSA Get IQ Components (Complex WDT)

Returns the I and Q data as a waveform data type.



- data specifies the complex waveform to split into I and Q components.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **I** returns the I component of the **data**.
- **Q** returns the Q component of the **data**.
- **number of samples** returns the number of samples in the input waveform.
- error out contains error information. If error in indicates that an error occurred before this VI ran, error out contains the same error information. Otherwise, it describes the error status that this VI produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.

### niRFSA Get Fetch Backlog

Returns the number of points acquired that have not been fetched yet..



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- **record number** specifies the record from which to read the backlog
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- error out contains error information. If error in indicates that an error occurred before this VI ran, error out contains the same error information. Otherwise, it describes the error status that this VI produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** the error or warning code. If status is TRUE, code is a non-zero error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning.
- **backlog** returns the number of samples available to read for the requested record.
- **instrument handle out** passes a reference to your instrument

### niRFSA Self Test

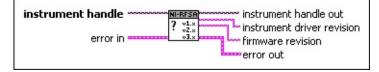
Performs a self-test on the NI-RFSA device and returns the test result. This VI performs a simple series of tests ensuring the NI-RFSA device is powered up and responding.



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **instrument handle out** passes a reference to your instrument session to the next VI. **instrument handle** is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- **self test result** returns the value from the device self-test. Zero means success.
- **self test message** returns the self-test response string from the NI-RFSA device.
- error out contains error information. If error in indicates that an error occurred before this VI ran, error out contains the same error information. Otherwise, it describes the error status that this VI produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** the error or warning code. If status is TRUE, code is a

### niRFSA Revision Query

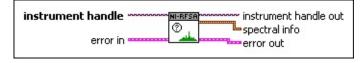
Returns the revision numbers of the NI-RFSA driver.



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **instrument handle out** passes a reference to your instrument session to the next VI. **instrument handle** is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize</u> With Options VIs and identifies a particular instrument session.
- **instrument driver revision** returns the instrument driver software revision numbers in the form of a string. The value of the Specific Driver Revision property is returned.
- **Firmware revision** returns the instrument firmware revision numbers in the form of a string. The value of the Instrument Firmware Revision property is returned.
- error out contains error information. If error in indicates that an error occurred before this VI ran, error out contains the same error information. Otherwise, it describes the error status that this VI produces.
  - **status** is TRUE (X) if an error occurred or FALSE (checkmark) to indicate a warning or that no error occurred.
  - **code** the error or warning code. If status is TRUE, code is a

### niRFSA Get Spectral Info for SMT

Returns a cluster containing information about the power spectrum NI-RFSA computes that is needed by the Spectral Measurements Toolkit (SMT).



- instrument handle identifies your instrument session. instrument handle is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA</u> <u>Initialize With Options</u> VIs and identifies a particular instrument session.
- error in (no error) describes error conditions that occur before this VI runs.
  - **Status** is TRUE (X) if an error occurred before this VI ran or FALSE (checkmark) to indicate a warning or that no error occurred before this VI ran. The default is FALSE.
  - **code** is the error or warning code. The default is 0. If status is TRUE, code is a negative error code. If status is FALSE, code is 0 or a warning code.
  - **source** describes the origin of the error or warning and is, in most cases, the name of the VI that produced the error or warning. The default is an empty string.
- **instrument handle out** passes a reference to your instrument session to the next VI. **instrument handle** is obtained from the <u>niRFSA Initialize</u> or the <u>niRFSA Initialize With Options</u> VIs and identifies a particular instrument session.
- **spectral info** returns properties of the computed spectrum such as spectrum type, spectrum scale (linear or dB), the window type used by the VI to compute the spectrum, window size, and FFT size. Connect this parameter to subsequent VIs that contain the **spectral info** parameter. Do not modify the values.
  - 💷 spectrum type
  - linear/dB specifies whether the spectrum scale is linear or in dB.
  - window specifies the time-domain window the VI uses.
  - window size
  - **FFT** size

## niRFSA Property Node

The niRFSA Property Node is used to set or get properties.

Some NI-RFSA properties are channel based. When a property is channel based, you must specify an active channel before setting or getting properties.

## NI-RFSA Express (IQ)

Configures and acquires data from National Instruments RF signal analyzers using NI-RFSA in IQ mode.

**Dialog Box Options** 

**Block Diagram Inputs** 

**Block Diagram Outputs** 

#### **Dialog Box Options**

Parameter	Description
	<ul> <li>Description</li> <li>Contains the following options:</li> <li>Device—Specifies the RF signal analyzer used. This ring control lists all NI RF signal analyzers installed on this computer that can be used by this Express VI. If you relaunch the VI and the selected RF signal analyzer is dimmed, it is no longer available.</li> <li>Carrier frequency (Hz)—Specifies the carrier frequency of the acquired RF signal.</li> <li>Reference level (dBm)—Specifies the maximum expected power of the RF signal to be acquired.</li> <li>Attenuation—Specifies whether the attenuation is automatically determined or user-specified.</li> <li>Attenuation (dB)—Specifies the downconverter's overall attenuation of the input RF signal.</li> <li>IQ rate (Samples/s)—Specifies the IQ rate of the waveform.</li> <li>Acquire—Specifies whether to acquire a finite number of samples or to acquire samples continuously until the acquisition is stopped.</li> <li>IQ samples—For a finite acquisition, specifies the number of samples to be acquired. For a</li> </ul>
	continuous acquisition, specifies the number of samples to be fetched every time the data is read.
Trigger	<ul> <li>Contains the following options:</li> <li>Reference trigger type—Specifies the reference trigger type for the acquisition.</li> <li>Pretrigger samples—Specifies the number of pretrigger samples the device must receive before the Reference trigger is acknowledged.</li> <li>Of the total number of samples acquired, the number of samples configured as Pretrigger</li> </ul>

	<ul> <li>samples are acquired immediately prior to the trigger. The remaining samples are acquired immediately after the trigger.</li> <li>Reference trigger source—Specifies the trigger source. The RF signal analyzer waits for the specified trigger to start the acquisition.</li> <li>Edge—Specifies whether to trigger on a rising or falling edge of the trigger signal.</li> <li>Level (dBm)—Specifies the trigger threshold for the IQ power edge trigger.</li> <li>Minimum quiet time (s)—Configures the minimum time the signal must be above or below the trigger level before the IQ Power Edge trigger is armed. If Slope is set to Rising, the signal must be below the trigger level for the specified time. If Slope is set to Falling, the signal must be above the trigger level for the specified time. If Slope is not the trigger level for the specified time. If Slope is not the trigger level for the specified time. Set this control when triggering on burst signals to avoid triggering in the middle of a burst.</li> <li>Max time (s)—Specifies how long to allow for the acquisition to complete before reporting a timeout error.</li> </ul>
Advanced	<ul> <li>Contains the following options:</li> <li>Reference clock source—Specifies the source of the reference clock signal. Only certain combinations of Reference clock source and PXI Chassis Clk10 source are valid.</li> <li>PXI chassis Clk10 source—Specifies the signal driven to the 10 MHz reference clock on the PXI backplane. Only certain combinations of Reference clock source and PXI Chassis Clk10 source are valid.</li> <li>Use relative initial time—Specifies if the timestamp value of the waveform is absolute or relative to the trigger point.</li> <li>Digitizer sample clock source—Specifies the</li> </ul>

	digitizer sample clock source. Select one of the following options:
	<ul> <li>Onboard clock—Uses the onboard sample clock of the digitizer.</li> </ul>
	<ul> <li>External—Coerce IQ rate—Uses an external sample clock. Select this option if your external clock has fixed rates. The IQ rate is coerced based upon the rate of the external clock.</li> </ul>
	<ul> <li>External—Coerce external clock—Uses an external sample clock. Select this option if your external clock has a flexible rate. The clock rate is calculated from the IQ rate and the external clock should be set accordingly.</li> </ul>
	<ul> <li>External clock rate (Hz)—This parameter is defined as one of the following:</li> </ul>
	<ul> <li>When Digitizer sample clock source is set to External—Coerce IQ Rate, specifies the external clock rate. The IQ rate is coerced based upon this value.</li> </ul>
	<ul> <li>When Digitizer sample clock source is set to External—Coerce external clock, displays the external clock rate, which is calculated from the IQ rate. The external clock should be set to this value.</li> </ul>
	<ul> <li>Output data type—Specifies the data type of the acquired waveform.</li> </ul>
Graph view	Specifies how the acquired waveform is displayed on the graph. Select from the following options: <ul> <li>I vs. Time</li> </ul>
	<ul><li> Q vs. Time</li><li> Power vs. Time</li></ul>
Autoscale graph	Specifies whether to autoscale the Y scale of the graph. Checked—The Y scale of the graph is autoscaled every time the graph is updated. Unchecked—The Y scale of the graph remains

unchanged when the graph is updated.	
--------------------------------------	--

# **Block Diagram Inputs**

Parameter	Description
close	Determines whether the instrument session remains open when the VI finishes execution. Use this parameter for loop optimization by setting it to FALSE on all iterations except the last. The default is TRUE.
	Note: This input is not intended for sharing the session between Express VIs. If a loop contains multiple Express VIs that use the same device, you must wire in TRUE for this input.
max time	Specifies the timeout value for the Express VI.
error in	Describes error conditions that occur before this Express VI runs.

# Block Diagram Outputs

Parameter	Description
data	Contains the data acquired by the device.
	Contains error information. If <b>error in</b> indicates that an error occurred before this Express VI runs, <b>error out</b> contains the same error information. Otherwise, it describes the error status that this Express VI produces.

# **NI-RFSA Express (Spectrum)**

Configures and acquires data from National Instruments RF signal analyzers using NI-RFSA in Spectrum mode.

Dialog Box Options Block Diagram Inputs Block Diagram Outputs

## **Dialog Box Options**

Deverseter	
Parameter	Description
Configuration	Contains the following options:
	• <b>Device</b> —Specifies the RF signal analyzer used. This ring control lists all NI RF signal analyzers installed on this computer that can be used by this Express VI. If you relaunch the VI and the selected RF signal analyzer is dimmed, it is no longer available.
	• <b>Resolution bandwidth (Hz)</b> —Specifies the resolution bandwidth of the spectrum. Resolution bandwidth controls the width of the frequency bins in the power spectrum computed by NI-RFSA. A larger value means the frequency bins are wider, thus you get fewer bins or spectral lines.
	Resolution bandwidth is calculated as the 3 dB bandwidth of the window frequency response.
	<ul> <li>Units—Specifies the units of the acquired spectrum.</li> </ul>
	<ul> <li>Reference level (dBm)—Specifies the maximum expected power of the RF signal to be acquired.</li> </ul>
	<ul> <li>Attenuation—Specifies whether the attenuation is automatically determined or user-specified.</li> </ul>
	<ul> <li>Attenuation (dB)—Specifies the downconverter's overall attenuation of the input RF signal.</li> </ul>
	• <b>Specification method</b> —Specifies whether the acquisition frequencies are defined in terms of a start and stop frequency or a center frequency and span.
	<ul> <li>Start frequency (Hz)—Specifies the start frequency of the acquired RF signal. The acquisition consists of a span of data from the start frequency to the stop frequency.</li> </ul>
	<ul> <li>Stop frequency (Hz)—Specifies the stop</li> </ul>

	<ul> <li>frequency of the acquired RF signal. The acquisition consists of a span of data from the start frequency to the stop frequency.</li> <li>Center frequency (Hz)—Specifies the center frequency of the acquired RF signal. The acquisition consists of a span of data surrounding the center frequency.</li> <li>Span (Hz)—Specifies the span of the acquired RF signal. The acquisition consists of a span of data surrounding the center frequency.</li> </ul>
Timing	<ul> <li>Contains the following options:</li> <li>Reference clock source—Specifies the source of the reference clock signal.</li> <li>PXI Chassis Clk10 source—Specifies the signal driven to the 10 MHz reference clock on the PXI backplane. Only certain combinations of Reference clock source and PXI Chassis Clk10 source are valid.</li> </ul>
Advanced Spectrum	<ul> <li>Contains the following options:</li> <li>Averaging mode—Specifies the averaging mode. Select from RMS, Vector, Peak Hold, or None.</li> <li>Number of averages—Specifies the number of acquisitions to average.</li> <li>Window type—Specifies the FFT window type.</li> </ul>
Autoscale graph	Specifies whether to autoscale the Y scale of the graph. Checked—The Y scale of the graph is autoscaled every time the graph is updated. Unchecked—The Y scale of the graph remains unchanged when the graph is updated.

## **Block Diagram Inputs**

Parameter	Description
close	Determines whether the instrument session remains open when the VI finishes execution. Use this parameter for loop optimization by setting it to FALSE on all iterations except the last. The default is TRUE.
	Note: This input is not intended for sharing the session between Express VIs. If a loop contains multiple Express VIs that use the same device, you must wire in TRUE for this input.
error in	Describes error conditions that occur before this Express VI runs.

# Block Diagram Outputs

Parameter	Description
data	Contains the data acquired by the device.
spectral info	Contains properties of the acquired spectrum. This output is used by VIs in the Spectral Measurements Toolkit.
error out	Contains error information. If <b>error in</b> indicates that an error occurred before this Express VI runs, <b>error out</b> contains the same error information. Otherwise, it describes the error status that this Express VI produces.

# Acquisition Type

#### Short Name: AcquisitionType

Configures whether the session acquires IQ data or computes a power spectrum over the specified frequency range.

<b>IQ</b> (100)	Configures the driver for IQ acquisitions.
Spectrum (101)	Configures the driver for spectrum acquisitions.

Data Type	Vilnt32
Permissions	R/W
High-Level VI	niRFSA Configure Acquisition Type

# **Vertical:Reference Level (dBm)**

#### Short Name: ReferenceLevel

Specifies the reference level. The value is expressed in dBm. The reference level represents the maximum expected power of an input RF signal. Reference level, attenuation, and mixer level are related by the following relationship:

attenuation = reference level – mixer level mixer level < reference level reference level – mixer level <= 50 dB

Data Type	ViReal64
Permissions	R/W
High-Level VI	niRFSA Configure Reference Level

# Vertical:Attenuation (dB)

#### Short Name: Attenuation

Specifies the downconverter module attenuation setting in dB.

Calculate the attenuation setting using desired <u>Reference Level</u> and <u>Mixer Level</u> settings, according to the following formula:

attenuation = reference level – mixer level

For example, when using a reference level of 0 dBm (default) with moderate distortion and low noise, specify an attenuation value of 20 dB, as shown by the following calculation:

```
attenuation = (0 dB reference level) – (–20 dB mixer level)
```

Data Type	ViReal64
Permissions	R/W
High-Level VI	None

# Vertical:Mixer Level (dBm)

#### Short Name: Mixer Level

Specifies the mixer level. The value is expressed in dBm. The mixer level represents the attenuation to apply to the input RF signal as it reaches the first mixer in the signal chain. NI-RFSA automatically selects an optimal mixer level value given the reference level if you do not configure this property. Reference level, attenuation, and mixer level are related by the following relationship:

attenuation = reference level – mixer level mixer level < reference level reference level – mixer level <= 50 dB

The following table shows the relationship between mixer level and noise and distortion.

Mixer Level	Noise and Distortion Effects
–20 dB	Moderate distortion, low noise
–30 dB	Best compromise between distortion and noise
–40 dB	Low distortion, high noise

Data Type	ViReal64
Permissions	R/W
High-Level VI	None

# **Acquisition: IQ: IQ Carrier Frequency**

#### Short Name: IQ.Carrier Frequency

Specifies the expected carrier frequency of the incoming signal for demodulation. The RF signal analyzer tunes to this frequency. This value may be coerced based on hardware settings and downconversion specifications.

Data Type	ViReal64
Permissions	R/W
High-Level VI	None

# Acquisition:IQ:IQ Rate (S/s)

#### Short Name: IQ.Rate

Specifies the IQ rate for the acquisition. The value is expressed in S/s.



**Notes** Bandwidth is equal to the coerced IQ rate times 0.8.

You should not need to configure an IQ rate higher than 25 MHz, since the NI PXI-5600 downconverter bandwidth is 20 MHz. If you choose to configure a higher IQ rate, you may see aliasing effects at negative frequencies because the IF frequency of the downconverter is at 15 MHz.

Data Type	ViReal64
Permissions	R/W
High-Level VI	None

# Acquisition:IQ:Number Of Samples Is Finite

#### Short Name: IQ.NumSampsIsFinite

Specifies whether to configure the device to acquire a finite number of samples or to acquire samples continuously.

Data Type	ViBoolean
Permissions	R/W
High-Level VI	niRFSA Configure Number of Samples

# Acquisition:IQ:Number Of Samples

**Short Name:** IQ.NumSamps Configures the number of samples.

Data Type	Vilnt64
Permissions	R/W
High-Level VI	None

# Acquisition:IQ:Number Of Records Is Finite

#### Short Name: IQ.NumRecordsIsFinite

Configures the device to stop after acquiring the specified number of records. Set to FALSE to acquire records continuously until you <u>abort</u> the acquisition.

Data Type	ViBoolean
Permissions	R/W
High-Level VI	None

# Acquisition:IQ:Number Of Records

#### Short Name: IQ.NumRecords

Specifies the number of records to acquire if the <u>Number of Records Is</u> <u>Finite</u> property is set to TRUE.

Data Type	Vilnt64
Permissions	R/W
High-Level VI	None

# **Acquisition:Spectrum:Center Frequency**

#### Short Name: Spectrum.CenterFrequency

Specifies the center frequency in a spectrum acquisition. The value is expressed in hertz. An acquisition consists of a span of data surrounding the center frequency.

Data Type	ViReal64
Permissions	R/W
High-Level VI	niRFSA Configure Spectrum Frequency

# Acquisition:Spectrum:Span

#### Short Name: Spectrum.Span

Specifies the frequency range of the computed spectrum. If you specify a center frequency of 1 GHz and span of 100 MHz, the spectrum ranges from 950 MHz to 1050 MHz after zoom processing. This value may be coerced based on hardware settings and downconversion specifications.



**Note** If you configure the spectrum span (stop frequency – start frequency) to a value larger than 20 MHz, RFSA performs multiple acquisitions and combines them into a spectrum of the size you requested.

Data Type	ViReal64
Permissions	R/W
High-Level VI	niRFSA Configure Spectrum Frequency

# Acquisition:Spectrum:Power Spectrum Units

#### Short Name: Spectrum.Units

Specifies the units of the spectrum.

<b>dBm</b> (200)	Units are dB with reference to 1 mW.
Volts Squared (201)	Units are in V <sup>2</sup> RMS.
<b>dBmV</b> (202)	Units are dB with reference to 1 millivolt.
<b>dBuV</b> (203)	Units are dB with reference to 1 microvolt.

Data Type	Vilnt32
Permissions	R/W
High-Level VI	None

# Acquisition:Spectrum:Resolution Bandwidth (Hz)

Short Name: Spectrum.ResolutionBandwidth

Specifies the resolution along the X axis of the spectrum. NI-RFSA uses the resolution bandwidth value to determine the acquisition size. If the <u>Number of Spectral Lines</u> property is specified, that value takes precedence over this value. If both properties are set to -1, the spectrum uses a default of 400 spectral lines.

Data Type	ViReal64
Permissions	R/W
High-Level VI	None

# Acquisition:Spectrum:Resolution Bandwidth Type

**Short Name:** Spectrum.ResolutionBandwidthType

Specifies the definition of the <u>Resolution Bandwidth</u> property.

<b>3dB</b> (300)	Defines the RBW in terms of the 3 dB bandwidth of the window specified by the <u>FFT Window Type</u> property.
<b>6dB</b> (301)	Defines the resolution bandwidth in terms of the 6 dB bandwidth of the window specified by the <u>FFT Window Type</u> property.
Bin Width (302)	Defines the resolution bandwidth in terms of the display resolution, which is the ratio of the sampling frequency to the number of samples that you acquire.
<b>ENBW</b> (303)	Defines the resolution bandwidth in terms of the Equivalent Noise Bandwidth (ENBW) of the window specified by the <u>FFT</u> <u>Window Type</u> property.

Data Type	Vilnt32
Permissions	R/W
High-Level VI	None

## Acquisition:Spectrum:Number Of Spectral Lines

Short Name: Spectrum.NumSpectralLines

Configures the number of spectral lines expected with the current power spectrum configuration. If you do not configure this property, NI-RFSA selects an appropriate value based on the <u>Resolution Bandwidth</u> property. If you configure this property, NI-RFSA coerces the Resolution Bandwidth value based on the number of spectral lines requested and the <u>acquisition span</u>.

Data Type	Vilnt32
Permissions	R/W
High-Level VI	None

# Acquisition:Spectrum:Averaging Mode

### Short Name: Spectrum.AveragingMode

Specifies the averaging mode for the spectrum acquisition.

<b>None</b> (400)	Configures the driver to perform no averaging on acquisitions.
<b>RMS</b> (401)	Configures the driver for RMS averaging. RMS averaging reduces signal fluctuations but not the noise floor. RMS averaging averages the energy or power of the signal, which prevents noise floor reduction and gives averaged rms quantities of single-channel measurements zero phase. RMS averaging for dual-channel measurements preserves important phase information.
<b>Vector</b> (402)	Configures the driver for vector averaging. Vector averaging reduces noise from synchronous signals. Vector averaging computes the average of complex quantities directly, which means that it allows separate averaging for real and imaginary parts. Complex averaging such as vector averaging reduces noise and usually requires a trigger to improve block-to-block phase coherence.
Peak Hold (403)	Configures the driver for peak hold averaging. Peak hold averaging retains the RMS peak levels of the averaged quantities. The peak hold averaging process performs peak hold at each frequency bin separately to retain peak rms levels from one FFT record to the next.

Data Type	Vilnt32
Permissions	R/W
High-Level VI	None

# Acquisition:Spectrum:Number Of Averages

#### Short Name: Spectrum.NumAverages

Specifies the number of averages to complete for linear weighting. The averaging process returns the final result after the number of averages is complete.

Data Type	Vilnt32
Permissions	R/W
High-Level VI	None

# Acquisition:Spectrum:FFT Window Type

Short Name: Spectrum.FFTWindowType

Specifies the time-domain window type.

**Uniform** (500)

Hanning (501)

Hamming (502)

Blackman-Harris (503)

Exact Blackman (504)

Blackman (505)

**Flat Top** (506)

4-term Blackman Harris (507)

7-term Blackman Harris (508)

Low Side Lobe (509)

Data Type	Vilnt32
Permissions	R/W
High-Level VI	None

# Acquisition:Spectrum:FFT Window Size

Short Name: Spectrum.FFTWindowSize

Returns the size of the window used in the Fast Fourier Transform.

Data Type	Vilnt32
Permissions	RO
High-Level VI	None

# Acquisition:Spectrum:FFT Size

**Short Name:** Spectrum.FFTSize Returns the size of the Fast Fourier Transform.

Data Type	Vilnt32
Permissions	RO
High-Level VI	None

# Acquisition:Fetch:Fetch Relative To

### Short Name: FetchRelativeTo

Specifies the absolute location within the acquired record from which to begin fetching.

Most recent sample (700)	Specifies that fetching occur relative to the most recently acquired data. The <u>Fetch Offset</u> property must be negative.
First sample (701)	Specifies that fetching occurs at the first sample acquired by the device. If the device wraps its buffer, then the first sample is no longer available. In this case, NI-RFSA returns an error if the fetch offset is in the overwritten data.
Reference trigger (702)	Specifies that fetching occur relative to the Reference trigger. This value behaves like <b>First Sample</b> if no Reference trigger is configured.
First pretrigger sample (703)	Specifies that fetching occur relative to the first pretrigger sample acquired. This value behaves like <b>First Sample</b> if no Reference trigger is configured.
Current read position (704)	Specifies that fetching occur after the last fetched sample.

Data Type	Vilnt32
Permissions	RO
High-Level VI	None

# Acquisition:Fetch:Fetch Offset

#### Short Name: FetchOffset

Specifies the offset relative to the position specified by the <u>Fetch Relative</u> To property from which to start fetching data. Offset can be a positive or negative value.

Data Type	Vilnt32
Permissions	R/W
High-Level VI	None

# Acquisition:Fetch:Records Done

#### Short Name: RecordsDone

Returns the number of records the RF signal analyzer has acquired.

Data Type	Vilnt32
Permissions	RO
High-Level VI	None

# **Clocking:Ref Clock Source**

#### Short Name: RefClockSrc

Specifies the reference clock source.

OnboardClock	Lock the NI-RFSA device to the NI PXI-5600 onboard clock.
	Lock the NI-RFSA device to the external REF IN connector on the NI PXI-5600. You must install the NI PXI- 5600 in Slot 2 of your PXI chassis to use this option.
	Lock the NI-RFSA device to the PXI backplane clock using the NI PXI-5600. You must connect the PXI 10 MHz connector to the REF IN connector on the NI PXI-5600 front panel to use this option.

Data Type	ViString
Permissions	R/W
High-Level VI	niRFSA Configure Ref Clock

# **Clocking:Ref Clock Rate**

#### Short Name: RefClockRate

Specifies the rate of the reference clock. The value is expressed in hertz.

NI-RFSA only supports a reference clock rate of 10 MHz.

Data Type	ViReal64
Permissions	R/W
High-Level VI	niRFSA Configure Ref Clock

## Clocking:Digitizer Sample Clock Timebase Source

Short Name: DigitizerSampClkTimebaseSrc

Specifies the source of the Sample clock timebase, which is the timebase used to control waveform sampling.

OnboardClock	The digitizer will use its onboard clock as the Sample clock timebase.	
	The digitizer will use the signal present on the CLK IN connector as the Sample clock timebase.	

Data Type	ViString
Permissions	R/W
High-Level VI	None

## **Clocking:Digitizer Sample Clock Timebase Rate**

Short Name: DigitizerSampClkTimebaseRate

Specifies the frequency, in hertz, of the external clock used as the timebase source if the <u>Digitizer Sample Clock Timebase Source</u> is an external source.

If timebase rate is set to a value below 60 MHz, signals at frequencies just above the 20 MHz passband of the downconverter may be aliased back into the passband. This aliasing occurs because the IF frequency of the downconverter is at 15 MHz, and the upper end of the passband is at 25 MHz. At sampling rates below 60 MHz, the Nyquist frequency is close to the end of the passband and creates aliases that are not effectively filtered by the downconverter.

Data Type	ViReal64
Permissions	R/W
High-Level VI	None

# **Clocking:PXI Chassis Clk10 Source**

#### Short Name: PXIChassisClk10Src

Specifies the signal to drive the 10 MHz reference clock on the PXI backplane. This option can only be configured when the NI PXI-5600 is in Slot 2 of the PXI chassis.

The device does not drive the PXI 10 MHz backplane reference clock.
The device drives the PXI 10 MHz backplane reference clock with the NI PXI-5600 onboard clock. You must connect the 10 MHz OUT connector to the PXI 10 MHz I/O on the NI PXI 5600 front panel to use this option.
The device drives the PXI 10 MHz backplane reference clock with the reference source attached to the NI PXI- 5600 REF IN connector. You must connect the 10 MHz OUT connector to the PXI 10 MHz I/O on the NI PXI 5600 front panel to use this option.

Data Type	ViString
Permissions	R/W
High-Level VI	niRFSA Configure PXI Chassis Clk10

# Triggers:Start:Type

### Short Name: StartTrig.Type

Specifies whether you want the Start trigger to be a digital edge or software trigger.

<b>None</b> (600)	No Start trigger is configured.
Digital Edge (601)	The Start trigger is not asserted until a digital edge is detected. The source of the digital edge is specified with the <u>Start Trigger Digital Edge Source</u> property.
	The Start trigger is not asserted until a software trigger occurs. You can assert the software trigger by calling the niRFSA Send Software Edge Trigger VI with and selecting Start Trigger as the trigger parameter.

Data Type	Vilnt32
Permissions	R/W
High-Level VI	niRFSA Configure Trigger

# Triggers:Start:Digital Edge:Source

#### Short Name: StartTrig.DigEdge.Src

Specifies the source terminal for the digital edge Start trigger. This property is used only when the <u>Start Trigger Type</u> property is set to **Digital Edge**.

PFI0The trigger is received on PFI 0.PFI1The trigger is received on PFI 1.PXI_Trig0The trigger is received on PXI trigger line 0.PXI_Trig1The trigger is received on PXI trigger line 1.PXI_Trig2The trigger is received on PXI trigger line 2.PXI_Trig3The trigger is received on PXI trigger line 3.PXI_Trig4The trigger is received on PXI trigger line 4.PXI_Trig5The trigger is received on PXI trigger line 5.PXI_Trig6The trigger is received on PXI trigger line 6.PXI_Trig7The trigger is received on PXI trigger line 7.PXI_STARThe trigger is received on the PXI star trigger line.		
PXI_Trig0The trigger is received on PXI trigger line 0.PXI_Trig1The trigger is received on PXI trigger line 1.PXI_Trig2The trigger is received on PXI trigger line 2.PXI_Trig3The trigger is received on PXI trigger line 3.PXI_Trig4The trigger is received on PXI trigger line 4.PXI_Trig5The trigger is received on PXI trigger line 5.PXI_Trig6The trigger is received on PXI trigger line 6.PXI_Trig7The trigger is received on PXI trigger line 7.	PFI0	The trigger is received on PFI 0.
PXI_Trig1The trigger is received on PXI trigger line 1.PXI_Trig2The trigger is received on PXI trigger line 2.PXI_Trig3The trigger is received on PXI trigger line 3.PXI_Trig4The trigger is received on PXI trigger line 4.PXI_Trig5The trigger is received on PXI trigger line 5.PXI_Trig6The trigger is received on PXI trigger line 6.PXI_Trig7The trigger is received on PXI trigger line 7.	PFI1	The trigger is received on PFI 1.
PXI_Trig2The trigger is received on PXI trigger line 2.PXI_Trig3The trigger is received on PXI trigger line 3.PXI_Trig4The trigger is received on PXI trigger line 4.PXI_Trig5The trigger is received on PXI trigger line 5.PXI_Trig6The trigger is received on PXI trigger line 6.PXI_Trig7The trigger is received on PXI trigger line 7.	PXI_Trig0	The trigger is received on PXI trigger line 0.
PXI_Trig3The trigger is received on PXI trigger line 3.PXI_Trig4The trigger is received on PXI trigger line 4.PXI_Trig5The trigger is received on PXI trigger line 5.PXI_Trig6The trigger is received on PXI trigger line 6.PXI_Trig7The trigger is received on PXI trigger line 7.	PXI_Trig1	The trigger is received on PXI trigger line 1.
PXI_Trig4The trigger is received on PXI trigger line 4.PXI_Trig5The trigger is received on PXI trigger line 5.PXI_Trig6The trigger is received on PXI trigger line 6.PXI_Trig7The trigger is received on PXI trigger line 7.	PXI_Trig2	The trigger is received on PXI trigger line 2.
PXI_Trig5The trigger is received on PXI trigger line 5.PXI_Trig6The trigger is received on PXI trigger line 6.PXI_Trig7The trigger is received on PXI trigger line 7.	PXI_Trig3	The trigger is received on PXI trigger line 3.
PXI_Trig6The trigger is received on PXI trigger line 6.PXI_Trig7The trigger is received on PXI trigger line 7.	PXI_Trig4	The trigger is received on PXI trigger line 4.
PXI_Trig7 The trigger is received on PXI trigger line 7.	PXI_Trig5	The trigger is received on PXI trigger line 5.
	PXI_Trig6	The trigger is received on PXI trigger line 6.
PXI_STAR The trigger is received on the PXI star trigger line.	PXI_Trig7	The trigger is received on PXI trigger line 7.
	PXI_STAR	The trigger is received on the PXI star trigger line.

Data Type	ViString
Permissions	R/W
High-Level VI	niRFSA Configure Trigger

## Triggers:Start:Digital Edge:Edge

Short Name: StartTrig.DigEdge.Edge

Specifies the active edge for the Start trigger. This property is used only when the <u>Start Trigger Type</u> property is set to **Digital Edge**.

**Rising Edge** (900)The trigger asserts on the rising edge of the signal.**Falling Edge** (901)The trigger asserts on the falling edge of the signal

Data Type	Vilnt32
Permissions	R/W
High-Level VI	niRFSA Configure Trigger

# Triggers:Start:Export:Output Terminal

Short Name: ExportedStartTrig.OutputTerm

Specifies the destination terminal for the exported Start trigger.

	The signal is not exported.
PFI0	PFI 0 on the front panel SMB jack connector.
PFI1	PFI 1 on the front panel DDC connector.
PXI_Trig0	PXI trigger line 0.
PXI_Trig1	PXI trigger line 1.
PXI_Trig2	PXI trigger line 2.
PXI_Trig3	PXI trigger line 3.
PXI_Trig4	PXI trigger line 4.
PXI_Trig5	PXI trigger line 5.
PXI_Trig6	PXI trigger line 6.
PXI_Trig6	PXI trigger line 7.
PXI_STAR	PXI star trigger line.

Data Type	ViString
Permissions	R/W
High-Level VI	<u>niRFSA Export Signal</u>

# Triggers:Ref:Type

#### Short Name: RefTrig.Type

Specifies whether you want the Reference trigger to be a digital edge, IQ power edge, or software trigger.

<b>None</b> (600)	No Reference trigger is configured.
Digital Edge (601)	The Reference trigger is not asserted until a digital edge is detected. The source of the digital edge is specified with the Reference Trigger Digital Edge Source property.
IQ Power Edge (603)	The Reference trigger is asserted when the signal is changing past the level specified with the slope (rising or falling) configured with the <u>IQ Power Edge Slope</u> property.
<b>Software</b> (604)	The Reference trigger is not asserted until a software trigger occurs. You can assert the software trigger by calling the <u>niRFSA Send Software Edge Trigger</u> VI with and selecting <b>Reference Trigger</b> as the <b>trigger</b> parameter.

Data Type	Vilnt32
Permissions	R/W
High-Level VI	niRFSA Configure Trigger

## **Triggers:Ref:Pretrigger Samples**

#### Short Name: RefTrig.PretrigSamples

Specifies the number of pretrigger samples, the samples acquired before the Reference trigger is received, to be acquired per record.

Data Type	Vilnt32
Permissions	R/W
High-Level VI	niRFSA Configure Trigger

# Triggers:Ref:Digital Edge:Source

#### Short Name: RefTrig.DigEdge.Src

Specifies the source terminal for the digital edge Reference trigger. This property is used only when the <u>Reference Trigger Type</u> property is set to **Digital Edge**.

The trigger is received on PFI 0.
The trigger is received on PFI 1.
The trigger is received on PXI trigger line 0.
The trigger is received on PXI trigger line 1.
The trigger is received on PXI trigger line 2.
The trigger is received on PXI trigger line 3.
The trigger is received on PXI trigger line 4.
The trigger is received on PXI trigger line 5.
The trigger is received on PXI trigger line 6.
The trigger is received on PXI trigger line 7.
The trigger is received on the PXI star trigger line.

Data Type	ViString
Permissions	R/W
High-Level VI	niRFSA Configure Trigger

## Triggers:Ref:Digital Edge:Edge

Short Name: RefTrig.DigEdge.Edge

Specify the active edge for the Reference trigger. This property is used only when the <u>Ref Trigger Type</u> property is set to **Digital Edge**.

**Rising Edge** (900)The trigger asserts on the rising edge of the signal.**Falling Edge** (901)The trigger asserts on the falling edge of the signal.

Data Type	Vilnt32
Permissions	R/W
High-Level VI	niRFSA Configure Trigger

# Triggers:Ref:IQ Power Edge:Source

#### Short Name: RefTrig.IQPwrEdge.Src

Specifies the channel from which the device will monitor the trigger. The only valid input for this attribute is "0" at this time.

Data Type	ViString
Permissions	R/W
High-Level VI	niRFSA Configure Trigger

## Triggers:Ref:IQ Power Edge:Level

#### Short Name: RefTrig.IQPwrEdge.Lvl

Specifies the power level in dBm at which the device will trigger. The device asserts the trigger when the signal exceeds the level specified by the value of this property.

Data Type	ViReal64
Permissions	R/W
High-Level VI	niRFSA Configure Trigger

# Triggers:Ref:IQ Power Edge:Slope

#### Short Name: RefTrig.IQPwrEdge.Slope

Specifies whether the device asserts the trigger when the signal power is rising or falling. When the trigger is configured for IQ power edge the device asserts the trigger when the power exceeds the specified level with the slope you specify.

Rising Slope (1000)	The trigger asserts when the signal power is rising.
Falling Slope	The trigger asserts when the signal power is
(1001)	falling.

Data Type	Vilnt32
Permissions	R/W
High-Level VI	niRFSA Configure Trigger

### Triggers:Ref:IQ Power Edge:Minimum Quiet Time

Short Name: RefTrig.IQPwrEdge.MinQuietTime

Specifies a time duration for which the signal must be quiet before the device arms the trigger. The signal is quiet when it is below the trigger level if the trigger slope, specified by the <u>Ref Trigger IQ Power Edge</u> <u>Slope</u> property, is set to **Rising Slope** or above the trigger level if the trigger slope is set to **Falling Slope**.

By default this value is set to 0, which means the device does not wait for a quiet time before arming the trigger. This property is useful to trigger the acquisition on signals containing repeated bursts, but for which each burst may have large changes in signal power within itself. By configuring the minimum quiet time to the time between bursts, you can ensure that the trigger occurs at the beginning of a burst rather than in signal power change within a burst.

Data Type	ViReal64
Permissions	R/W
High-Level VI	None

# Triggers:Ref:Export:Output Terminal

Short Name: ExportedRefTrig.OutputTerm

Specifies the destination terminal for the exported Reference trigger.

	The signal is not exported.
PFI0	PFI 0 on the front panel SMB jack connector.
PFI1	PFI 1 on the front panel DDC connector.
PXI_Trig0	PXI trigger line 0.
PXI_Trig1	PXI trigger line 1.
PXI_Trig2	PXI trigger line 2.
PXI_Trig3	PXI trigger line 3.
PXI_Trig4	PXI trigger line 4.
PXI_Trig5	PXI trigger line 5.
PXI_Trig6	PXI trigger line 6.
PXI_Trig6	PXI trigger line 7.
PXI_STAR	PXI star trigger line.

Data Type	ViString
Permissions	R/W
High-Level VI	<u>niRFSA Export Signal</u>

## Triggers:Ref:Advanced:Ref To Ref Trigger Holdoff (s)

Short Name: RefToRefHoldoff

Specifies the minimum time in seconds that must elapse after the Reference trigger for one record is received before the device will recognize the Reference trigger for the next record.

Data Type	ViReal64
Permissions	R/W
High-Level VI	None

## Triggers:Ref:Advanced:Start To Ref Trigger Holdoff (s)

Short Name: StartToRefHoldoff

Specifies the minimum time in seconds that must elapse after the Start trigger is received before the device recognizes a Reference trigger.

Data Type	ViReal64
Permissions	R/W
High-Level VI	None

# Triggers:Advance:Type

#### Short Name: AdvanceTrig.Type

Specifies whether you want the Advance trigger to be a digital edge or pattern match trigger.

<b>None</b> (600)	No Advance trigger is configured.
•	The Advance trigger is not asserted until a digital edge is detected. The source of the digital edge is specified with the Digital Edge Advance Trigger Source property.
	The Advance trigger is not asserted until a software trigger occurs. You can assert the software trigger by calling the niRFSA Send Software Edge Trigger VI with and selecting Start Trigger as the trigger parameter.

Data Type	Vilnt32
Permissions	R/W
High-Level VI	niRFSA Configure Trigger

## Triggers:Advance:Digital Edge:Source

Short Name: AdvanceTrig.DigEdge.Src

Specifies the source terminal for the Advance trigger. This property is used only when <u>Advance Trigger Type</u> is set to **Digital Edge**.

PFI0	The trigger is received on PFI 0.
PFI1	The trigger is received on PFI 1.
PXI_Trig0	The trigger is received on PXI trigger line 0.
PXI_Trig1	The trigger is received on PXI trigger line 1.
PXI_Trig2	The trigger is received on PXI trigger line 2.
PXI_Trig3	The trigger is received on PXI trigger line 3.
PXI_Trig4	The trigger is received on PXI trigger line 4.
PXI_Trig5	The trigger is received on PXI trigger line 5.
PXI_Trig6	The trigger is received on PXI trigger line 6.
PXI_Trig7	The trigger is received on PXI trigger line 7.
PXI_STAR	The trigger is received on the PXI star trigger line.

Data Type	ViString
Permissions	R/W
High-Level VI	niRFSA Configure Trigger

# Triggers:Advance:Export:Output Terminal

Short Name: ExportedAdvanceTrig.OutputTerm

Specifies the destination terminal for the exported Advance trigger.

	The signal is not exported.
PFI0	PFI 0 on the front panel SMB jack connector.
PFI1	PFI 1 on the front panel DDC connector.
PXI_Trig0	PXI trigger line 0.
PXI_Trig1	PXI trigger line 1.
PXI_Trig2	PXI trigger line 2.
PXI_Trig3	PXI trigger line 3.
PXI_Trig4	PXI trigger line 4.
PXI_Trig5	PXI trigger line 5.
PXI_Trig6	PXI trigger line 6.
PXI_Trig6	PXI trigger line 7.
PXI_STAR	PXI star trigger line.

Data Type	ViString
Permissions	R/W
High-Level VI	<u>niRFSA Export Signal</u>

# Triggers:Arm Ref:Type

#### Short Name: ArmRefTrig.Type

Specifies whether you want the Arm Reference trigger to be a digital edge or software trigger.

<b>None</b> (600)	No Arm Reference trigger is configured.
Digital Edge (601)	The Arm Reference trigger is not asserted until a digital edge is detected. The source of the digital edge is specified with the <u>Arm Ref Trigger Digital Edge Source</u> property.
(604)	The Arm Reference trigger is not asserted until a software trigger occurs. You can assert the software trigger by calling the niRFSA Send Software Edge Trigger VI with and selecting <b>Arm Ref Trigger</b> as the <b>trigger</b> parameter.

Data Type	Vilnt32
Permissions	R/W
High-Level VI	None

# Triggers:ArmRef:Digital Edge:Source

#### Short Name: ArmRefTrig.DigEdge.Src

Specifies the source terminal for the digital edge Arm Reference trigger. This property is used only when the <u>Arm Ref Trigger Type</u> property is set to **Digital Edge**.

PFI0	The trigger is received on PFI 0.
PFI1	The trigger is received on PFI 1.
PXI_Trig0	The trigger is received on PXI trigger line 0.
PXI_Trig1	The trigger is received on PXI trigger line 1.
PXI_Trig2	The trigger is received on PXI trigger line 2.
PXI_Trig3	The trigger is received on PXI trigger line 3.
PXI_Trig4	The trigger is received on PXI trigger line 4.
PXI_Trig5	The trigger is received on PXI trigger line 5.
PXI_Trig6	The trigger is received on PXI trigger line 6.
PXI_Trig7	The trigger is received on PXI trigger line 7.
PXI_STAR	The trigger is received on the PXI star trigger line.

Data Type	ViString
Permissions	R/W
High-Level VI	None

## **Events:Ready For Start:Output Terminal**

Short Name: RdyForStartEvent.OutputTerm

Specifies the destination terminal for the Ready for Start event.

	The signal is not exported.
PFI0	PFI 0 on the front panel SMB jack connector.
PFI1	PFI 1 on the front panel DDC connector.
PXI_Trig0	PXI trigger line 0.
PXI_Trig1	PXI trigger line 1.
PXI_Trig2	PXI trigger line 2.
PXI_Trig3	PXI trigger line 3.
PXI_Trig4	PXI trigger line 4.
PXI_Trig5	PXI trigger line 5.
PXI_Trig6	PXI trigger line 6.
PXI_Trig6	PXI trigger line 7.
PXI_STAR	PXI star trigger line.

Data Type	ViString
Permissions	R/W
High-Level VI	<u>niRFSA Export Signal</u>

## **Events:Ready For Advance:Output Terminal**

Short Name: RdyForAdvanceEvent.OutputTerm

Specifies the destination terminal for the Ready for Advance event.

	The signal is not exported.
PFI0	PFI 0 on the front panel SMB jack connector.
PFI1	PFI 1 on the front panel DDC connector.
PXI_Trig0	PXI trigger line 0.
PXI_Trig1	PXI trigger line 1.
PXI_Trig2	PXI trigger line 2.
PXI_Trig3	PXI trigger line 3.
PXI_Trig4	PXI trigger line 4.
PXI_Trig5	PXI trigger line 5.
PXI_Trig6	PXI trigger line 6.
PXI_Trig6	PXI trigger line 7.
PXI_STAR	PXI star trigger line.

Data Type	ViString
Permissions	R/W
High-Level VI	<u>niRFSA Export Signal</u>

# **Events:Ready For Ref:Output Terminal**

Short Name: RdyForRefEvent.OutputTerm

Specifies the destination terminal for the Ready for Reference event.

	The signal is not exported.
PFI0	The signal is exported to PFI 0.
PFI1	The signal is exported to PFI 1.
PXI_Trig0	The signal is exported to PXI trigger line 0.
PXI_Trig1	The signal is exported to PXI trigger line 1.
PXI_Trig2	The signal is exported to PXI trigger line 2.
PXI_Trig3	The signal is exported to PXI trigger line 3.
PXI_Trig4	The signal is exported to PXI trigger line 4.
PXI_Trig5	The signal is exported to PXI trigger line 5.
PXI_Trig6	The signal is exported to PXI trigger line 6.
PXI_Trig6	The signal is exported to PXI trigger line 7.
PXI_STAR	The signal is exported to the PXI star trigger line.

Data Type	ViString
Permissions	R/W
High-Level VI	<u>niRFSA Export Signal</u>

# **Events:End Of Record:Output Terminal**

Short Name: EndOfRecEvent.OutputTerm

Specifies the destination terminal for the End of Record event.

	The signal is not exported.	
PFI0	PFI 0 on the front panel SMB jack connector.	
PFI1	PFI 1 on the front panel DDC connector.	
PXI_Trig0	PXI trigger line 0.	
PXI_Trig1	PXI trigger line 1.	
PXI_Trig2	PXI trigger line 2.	
PXI_Trig3	PXI trigger line 3.	
PXI_Trig4	PXI trigger line 4.	
PXI_Trig5	PXI trigger line 5.	
PXI_Trig6	PXI trigger line 6.	
PXI_Trig6	PXI trigger line 7.	
PXI_STAR	PXI star trigger line.	

Data Type	ViString
Permissions	R/W
High-Level VI	<u>niRFSA Export Signal</u>

## **Advanced:Digital IF Equalization Enabled**

**Short Name:** DigitalIFEqualizationEnabled Toggles use of the digital equalization filter for the NI 5600.

Data Type	ViBoolean
Permissions	R/W
High-Level VI	None

## **Device Characteristics:Serial Number**

Short Name: SerialNum

Data Type	ViString
Permissions	RO
High-Level VI	None

## **Device Characteristics:Device Temperature (°C)**

Short Name: DeviceTemp

Returns the current temperature of the NI 5600 downconverter module.

Data Type	ViReal64
Permissions	RO
High-Level VI	None

## Inherent IVI Attributes:User Options:Cache

#### Short Name: Cache

Specifies whether to cache the value of properties. When caching is enabled, NI-RFSA tracks the current NI-RFSA device settings and avoids sending redundant commands to the device.

NI-RFSA can always cache or never cache particular properties, regardless of the setting of this property.

Use the <u>niRFSA Initialize With Options</u> VI to override the default value.

Data Type	ViBoolean
Permissions	R/W
High-Level VI	None

## Inherent IVI Attributes:User Options:Interchange Check

Short Name: Interchange Check

Specifies whether to perform interchangeability checking and retrieve interchangeability warnings.

Data Type	ViBoolean
Permissions	R/W
High-Level VI	None

# Inherent IVI Attributes:User Options:Driver Setup

#### Short Name: Revision

The DriverSetup string is used to set the initial values for attributes that are specific to NI-RFSA.

The format of the Driver Setup string is:

Tag: Value

*Tag* is the name of the DriverSetup string attribute. *Value* is the value set to the attribute. To set multiple attributes, separate their assignments with a semicolon.

The DriverSetup string can include the following tags:

Digitizer—Specifies the resource name of the digitizer to use for this session. If this DriverSetup tag is not specified, the resource name for the downconverter associated in MAX is used, for example, DriverSetup=Digitizer:pxi1slot4

Refer to <u>niRFSA Initialize With Options</u> for additional information about the optionsString parameter. Refer to the <u>NI RF Vector Signal Analyzers</u> <u>Getting Started Guide</u> for information on MAX setup.

Default Value: "" (empty string)

Data Type	ViString
Permissions	RO
High-Level VI	None

## Inherent IVI Attributes:User Options:Query Instrument Status

Short Name: Query Instrument Status

Specifies whether NI-RFSA queries the NI-RFSA device status after each operation. Querying the device status is useful for debugging. After you validate your program, you can set this property to FALSE to disable status checking and maximize performance.

NI-RFSA can choose to ignore status checking for particular properties, regardless of the setting of this property.

Note: Use the niRFSA Initialize with Options VI to override this value.

Data Type	ViBoolean
Permissions	R/W
High-Level VI	None

## Inherent IVI Attributes:User Options:Range Check

#### Short Name: RangeCheck

Specifies whether to validate attribute values and function parameters. If enabled, NI-RFSA validates the parameter values that you pass to NI-RFSA functions. Range checking parameters is very useful for debugging. After you validate your program, you can set this attribute to FALSE to disable range checking and maximize performance.



**Note** Use the <u>niRFSA Initialize With Options</u> VI to override this value.

## **Defined Values:**

NI-RFSA validates attribute values and function parameters. This is the default value.
NI-RFSA does not validate attribute values and function parameters.

Data Type	ViBoolean
Permissions	R/W
High-Level VI	None

# Inherent IVI Attributes:User Options:Record Value Coercions

Short Name: Record Value Coercions

Specifies whether the IVI engine keeps a list of the value coercions it makes for integer and real type properties.

Note Record Value Coercions is not supported.

Data Type	ViBoolean
Permissions	R/W
High-Level VI	None

## Inherent IVI Attributes:User Options:Simulate

#### Short Name: Simulate

Specifies whether NI-RFSA simulates I/O operations. This is useful for debugging applications without using hardware. Once a session is opened, you cannot change the simulation state. Use the niRFSA Initialize with Options VI to enable simulation.

Data Type	ViBoolean
Permissions	RO
High-Level VI	None

# Inherent IVI Attributes:Driver Identification:Description

Short Name: Description

A string that contains a brief description of NI-RFSA.

Data Type	ViString
Permissions	RO
High-Level VI	None

## Inherent IVI Attributes:Driver Identification:Driver Prefix

Short Name: Driver Prefix

A string that contains the prefix for NI-RFSA. The name of each usercallable function in NI-RFSA starts with this prefix.

Data Type	ViString
Permissions	RO
High-Level VI	None

## Inherent IVI Attributes:Driver Identification:Driver Vendor

Short Name: Driver Vendor

A string that contains the name of the vendor that supplies NI-RFSA.

Data Type	ViString
Permissions	RO
High-Level VI	None

## Inherent IVI Attributes:Driver Identification:Revision

Short Name: Revision

A string that contains additional version information about NI-RFSA.

Data Type	ViString
Permissions	RO
High-Level VI	None

## Inherent IVI Attributes:Driver Capabilities:Supported Instrument Models

Short Name: Supported Instrument Models

Contains a model code of the NI-RFSA device. For drivers that support more than one device, this property contains a comma-separated list of supported devices.

## Remarks

The following table lists the characteristics of this property.

Data Type	ViString
Permissions	RO
High-Level VI	None

# Inherent IVI Attributes:Instrument Identification:Manufacturer

Short Name: Manufacturer

A string that contains the name of the manufacturer for the NI-RFSA device you are currently using.

## Remarks

The following table lists the characteristics of this property.

Data Type	ViString
Permissions	RO
High-Level VI	None

# Inherent IVI Attributes:Instrument Identification:Model

Short Name: Model

A string that contains the model number or name of the NI-RFSA device you are currently using.

## Remarks

The following table lists the characteristics of this property.

Data Type	ViString
Permissions	RO
High-Level VI	None

# Inherent IVI Attributes:Instrument Identification:Firmware Revision

Short Name: Firmware Revision

A string that contains the firmware revision information for the NI-RFSA device you are currently using.

## Remarks

The following table lists the characteristics of this property.

Data Type	ViString
Permissions	RO
High-Level VI	None

# Inherent IVI Attributes:Advanced Session Information:Logical Name

Short Name: Logical Name

Contains the logical name you specified when opening the current IVI session. You may pass a logical name to the <u>niRFSA Initialize</u> or <u>niRFSA</u> <u>Initialize with Options</u> VIs. The IVI Configuration Utility must contain an entry for the logical name. The logical name entry refers to a driver session section in the IVI Configuration file. The driver session section specifies a physical device and initial user options.

## Remarks

The following table lists the characteristics of this property.

Data Type	ViString
Permissions	RO
High-Level VI	None

# Inherent IVI Attributes:Advanced Session Information:Resource Descriptor

Short Name: Resource Descriptor

Indicates the resource descriptor NI-RFSA uses to identify the physical device. If you initialize NI-RFSA with a logical name, this property contains the resource name that corresponds to the entry in the IVI Configuration Utility.

If you initialize NI-RFSA with the resource name, this property contains that value.

## Remarks

The following table lists the characteristics of this property.

Data Type	ViString
Permissions	RO
High-Level VI	None

# niRFSA\_init

ViStatus = niRFSA\_init (ViRsrc resourceName, ViBoolean IDQuery, ViBoolean reset, ViSession\* vi);

### Purpose

Creates a new session for the device. This function performs the following initialization actions:

- Creates a new instrument driver session to the RF signal analyzer, using the downconverter resource name you specify.
- Sends initialization commands to reset both hardware modules to a known state necessary for NI-RFSA operation.
- Note Before initializing the NI 5661, an NI 5142 IF digitizer module must be associated with the NI 5600 downconverter module in MAX. After association, pass the NI 5600 device name to this VI to initialize both modules. To change the digitizer association, modify the NI 5600 Properties page in MAX, or use this function to override the association in MAX. Refer to the *NI RF Vector Signal Analyzers Getting Started Guide*, installed at Start»Programs»National Instruments»NI-RFSA»Documentation for information on MAX association.

#### **Parameters**

Input

Name

Type

resourceName ViRsrc

Description

Specifies the resource name of the device to initialize.

Example #	Device Type	Syntax
1	myDAQmxDevice	NI-DAQmx device device name = "myDAQmxDevic
2	myLogicalName	IVI logical name c virtual instrument name = "myLogicalName'

For NI-DAQmx devices, the syntax is the device name specified in MAX, as shown in Example Typical default names for NI-DAQmx devices i MAX are Dev1 or PXI1Slot1. You can rename an NI-DAQmx device by right-clicking on the name in MAX and entering a new name. You can also pass in the name of an IVI logical name configured with the IVI Configuration utility. For additional information, refer to the IN topic in the Measurement & Automation Explorer Help.



#### 🔥 Caution

NI-DAQmx device names are not casesensitive. However, all IVI names, such as logical names, are case-sensitive. If you use an IVI logical name, make sure the name is identical to the name showr in the IVI Configuration Utility.

IDQuery

ViBoolean Specifies whether NI-RFSA performs an ID

		query.
reset	ViBoolean	Specifies whether you want the to reset the NI RFSA device during the initialization procedure TRUE means that the device is reset; FALSE means that the device is not reset. FALSE is the default value.
Output		
Name	Туре	Description
vi	ViSession	Identifies your instrument session.

## **Return Value**

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# niRFSA\_InitWithOptions

ViStatus = niRFSA\_InitWithOptions( ViRsrc resourceName, ViBoolean IDQuery, ViBoolean reset, ViConstString optionString, ViSession\* newVi);

### Purpose

Creates a new session for the device. This function performs the following initialization actions:

• Creates a new instrument driver session to the RF signal analyzer, using the downconverter resource name you specify.

- Sends initialization commands to reset both hardware modules to a known state necessary for NI-RFSA operation.
- Note Before initializing the NI 5661, an NI 5142 IF digitizer module must be associated with the NI 5600 downconverter module in MAX. After association, pass the NI 5600 device name to this VI to initialize both modules. To change the digitizer association, modify the NI 5600 Properties page in MAX, or use the <u>niRFSA\_InitWithOptions</u> function to override the association in MAX. Refer to the *NI RF Vector Signal Analyzers Getting Started Guide*, installed at Start»Programs»National Instruments»NI-RFSA»Documentation for information on MAX association.

### **Parameters**

Input

Name Type

resourceName ViRsrc

Description

Specifies the resource name of the device

Example #	Device Type	
1	myDAQmxDevice	NI-DAQmx de "myDAQmxDŧ
2	myLogicalName	IVI logical nan "myLogicalNa

For NI-DAQmx devices, the syntax is the d shown in Example 1. Typical default name: Dev1 or PXI1Slot1. You can rename an NIthe name in MAX and entering a new name an IVI logical name configured with the IVI information, refer to the IVI topic in the Mea Help.



#### Caution

NI-DAQmx device names are not cas names, such as logical names, are c logical name, make sure the name is **IVI** Configuration Utility.

Specifies whether NI-RFSA performs an IC **IDQuery** ViBoolean

- Specifies whether you want the to reset the ViBoolean reset initialization procedure. TRUE means that that the device is not reset. FALSE is the d
- optionString ViConstString Sets the initial value of certain attributes fo lists the attributes and the name you pass attribute.

Name	Att
RangeCheck	NIRFSG_ATTR_RANG

QueryInstrStatus	NIRFSG_ATTR_QUER`
Cache	NIRFSG_ATTR_CACHI
RecordCoercions	NIRFSG_ATTR_RECO
DriverSetup	NIRFSG_ATTR_DRIVE

The format of this string is, "*AttributeName* name of the attribute and *Value* is the value To set multiple attributes, separate their as

Example Option String: "RangeCheck=1,QueryInstrStatus=0,Cach

*Output* Name

Type Description

vi ViSession Identifies your instrument session.

## **Return Value**

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# niRFSA\_ConfigureAcquisitionType

ViStatus = niRFSA\_ConfigureAcquisitionType( ViSession vi, ViInt32 acquisitionType);

### Purpose

Configures whether the session acquires IQ data or computes a power spectrum over the specified frequency range.

### Parameters

Input

•			
Name	Туре	Description	
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.	
acquisitionType	Vilnt32	Configures	the type of acquisition.
		IQ	Configures the driver for IQ acquisitions.
		Spectrum	Configures the driver for spectrum acquisitions.

## **Return Value**

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# niRFSA\_ConfigureReferenceLevel

ViStatus = niRFSA\_ConfigureReferenceLevel( ViSession vi, ViConstString channelList, ViReal64 referenceLevel);

### Purpose

Configures the reference level. The reference level represents the maximum expected power of an input RF signal.

### Parameters

Input

Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
channelList	ViConstString	Identifies channels to apply settings. Use "" or VI_NULL to specify all channels.
referenceLevel	ViReal64	Specifies the expected total integrated power of the RF input signal in dBm.

## **Return Value**

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# niRFSA\_ConfigureIQCarrierFrequency

ViStatus = niRFSA\_ConfigureIQCarrierFrequency(ViSession vi, ViConstString channelList, ViReal64 carrierFrequency);

### Purpose

This function configures the carrier frequency of the RF vector signal analyzer hardware during an IQ acquisition.

### Parameters

Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
channelList	ViConstString	Identifies channels to apply settings. Use "" or VI_NULL to specify all channels.
carrierFrequency	ViReal64	Specifies the expected carrier frequency of the incoming signal for demodulation. The RF vector signal analyzer tunes to this frequency. This value may be coerced based on hardware settings and downconversion specifications.

## **Return Value**

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

0	Success	
Positive Values	Warnings	
Negative Values	Errors	

# niRFSA\_ConfigurelQRate

ViStatus = niRFSA\_ConfigureIQRate( ViSession vi, ViConstString channelList, ViReal64 iqRate);

### Purpose

Configures the rate at which the device samples IQ values. Bandwidth is equal to the coerced **iqRate** times 0.8.



**Note** You should not need to configure an IQ rate higher than 25 MHz, since the NI PXI-5600 downconverter bandwidth is 20 MHz. If you choose to configure a higher IQ rate, you may see aliasing effects at negative frequencies because the IF frequency of the downconverter is at 15 MHz.

### Parameters

Input		
Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
channelList	ViConstString	Identifies channels to apply settings. Use "" or VI_NULL to specify all channels.
iqRate	ViReal64	Specifies the IQ rate for the acquisition. The value is expressed in S/s.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# niRFSA\_ConfigureNumberOfSamples

ViStatus = niRFSA\_ConfigureNumberOfSamples( ViSession vi, ViConstString channelList, ViBoolean numberOfSamplesIsFinite, ViInt64 samplesPerRecord);

Configures the number of samples in a finite acquisition or configures the device to continuously acquire samples. If you configure the device for finite acquisition, it acquires the specified number of samples and stops the acquisition. You can configure the device to acquire multiple records using the <u>niRFSA\_ConfigureNumberOfRecords</u> function, each record containing the number of samples specified in this function. The default number of records to acquire is 1.

If the device is configured to continuously acquire samples, it continues acquiring data until you abort the acquisition. The device stores data in onboard memory in a circular fashion. Once the device fills the memory, it starts overwriting previously acquired data from the beginning of the memory buffer. Retrieve the samples using the niRFSA fetch IQ functions as they are being acquired to avoid data being overwritten before you can retrieve it.

Input		
Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
channelList	ViConstString	Identifies channels to apply settings. Use "" or VI_NULL to specify all channels.
numberOfSamplesIsFinite	ViBoolean	Specifies whether to configure the device to acquire a finite number of samples or to acquire samples continuously. VI_TRUE indicates that the device acquires a finite number of samples, while VI_FALSE indicates that the device continuously acquires samples.
samplesPerRecord	Vilnt64	Specifies the number of samples per record if <b>numberOfsamplesIsFinite</b> is set to VI_TRUE.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# niRFSA\_ConfigureNumberOfRecords

ViStatus = niRFSA\_ConfigureNumberOfRecords( ViSession vi, ViConstString channelList, ViBoolean numberOfRecordsIsFinite, ViInt64 numberOfRecords);

Configures the number of records in a finite acquisition or configures the device to continuously acquire records. Notice that you can only configure the device to acquire multiple records if **numberOfSamplesIsFinite** is set to VI\_TRUE.

If you configure the device to acquire records continuously, it continues acquiring records until you abort the acquisition. The device stores records in onboard memory in a circular fashion. Once the device fills the memory, it starts overwriting previously acquired records from the beginning of the memory buffer. Fetch the records as they are being acquired to avoid data being overwritten before you can retrieve it.

Input		
Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
channelList	ViConstString	Identifies channels to apply settings. Use "" or VI_NULL to specify all channels.
numberOfRecordsIsFinite	ViBoolean	Set to VI_TRUE to configure the device to stop after acquiring the specified number of records. Set to VI_FALSE to acquire records continuously until you abort the acquisition.
numberOfRecords	Vilnt64	Specifies the number of records to acquire if number of records is finite is set to VI_TRUE.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# niRFSA\_ConfigureSpectrumFrequencyCenterSpa

ViStatus = niRFSA\_ConfigureSpectrumFrequencyCenterSpan( ViSession vi, ViConstString channelList, ViReal64 centerFrequency, ViReal64 span);

Configures the span and center frequency of a spectrum acquisition. An acquisition consists of a span of data surrounding the center frequency.



**Note** If you configure the spectrum span (stop frequency – start frequency) to a value larger than 20 MHz, RFSA performs multiple acquisitions and combines them into a spectrum of the size you requested.

Input

Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
channelList	ViConstString	Identifies channels to apply settings. Use "" or VI_NULL to specify all channels.
centerFrequency	ViReal64	Specifies the center frequency in a spectrum acquisition. The value is expressed in Hertz.
span	ViReal64	Specifies the span of a spectrum acquisition. The value is expressed in Hertz.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# niRFSA\_ConfigureSpectrumFrequencyStartStop

ViStatus = niRFSA\_ConfigureSpectrumFrequencyStartStop( ViSession vi, ViConstString channelList, ViReal64 startFrequency, ViReal64 stopFrequency);

Configures the start and stop frequency of a spectrum acquisition.

Note If you configure the spectrum span (**stopFrequency** – **startFrequency**) to a value larger than 20 MHz, RFSA performs multiple acquisitions and combines them into a spectrum of the size you requested.

Input

Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
channelList	ViConstString	Identifies channels to apply settings. Use "" or VI_NULL to specify all channels.
startFrequency	ViReal64	Specifies the lower band of a span of frequencies.
stopFrequency	ViReal64	Specifies the upper band of a span of frequencies.

Name Type Description

status ViStatus Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.

To obtain a text description of the status code, call the <u>niRFSA\_error\_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA\_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA\_ClearError</u> function.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Positive Values	Warnings
Negative Values	Errors

# niRFSA\_ConfigureResolutionBandwidth

ViStatus = niRFSA\_ConfigureResolutionBandwidth( ViSession vi, ViConstString channelList, ViReal64 resolutionBandwidth);

Configures the resolution bandwidth of a spectrum acquisition. The resolution bandwidth controls the width of the frequency bins in the power spectrum computed by NI-RFSA. A larger value for resolution bandwidth means the frequency bins are wider, and hence you get fewer bins or spectral lines.

By default, the resolution bandwidth value corresponds to the 3 dB bandwidth of the window type NI-RFSA uses to compute the spectrum. To specify the frequency bin width directly, change the resolution bandwidth type attribute to bin width. Refer to the NIRFSA\_ATTR\_RESOLUTION\_BANDWIDTH\_TYPE attribute for more information.

Input

Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> the <u>niRFSA_init</u> or the <u>niRFSA_initW</u> functions and identifies a particular in session.
channelList	ViConstString	Identifies channels to apply settings. VI_NULL to specify all channels.
resolutionBandwidth	ViReal64	Specifies the resolution bandwidth of acquisition. The value is expressed ir Configure the type of resolution band NIRFSA_ATTR_RESOLUTION_BAN attribute.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# niRFSA\_ConfigureDigitalEdgeStartTrigger

ViStatus = niRFSA\_ConfigureDigitalEdgeStartTrigger( ViSession vi, ViConstString source, ViInt32 edge);

Configures the device to wait for a digital edge Start trigger at the beginning of the acquisition.

Input

Name Type Description

vi ViSession Identifies your instrument session. **vi** is obtained from the <u>niRFSA\_init</u> or the <u>niRFSA\_initWithOptions</u> functions and identifies a particular instrument session.

source ViConstString Specifies the source of the digital edge for the Start trigger.

The trigger is received on PFI 0.
The trigger is received on PFI 1.
The trigger is received on PXI trigger line 0.
The trigger is received on PXI trigger line 1.
The trigger is received on PXI trigger line 2.
The trigger is received on PXI trigger line 3.
The trigger is received on PXI trigger line 4.
The trigger is received on PXI trigger line 5.
The trigger is received on PXI trigger line 6.
The trigger is received on PXI trigger line 7.
The trigger is received on the PXI star trigger line.

edge Vilnt32

Specifies the edge to detect. You can choose **Rising Edge** or **Falling Edge**.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# niRFSA\_ConfigureSoftwareEdgeStartTrigger

ViStatus = niRFSA\_ConfigureSoftwareEdgeStartTrigger( ViSession vi);

Configures the device to wait for a software Start trigger at the beginning of the acquisition. The device waits until you call the <a href="mailto:niRFSA\_SendSoftwareEdgeTrigger">niRFSA\_SendSoftwareEdgeTrigger</a> function to assert the trigger.

Input

- Name Type Description
- vi ViSession Identifies your instrument session. **vi** is obtained from the <u>niRFSA\_init</u> or the <u>niRFSA\_initWithOptions</u> functions and identifies a particular instrument session.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# niRFSA\_DisableStartTrigger

ViStatus = niRFSA\_DisableStartTrigger( ViSession vi);

Configures the device to not wait for a Start trigger at the beginning of the acquisition.

Input

- Name Type Description
- vi ViSession Identifies your instrument session. **vi** is obtained from the <u>niRFSA\_init</u> or the <u>niRFSA\_initWithOptions</u> functions and identifies a particular instrument session.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# niRFSA\_ConfigureDigitalEdgeRefTrigger

ViStatus = niRFSA\_ConfigureDigitalEdgeRefTrigger( ViSession vi, ViConstString source, ViInt32 edge, ViInt64 pretriggerSamples);

Configures the device to wait for a digital edge Reference trigger to mark a reference point within the record.

Input

Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
source	ViConstString	Specifies the source of the digital edge for the Reference trigger.

	The trigger is reached an
PFI0	The trigger is received on PFI 0.
PFI1	The trigger is received on PFI 1.
PXI_Trig0	The trigger is received on PXI trigger line 0.
PXI_Trig1	The trigger is received on PXI trigger line 1.
PXI_Trig2	The trigger is received on PXI trigger line 2.
PXI_Trig3	The trigger is received on PXI trigger line 3.
PXI_Trig4	The trigger is received on PXI trigger line 4.
PXI_Trig5	The trigger is received on PXI trigger line 5.
PXI_Trig6	The trigger is received on PXI trigger line 6.
PXI_Trig7	The trigger is received on PXI trigger line 7.
PXI_STAR	The trigger is received on the PXI star trigger line.

edge	Vilnt32	Specifies the edge to detect. You can choose <b>Rising Edge</b> or <b>Falling Edge</b> .
pretriggerSamples	s Vilnt64	Specifies the number of samples to store for each record that were acquired in the time period immediately before the trigger occurred.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

## niRFSA\_ConfigureIQPowerEdgeRefTrigger

ViStatus = niRFSA\_ConfigureIQPowerEdgeRefTrigger( ViSession vi, ViConstString source, ViReal64 level, ViInt32 slope, ViInt64 pretriggerSamples);

Configures the device to wait for the complex power of the IQ data to cross the specified threshold to mark a reference point within the record.

Input

Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
source	ViConstString	Specifies the source of the RF signal for the power edge Reference trigger. The only supported value is "0."
level	ViReal64	Specifies the threshold above or below which the device will trigger.
slope	Vilnt32	Specifies whether the device detects a rising or falling slope on the trigger signal.
pretriggerSamples	Vilnt64	Specifies the number of samples to store for each record that were acquired in the time period immediately before the trigger occurred.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

## niRFSA\_ConfigureSoftwareEdgeRefTrigger

ViStatus = niRFSA\_ConfigureSoftwareEdgeRefTrigger( ViSession vi, ViInt64 pretriggerSamples);

Configures the device to wait for a software Reference trigger to mark a reference point within the record. The device waits until you call the <u>niRFSA\_SendSoftwareEdgeTrigger</u> function to assert the trigger.

InputNameTypeDescriptionviViSessionIdentifies your instrument session. vi is<br/>obtained from the niRFSA\_init or the<br/>niRFSA\_initWithOptions functions and<br/>identifies a particular instrument session.pretriggerSamplesViInt64Specifies the number of samples to store<br/>for each record that were acquired in the<br/>time period immediately before the trigger<br/>occurred.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# niRFSA\_DisableRefTrigger

ViStatus = niRFSA\_DisableRefTrigger( ViSession vi);

Configures the device to not wait for a Reference trigger to mark a reference point within a record.

Input

- Name Type Description
- vi ViSession Identifies your instrument session. **vi** is obtained from the <u>niRFSA\_init</u> or the <u>niRFSA\_initWithOptions</u> functions and identifies a particular instrument session.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# niRFSA\_ConfigureDigitalEdgeAdvanceTrigger

ViStatus = niRFSA\_ConfigureDigitalEdgeAdvanceTrigger( ViSession vi, ViConstString source, ViInt32 edge);

Configures the device to wait for a digital edge Advance trigger between records.

Input				
Name	Туре	Description		
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.		
source	ViConstString	Specifies th Advance tri	e source of the digital edge for the gger.	
		PFI0	The trigger is received on PFI 0.	
		PFI1	The trigger is received on PFI 1.	
		PXI_Trig0	The trigger is received on PXI trigger line 0.	
		PXI_Trig1	The trigger is received on PXI trigger line 1.	
		PXI_Trig2	The trigger is received on PXI trigger line 2.	
		PXI_Trig3	The trigger is received on PXI trigger line 3.	
		PXI_Trig4	The trigger is received on PXI trigger line 4.	
		PXI_Trig5	The trigger is received on PXI trigger line 5.	
		PXI_Trig6	The trigger is received on PXI trigger line 6.	
		PXI_Trig7	The trigger is received on PXI trigger line 7.	
		PXI_STAR	The trigger is received on the PXI star trigger line.	
edge	Vilnt32	Specifies th	e edge to detect. You can choose	

Rising Edge or Falling Edge.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

## niRFSA\_ConfigureSoftwareEdgeAdvanceTrigger

ViStatus = niRFSA\_ConfigureSoftwareEdgeAdvanceTrigger( ViSession vi);

Configures the device to wait for a software Advance trigger between records. The device waits until you call the <u>niRFSA\_SendSoftwareEdgeTrigger</u> function to assert the trigger.

Input

- Name Type Description
- vi ViSession Identifies your instrument session. **vi** is obtained from the <u>niRFSA\_init</u> or the <u>niRFSA\_initWithOptions</u> functions and identifies a particular instrument session.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

## niRFSA\_DisableAdvanceTrigger

ViStatus = niRFSA\_DisableAdvanceTrigger( ViSession vi);

Configures the device to not wait for an Advance trigger between records of a multirecord acquisition.

Input

- Name Type Description
- vi ViSession Identifies your instrument session. **vi** is obtained from the <u>niRFSA\_init</u> or the <u>niRFSA\_initWithOptions</u> functions and identifies a particular instrument session.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

## niRFSA\_SendSoftwareEdgeTrigger

ViStatus = niRFSA\_SendSoftwareEdgeTrigger( ViSession vi, ViInt32 trigger, ViConstString triggerIdentifier);

Sends a trigger to the device when you configure a software version of a supported trigger and the device is waiting for the trigger to be sent. This function also can be used to override a hardware trigger.

This function returns an error in the following situations:

- You configure an invalid trigger
- You are in spectrum mode
- You have not previously called the <u>niRFSA\_Initiate</u> function.

NI-Scope handles other errors.

Input		
Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
trigger	Vilnt32	Specifies the software signal to send. You can send a Start, Reference, Advance, or Arm Reference trigger.
triggerldentifier	ViConstString	Specifies a particular instance of a trigger. This parameter currently is not supported.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

## niRFSA\_ExportSignal

ViStatus = niRFSA\_ExportSignal(ViSession vi, ViInt32 signal, ViConstString signalIdentifier, ViConstString outputTerminal);

Routes signals to the specified output terminal.

If you export a signal with this VI and commit the session, the signal is routed to the output terminal you specify. If you then reconfigure the signal to have a different output terminal, the previous output terminal is tristated when the session is next committed. If you change the **outputTerminal** to NIRFSA\_VAL\_DO\_NOT\_EXPORT and commit, the previous output terminal is tristated.

Any signals, except for PXI trigger lines, that are exported within a session persist after the session closes to prevent signal glitches between sessions. PXI trigger lines are always set to tristate when a session is closed. If you wish to have the terminal that the signal was exported to tristated when the session closes, first change the **outputTerminal** for the exported signal to

NIRFSA\_VAL\_DO\_NOT\_EXPORT and commit the session again before closing it.

You can also tristate all PFI lines by setting the **resetDevice** parameter in the niRFSA\_Initialize function or by using the <u>niRFSA\_reset</u> function.

In	ทเ	ıt
	ρι	π

1		
Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
signal	Vilnt32	Specifies the type of signal to route. You can choose to export the Start, Reference, and Advance triggers and the Ready for Start, Ready for Advance, Ready for Ref, End of Record, and Done events.
signalldentifier	ViConstString	Specifies a particular instance of a trigger. This parameter currently is not supported.
outputTerminal	ViConstString	Specifies the terminal where the signal will be exported. You can choose not to export any signal.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

## niRFSA\_ConfigureRefClock

ViStatus = niRFSA\_ConfigureRefClock( ViSession vi, ViConstString clockSource, ViReal64 refClockRate);

Configures the NI-RFSA device reference clock.

Input

Name Type Description

vi ViSession Identifies your instrument session. **vi** is obtained from the <u>niRFSA\_init</u> or the <u>niRFSA\_initWithOptions</u> functions and identifies a particular instrument session.

refClockSource ViConstString Specifies the reference clock source.

OnboardClock	Lock the NI-RFSA device to the NI PXI- 5600 onboard clock.
RefIn	Lock the NI-RFSA device to the external REF IN connector on the NI PXI-5600.
PXI_Clk10	Lock the NI-RFSA device to the PXI backplane clock using the NI PXI-5600. You must connect the PXI 10 MHz connector to the REF IN connector on the NI PXI-5600 front panel to use this option.

refClockRate ViReal64 Specifies the reference clock rate, expressed in Hertz. The default value is 10 MHz, which is the only currently supported value.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# niRFSA\_ConfigurePXIChassisClk10

ViStatus = niRFSA\_ConfigurePXIChassisClk10( ViSession vi, ViConstString pxiClk10Source);

Specifies the signal to drive the 10 MHz reference clock on the PXI backplane. This option can only be configured when the NI PXI-5600 is in Slot 2 of the PXI chassis.

Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
PXIClk10Source	ViConstString	Specifies the signal to drive the 10 MHz reference clock on the PXI backplane. This option can only be configured when the NI PXI-5600 is in Slot 2 of the PXI chassis.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

## niRFSA\_SetAttributeViInt32

ViStatus = niRFSA\_SetAttributeViInt32 (ViSession vi, ViConstString channelName, ViAttr attributeId, ViInt32 value);

Sets the value of a ViInt32 attribute.

You can use this low-level function to set the values of inherent IVI attributes, class-defined attributes, and instrument-specific attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled, and the currently cached value is invalid or is different than the value you specify.

NI-RFSA contains high-level functions that set most of the instrument attributes. It is best to use the high-level driver functions as much as possible. They handle order dependencies and multithread locking for you. In addition, they perform status checking only after setting all of the attributes. In contrast, when you set multiple attributes using the SetAttribute functions, the functions check the instrument status after each call.

Also, when state caching is enabled, the high-level functions that configure multiple attributes perform instrument I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant instrument I/O.

Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
channelName	ViConstString	If the attribute is channel based, this parameter specifies the channel to which the attribute applies. If the attribute is not channel based, set this parameter to "" (empty string) or VI_NULL.
attributeID	ViAttr	Pass the ID of an attribute.
attributeValue	Vilnt32	Pass the value to which you want to set the attribute.
		Note Some of the values might not be valid depending on the current settings of the instrument session.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

## niRFSA\_SetAttributeViInt64

ViStatus = niRFSA\_SetAttributeViInt64 (ViSession vi, ViConstString channelName, ViAttr attributeId, ViInt64 value);

Sets the value of a ViInt64 attribute.

You can use this low-level function to set the values of inherent IVI attributes, class-defined attributes, and instrument-specific attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled, and the currently cached value is invalid or is different than the value you specify.

NI-RFSA contains high-level functions that set most of the instrument attributes. It is best to use the high-level driver functions as much as possible. They handle order dependencies and multithread locking for you. In addition, they perform status checking only after setting all of the attributes. In contrast, when you set multiple attributes using the SetAttribute functions, the functions check the instrument status after each call.

Also, when state caching is enabled, the high-level functions that configure multiple attributes perform instrument I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant instrument I/O.

Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
channelName	ViConstString	If the attribute is channel based, this parameter specifies the channel to which the attribute applies. If the attribute is not channel based, set this parameter to "" (empty string) or VI_NULL.
attributeID	ViAttr	Pass the ID of an attribute.
attributeValue	Vilnt32	Pass the value to which you want to set the attribute.
		Note Some of the values might not be valid depending on the current settings of the instrument session.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

## niRFSA\_SetAttributeViReal64

ViStatus = niRFSA\_SetAttributeViReal64 (ViSession vi, ViConstString channelName, ViAttr attributeId, ViReal64 value);

Sets the value of a ViReal64 attribute.

You can use this low-level function to set the values of inherent IVI attributes, class-defined attributes, and instrument-specific attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled, and the currently cached value is invalid or is different than the value you specify.

NI-RFSA contains high-level functions that set most of the instrument attributes. It is best to use the high-level driver functions as much as possible. They handle order dependencies and multithread locking for you. In addition, they perform status checking only after setting all of the attributes. In contrast, when you set multiple attributes using the SetAttribute functions, the functions check the instrument status after each call.

Also, when state caching is enabled, the high-level functions that configure multiple attributes perform instrument I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant instrument I/O.

Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
channelName	ViConstString	If the attribute is channel based, this parameter specifies the channel to which the attribute applies. If the attribute is not channel based, set this parameter to "" (empty string) or VI_NULL.
attributeID	ViAttr	Pass the ID of an attribute.
attributeValue	Vilnt32	Pass the value to which you want to set the attribute.
		Note Some of the values might not be valid depending on the current settings of the instrument session.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

## niRFSA\_SetAttributeViString

ViStatus = niRFSA\_SetAttributeViString (ViSession vi, ViConstString channelName, ViAttr attributeId, ViString value);

Sets the value of a ViString attribute.

You can use this low-level function to set the values of inherent IVI attributes, class-defined attributes, and instrument-specific attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled, and the currently cached value is invalid or is different than the value you specify.

NI-RFSA contains high-level functions that set most of the instrument attributes. It is best to use the high-level driver functions as much as possible. They handle order dependencies and multithread locking for you. In addition, they perform status checking only after setting all of the attributes. In contrast, when you set multiple attributes using the SetAttribute functions, the functions check the instrument status after each call.

Also, when state caching is enabled, the high-level functions that configure multiple attributes perform instrument I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant instrument I/O.

Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
channelName	ViConstString	If the attribute is channel based, this parameter specifies the channel to which the attribute applies. If the attribute is not channel based, set this parameter to "" (empty string) or VI_NULL.
attributeID	ViAttr	Pass the ID of an attribute.
attributeValue	Vilnt32	Pass the value to which you want to set the attribute.
		Note Some of the values might not be valid depending on the current settings of the instrument session.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

## niRFSA\_SetAttributeViBoolean

ViStatus = niRFSA\_SetAttributeViBoolean (ViSession vi, ViConstString channelName, ViAttr attributeId, ViBoolean value);

Sets the value of a ViBoolean attribute.

You can use this low-level function to set the values of inherent IVI attributes, class-defined attributes, and instrument-specific attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled, and the currently cached value is invalid or is different than the value you specify.

NI-RFSA contains high-level functions that set most of the instrument attributes. It is best to use the high-level driver functions as much as possible. They handle order dependencies and multithread locking for you. In addition, they perform status checking only after setting all of the attributes. In contrast, when you set multiple attributes using the SetAttribute functions, the functions check the instrument status after each call.

Also, when state caching is enabled, the high-level functions that configure multiple attributes perform instrument I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant instrument I/O.

Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
channelName	ViConstString	If the attribute is channel based, this parameter specifies the channel to which the attribute applies. If the attribute is not channel based, set this parameter to "" (empty string) or VI_NULL.
attributeID	ViAttr	Pass the ID of an attribute.
attributeValue	Vilnt32	Pass the value to which you want to set the attribute.
		Note Some of the values might not be valid depending on the current settings of the instrument session.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

## niRFSA\_SetAttributeViSession

ViStatus = niRFSA\_SetAttributeViSession (ViSession vi, ViConstString channelName, ViAttr attributeId, ViSession value);

Sets the value of a ViSession attribute.

You can use this low-level function to set the values of inherent IVI attributes, class-defined attributes, and instrument-specific attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled, and the currently cached value is invalid or is different than the value you specify.

NI-RFSA contains high-level functions that set most of the instrument attributes. It is best to use the high-level driver functions as much as possible. They handle order dependencies and multithread locking for you. In addition, they perform status checking only after setting all of the attributes. In contrast, when you set multiple attributes using the SetAttribute functions, the functions check the instrument status after each call.

Also, when state caching is enabled, the high-level functions that configure multiple attributes perform instrument I/O only for the attributes whose value you change. Thus, you can safely call the high-level functions without the penalty of redundant instrument I/O.

Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
channelName	ViConstString	If the attribute is channel based, this parameter specifies the channel to which the attribute applies. If the attribute is not channel based, set this parameter to "" (empty string) or VI_NULL.
attributeID	ViAttr	Pass the ID of an attribute.
attributeValue	Vilnt32	Pass the value to which you want to set the attribute.
		Note Some of the values might not be valid depending on the current settings of the instrument session.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

## niRFSA\_GetAttributeViInt32

ViStatus = niRFSA\_GetAttributeViInt32 (ViSession vi, ViConstString channelName, ViAttr attributeId, ViInt32 \*value);

Queries the value of a ViInt32 attribute.

You can use this low-level function to get the values of inherent IVI attributes, class-defined attributes, and instrument-specific attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled, and the currently cached value is invalid.

Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
channelName	ViConstString	If the attribute is channel based, this parameter specifies the channel to which the attribute applies. If the attribute is not channel based, set this parameter to "" (empty string) or VI_NULL.
attributeID	ViAttr	Pass the ID of an attribute.
Output		
Name	Туре	Description
attributeValue	Vilnt32*	Returns the current value of the attribute. Pass the address of a ViInt32 variable.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

0	Success	
Positive Values	Warnings	
Negative Values	Errors	

## niRFSA\_GetAttributeViInt64

ViStatus = niRFSA\_GetAttributeViInt64 (ViSession vi, ViConstString channelName, ViAttr attributeId, ViInt64 \*value);

Queries the value of a ViInt64 attribute.

You can use this low-level function to get the values of inherent IVI attributes, class-defined attributes, and instrument-specific attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled, and the currently cached value is invalid.

Input

Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
channelName	ViConstString	If the attribute is channel based, this parameter specifies the channel to which the attribute applies. If the attribute is not channel based, set this parameter to "" (empty string) or VI_NULL.
attributeID	ViAttr	Pass the ID of an attribute.
Output		
Name	Туре	Description
attributeValue	Vilnt32*	Returns the current value of the attribute. Pass the address of a ViInt32 variable.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

## niRFSA\_GetAttributeViReal64

ViStatus = niRFSA\_GetAttributeViReal64 (ViSession vi, ViConstString channelName, ViAttr attributeId, ViReal64 \*value);

Queries the value of a ViReal64 attribute.

You can use this low-level function to get the values of inherent IVI attributes, class-defined attributes, and instrument-specific attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled, and the currently cached value is invalid.

Input

Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
channelName	ViConstString	If the attribute is channel based, this parameter specifies the channel to which the attribute applies. If the attribute is not channel based, set this parameter to "" (empty string) or VI_NULL.
attributeID	ViAttr	Pass the ID of an attribute.
Output		
Name	Туре	Description
attributeValue	Vilnt32*	Returns the current value of the attribute. Pass the address of a ViInt32 variable.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# niRFSA\_GetAttributeViString

ViStatus = niRFSA\_GetAttributeViString (ViSession vi, ViConstString channelName, ViAttr attributeId, ViString \*value);

Queries the value of a ViString attribute.

You can use this low-level function to get the values of inherent IVI attributes, class-defined attributes, and instrument-specific attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled, and the currently cached value is invalid.

You must provide a ViString (ViChar array) to serve as a buffer for the value. You pass the number of bytes in the buffer as the **bufferSize** parameter. If the current value of the attribute, including the terminating NULL byte, is larger than the size you indicate in the **bufferSize** parameter, the function copies buffer size minus 1 bytes into the buffer, places an ASCII NULL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the buffer size is 4, the function places "123" into the buffer and returns 7.

If you want to call this function just to get the required buffer size, you can pass 0 for the **bufferSize** and VI\_NULL for the **attributeValue** buffer.

If you want the function to fill in the buffer regardless of the number of bytes in the value, pass a negative number for the buffer size parameter.

Input

mpat		
Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
channelName	ViConstString	If the attribute is channel based, this parameter specifies the channel to which the attribute applies. If the attribute is not channel based, set this parameter to "" (empty string) or VI_NULL.
attributeID	ViAttr	Pass the ID of an attribute.
bufferSize	Vilnt32	Pass the number of bytes in the ViChar buffer you specify for the attribute value parameter.
		If you pass a negative number, the function copies the value to the buffer regardless of the number of bytes in the value.
		If you pass 0, you can pass VI_NULL for the attribute value buffer parameter.
Output		
Name	Туре	Description
attributeValue	Vilnt32*	Returns the current value of the attribute. Pass the address of a ViInt32 variable.

Name	Туре	Description	
statusOrRequiredSize	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the niRFSA_error_message function. To obtain additional information about the error condition, call the niRFSA_GetError function. To clear the error information from the driver, call the niRFSA_ClearError function.	Meaning
		The general meaning of the status code is as follows:	
		Value	
0	Success		

0	Success
Positive Values	Warnings
Negative Values	Errors

# niRFSA\_GetAttributeViBoolean

ViStatus = niRFSA\_GetAttributeViBoolean (ViSession vi, ViConstString channelName, ViAttr attributeId, ViBoolean \*value);

Queries the value of a ViBoolean attribute.

You can use this low-level function to get the values of inherent IVI attributes, class-defined attributes, and instrument-specific attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled, and the currently cached value is invalid.

Input

1		
Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
channelName	ViConstString	If the attribute is channel based, this parameter specifies the channel to which the attribute applies. If the attribute is not channel based, set this parameter to "" (empty string) or VI_NULL.
attributeID	ViAttr	Pass the ID of an attribute.
Output		
Name	Туре	Description
attributeValue	Vilnt32*	Returns the current value of the attribute. Pass the address of a ViInt32 variable.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

## niRFSA\_GetAttributeViSession

ViStatus = niRFSA\_GetAttributeViSession (ViSession vi, ViConstString channelName, ViAttr attributeId, ViSession \*value);

Queries the value of a ViSession attribute.

You can use this low-level function to get the values of inherent IVI attributes, class-defined attributes, and instrument-specific attributes. If the attribute represents an instrument state, this function performs instrument I/O in the following cases:

- State caching is disabled for the entire session or for the particular attribute.
- State caching is enabled, and the currently cached value is invalid.

Input

Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
channelName	ViConstString	If the attribute is channel based, this parameter specifies the channel to which the attribute applies. If the attribute is not channel based, set this parameter to "" (empty string) or VI_NULL.
attributeID	ViAttr	Pass the ID of an attribute.
Output		
Name	Туре	Description
attributeValue	Vilnt32*	Returns the current value of the attribute. Pass the address of a ViInt32 variable.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# niRFSA\_ResetAttribute

ViStatus = niRFSA\_ResetAttribute (ViSession vi, ViConstString channelName, ViAttr attributeId);

Resets the attribute to its default value.

Input		
Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
channelName	ViConstString	If the attribute is channel based, this parameter specifies the channel to which the attribute applies. If the attribute is not channel based, set this parameter to "" (empty string) or VI_NULL.
attributeID	ViAttr	Pass the ID of an attribute.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

## niRFSA\_Initiate

ViStatus = niRFSA\_Initiate( ViSession vi);

Starts an IQ acquisition. You may use this function in conjunction with the NI-RFSA fetch IQ functions to retrieve acquired IQ data, or use the NI-RFSA read IQ functions to both initiate the acquisition and retrieve IQ data at one time.

Input

Name Type Description

vi ViSession Passes a reference to your instrument session to the next function. **vi** is obtained from the <u>niRFSA\_init</u> or the <u>niRFSA\_initWithOptions</u> functions and identifies a particular instrument session.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# niRFSA\_FetchIQSingleRecordComplexI16

ViStatus = niRFSA\_FetchIQSingleRecordComplexI16(ViSession vi, ViConstString channelList, ViInt64 recordNumber, ViInt64 numberOfSamples, ViReal64 timeout, NIComplexI16\* data, niRFSA\_wfmInfo\* wfmInfo);

Fetches binary IQ data from a single record in an acquisition. The fetch transfers acquired waveform data from device memory to PC memory. The data was acquired to onboard memory previously by the hardware after it was initiated.

This function is not necessary if you use the <u>niRFSA\_ReadIQSingleRecordComplexF64</u> function, as the fetch is performed as part of that function.

Input

, Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
channelList	ViConstString	Specifies channels from which to fetch data.
recordNumber	Vilnt64	Specifies the record to fetch. Record numbers are zero- indexed.
numberOfSamples	Vilnt64	Specifies the number of samples to fetch.
timeout	ViReal64	Specifies in seconds the time allotted for the function to complete before returning a timeout error. A value of -1 specifies the function waits until all data is available. A value of 0 specifies the function returns available data immediately.
Output		
Name	Туре	Description
data	NIComplexI16*	Returns the acquired waveform.
wfmInfo	niRFSA_wfmInfo*	Contains the absolute and relative timestamp for the operation, the dt, and the actual number of samples read.
		The following list provides more information about each of these

properties:

- absolute timestamp specifies the timestamp in seconds of the first fetched sample that is comparable between records and acquisitions.
- relative timestamp returns a timestamp that corresponds to the difference in seconds between the first sample returned and the Reference trigger location.
- dt—returns the time interval between data points in the acquired signal. The IQ data sampling rate is the reciprocal of this value.
- actual samples read returns an integer representing the number of samples in the waveform.
- offset—specifies the offset to scale data in mx+b form.
- gain—specifies the gain to scale data in mx+b form.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# niRFSA\_FetchIQSingleRecordComplexF64

ViStatus = niRFSA\_FetchIQSingleRecordComplexF64(ViSession vi, ViConstString channelList, ViInt64 recordNumber, ViInt64 numberOfSamples, ViReal64 timeout, NIComplexNumber\* data, niRFSA\_wfmInfo\* wfmInfo);

Fetches IQ data from a single record in an acquisition. The fetch transfers acquired waveform data from device memory to PC memory. The data was acquired to onboard memory previously by the hardware after it was initiated.

This function is not necessary if you use the <u>niRFSA\_ReadIQSingleRecordComplexF64</u> function, as the fetch is performed as part of that function.

Input

mpat		
Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
channelList	ViConstString	Specifies channels from which to fetch data.
recordNumber	Vilnt64	Specifies the record to fetch. Record numbers are zero- indexed.
numberOfSamples	Vilnt64	Specifies the number of samples to fetch.
timeout	ViReal64	Specifies in seconds the time allotted for the function to complete before returning a timeout error. A value of -1 specifies the function waits until all data is available. A value of 0 specifies the function returns available data immediately.
Output		
Name	Туре	Description
data	NIComplexNumber*	Returns the acquired waveform.
wfmInfo	niRFSA_wfmInfo*	Returns the absolute and relative timestamp for the operation, the dt, and the actual number of samples read.

The following list provides more information about each of these properties:

- absolute timestamp specifies the timestamp in seconds of the first fetched sample that is comparable between records and acquisitions.
- relative timestamp returns a timestamp that corresponds to the difference in seconds between the first sample returned and the Reference trigger location.
- dt—returns the time interval between data points in the acquired signal. The IQ data sampling rate is the reciprocal of this value.
- actual samples read returns an integer representing the number of samples in the waveform.
- offset—specifies the offset to scale data in mx+b form.
- gain—specifies the gain to scale data in mx+b form.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

0	Success	
Positive Values	Warnings	
Negative Values	Errors	

# niRFSA\_ReadIQSingleRecordComplexF64

ViStatus = niRFSA\_ReadIQSingleRecordComplexF64( ViSession vi, ViConstString channelList, ViReal64 timeout, NIComplexNumber\* data, ViInt64 dataArraySize, niRFSA\_wfmInfo\* wfmInfo);

Initiates an acquisition and fetches a single IQ data record. Do not use this function if you have configured the device to continuously acquire data samples or to acquire multiple records.

Input

mput		
Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> function: and identifies a particular instrument session.
channelList	ViConstString	Specifies channels from which to fetch data.
timeout	ViReal64	Specifies in seconds the time allottec for the function to complete before returning a timeout error. A value of - specifies the function waits until all data is available.
dataArraySize	Vilnt32	Specifies the size of the array for the <b>data</b> parameter. The array needs to be at least as large as the number of samples configured in the <u>niRFSA_ConfigureNumberOfSample</u> function.
Output		
Name	Туре	Description
data	NIComplexNumber*	Returns the acquired waveform.
wfmInfo*	niRFSA_wfmInfo	Returns additional information about the <b>data</b> array.
		The following list provides more information about each of these properties: • absolute timestamp—specifies
		the timestamp in seconds of the first fetched sample that is comparable between records

and acquisitions.

- relative timestamp—returns a timestamp that corresponds to the difference in seconds between the first sample returned and the Reference trigger location.
- dt—returns the time interval between data points in the acquired signal. The IQ data sampling rate is the reciprocal of this value.
- actual samples read—returns an integer representing the number of samples in the waveform.
- offset—specifies the offset to scale data in mx+b form.
- gain—specifies the gain to scale data in mx+b form.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# niRFSA\_FetchIQMultiRecordComplexI16

ViStatus = niRFSA\_FetchIQMultiRecordComplexI16 (ViSession vi, ViConstString channelList, ViInt64 startingRecord, ViInt64 numberOfRecords, ViInt64 numberOfSamples, ViReal64 timeout, NIComplexI16\* data, niRFSA\_wfmInfo\* wfmInfo);

Fetches binary IQ data from multiple records in an acquisition. Fetching transfers acquired waveform data from device memory to PC memory. The data was acquired to onboard memory previously by the hardware after it was initiated.

This function is not necessary if you use the <u>niRFSA\_ReadIQSingleRecordComplexF64</u> function, as the fetch is performed as part of that function.

Input

mpat		
Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
channelList	ViConstString	Specifies channels from which to fetch data.
startingRecord	Vilnt64	Specifies the first record to retrieve.
numberOfRecords	Vilnt64	Specifies the number of records to fetch. A value of -1 specifies that NI-RFSA fetches all records in an acquisition starting with the record specified by <b>startingRecord</b> . Record numbers are zero-indexed.
numberOfSamples	Vilnt64	Specifies the number of samples to fetch.
timeout	ViReal64	Specifies in seconds the time allotted for the function to complete before returning a timeout error. A value of -1 specifies the function waits until all data is available. A value of 0 specifies the function returns available data immediately.
Output		
Name	Туре	Description
data	NIComplexI16*	Returns the acquired waveform for each record fetched. The

		waveforms are written sequentially in the array. Allocate an array at least as large as <b>numberOfSamples</b> times <b>numberOfRecords</b> for this parameter.
wfmInfo	niRFSA_wfmInfo*	Returns an array of structures containing information about each record fetched. Each structure contains the absolute and relative timestamp, the dt, and the actual number of samples read for the corresponding record.
		<ul> <li>The following list provides more information about each of these properties:</li> <li>absolute timestamp— specifies the timestamp in seconds of the first fetched sample that is comparable between records and acquisitions.</li> <li>relative timestamp— returns a timestamp that corresponds to the difference in seconds between the first sample returned and the Reference trigger location.</li> <li>dt—returns the time interval between data points in the acquired signal. The IQ data sampling rate is the reciprocal of this value.</li> </ul>

- actual samples read returns an integer representing the number of samples in the waveform.
- offset—specifies the offset to scale data in mx+b form.
- gain—specifies the gain to scale data in mx+b form.
- Note Allocate an array of structures at least as large as numberOfRecords for this parameter.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# niRFSA\_FetchIQMultiRecordComplexF64

ViStatus = niRFSA\_FetchIQMultiRecordComplexF64(ViSession vi, ViConstString channelList, ViInt64 startingRecord, ViInt64 numberOfRecords, ViInt64 numberOfSamples, ViReal64 timeout, NIComplexNumber\* data, niRFSA\_wfmInfo\* wfmInfo);

Fetches IQ data from multiple records in an acquisition. A fetch transfers acquired waveform data from device memory to PC memory. The data was acquired to onboard memory previously by the hardware after it was initiated.

This function is not necessary if you use the <u>niRFSA\_ReadIQSingleRecordComplexF64</u> function, as the fetch is performed as part of that function.

Input

mpat		
Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
channelList	ViConstString	Specifies channels from which to fetch data.
startingRecord	Vilnt64	Specifies the first record to retrieve.
numberOfRecords	Vilnt64	Specifies the number of records to fetch. A value of -1 specifies that NI-RFSA fetches all records in an acquisition starting with the record specified by <b>startingRecord</b> . Record numbers are zero- indexed.
numberOfSamples	Vilnt64	Specifies the number of samples per record.
timeout	ViReal64	Specifies in seconds the time allotted for the function to complete before returning a timeout error. A value of -1 specifies the function waits until all data is available. A value of 0 specifies the function returns available data immediately.
Output		
Name	Туре	Description

data	NIComplexNumber*	Returns the acquired waveform for each record fetched. The waveforms are written sequentially in the array. Allocate an array at least as large as <b>numberOfSamples</b> times <b>numberOfRecords</b> for this parameter.
wfmInfo	niRFSA_wfmInfo*	<ul> <li>Returns an array of structures containing information about each record fetched. Each structure contains the absolute and relative timestamp, the dt, and the actual number of samples read for the corresponding record.</li> <li>The following list provides more information about each of these properties: <ul> <li>absolute timestamp— specifies the timestamp in seconds of the first fetched sample that is comparable between records and acquisitions.</li> <li>relative timestamp— returns a timestamp that corresponds to the difference in seconds between the first sample returned and the Reference trigger location.</li> <li>dt—returns the time interval between data points in the acquired signal. The IQ data</li> </ul> </li> </ul>

sampling rate is the reciprocal of this value.

- actual samples read returns an integer representing the number of samples in the waveform.
- offset—specifies the offset to scale data in mx+b form.
- gain—specifies the gain to scale data in mx+b form.
- Note Allocate an array of structures at least as large as numberOfRecords for this parameter.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

## niRFSA\_Abort

ViStatus = niRFSA\_Abort( ViSession vi);

Stops an acquisition previously started with the <u>niRFSA\_Initiate</u> function. Unless you want to stop an acquisition before it is complete or you are continuously acquiring data, calling this function is optional.

Input

- Name Type Description
- vi ViSession Identifies your instrument session. **vi** is obtained from the <u>niRFSA\_init</u> or the <u>niRFSA\_initWithOptions</u> functions and identifies a particular instrument session.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	

## niRFSA\_ReadPowerSpectrumF64

ViStatus = niRFSA\_ReadPowerSpectrumF64( ViSession vi, ViConstString channelList, ViReal64 timeout, ViReal64 powerSpectrumData[], ViInt32 dataArraySize, niRFSA\_spectrumInfo\* spectrumInfo);

Initiates a spectrum acquisition and returns power spectrum data.

Input

	<b>T</b>	Description
Name	Туре	Description
vi	ViSession	Identifies your instrument : is obtained from the <u>niRFS</u> the <u>niRFSA_initWithOptior</u> functions and identifies a p instrument session.
channelList	ViConstString	Specifies channels from w fetch data.
timeout	ViReal64	Specifies in seconds the ti allotted for the function to before returning a timeout value of -1 specifies the fu waits until all data is availa
dataArraySize	Vilnt64	Specifies the size of the air specify for the <b>powerSpec</b> parameter. Use the <u>niRFSA_GetNumberOfSpe</u> function to learn the array need to allocate. The array at least as large as the nuir spectral lines that NI-RFSP computes for the power sp
Output		
Name	Туре	Description
powerSpectrumData[]	ViReal64	Returns power spectrum c
spectrumInfo	niRFSA_spectrumInfo*	Returns additional informa the <b>powerSpectrumData</b> This information includes t frequency corresponding t element in the array, the fr increment between adjace elements, and the number

spectral lines the function

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	

# niRFSA\_GetNumberOfSpectralLines

ViStatus = niRFSA\_GetNumberOfSpectralLines( ViSession vi, ViConstString channelList, ViInt32\* numberOfSpectralLines);

Returns the number of spectral lines that NI-RFSA will compute with the current power spectrum configuration.

Input Name Туре Description ViSession Identifies your instrument session vi from the niRFSA init or the niRFSA initWithOptions functions particular instrument session. channelList ViConstString Identifies channels to apply setting VI\_NULL to specify all channels. Output Name Туре Description

numberOfSpectralLines ViInt32\*

Returns the value of the <u>NIRFSA\_ATTR\_NUMBER\_OF\_S</u> attribute.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	

# niRFSA\_CheckAcquisitionStatus

ViStatus = niRFSA\_CheckAcquisitionStatus( ViSession vi, ViBoolean\* isDone);

Checks the status of the acquisition. Use this VI to check for any errors that may occur during signal acquisition or to check whether the device has completed the acquisition operation.

Input		
Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
Output		
Name	Туре	Description
isDone	ViBoolean*	Returns VI_TRUE when signal acquisition is complete.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	

# niRFSA\_reset

ViStatus = niRFSA\_reset( ViSession vi);

Resets the device to a known initialization state.

Input

- Name Type Description
- vi ViSession Identifies your instrument session. **vi** is obtained from the <u>niRFSA\_init</u> or the <u>niRFSA\_initWithOptions</u> functions and identifies a particular instrument session.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

## niRFSA\_Commit

ViStatus = niRFSA\_Commit( ViSession vi);

Commits settings to hardware. Calling this function is optional. Settings are automatically committed to hardware when you call the <u>niRFSA\_Initiate</u>, or a read IQ or read power spectrum function.

Input

- Name Type Description
- vi ViSession Identifies your instrument session. **vi** is obtained from the <u>niRFSA\_init</u> or the <u>niRFSA\_initWithOptions</u> functions and identifies a particular instrument session.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

## niRFSA\_self\_test

ViStatus = niRFSA\_self\_test( ViSession vi, ViInt16 \*testResult,/\*Output\*/ ViChar testMessage[]);

Performs a self-test on the NI-RFSA device and returns the test result. This function performs a simple series of tests ensuring the NI-RFSA device is powered up and responding.

Input		
Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
Output		
Name	Туре	Description
testResult	Vilnt16*	Returns the value from the device self-test. Zero means success.
testMessage[]	ViChar	Returns the self-test response string from the NI-RFSA device.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# niRFSA\_PerformThermalCorrection

ViStatus = niRFSA\_PerformThermalCorrection( ViSession vi);

Measurements are affected by changes in temperature. NI-RFSA internally acquires the temperature every time you initiate an acquisition. If you are performing a very long continuous acquisition, National Instruments recommends calling this function once every 10 minutes in a stable temperature environment to periodically update temperature calibration.

Input

- Name Type Description
- vi ViSession Identifies your instrument session. **vi** is obtained from the <u>niRFSA\_init</u> or the <u>niRFSA\_initWithOptions</u> functions and identifies a particular instrument session.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

## niRFSA\_GetFetchBacklog

ViStatus = niRFSA\_GetFetchBacklog( ViSession vi, ViConstString channelList, ViInt64 recordNumber, ViInt64\* backlog);

Returns the number of points acquired that have not been fetched yet.

Input

Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
channelList	ViConstString	Identifies channels to apply settings. Use "" or VI_NULL to specify all channels.
recordNumber	Vilnt64	Specifies the record from which to read the backlog.
Output		
Name	Туре	Description
backlog	Vilnt64*	Returns the number of samples available to read for the requested record.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# niRFSA\_revision\_query

ViStatus = niRFSA\_revision\_query( ViSession vi, ViChar driverRev[], ViChar instrRev[]);

Returns the revision numbers of the NI-RFSA driver.

Input		
Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained fro the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> func and identifies a particular instrument session.
Output		
Name	Туре	Description
driverRev	ViChar[]	Returns the value of <a href="https://www.watering.com">NIRFSA_ATTR_SPECIFIC_DRIVER_REVISION</a> in form of a string.
		You must pass a ViChar array with at least 256 byte
instRev	ViChar[]	Returns the value of <u>NIRFSA_ATTR_INSTRUMENT_FIRMWARE_REVI</u> in the form of a string.
		You must pass a ViChar array with at least 256 byte

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# niRFSA\_GetSpectralInfoForSMT

ViStatus = niRFSA\_GetSpectralInfoForSMT( ViSession vi, SmtSpectrumInfo\* spectrumInfo);

Returns a cluster containing information about the power spectrum NI-RFSA computes that is needed by the Spectral Measurements Toolkit (SMT).

Input		
Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
Output		
Name	Туре	Description
spectrumInfo	SmtSpectrumInfo*	Returns a cluster containing information about the power spectrum NI-RFSA computes that is needed by the Spectral Measurements Toolkit (SMT).

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

## niRFSA\_error\_message

ViStatus = niRFSA\_error\_message( ViSession vi, ViStatus statusCode, ViChar message[]);

Converts a status code returned by an NI-RFSA function into a user-readable string.

Input		
Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
errorCode	ViStatus	Pass the status parameter that is returned from any NI-RFSA function. The default value is 0 (VI_SUCCESS).
Output		
Name	Туре	Description
errorMessage	ViChar[]	Returns the user-readable message string that corresponds to the status code you specify.
		You must pass a ViChar array with at least 256 bytes to this parameter.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

## niRFSA\_GetError

ViStatus = niRFSA\_GetError( ViSession vi, ViStatus \*errorCode, ViInt32 errorDescriptionBufferSize, ViChar errorDescription[]);

Retrieves and then clears the IVI error information for the session or the current execution thread.



**Note** If the **errorDescriptionBufferSize** parameter is 0, this function does not clear the error information. By passing 0 for the buffer size, you can determine the buffer size required to get the entire error description string and then call this function again with a sufficiently large buffer.

If you specify a valid IVI session for the **vi** parameter, this function retrieves and then clears the error information for the session. If you pass VI\_NULL for **vi**, this function retrieves and then clears the error information for the current execution thread. If **vi** is an invalid session, this function does nothing and returns an error. Normally, the error information describes the first error that occurred since the user last called this function or <u>niRFSA\_ClearError</u>.

Decorintion
Description
Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
Pass the number of bytes in the ViChar array you specify for the description parameter.
If the error description, including the terminating NULL byte, contains more bytes than you indicate in this parameter, the function copies buffer size - 1 bytes into the buffer, places an ASCII NULL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the buffer size is 4, the function places "123" into the buffer and returns 7. If you pass a negative number, the function copies the value to the buffer regardless of the number of bytes in the value. If you pass 0, you can pass VI_NULL for the <b>errorDescription</b> parameter. Default Value: None

Name errorCode	Type ViStatus*	Description Returns the error code for the session or execution thread. If you pass 0 for the <b>errorDescriptionBufferSize</b> parameter, you can pass VI_NULL for this parameter.
errorDescription	ViChar	Returns the error description for the IVI session or execution thread.
		If there is no description, this function returns an empty string. The buffer must contain at least as many elements as the value you specify with the buffer size parameter. If the error description, including the terminating NULL byte, contains more bytes than you indicate with the buffer size parameter, the function copies buffer size - 1 bytes into the buffer, places an ASCII NULL byte at the end of the buffer, and returns the buffer size you must pass to get the entire value. For example, if the value is "123456" and the buffer size is 4, the function places "123" into the buffer and returns 7. If you pass 0 for the buffer size, you can pass VI_NULL for this parameter.

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

0	Success	
Positive Values	Warnings	
Negative Values	Errors	

## niRFSA\_ClearError

ViStatus = niRFSA\_ClearError( ViSession vi);

#### Purpose

Clears the error information associated with the session. If you pass VI\_NULL for the **vi** parameter, the niRFSA\_ClearError function clears the error information for the current execution thread.



**Note** niRFSA\_GetError clears the error information after it is retrieved. A call to <u>niRFSA\_ClearError</u> is only necessary when a call to <u>niRFSA\_GetError</u> is not used to retrieve error information.

The error information includes a primary error, secondary error, and an error elaboration string.

#### Parameters

Input

- Name Type Description
- vi ViSession Identifies your instrument session. **vi** is obtained from the <u>niRFSA\_init</u> or the <u>niRFSA\_initWithOptions</u> functions and identifies a particular instrument session.

### **Return Value**

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# niRFSA\_LockSession

ViStatus = niRFSA\_LockSession( ViSession vi, ViBoolean \*callerHasLock)

#### Purpose

Obtains a multithread lock on the instrument session. Before doing so, this function waits until all other execution threads have released their locks on the instrument session.

Other threads might have obtained a lock on this session in the following ways:

- Your application already called the niRFSA\_LockSession function.
- A call to NI-RFSA locked the session.

After the call to this function returns successfully, no other threads can access the instrument session until you call the <u>niRFSA\_UnlockSession</u> function. Use niRFSA\_LockSession and niRFSA\_UnlockSession around a sequence of calls to NI-RFSA functions if you require that the NI-RFSA device retain its settings through the end of the sequence.

You can safely make nested calls to the niRFSA\_LockSession function within the same thread. To completely unlock the session, balance each call to niRFSA\_LockSession with a call to niRFSA\_UnlockSession. If, however, you use the **callerHasLock** parameter in all calls to niRFSA\_LockSession and niRFSA\_UnlockSession within a function, the IVI Library locks the session only once within the function regardless of the number of calls you make to niRFSA\_LockSession. Locking the session only once allows you to call niRFSA\_UnlockSession just once at the end of the function.

#### Parameters

Input

mpat		
Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
Output		
Name	Туре	Description
callerHasLock	ViBoolean*	This parameter serves as a convenience. If you do not want to use this parameter, pass VI_NULL.
		Use this parameter in complex functions to keep track of whether you obtain a lock and therefore need to unlock the session. Pass the address of a local ViBoolean variable. In the declaration of the local variable, initialize it to VI_FALSE. Pass the address of the same local variable to any other calls you make to this function or niRFSA_UnlockSession in the same function.
		<ul> <li>niRFSA_LockSession and niRFSA_UnlockSession each inspect the current value and take the following actions:</li> <li>If the value is VI_TRUE, niRFSA_LockSession does not lock the session again. If the value is VI_FALSE, niRFSA_LockSession obtains the lock and sets the value of the parameter to VI_TRUE.</li> <li>If the value is VI_FALSE, niRFSA_UnlockSession does not</li> </ul>

attempt to unlock the session. If the value is VI\_TRUE,

niRFSA\_UnlockSession releases the lock and sets the value of the parameter to VI\_FALSE.

Thus, you can call niRFSA\_UnlockSession at the end of your function regardless of whether you actually have the lock.

### **Return Value**

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# niRFSA\_UnlockSession

ViStatus = niRFSA\_UnlockSession( ViSession vi, ViBoolean \*callerHasLock);

#### Purpose

Releases a lock obtained on an NI-RFSA device session by calling the <u>niRFSA\_LockSession</u> function.

#### Parameters

Input

mpat		
Name	Туре	Description
vi	ViSession	Identifies your instrument session. <b>vi</b> is obtained from the <u>niRFSA_init</u> or the <u>niRFSA_initWithOptions</u> functions and identifies a particular instrument session.
Output		
Name	Туре	Description
callerHasLock	ViBoolean*	This parameter serves as a convenience. If you do not want to use this parameter, pass VI_NULL.
		Use this parameter in complex functions to keep track of whether you obtain a lock and therefore need to unlock the session. Pass the address of a local ViBoolean variable. In the declaration of the local variable, initialize it to VI_FALSE. Pass the address of the same local variable to any other calls you make to this function or niRFSA_UnlockSession in the same function.
		<ul> <li>niRFSA_LockSession and niRFSA_UnlockSession each inspect the current value and take the following actions:</li> <li>If the value is VI_TRUE, niRFSA_LockSession does not lock the session again. If the value is VI_FALSE, niRFSA_LockSession obtains the lock and sets the value of the parameter to VI_TRUE.</li> <li>If the value is VI_FALSE, niRFSA_UnlockSession does not</li> </ul>

attempt to unlock the session. If the value is VI\_TRUE,

niRFSa\_UnlockSession releases the lock and sets the value of the parameter to VI\_FALSE.

Thus, you can call niRFSA\_UnlockSession at the end of your function regardless of whether you actually have the lock.

### **Return Value**

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# niRFSA\_close

ViStatus = niRFSA\_close(ViSession vi);

### Purpose

Closes the session to the device.

#### Parameters

Input

- Name Type Description
- vi ViSession Identifies your instrument session. **vi** is obtained from the <u>niRFSA\_init</u> or the <u>niRFSA\_initWithOptions</u> functions and identifies a particular instrument session.

### **Return Value**

Name	Туре	Description	
status	ViStatus	Returns the status code of this operation. The status code either indicates success or describes an error or warning condition. You examine the status code from each call to an instrument driver function to determine if an error occurred.	
		To obtain a text description of the status code, call the <u>niRFSA_error_message</u> function. To obtain additional information about the error condition, call the <u>niRFSA_GetError</u> function. To clear the error information from the driver, call the <u>niRFSA_ClearError</u> function.	Meaning
		The general meaning of the status code is as follows:	
		Value	

Success	
Warnings	
Errors	
	Warnings

# NIRFSA\_ATTR\_CENTER\_FREQUENCY

#### **Specific Attribute**

#### Data type Access High Level Functions

ViReal64 R/W <u>niRFSA\_ConfigureSpectrumFrequencyCenterSpan</u>

Specifies the center frequency in a spectrum acquisition. The value is expressed in hertz. An acquisition consists of a span of data surrounding the center frequency.

## NIRFSA\_ATTR\_SPAN

#### **Specific Attribute**

#### Data type Access High Level Functions

ViReal64 R/W <u>niRFSA\_ConfigureSpectrumFrequencyCenterSpan</u>

Specifies the frequency range of the computed spectrum. If you specify a center frequency of 1 GHz and span of 100 MHz, the spectrum ranges from 950 MHz to 1050 MHz after zoom processing. This value may be coerced based on hardware settings and downconversion specifications.

Note If you configure the spectrum span (stop frequency – start frequency) to a value larger than 20 MHz, RFSA performs multiple acquisitions and combines them into a spectrum of the size you requested.

# NIRFSA\_ATTR\_REFERENCE\_LEVEL

### **Specific Attribute**

Data type	Access	High Level Functions
ViReal64	R/W	niRFSA_ConfigureReferenceLevel

Specifies the reference level. The value is expressed in dBm. The reference level represents the maximum expected power of an input RF signal.

# NIRFSA\_ATTR\_ATTENUATION

## Specific Attribute

Data type	Access	High Level Functions
ViReal64	R/W	None

Specifies the downconverter module attenuation setting in dB.

Calculate the attenuation setting using desired <u>NIRFSA\_ATTR\_REFERENCE\_LEVEL</u> and <u>NIRFSA\_ATTR\_MIXER\_LEVEL</u> settings, according to the following formula:

attenuation = reference level – mixer level

For example, when using a reference level of 0 dBm (default) with moderate distortion and low noise, specify an attenuation value of 20 dB, as shown by the following calculation:

attenuation = (0 dB reference level) – (-20 dB mixer level

# NIRFSA\_ATTR\_MIXER\_LEVEL

## Specific Attribute

Data type	Access	High Level Functions
ViReal64	R/W	None

Specifies the mixer level. The value is expressed in dBm. The mixer level represents the attenuation to apply to the input RF signal as it reaches the first mixer in the signal chain. NI-RFSA automatically selects an optimal mixer level value given the reference level if you do not configure this attribute.

# NIRFSA\_ATTR\_ACQUISITION\_TYPE

## Specific Attribute

Data type	Access	High Level Functions
Vilnt32	R/W	niRFSA_ConfigureAcquisitionType

Configures whether the session acquires IQ data or computes a power spectrum over the specified frequency range.

### **Defined Values:**

NIRFSA_VAL_IQ (100)	Configures the driver for IQ acquisitions.
NIRFSA_VAL_SPECTRUM (101)	Configures the driver for spectrum acquisitions.

# NIRFSA\_ATTR\_IQ\_RATE

## Specific Attribute

Data type	Access	High Level Functions
ViReal64	R/W	niRFSA_ConfigureIQRate

Specifies the IQ rate for the acquisition. The value is expressed in S/s.

Notes Bandwidth is equal to the coerced IQ rate times 0.8.

You should not need to configure an IQ rate higher than 25 MHz, since the NI PXI-5600 downconverter bandwidth is 20 MHz. If you choose to configure a higher IQ rate, you may see aliasing effects at negative frequencies because the IF frequency of the downconverter is at 15 MHz.

# NIRFSA\_ATTR\_NUM\_SAMPLES\_IS\_FINITE

# Specific Attribute

Data type	Access	High Level Functions
ViBoolean	R/W	niRFSA ConfigureNumberOfSamples

Configures the device to stop after acquiring the specified number of samples or to acquire continuously.

### **Defined Values:**

	Configures the device to stop after acquiring the specified number of samples.
VI_FALSE	Acquire continuously until you abort the acquisition.

# NIRFSA\_ATTR\_NUM\_SAMPLES

## Specific Attribute

Data type	Access	High Level Functions
Vilnt64	R/W	niRFSA_ConfigureNumberOfSamples

Configures the number of samples.

## NIRFSA\_ATTR\_NUM\_RECORDS\_IS\_FINITE

### **Specific Attribute**

Data type	Access	High Level Functions

ViBoolean R/W <u>niRFSA\_ConfigureNumberOfRecords</u>

Configures the device to stop after acquiring the specified number of records or to acquire records continuously.

#### **Defined Values:**

_	Configures the device to stop after acquiring the specified number of records.
VI_FALSE	Acquire records continuously until you abort the acquisition.

# NIRFSA\_ATTR\_NUM\_RECORDS

Data type	Access	High Level Functions
Vilnt64	R/W	niRFSA_ConfigureNumberOfRecords

Specifies the number of records to acquire if <u>NIRFSA\_ATTR\_NUM\_RECORDS\_IS\_FINITE</u> is set to VI\_TRUE.

## NIRFSA\_ATTR\_POWER\_SPECTRUM\_UNITS

Data type	Access	High Level Functions	
Vilnt32	R/W	None	

Specifies the units of the spectrum.

### **Defined Values:**

NIRFSA_VAL_DBM (200)	Units are dB with reference to 1 milliwatt.
NIRFSA_VAL_VOLTS_SQUARED (201)	Units are in V <sup>2</sup> RMS.
NIRFSA_VAL_DBMV (202)	Units are dB with reference to 1 millivolt.
NIRFSA_VAL_DBUV (203)	Units are dB with reference to 1 microvolt.

## NIRFSA\_ATTR\_RESOLUTION\_BANDWIDTH

Data type	Access	High Level Functions	
ViReal64	R/W	niRFSA ConfigureResolutionBandwidth	

Specifies the resolution along the x-axis of the spectrum. NI-RFSA uses the resolution bandwidth value to determine the acquisition size. If <u>NIRFSA\_ATTR\_NUMBER\_OF\_SPECTRAL\_LINES</u> is specified, that value takes precedence over this value. If both attributes are set to 1, the spectrum uses a default of 400 spectral lines.

## NIRFSA\_ATTR\_RESOLUTION\_BANDWIDTH\_TYP

Data type	Access	High Level Functions
Vilnt32	R/W	None

Specifies the definition of <u>NIRFSA\_ATTR\_RESOLUTION\_BANDWIDTH</u>.

### **Defined Values:**

NIRFSA_VAL_RBW_3DB (300)	Defines the RBW in terms of the 3 dB bandwidth of the window specified by NIRFSA_ATTR_FFT_WINDOW_TYPE
NIRFSA_VAL_RBW_6DB (301)	Defines the resolution bandwidth in terms of the 6 dB bandwidth of the window specified by <u>NIRFSA_ATTR_FFT_WINDOW_TYPE</u>
NIRFSA_VAL_RBW_BIN_WIDTH (302)	Defines the resolution bandwidth in terms of the display resolution, which is the ratio of the sampling frequency to the number of samples that you acquire.
NIRFSA_VAL_RBW_ENBW (303)	Defines the resolution bandwidth in terms of the Equivalent Noise Bandwidth (ENBW) of the window specified by <u>NIRFSA_ATTR_FFT_WINDOW_TYPE</u>

## NIRFSA\_ATTR\_NUMBER\_OF\_SPECTRAL\_LINES

Data type	Access	High Level Functions
Vilnt32	R/W	None

Configures the number of spectral lines expected with the current power spectrum configuration. If you do not configure this attribute, NI-RFSA selects an appropriate value based on the

NIRFSA\_ATTR\_RESOLUTION\_BANDWIDTH attribute. If you configure this attribute, NI-RFSA coerces the resolution bandwidth value based on the number of spectral lines requested and the acquisition span.

## NIRFSA\_ATTR\_SPECTRUM\_AVERAGING\_MODE

Data type	Access	High Level Functions
Vilnt32	R/W	None

Specifies the averaging mode for the spectrum acquisition.

### **Defined Values:**

NIRFSA_VAL_NO_AVERAGING (400)	Configures the driver to perform no averaging on acquisitions.
NIRFSA_VAL_RMS_AVERAGING (401)	Configures the driver for RMS averaging. RMS averaging reduces signal fluctuations but not the noise floor. RMS averaging averages the energy or power of the signal, which prevents noise floor reduction and gives averaged rms quantities of single-channel measurements zero phase. RMS averaging for dual- channel measurements preserves important phase information.
NIRFSA_VAL_VECTOR_AVERAGING (402)	Configures the driver for vector averaging. Vector averaging reduces noise from synchronous signals. Vector averaging computes the average of complex quantities directly, which means that it allows separate averaging for real and imaginary parts. Complex averaging such as vector averaging reduces noise and usually requires a trigger to improve block-to-block phase coherence.
NIRFSA_VAL_PEAK_HOLD_AVERAGING (403)	Configures the driver for peak hold averaging. Peak

hold averaging retains the RMS peak levels of the averaged quantities. The peak hold averaging process performs peak hold at each frequency bin separately to retain peak rms levels from one FFT record to the next.

## NIRFSA\_ATTR\_SPECTRUM\_NUM\_AVERAGES

Data type	Access	High Level Functions	
Vilnt32	R/W	None	

Specifies the number of averages to complete for linear weighting. The averaging process returns the final result after the number of averages is complete.

## NIRFSA\_ATTR\_FFT\_WINDOW\_TYPE

Data type	Access	High Level Functions
Vilnt32	R/W	None

Specifies the time-domain window type.

#### **Defined Values:**

NIRFSA_VAL_UNIFORM (500)
NIRFSA_VAL_HANNING (501)
NIRFSA_VAL_HAMMING (502)
NIRFSA_VAL_BLACKMAN_HARRIS (503)
NIRFSA_VAL_EXACT_BLACKMAN (504)
NIRFSA_VAL_BLACKMAN (505)
NIRFSA_VAL_FLAT_TOP (506)
NIRFSA_VAL_4_TERM_BLACKMAN_HARRIS (507)
NIRFSA_VAL_7_TERM_BLACKMAN_HARRIS (508)
NIRFSA_VAL_LOW_SIDE_LOBE (509)

## NIRFSA\_ATTR\_FFT\_WINDOW\_SIZE

Data type	Access	High Level Functions
Vilnt32	RO	None

Returns the size of the window used in the Fast Fourier Transform.

# NIRFSA\_ATTR\_FFT\_SIZE

Data type	Access	High Level Functions
Vilnt32	RO	None

Returns the size of the Fast Fourier Transform.

# NIRFSA\_ATTR\_FETCH\_RELATIVE\_TO

Data type	Access	High Level Functions
Vilnt32	R/W	None

Specifies the absolute location within the acquired record from which to begin fetching.

#### **Defined Values:**

NIRFSA_VAL_MOST_RECENT_SAMPLE (700)	Specifies that fetchir relative to the most r acquired data. <u>NIRFSA_ATTR_FET</u> must be negative.
NIRFSA_VAL_FIRST_SAMPLE (701)	Specifies that fetchir the first sample acqu device. If the device buffer, then the first s longer available. In t RFSA returns an erro offset is in the overw
NIRFSA_VAL_REF_TRIGGER (702)	Specifies that fetchir relative to the Refere This value behaves   NIRFSA_VAL_FIRST no Reference trigger configured.
NIRFSA_VAL_FIRST_PRETRIGGER_SAMPLE (703)	Specifies that fetchir relative to the first pr sample acquired. Th behaves like NIRFSA_VAL_FIRST no Reference trigger configured.
NIRFSA_VAL_CURRENT_READ_POSITION (704)	Specifies that fetchir the last fetched sam

# NIRFSA\_ATTR\_FETCH\_OFFSET

Data type	Access	High Level Functions
Vilnt64	R/W	None

Specifies the offset relative to the position specified by <u>NIRFSA\_ATTR\_FETCH\_RELATIVE\_TO</u> from which to start fetching data. Offset can be a positive or negative value.

## NIRFSA\_ATTR\_RECORDS\_DONE

Data type	Access	High Level Functions
Vilnt32	RO	None

Returns the number of records the RF signal analyzer has acquired.

# NIRFSA\_ATTR\_REF\_CLOCK\_SOURCE

Data type	Access	High Level Functions
ViString	R/W	niRFSA_ConfigureRefClock

Specifies the Reference clock source.

### **Defined Values:**

NIRFSA_VAL_ONBOARD_CLOCK_STR ("OnboardClock")	Lock the NI-RFSA device to the NI PXI-5600 onboard clock.
	Lock the NI-RFSA device to the external REF IN connector on the NI PXI-5600. You must install the NI PXI-5600 in Slot 2 of your PXI chassis to use this option.
NIRFSA_VAL_PXI_CLK10_STR ("PXI_Clk10")	Lock the NI-RFSA device to the PXI backplane clock using the NI PXI-5600.

# NIRFSA\_ATTR\_REF\_CLOCK\_RATE

Data type	Access	High Level Functions
ViReal64	R/W	niRFSA_ConfigureRefClock

Specifies the rate of the reference clock. The value is expressed in hertz. NI-RFSA only supports a reference clock rate of 10 MHz.

# NIRFSA\_ATTR\_DIGITIZER\_SAMPLE\_CLOCK\_TIN

Data type	Access	High Level Functions
ViString	R/W	None

Specifies the digitizer's sample clock timebase source.

### **Defined Values:**

NIRFSA_VAL_NONE_STR	The digitizer uses its onboard clock as the Sample clock timebase source.
	The digitizer uses the signal at the CLK IN input terminal as the Sample clock timebase source.
NIRFSA_VAL_PXI_STAR_STR	The digitizer uses the signal on the PXI star trigger line as the Sample clock timebase source.

# NIRFSA\_ATTR\_DIGITIZER\_SAMPLE\_CLOCK\_TIN

Data type	Access	High Level Functions
ViReal64	R/W	N/A

Specifies the frequency, in hertz, of the external clock used as the timebase source if

NIRFSA\_ATTR\_DIGITIZER\_SAMPLE\_CLOCK\_TIMEBASE\_SOURCE is an external source.

If timebase rate is set to a value below 60 MHz, signals at frequencies just above the 20 MHz passband of the downconverter may be aliased back into the passband. This aliasing occurs because the IF frequency of the downconverter is at 15 MHz, and the upper end of the passband is at 25 MHz. At sampling rates below 60 MHz, the Nyquist frequency is close to the end of the passband and creates aliases that are not effectively filtered by the downconverter.

# NIRFSA\_ATTR\_PXI\_CHASSIS\_CLK10\_SOURCE

Data type	Access	High Level Functions
ViString	R/W	niRFSA_ConfigurePXIChassisClk10

Specifies the signal to drive the 10 MHz reference clock on the PXI backplane. This option can only be configured when the NI PXI-5600 is in Slot 2 of the PXI chassis.

### **Defined Values:**

NIRFSA_VAL_NONE_STR ("None")	The device does not drive the PXI 10 MHz backplane reference clock.
NIRFSA_VAL_ONBOARD_CLOCK_STR ("OnboardClock")	The device drives the PXI 10 MHz backplane reference clock with the NI PXI-5600 onboard clock. You must connect the 10 MHz OUT connector to the PXI 10 MHz I/O on the NI PXI 5600 front panel to use this option.
NIRFSA_VAL_REF_IN_STR ("Refln")	The device drives the PXI 10 MHz backplane reference clock with the reference source attached to the NI PXI-5600 REF IN connector. You must connect the 10 MHz OUT connector to the PXI 10 MHz I/O on the NI PXI 5600 front panel to use this option.

# NIRFSA\_ATTR\_START\_TRIGGER\_TYPE

Data type	Access	High Level Functions
Vilnt32	R/W	None

Specifies whether you want the Start trigger to be a digital edge or software trigger.

### **Defined Values:**

NIRFSA_VAL_NONE (600)	No Advance trigger is configured.
NIRFSA_VAL_DIGITAL_EDGE (601)	The Start trigger is not asserted until a dic The source of the digital edge is specified <u>NIRFSA_ATTR_DIGITAL_EDGE_START</u>
NIRFSA_VAL_SOFTWARE (604)	The Advance trigger is not asserted until a occurs. You can assert the software trigger <u>niRFSA_SendSoftwareEdgeTrigger</u> function NIRFSA_VAL_START_TRIGGER as the <b>tr</b>

## NIRFSA\_ATTR\_DIGITAL\_EDGE\_START\_TRIGGEF

#### **Specific Attribute**

Data type Access High Level Functions

ViString R/W <u>niRFSA\_ConfigureDigitalEdgeStartTrigger</u>

Specifies the source terminal for the Start trigger. This attribute is used only when <u>NIRFSA\_ATTR\_START\_TRIGGER\_TYPE</u> is set to NIRFSA\_VAL\_DIGITAL\_EDGE.

PFI0	The trigger is received on PFI 0.
PFI1	The trigger is received on PFI 1.
PXI_Trig0	The trigger is received on PXI trigger line 0.
PXI_Trig1	The trigger is received on PXI trigger line 1.
PXI_Trig2	The trigger is received on PXI trigger line 2.
PXI_Trig3	The trigger is received on PXI trigger line 3.
PXI_Trig4	The trigger is received on PXI trigger line 4.
PXI_Trig5	The trigger is received on PXI trigger line 5.
PXI_Trig6	The trigger is received on PXI trigger line 6.
PXI_Trig7	The trigger is received on PXI trigger line 7.
PXI_STAR	The trigger is received on the PXI star trigger line.

## NIRFSA\_ATTR\_DIGITAL\_EDGE\_START\_TRIGGEF

#### **Specific Attribute**

Data Access High Level Functions

ViInt32 R/W <u>niRFSA\_ConfigureDigitalEdgeStartTrigger</u>

Specify the active edge for the Start trigger. This property is used only when <u>NIRFSA\_ATTR\_START\_TRIGGER\_TYPE</u> is set to NIRFSA\_VAL\_DIGITAL\_EDGE.

The trigger asserts on the rising edge of the signal.
The trigger asserts on the falling edge of the signal

## NIRFSA\_ATTR\_EXPORTED\_START\_TRIGGER\_O

Data type	Access	High Level Functions
ViString	R/W	niRFSA_ExportSignal

Specifies the destination terminal for the exported Start trigger.

### **Defined Values:**

NIRFSA_VAL_DO_NOT_EXPORT_STR ("")	The signal is not exported.
NIRFSA_VAL_PFI0_STR ("PFI0")	The signal is exported on PFI 0.
NIRFSA_VAL_PFI1_STR ("PFI1")	The signal is exported on PFI 1.
NIRFSA_VAL_PXI_TRIG0_STR ("PXI_Trig0")	The signal is exported on PXI trigger line 0.
NIRFSA_VAL_PXI_TRIG1_STR ("PXI_Trig1")	The signal is exported on PXI trigger line 1.
NIRFSA_VAL_PXI_TRIG2_STR ("PXI_Trig2")	The signal is exported on PXI trigger line 2.
NIRFSA_VAL_PXI_TRIG3_STR ("PXI_Trig3")	The signal is exported on PXI trigger line 3.
NIRFSA_VAL_PXI_TRIG4_STR ("PXI_Trig4")	The signal is exported on PXI trigger line 4.
NIRFSA_VAL_PXI_TRIG5_STR ("PXI_Trig5")	The signal is exported on PXI trigger line 5.
NIRFSA_VAL_PXI_TRIG6_STR ("PXI_Trig6")	The signal is exported on PXI trigger line 6.
NIRFSA_VAL_PXI_TRIG7_STR ("PXI_Trig7")	The signal is exported on PXI trigger line 7.
NIRFSA_VAL_PXI_STAR_STR ("PXI_STAR")	The signal is exported on the PXI star trigger line.

# NIRFSA\_ATTR\_REF\_TRIGGER\_TYPE

Data type	Access	High Level Functions
Vilnt32	R/W	None

Specifies whether you want the Reference trigger to be a digital edge, IQ power edge, or software trigger.

### **Defined Values:**

NIRFSA_VAL_NONE (600)	No Reference trigger is configured.
NIRFSA_VAL_DIGITAL_EDGE (601)	The Reference trigger is not asserted u detected. The source of the digital edge NIRFSA_ATTR_DIGITAL_EDGE_REF attribute.
NIRFSA_VAL_IQ_POWER_EDGE (603)	
NIRFSA_VAL_SOFTWARE (604)	The Arm Reference trigger is not asser trigger occurs. You can assert the softw the <u>niRFSA_SendSoftwareEdgeTrigger</u> selecting NIRFSA_VAL_REF_TRIGGER parameter.

## NIRFSA\_ATTR\_REF\_TRIGGER\_PRETRIGGER\_S/

Data type	Access	High Level Functions
Vilnt64	R/W	niRFSA_ConfigureDigitalEdgeRefTrigger niRFSA_ConfigureSoftwareEdgeRefTrigger niRFSA_ConfigureIQPowerEdgeRefTrigger

Specifies the number of pretrigger samples, the samples acquired before the Reference trigger is received, to be acquired per record.

## NIRFSA\_ATTR\_DIGITAL\_EDGE\_REF\_TRIGGER\_\$

#### **Specific Attribute**

Data type Access High Level Functions

ViString R/W <u>niRFSA\_ConfigureDigitalEdgeRefTrigger</u>

Specifies the source terminal for the Reference trigger. This attribute is used only when <u>NIRFSA\_ATTR\_REF\_TRIGGER\_TYPE</u> is set to NIRFSA\_VAL\_DIGITAL\_EDGE.

PFI0	The trigger is received on PFI 0.
PFI1	The trigger is received on PFI 1.
PXI_Trig0	The trigger is received on PXI trigger line 0.
PXI_Trig1	The trigger is received on PXI trigger line 1.
PXI_Trig2	The trigger is received on PXI trigger line 2.
PXI_Trig3	The trigger is received on PXI trigger line 3.
PXI_Trig4	The trigger is received on PXI trigger line 4.
PXI_Trig5	The trigger is received on PXI trigger line 5.
PXI_Trig6	The trigger is received on PXI trigger line 6.
PXI_Trig7	The trigger is received on PXI trigger line 7.
PXI_STAR	The trigger is received on the PXI star trigger line.

## NIRFSA\_ATTR\_DIGITAL\_EDGE\_REF\_TRIGGER\_I

#### **Specific Attribute**

\_

Data	٨٠٢٩٩٩	<b>High Level Functions</b>
type	ALLESS	The real functions

ViInt32 R/W <u>niRFSA\_ConfigureDigitalEdgeRefTrigger</u>

Specify the active edge for the Reference trigger. This property is used only when <u>NIRFSA\_ATTR\_REF\_TRIGGER\_TYPE</u> is set to NIRFSA\_VAL\_DIGITAL\_EDGE.

 The trigger asserts on the rising edge of the signal.
 The trigger asserts on the falling edge of the signal

# NIRFSA\_ATTR\_IQ\_POWER\_EDGE\_REF\_TRIGGE

#### **Specific Attribute**

Data type Access High Level Functions

ViString R/W <u>niRFSA\_ConfigureIQPowerEdgeRefTrigger</u>

Specifies the channel from which the device will monitor the trigger. The only valid input for this attribute is "0" at this time.

# NIRFSA\_ATTR\_IQ\_POWER\_EDGE\_REF\_TRIGGE

#### **Specific Attribute**

Data type Access High Level Functions

ViReal64 R/W <u>niRFSA\_ConfigureIQPowerEdgeRefTrigger</u>

Specifies the power level in dBm at which the device will trigger. The device asserts the trigger when the signal exceeds the level specified by the value of this attribute.

# NIRFSA\_ATTR\_IQ\_POWER\_EDGE\_REF\_TRIGGE

#### **Specific Attribute**

Data Access High Level Functions

ViInt32 R/W <u>niRFSA\_ConfigureIQPowerEdgeRefTrigger</u>

Specifies whether the device asserts the trigger when the signal power is rising or falling. When the trigger is configured for IQ power edge the device asserts the trigger when the power exceeds the specified level with the slope you specify.

 The trigger asserts when the signal power is rising.
 The trigger asserts when the signal power is falling.

# NIRFSA\_ATTR\_IQ\_POWER\_EDGE\_REF\_TRIGGE

### **Specific Attribute**

Data	Access	<b>High Level Functions</b>
type	AUCC33	

ViReal64 R/W

None

Specifies a time duration for which the signal must be quiet before the device arms the trigger. The signal is quiet when it is below the trigger level if the trigger slope, specified by

NIRFSA\_ATTR\_IQ\_POWER\_EDGE\_REF\_TRIGGER\_SLOPE, is set to NIRFSA\_VAL\_RISING\_SLOPE or above the trigger level if the trigger slope is set to NIRFSA\_VAL\_FALLING\_SLOPE.

By default this value is set to 0, which means the device does not wait for a quiet time before arming the trigger. This attribute is useful to trigger the acquisition on signals containing repeated bursts, but for which each burst may have large changes in signal power within itself. By configuring the minimum quiet time to the time between bursts, you can ensure that the trigger occurs at the beginning of a burst rather than in signal power change within a burst.

### NIRFSA\_ATTR\_EXPORTED\_REF\_TRIGGER\_OUT

Data type	Access	High Level Functions
ViString	R/W	niRFSA_ExportSignal

Specifies the destination terminal for the exported Reference trigger.

NIRFSA_VAL_DO_NOT_EXPORT_STR ("")	The signal is not exported.
NIRFSA_VAL_PFI0_STR ("PFI0")	The signal is exported on PFI 0.
NIRFSA_VAL_PFI1_STR ("PFI1")	The signal is exported on PFI 1.
NIRFSA_VAL_PXI_TRIG0_STR ("PXI_Trig0")	The signal is exported on PXI trigger line 0.
NIRFSA_VAL_PXI_TRIG1_STR ("PXI_Trig1")	The signal is exported on PXI trigger line 1.
NIRFSA_VAL_PXI_TRIG2_STR ("PXI_Trig2")	The signal is exported on PXI trigger line 2.
NIRFSA_VAL_PXI_TRIG3_STR ("PXI_Trig3")	The signal is exported on PXI trigger line 3.
NIRFSA_VAL_PXI_TRIG4_STR ("PXI_Trig4")	The signal is exported on PXI trigger line 4.
NIRFSA_VAL_PXI_TRIG5_STR ("PXI_Trig5")	The signal is exported on PXI trigger line 5.
NIRFSA_VAL_PXI_TRIG6_STR ("PXI_Trig6")	The signal is exported on PXI trigger line 6.
NIRFSA_VAL_PXI_TRIG7_STR ("PXI_Trig7")	The signal is exported on PXI trigger line 7.
NIRFSA_VAL_PXI_STAR_STR ("PXI_STAR")	The signal is exported on the PXI star trigger line.

# NIRFSA\_ATTR\_START\_TO\_REF\_TRIGGER\_HOLI

**Specific Attribute** 

Data type Access High-Level Functions

ViReal64 R/W None

Specifies the minimum time in seconds that must elapse after the Start trigger is received before the device recognizes a Reference trigger.

Units: seconds

# NIRFSA\_ATTR\_REF\_TO\_REF\_TRIGGER\_HOLDO

**Specific Attribute** 

Data type Access High-Level Functions

ViReal64 R/W None

Specifies the minimum time in seconds that must elapse after the Reference trigger for one record is received before the device will recognize the Reference trigger for the next record.

Units: seconds

# NIRFSA\_ATTR\_ADVANCE\_TRIGGER\_TYPE

Data type	Access	High Level Functions
Vilnt32	R/W	None

Specifies whether you want the Advance trigger to be a digital edge or pattern match trigger.

NIRFSA_VAL_NONE (600)	No Advance trigger is configured.
NIRFSA_VAL_DIGITAL_EDGE (601)	The Advance trigger is not asserted until a The source of the digital edge is specified NIRFSA_ATTR_DIGITAL_EDGE_ADVAN
NIRFSA_VAL_SOFTWARE (604)	The Advance trigger is not asserted until a You can assert the software trigger by cal <u>niRFSA_SendSoftwareEdgeTrigger</u> function NIRFSA_VAL_ADVANCE_TRIGGER as the software trigger as the software tr

### NIRFSA\_ATTR\_DIGITAL\_EDGE\_ADVANCE\_TRIG

#### **Specific Attribute**

Data type Access High Level Functions

ViString R/W <u>niRFSA\_ConfigureDigitalEdgeAdvanceTrigger</u>

Specifies the source terminal for the Advance trigger. This attribute is used only when <u>NIRFSA\_ATTR\_ADVANCE\_TRIGGER\_TYPE</u> is set to NIRFSA\_VAL\_DIGITAL\_EDGE.

PFI0	The trigger is received on PFI 0.
PFI1	The trigger is received on PFI 1.
PXI_Trig0	The trigger is received on PXI trigger line 0.
PXI_Trig1	The trigger is received on PXI trigger line 1.
PXI_Trig2	The trigger is received on PXI trigger line 2.
PXI_Trig3	The trigger is received on PXI trigger line 3.
PXI_Trig4	The trigger is received on PXI trigger line 4.
PXI_Trig5	The trigger is received on PXI trigger line 5.
PXI_Trig6	The trigger is received on PXI trigger line 6.
PXI_Trig7	The trigger is received on PXI trigger line 7.
PXI_STAR	The trigger is received on the PXI star trigger line.

### NIRFSA\_ATTR\_EXPORTED\_ADVANCE\_TRIGGEF

Data type	Access	High Level Functions
ViString	R/W	niRFSA_ExportSignal

Specifies the destination terminal for the exported Advance trigger.

NIRFSA_VAL_DO_NOT_EXPORT_STR ("")	The signal is not exported.
NIRFSA_VAL_PFI0_STR ("PFI0")	The signal is exported on PFI 0.
NIRFSA_VAL_PFI1_STR ("PFI1")	The signal is exported on PFI 1.
NIRFSA_VAL_PXI_TRIG0_STR ("PXI_Trig0")	The signal is exported on PXI trigger line 0.
NIRFSA_VAL_PXI_TRIG1_STR ("PXI_Trig1")	The signal is exported on PXI trigger line 1.
NIRFSA_VAL_PXI_TRIG2_STR ("PXI_Trig2")	The signal is exported on PXI trigger line 2.
NIRFSA_VAL_PXI_TRIG3_STR ("PXI_Trig3")	The signal is exported on PXI trigger line 3.
NIRFSA_VAL_PXI_TRIG4_STR ("PXI_Trig4")	The signal is exported on PXI trigger line 4.
NIRFSA_VAL_PXI_TRIG5_STR ("PXI_Trig5")	The signal is exported on PXI trigger line 5.
NIRFSA_VAL_PXI_TRIG6_STR ("PXI_Trig6")	The signal is exported on PXI trigger line 6.
NIRFSA_VAL_PXI_TRIG7_STR ("PXI_Trig7")	The signal is exported on PXI trigger line 7.
NIRFSA_VAL_PXI_STAR_STR ("PXI_STAR")	The signal is exported on the PXI star trigger line.

# NIRFSA\_ATTR\_ARM\_REF\_TRIGGER\_TYPE

Data type	Access	High Level Functions
Vilnt32	R/W	None

Specifies whether you want the Arm Reference trigger to be a digital edge or software trigger.

NIRFSA_VAL_NONE (600)	No Arm Reference trigger is configured.
NIRFSA_VAL_DIGITAL_EDGE (601)	The Arm Reference trigger is not asserted detected. The source of the digital edge is NIRFSA_ATTR_DIGITAL_EDGE_ARM_F attribute.
NIRFSA_VAL_SOFTWARE (604)	The Arm Reference trigger is not asserted occurs. You can assert the software triggen niRFSA_SendSoftwareEdgeTrigger function NIRFSA_VAL_ARM_REF_TRIGGER as th

# NIRFSA\_ATTR\_DIGITAL\_EDGE\_ARM\_REF\_TRIG

Data type	Access	High Level Functions
ViString	R/W	None

Specifies the source terminal for the Arm Reference trigger. This attribute is used only when <u>NIRFSA\_ATTR\_ARM\_REF\_TRIGGER\_TYPE</u> is set to NIRFSA\_VAL\_DIGITAL\_EDGE.

PFI0	The trigger is received on PFI 0.
PFI1	The trigger is received on PFI 1.
PXI_Trig0	The trigger is received on PXI trigger line 0.
PXI_Trig1	The trigger is received on PXI trigger line 1.
PXI_Trig2	The trigger is received on PXI trigger line 2.
PXI_Trig3	The trigger is received on PXI trigger line 3.
PXI_Trig4	The trigger is received on PXI trigger line 4.
PXI_Trig5	The trigger is received on PXI trigger line 5.
PXI_Trig6	The trigger is received on PXI trigger line 6.
PXI_Trig7	The trigger is received on PXI trigger line 7.
PXI_STAR	The trigger is received on the PXI star trigger line.

# NIRFSA\_ATTR\_EXPORTED\_READY\_FOR\_STAR1

Data type	Access	High Level Functions
ViString	R/W	niRFSA_ExportSignal

Identifies the hardware signal line on which the digital pulse for the Ready for Start event is generated.

NIRFSA_VAL_DO_NOT_EXPORT_STR ("")	The signal is not exported.
NIRFSA_VAL_PFI0_STR ("PFI0")	The signal is exported on PFI 0.
NIRFSA_VAL_PFI1_STR ("PFI1")	The signal is exported on PFI 1.
NIRFSA_VAL_PXI_TRIG0_STR ("PXI_Trig0")	The signal is exported on PXI trigger line 0.
NIRFSA_VAL_PXI_TRIG1_STR ("PXI_Trig1")	The signal is exported on PXI trigger line 1.
NIRFSA_VAL_PXI_TRIG2_STR ("PXI_Trig2")	The signal is exported on PXI trigger line 2.
NIRFSA_VAL_PXI_TRIG3_STR ("PXI_Trig3")	The signal is exported on PXI trigger line 3.
NIRFSA_VAL_PXI_TRIG4_STR ("PXI_Trig4")	The signal is exported on PXI trigger line 4.
NIRFSA_VAL_PXI_TRIG5_STR ("PXI_Trig5")	The signal is exported on PXI trigger line 5.
NIRFSA_VAL_PXI_TRIG6_STR ("PXI_Trig6")	The signal is exported on PXI trigger line 6.
NIRFSA_VAL_PXI_TRIG7_STR ("PXI_Trig7")	The signal is exported on PXI trigger line 7.
NIRFSA_VAL_PXI_STAR_STR ("PXI_STAR")	The signal is exported on the PXI star trigger line.

# NIRFSA\_ATTR\_EXPORTED\_READY\_FOR\_ADVAI

Data type	Access	High Level Functions
ViString	R/W	niRFSA_ExportSignal

Identifies the hardware signal line on which the digital pulse for the Ready for Advance event is generated.

NIRFSA_VAL_DO_NOT_EXPORT_STR ("")	The signal is not exported.
NIRFSA_VAL_PFI0_STR ("PFI0")	The signal is exported on PFI 0.
NIRFSA_VAL_PFI1_STR ("PFI1")	The signal is exported on PFI 1.
NIRFSA_VAL_PXI_TRIG0_STR ("PXI_Trig0")	The signal is exported on PXI trigger line 0.
NIRFSA_VAL_PXI_TRIG1_STR ("PXI_Trig1")	The signal is exported on PXI trigger line 1.
NIRFSA_VAL_PXI_TRIG2_STR ("PXI_Trig2")	The signal is exported on PXI trigger line 2.
NIRFSA_VAL_PXI_TRIG3_STR ("PXI_Trig3")	The signal is exported on PXI trigger line 3.
NIRFSA_VAL_PXI_TRIG4_STR ("PXI_Trig4")	The signal is exported on PXI trigger line 4.
NIRFSA_VAL_PXI_TRIG5_STR ("PXI_Trig5")	The signal is exported on PXI trigger line 5.
NIRFSA_VAL_PXI_TRIG6_STR ("PXI_Trig6")	The signal is exported on PXI trigger line 6.
NIRFSA_VAL_PXI_TRIG7_STR ("PXI_Trig7")	The signal is exported on PXI trigger line 7.
NIRFSA_VAL_PXI_STAR_STR ("PXI_STAR")	The signal is exported on the PXI star trigger line.

# NIRFSA\_ATTR\_EXPORTED\_READY\_FOR\_REF\_E

Data type	Access	High Level Functions
ViString	R/W	niRFSA_ExportSignal

Identifies the hardware signal line on which the digital pulse for the Ready for Ref event is generated.

NIRFSA_VAL_DO_NOT_EXPORT_STR ("")	The signal is not exported.
NIRFSA_VAL_PFI0_STR ("PFI0")	The signal is exported on PFI 0.
NIRFSA_VAL_PFI1_STR ("PFI1")	The signal is exported on PFI 1.
NIRFSA_VAL_PXI_TRIG0_STR ("PXI_Trig0")	The signal is exported on PXI trigger line 0.
NIRFSA_VAL_PXI_TRIG1_STR ("PXI_Trig1")	The signal is exported on PXI trigger line 1.
NIRFSA_VAL_PXI_TRIG2_STR ("PXI_Trig2")	The signal is exported on PXI trigger line 2.
NIRFSA_VAL_PXI_TRIG3_STR ("PXI_Trig3")	The signal is exported on PXI trigger line 3.
NIRFSA_VAL_PXI_TRIG4_STR ("PXI_Trig4")	The signal is exported on PXI trigger line 4.
NIRFSA_VAL_PXI_TRIG5_STR ("PXI_Trig5")	The signal is exported on PXI trigger line 5.
NIRFSA_VAL_PXI_TRIG6_STR ("PXI_Trig6")	The signal is exported on PXI trigger line 6.
NIRFSA_VAL_PXI_TRIG7_STR ("PXI_Trig7")	The signal is exported on PXI trigger line 7.
NIRFSA_VAL_PXI_STAR_STR ("PXI_STAR")	The signal is exported on the PXI star trigger line.

# NIRFSA\_ATTR\_EXPORTED\_END\_OF\_RECORD\_I

Data type	Access	High Level Functions
ViString	R/W	niRFSA_ExportSignal

Identifies the hardware signal line on which the digital pulse for the End of Record event is generated.

NIRFSA_VAL_DO_NOT_EXPORT_STR ("")	The signal is not exported.
NIRFSA_VAL_PFI0_STR ("PFI0")	The signal is exported on PFI 0.
NIRFSA_VAL_PFI1_STR ("PFI1")	The signal is exported on PFI 1.
NIRFSA_VAL_PXI_TRIG0_STR ("PXI_Trig0")	The signal is exported on PXI trigger line 0.
NIRFSA_VAL_PXI_TRIG1_STR ("PXI_Trig1")	The signal is exported on PXI trigger line 1.
NIRFSA_VAL_PXI_TRIG2_STR ("PXI_Trig2")	The signal is exported on PXI trigger line 2.
NIRFSA_VAL_PXI_TRIG3_STR ("PXI_Trig3")	The signal is exported on PXI trigger line 3.
NIRFSA_VAL_PXI_TRIG4_STR ("PXI_Trig4")	The signal is exported on PXI trigger line 4.
NIRFSA_VAL_PXI_TRIG5_STR ("PXI_Trig5")	The signal is exported on PXI trigger line 5.
NIRFSA_VAL_PXI_TRIG6_STR ("PXI_Trig6")	The signal is exported on PXI trigger line 6.
NIRFSA_VAL_PXI_TRIG7_STR ("PXI_Trig7")	The signal is exported on PXI trigger line 7.
NIRFSA_VAL_PXI_STAR_STR ("PXI_STAR")	The signal is exported on the PXI star trigger line.

## NIRFSA\_ATTR\_DIGITAL\_IF\_EQUALIZATION\_ENA

### **Specific Attribute**

Data	Access	High Level Functions
type	ALLESS	right Level Functions

ViBoolean

R/W

None

Toggles use of the digital equalization filter for the NI 5600.

VI_TRUE	Enables digital IF equalization on the NI 5600.
VI_FALSE	Disables digital IF equalization on the NI 5600.

## NIRFSA\_ATTR\_SERIAL\_NUMBER

Data type	Access	High Level Functions
ViString	RO	None

Returns the serial number of the NI 5600 downconverter module.

## NIRFSA\_ATTR\_TEMPERATURE

Data type	Access	High Level Functions
ViReal64	RO	None

Returns the current temperature of the NI 5600 downconverter module.

## NIRFSA\_ATTR\_RANGE\_CHECK

Data type	Access	High Level Functions
ViBoolean	R/W	None

Specifies whether to validate attribute values and function parameters. If enabled, NI-RFSA validates the parameter values that you pass to NI-RFSA functions. Range checking parameters is very useful for debugging. After you validate your program, you can set this attribute to VI\_FALSE to disable range checking and maximize performance.

**Note** Use <u>niRFSA\_InitWithOptions</u> to override this value.

VI_TRUE	NI-RFSA validates attribute values and function parameters. This is the default value.
VI_FALSE	NI-RFSA does not validate attribute values and function parameters.

## NIRFSA\_ATTR\_QUERY\_INSTRUMENT\_STATUS

### Specific Attribute

Data	Access	<b>High Level Functions</b>
type	AUUU33	

ViBoolean

R/W

None

Specifies whether NI-RFSA queries the device status after each operation. Querying the device status is useful for debugging. After you validate your program, you can set this attribute to VI\_FALSE to disable status checking and maximize performance.

NI-RFSA can choose to ignore status checking for particular attributes, regardless of the setting of this attribute.

**Note** Use <u>niRFSA\_InitWithOptions</u> to override this value.

VI_TRUE	NI-RFSA queries the device status after each operation.
VI_FALSE	NI-RFSA does not query the device status after each
	operation. This is the default value.

# NIRFSA\_ATTR\_CACHE

### **Specific Attribute**

Data type	Access	High Level Functions
--------------	--------	----------------------

ViBoolean

R/W

None

Specifies whether to cache the value of attributes. When caching is enabled , NI-RFSA tracks the current NI-RFSA device settings and avoids sending redundant commands to the device.

NI-RFSA can always cache or never cache particular attributes, regardless of the setting of this attribute.

Default Value: VI\_TRUE

Use <u>niRFSA\_InitWithOptions</u> to override the default value.

VI_TRUE	Caching is enabled.
VI_FALSE	Caching is disabled.

# NIRFSA\_ATTR\_SIMULATE

Data type	Access	High Level Functions
ViBoolean	RO	None

Specifies whether or not to simulate NI-RFSA I/O operations. This is useful for debugging applications without using hardware. Once a session is opened, you cannot change the simulation state. Use the <u>niRFSA\_InitWithOptions</u> function to enable simulation.

VI_	TRUE	NI-RFSA simulates NI-RFSA I/O operations.
VI_	FALSE	NI-RFSA does not simulate NI-RFSA I/O operations. This is
		the default value.

## NIRFSA\_ATTR\_RECORD\_COERCIONS

### **Specific Attribute**

Data	Access	High Level Functions
type	ALLESS	

ViBoolean

R/W

None

Specifies whether the IVI engine keeps a list of the value coercions it makes for integer and real type attributes.



**Note** This attribute is currently not supported.

The IVI engine keeps a list of the value coercions it makes for integer and real type attributes.
The IVI engine does not keep a list of the value coercions it makes for integer and real type attributes. This is the default value.

## NIRFSA\_ATTR\_INTERCHANGE\_CHECK

### **Specific Attribute**

Data	Access	High Level Functions
type	AUCESS	

ViBoolean

R/W

None

Specifies whether to perform interchangeability checking and retrieve interchangeability warnings.



**Note** Interchangeability check is unsupported.

The driver performs interchangeability checking and retrieves warnings.
The driver does not perform interchangeability checking or retrieve warnings. This is the default value.

# NIRFSA\_ATTR\_SPECIFIC\_DRIVER\_DESCRIPTIOI

Data type	Access	High Level Functions
ViString	RO	None

A string that contains a brief description of NI-RFSA.

# NIRFSA\_ATTR\_SPECIFIC\_DRIVER\_PREFIX

Data type	Access	High Level Functions
ViString	RO	None

A string that contains the prefix for NI-RFSA. The name of each usercallable function in NI-RFSA starts with this prefix.

## NIRFSA\_ATTR\_SPECIFIC\_DRIVER\_VENDOR

Data type	Access	High Level Functions
ViString	RO	None

A string that contains the name of the vendor that supplies NI-RFSA.

# NIRFSA\_ATTR\_SPECIFIC\_DRIVER\_REVISION

Data type	Access	High Level Functions
ViString	RO	None

A string that contains additional version information about NI-RFSA.

## NIRFSA\_ATTR\_SUPPORTED\_INSTRUMENT\_MOI

Data type	Access	High Level Functions
ViString	RO	None

Contains a model code of the NI-RFSA device. For drivers that support more than one device, this attribute contains a comma-separated list of supported devices.

### NIRFSA\_ATTR\_INSTRUMENT\_MANUFACTURER

Data type	Access	High Level Functions
ViString	RO	None

A string that contains the name of the manufacturer for the NI-RFSA device you are currently using.

## NIRFSA\_ATTR\_INSTRUMENT\_MODEL

Data type	Access	High Level Functions
ViString	RO	None

A string that contains the model number or name of the NI-RFSA device that you are currently using.

## NIRFSA\_ATTR\_INSTRUMENT\_FIRMWARE\_REVIS

Data type	Access	High Level Functions
ViString	RO	None

A string that contains the firmware revision information for the NI-RFSA device you are currently using.

## NIRFSA\_ATTR\_LOGICAL\_NAME

Data type	Access	High Level Functions
ViString	RO	None

A string containing the logical name you specified when opening the current IVI session. You may pass a logical name to <u>niRFSA\_init</u> or <u>niRFSA\_InitWithOptions</u>. The IVI Configuration Utility must contain an entry for the logical name. The logical name entry refers to a driver session section in the IVI Configuration file. The driver session section specifies a physical device and initial user options.

## NIRFSA\_ATTR\_IO\_RESOURCE\_DESCRIPTOR

Data type	Access	High Level Functions
ViString	RO	None

Indicates the resource name NI-RFSA uses to identify the physical device. If you initialize NI-RFSA with a logical name, this attribute contains the resource name that corresponds to the entry in the IVI Configuration Utility.

If you initialize NI-RFSA with the resource name, this attribute contains that value.

## NIRFSA\_ATTR\_DRIVER\_SETUP

Data type	Access	High Level Functions
ViString	RO	None

The DriverSetup string is used to set the initial values for attributes that are specific to NI-RFSA.

The format of the DriverSetup string is:

Tag: Value

*Tag* is the name of the DriverSetup string attribute. *Value* is the value set to the attribute. To set multiple attributes, separate their assignments with a semicolon.

The DriverSetup string can include the following tags:

Digitizer—Specifies the resource name of the digitizer to use for this session. If this DriverSetup tag is not specified, the resource name for the downconverter associated in MAX is used, for example, DriverSetup=Digitizer:pxi1slot4

Refer to <u>niRFSA\_InitWithOptions</u> for additional information about the **optionsString** parameter. Refer to the <u>NI RF Signal Analyzers Getting</u> <u>Started Guide</u> for information on MAX setup.

Default Value: "" (empty string)

## **Operating System Support**

For information about the supported operating system (OS) for your device, refer to the <u>NI RF Signal Analzyers Readme</u>.



**Note** Some devices are not supported under Windows Vista. Refer to the <u>NI RF Signal Analzyers Readme</u> for a complete list of products and their OS support.

## **Important Information**

Warranty Copyright Trademarks Patents Warning Regarding Use of NI Products

## **Technical Support and Professional Services**

Visit the following sections of the National Instruments Web site at ni.com for technical support and professional services:

- <u>Support</u>—Online technical support resources at ni.com/support include the following:
  - Self-Help Resources—For answers and solutions, visit the award-winning National Instruments Web site for software drivers and updates, a searchable <u>KnowledgeBase</u>, product <u>manuals</u>, step-by-step troubleshooting wizards, thousands of example programs, tutorials, application notes, instrument drivers, and so on.
  - Free Technical Support—All registered users receive free Basic Service, which includes access to hundreds of Applications Engineers worldwide in the <u>NI Discussion</u> Forums at ni.com/forums. National Instruments Applications Engineers make sure every question receives an answer.

For information about other <u>technical support options</u> in your area, visit ni.com/services or <u>contact</u> your local office at ni.com/contact.

- Training and Certification—Visit ni.com/training for self-paced training, eLearning virtual classrooms, interactive CDs, and Certification program information. You also can register for instructor-led, hands-on courses at locations around the world.
- <u>System Integration</u>—If you have time constraints, limited in-house technical resources, or other project challenges, National Instruments Alliance Partner members can help. To learn more, call your local NI office or visit ni.com/alliance.
- Declaration of Conformity (DoC)—A DoC is our claim of compliance with the Council of the European Communities using the manufacturers declaration of conformity. This system affords the user protection for electronic compatibility (EMC) and product safety. You can obtain the DoC for your product by visiting ni.com/certification.
- <u>Calibration Certificate</u>—If your product supports calibration, you can obtain the calibration certificate for your product at ni.com/calibration.

If you searched ni.com and could not find the answers you need, contact your local office or NI corporate headquarters. You also can visit the <u>Worldwide Offices</u> section of ni.com/niglobal to access the branch office Web sites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.

## Warranty

The NI PXI-5661 is warranted against defects in materials and workmanship for a period of one year from the date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace equipment that proves to be defective during the warranty period. This warranty includes parts and labor.

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this document is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

## Copyright

Under the copyright laws, this publication may not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of National Instruments Corporation.

National Instruments respects the intellectual property of others, and we ask our users to do the same. NI software is protected by copyright and other intellectual property laws. Where NI software may be used to reproduce software or other materials belonging to others, you may use NI software only to reproduce materials that you may reproduce in accordance with the terms of any applicable license or other legal restriction.

## Trademarks

National Instruments, NI, ni.com, and LabVIEW are trademarks of National Instruments Corporation. Refer to the *Terms of Use* section on ni.com/legal for more information about <u>National Instruments trademarks</u>.

FireWire® is the registered trademark of Apple Computer, Inc.

Handle Graphics®, MATLAB®, Real-Time Workshop®, Simulink®, Stateflow®, and xPC TargetBox® are registered trademarks, and TargetBox™ and Target Language Compiler™ are trademarks of The MathWorks, Inc.

Tektronix® and Tek are registered trademarks of Tektronix, Inc.

Other product and company names mentioned herein are trademarks or trade names of their respective companies.

Members of the National Instruments Alliance Partner Program are business entities independent from National Instruments and have no agency, partnership, or joint-venture relationship with National Instruments.

#### Patents

For patents covering National Instruments products, refer to the appropriate location: **Help**»**Patents** in your software, the patents.txt file on your CD, or <u>ni.com/patents</u>.

# WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS

(1) NATIONAL INSTRUMENTS PRODUCTS ARE NOT DESIGNED WITH COMPONENTS AND TESTING FOR A LEVEL OF RELIABILITY SUITABLE FOR USE IN OR IN CONNECTION WITH SURGICAL IMPLANTS OR AS CRITICAL COMPONENTS IN ANY LIFE SUPPORT SYSTEMS WHOSE FAILURE TO PERFORM CAN REASONABLY BE EXPECTED TO CAUSE SIGNIFICANT INJURY TO A HUMAN.

(2) IN ANY APPLICATION, INCLUDING THE ABOVE, RELIABILITY OF OPERATION OF THE SOFTWARE PRODUCTS CAN BE IMPAIRED BY ADVERSE FACTORS, INCLUDING BUT NOT LIMITED TO FLUCTUATIONS IN ELECTRICAL POWER SUPPLY, COMPUTER HARDWARE MALFUNCTIONS, COMPUTER OPERATING SYSTEM SOFTWARE FITNESS, FITNESS OF COMPILERS AND DEVELOPMENT SOFTWARE USED TO DEVELOP AN APPLICATION, INSTALLATION ERRORS. SOFTWARE AND HARDWARE COMPATIBILITY PROBLEMS, MALFUNCTIONS OR FAILURES OF ELECTRONIC MONITORING OR CONTROL DEVICES, TRANSIENT FAILURES OF ELECTRONIC SYSTEMS (HARDWARE AND/OR SOFTWARE), UNANTICIPATED USES OR MISUSES, OR ERRORS ON THE PART OF THE USER OR APPLICATIONS DESIGNER (ADVERSE FACTORS SUCH AS THESE ARE HEREAFTER COLLECTIVELY TERMED "SYSTEM FAILURES"). ANY APPLICATION WHERE A SYSTEM FAILURE WOULD CREATE A RISK OF HARM TO PROPERTY OR PERSONS (INCLUDING THE RISK OF BODILY INJURY AND DEATH) SHOULD NOT BE RELIANT SOLELY UPON ONE FORM OF ELECTRONIC SYSTEM DUE TO THE RISK OF SYSTEM FAILURE. TO AVOID DAMAGE. INJURY. OR DEATH. THE USER OR APPLICATION DESIGNER MUST TAKE REASONABLY PRUDENT STEPS TO PROTECT AGAINST SYSTEM FAILURES. INCLUDING BUT NOT LIMITED TO BACK-UP OR SHUT DOWN MECHANISMS. BECAUSE EACH END-USER SYSTEM IS CUSTOMIZED AND DIFFERS FROM NATIONAL INSTRUMENTS' TESTING PLATFORMS AND BECAUSE A USER OR APPLICATION DESIGNER MAY USE NATIONAL INSTRUMENTS PRODUCTS IN COMBINATION WITH OTHER PRODUCTS IN A MANNER NOT EVALUATED OR CONTEMPLATED BY NATIONAL INSTRUMENTS, THE USER OR

APPLICATION DESIGNER IS ULTIMATELY RESPONSIBLE FOR VERIFYING AND VALIDATING THE SUITABILITY OF NATIONAL INSTRUMENTS PRODUCTS WHENEVER NATIONAL INSTRUMENTS PRODUCTS ARE INCORPORATED IN A SYSTEM OR APPLICATION, INCLUDING, WITHOUT LIMITATION, THE APPROPRIATE DESIGN, PROCESS AND SAFETY LEVEL OF SUCH SYSTEM OR APPLICATION.

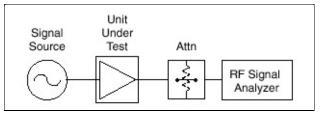
## **1 dB Gain Compression Measurement**

An amplifier maintains a constant gain for low-level input signals. However, at higher input levels, the amplifier goes into saturation and its gain decreases. The 1 dB compression point ( $P_{1dB}$ ) indicates the power level that causes the gain to drop by 1 dB from its small signal value.

#### **Measurement Setup**

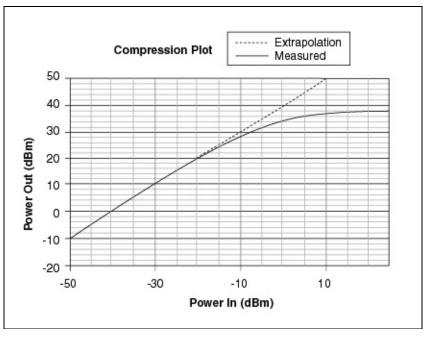
Measuring the 1 dB gain compression point of a device requires driving the UUT into compression without driving the RF Signal Analyzer into compression. This requires proper attenuation at the RF Signal Analyzer and a signal source of sufficient power to compress the UUT. You can apply attenuation by programming the internal input attenuators or by using external attenuation.

The 1 dB compression point is derived from the gain relationship between output power and input power. Using the measurement setup shown in the figure below, source amplitude is slowly increased while the UUT output is monitored.



#### Typical 1 dB Gain Compression Setup

Output power is plotted against input power as shown in the following figure.



#### Gain Compression Plot

The straight line on this graph is an extrapolation of the small signal gain

of the UUT. The input 1 dB compression point is the input power that causes the UUT gain to drop by 1 dB from this small signal value, or approximately 12 dBm in this case.

#### **Understanding RF Signal Analyzer Compression Limits**

Like all signal analysis devices, the RF Signal Analyzer is not completely linear and will eventually reach compression. However, the RF Signal Analyzer architecture possesses a high degree of linearity, and its compression point is typically 5 dBm or higher.

Ensure accurate UUT compression measurements by limiting the signal at the RF Signal Analyzer input mixer to 20 dB below the compression point listed in the *NI PXI-5660 RF Vector Signal Analyzer Specifications* document included in your RF Signal Analyzer kit.

## Choosing the Optimal RF Signal Analyzer Attenuation Setting

Choosing the optimal attenuation settings for a UUT compression measurement requires you take the following factors into account:

- The maximum output signal of your UUT must be attenuated to 1020 dB less than the compression point of the RF Signal Analyzer.
- The resolution bandwidth setting of the RF Signal Analyzer must be low enough that small signals used to determine the linear gain of the UUT are not overwhelmed with noise from the RF Signal Analyzer.

To set the proper RF Signal Analyzer attenuation level for a compression test on a UUT with known output compression estimate and known approximate gain, complete the following steps:

1. Set the RF Signal Analyzer mixer level to 20 dBm and its reference level to 10 dB above the estimated UUT compression point.

*mixer level = reference level – attenuation.* 

- 2. Set the RF Signal Analyzer center frequency to your intended testing frequency, its span to 1 MHz, and its resolution bandwidth to 1 kHz.
- 3. Inject a signal into the UUT small enough that its output level is at least 20 dB below the estimated UUT compression point. If the UUT output signal level is too close to the noise floor of the RF Signal Analyzer, decrease the RF Signal Analyzer resolution bandwidth.
- 4. Increase the input signal to the UUT. If the output signal has reached 5 dB below the RF Signal Analyzer reference level and compression of the UUT has not been reached, increase the reference level by 10 dB.
- 5. Repeat step 4 until compression appears in the UUT.

The setting you obtain is the optimal attenuation setting.

## **Branch Offices**

Office	<b>Telephone Number</b>
Australia	1800 300 800
Austria	43 662 457990-0
Belgium	32 (0) 2 757 0020
Brazil	55 11 3262 3599
Canada	800 433 3488
China	86 21 6555 7838
Czech Republic	420 224 235 774
Denmark	45 45 76 26 00
Finland	385 (0) 9 725 72511
France	33 (0) 1 48 14 24 24
Germany	49 89 7413130
India	91 80 41190000
Israel	972 0 3 6393737
Italy	39 02 413091
Japan	81 3 5472 2970
Korea	82 02 3451 3400
Lebanon	961 (0) 1 33 28 28
Malaysia	1800 887710
Mexico	01 800 010 0793
Netherlands	31 (0) 348 433 466
New Zealand	0800 553 322
Norway	47 (0) 66 90 76 60
Poland	48 22 3390150
Portugal	351 210 311 210
Russia	7 495 783 6851
Singapore	1800 226 5886
Slovenia	386 3 425 42 00

South Africa	27 0 11 805 8197
Spain	34 91 640 0085
Sweden	46 (0) 8 587 895 00
Switzerland	41 56 2005151
Taiwan	886 02 2377 2222
Thailand	662 278 6777
Turkey	90 212 279 3031
United Kingdom	44 (0) 1635 523545
United States (Corporate)	512 683 0100