Application Note

74-0042-180305

thinkRF

monitor. detect. analyze

Using the ThinkRF Real-Time Spectrum Analyzer with LabVIEW

Contents

Introduction

The ThinkRF Real-Time Spectrum Analyzer (RTSA) has the performance of traditional high-end lab spectrum analyzers at a fraction of the cost, size, weight and power consumption and is designed for distributed deployment.

LabVIEW is a system-design platform and development environment for a visual programming language from National Instruments. It is commonly used for instrument control, data acquisition, test and measurement and industrial automation on a variety of platforms including Microsoft Windows.

To make use of LabView system capability, ThinkRF provides RTSA LabVIEW API (Application Programming Interface), among other APIs, as part of ThinkRF RTSA solution. This Application Note will help you to easily and quickly integrate your ThinkRF RTSA into your existing or new NI LabVIEW system.

This Application Note will walk you through the ThinkRF provided RTSA LabVIEW API VI (Virtual Instrument) functions, as well as running an example. This AppNote also assumes you have some network knowledge and have access to a R5500. Otherwise, you can connect via the Internet to a ThinkRF's evaluation RTSA unit at www.thinkrf.com/demo.

Software and System Requirements

To use the RTSA LabVIEW API, the following software and system is required:

- Windows 7/higher 32-bit/64-bit operating system
- NI LabVIEW Full Development 2014 or later 32-bit/64-bit
- RTSAInterface.dll, a C++ DLL provided by ThinkRF (included in LabVIEW API release but might be download and updated separately).

The latest RTSA Firmware and DLL may be downloaded from http://www.thinkrf.com/firmware-updates/.

RTSA LabVIEW API VIs

All ThinkRF's RTSA LabVIEW API VIs have this icon  . The API could be open as a project in LabVIEW by selecting **File > Open Project...** and browse to the API folder to select *ThinkRF RTSA API.lvproj*. The API files are organized as shown in Figure Illustration and Illustration.

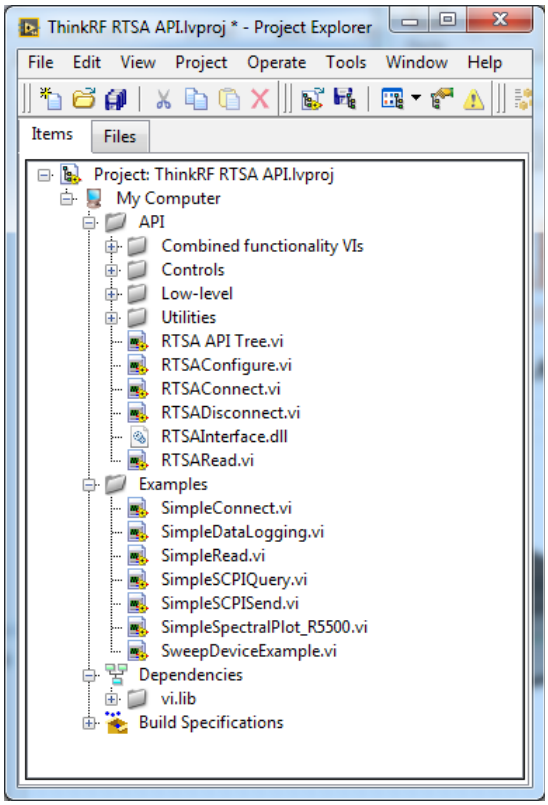

Figure 1: A tree view of the API files

RTSA API Tree.vi

 *RTSA API Tree.vi* is an illustrative VI (see Figure Illustration) that shows in block diagrams the organization of the RTSA VIs as well as how to use them. The VIs are divided into categories depending on their function.

This VI doesn't do any function and the Run button is normally shown as "broken" (i.e. has no effect). However, highly recommend to start with this VI to familiarize yourself with the API VIs. Clicking on a VI icon listed in this view would take you directly to that VI.

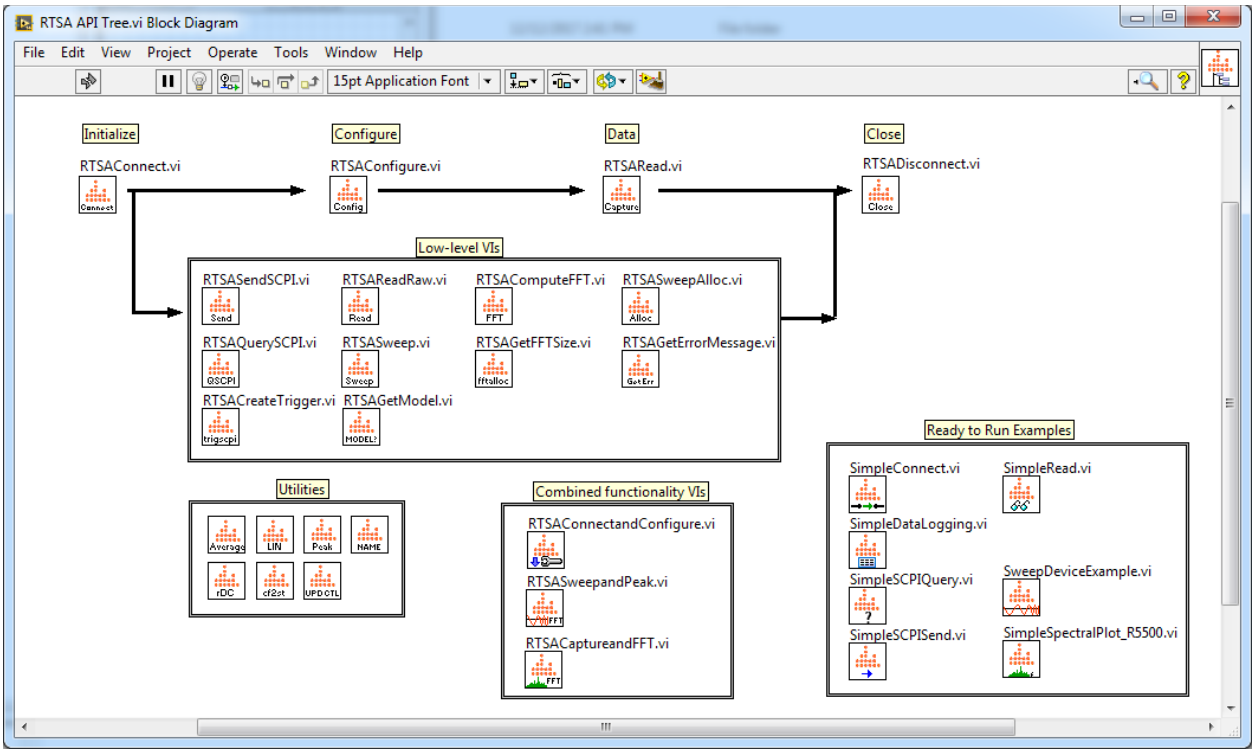

Figure 2: Content of *RTSA API Tree.vi*, illustrating the API Tree in blocks view

Main VIs

The following is list of the RTSA LabVIEW VIs that can be used to connect, configure, and acquire data from a RTSA.

1. *RTSAConnect.vi*

Establish a connection to the RTSA.

Inputs:

- RTSA IP Address (string): The IP Address of the RTSA.
- error in (LabVIEW error cluster): A LabVIEW error cluster to prevent the VI from executing if a previous VI had an error.

Outputs:

- RTSA_handle (64-bit integer): Represents the socket connection to the RTSA.
- error out (LabVIEW error cluster): Indicates whether an error has occurred.

2. *RTSAConfigure.vi*

Reset (*RST) the RTSA to default settings, get data acquisition access, and then verify and configure the RTSA to the provided settings.

Inputs:

- RTSA_handle (64-bit integer): The RTSA connection session.
- configuration cluster (LabVIEW Cluster): A cluster with the desired RTSA configuration. The configuration cluster's types are chosen with LabVIEW controls in mind. The cluster contains:
  - Frequency (64-bit integer): The center frequency (MHz)

- Mode (string): The RFE mode
- Sample Size (string): The sample size of each packet
- Number of packets (string): The number of packets to be captured
- PLL Reference (string): The PLL reference source (INT or EXT)
- Decimation (string): The decimation value
- Attenuator (string): The attenuation values in dB

- error in (LabVIEW error cluster): A LabVIEW error cluster to prevent the VI from executing if a previous VI had an error.

Outputs:

- RTSA_handle out (64-bit integer): The RTSA connection session returned.
- error out (LabVIEW error cluster): Indicates whether an error has occurred.

3. *RTSARead.vi*

Capture multiple IF data packets when the number_of_packets is greater than 1, and concatenate all the data packets into one LabVIEW Cluster.

Inputs:

- RTSA_handle (64-bit integer): The RTSA connection session.
- number_of_packets (32-bit integer): The number of packets to capture.
- timeout (unsigned 32-bit integer): The time delay before the socket stops trying to read data from the network socket.
- error in (LabVIEW error cluster): A LabVIEW error cluster to prevent the VI from executing if a previous VI had an error.

Outputs:

- RTSA_handle out (64-bit integer): The RTSA connection session returned.
- IQ Data (LabVIEW Cluster): The IF data stored in a cluster. Note multiple packets all concatenated if the number_of_packets is greater than 1. Please see *RTSAReadRaw.vi* for the data format.
- Context (LabVIEW Cluster): The context data of the IF packets. Please see *RTSAReadRaw.vi* for the format.
- error out (LabVIEW error Cluster): Indicates whether an error has occurred.

4. *RTSADisconnect.vi*

Close a connection to the RTSA.

Inputs:

- RTSA_handle (64-bit integer): Represents the socket connection to the RTSA.
- error in (LabVIEW error cluster): A LabVIEW error cluster to indicate if a previous error has occurred. Only the RTSADisconnect will attempt to disconnect and close the session with the RTSA unit even if a previous error has occurred.

Output:

- error out (LabVIEW error cluster): Indicates whether an error has occurred.

Low-Level VIs

This section consists of VIs that either are subset VIs to the main VIs mentioned before or perform other RTSA control, configuration and data processing functions.

1. *RTSASendSCPI.vi*

Send SCPI command to the RTSA. See RTSA's Programmer's Guide for the list of SCPI commands.

Inputs:

- RTSA_handle (64-bit integer): The RTSA connection session.
- SCPI_command (string): The SCPI command to be sent to the RTSA.
- error in (LabVIEW error cluster): A LabVIEW error cluster to prevent the VI from executing if a previous VI had an error.

Outputs:

- RTSA_handle out (64-bit integer): The RTSA connection session returned.
- error out (LabVIEW error cluster): Indicates whether an error has occurred.

2. *RTSAQuerySCPI.vi*

Send a SCPI query command to the RTSA and receive the response. See RTSA's Programmer's Guide for the list of SCPI query commands.

Inputs:

- RTSA_handle (64-bit integer): The RTSA connection session.
- SCPI_command (string): The SCPI command to be queried to the RTSA.
- error in (LabVIEW error cluster): A LabVIEW error cluster to prevent the VI from executing if a previous VI had an error.

Outputs:

- RTSA_handle out (64-bit integer): The RTSA connection session returned.
- Query_response (string): The returned value of the SCPI query.
- error out (LabVIEW error cluster): Indicates whether an error has occurred.

3. *RTSAGetModel.vi*

Get the model variant of the RTSA unit (Ex: 408, 418, 427).

Inputs:

- RTSA_handle (64-bit integer): The RTSA connection session.
- RTSA IP Address (string): The IP Address of RTSA.
- error in (LabVIEW error cluster): A LabVIEW error cluster to prevent the VI from executing if a previous VI had an error.

Outputs:

- RTSA_handle out (64-bit integer): The RTSA session connection returned.
- Model (string): The model variant of the RTSA unit (Ex: 408, 418, 427).
- error out (LabVIEW error cluster): Indicates whether an error has occurred.

4. *RTSAReadRaw.vi*

Read an I/Q IF data packet along with the relevant context packets. The data packet size is what being last set in the device or of power-up (or *RST) default of 1024.

Inputs:

- RTSA_handle (64-bit integer): The RTSA connection session.

- timeout (unsigned 32-bit integer): The time delay before the socket stops trying to read data from the network socket.
- error in (LabVIEW error cluster): A LabVIEW error cluster to prevent the VI from executing if a previous VI had an error.

Outputs:

- RTSA_handle out (64-bit integer): The RTSA connection session returned.
- IQ Data (LabVIEW Cluster): The IF data stored in a cluster. Note the cluster will have different values depending on the RFE mode of the RTSA. The cluster contains:
  - i_data (16-bit integer): 16-bit I data
  - q_data (16-bit integer): 16-bit Q data
  - i32_data (32-bit integer): 32-bit I-data for HDR mode
- Context (LabVIEW Cluster): The context data of the IF packet. The cluster contains:
  - stream_id (unsigned 32-bit integer): A VRT stream ID that identifies the data format as I16/Q16 or I32 (See the Programmer's Guide)
  - spec_inv (unsigned 8-bit integer): An integer indicating whether spectral inversion is present (0 - no inversion, 1 - there is inversion). See the Programmer's Guide for further explanation.
  - samples_per_packet (32-bit integer): The number of samples in the data packet returned
  - timestamp_sec (unsigned 32-bit_integer): The UTC timestamp value in seconds indicating when the data was captured
  - timestamp_psec (unsigned 64-bit integer:) The number of picoseconds passed since the last increment of the UTC timestamp seconds field
  - reference_level (integer 16-bit): A power reference level value, in dBm, to be applied to the computed spectral data calculated basing on the received data. See the Programmer's Guide for the usage.
  - bandwidth (64-bit integer): The full bandwidth of the IF data received, in Hz
  - center_freq (64-bit integer): The center frequency of the IF data received, in Hz
- error out (LabVIEW error cluster): Indicates whether an error has occurred.

5. *RTSAGetErrorMessage.vi*

Query the RTSA for an error message.

Inputs:

- error_code (16-bit integer): The code of the error to be investigated.
- error in (LabVIEW error cluster): A LabVIEW error cluster to prevent the VI from executing if a previous VI had an error.

Outputs:

- error_message (string): The returned error message.
- error out (LabVIEW error cluster): Indicates whether an error has occurred.

6. *RTSAGetFFTSize.vi*

Retrieve the size of a buffer required to store the FFT data.

Inputs:

- Context (LabVIEW Cluster): The context data of the IF packets, which could be retrieved from *RTSAReadRaw.vi*.

- error in (LabVIEW error cluster): A LabVIEW error cluster to prevent the VI from executing if a previous VI had an error.

Outputs:

- memory_size (32-bit integer) The size required for the FFT buffer.
- error out (LabVIEW error cluster): Indicates whether an error has occurred.

7. *RTSAComputeFFT.vi*

Compute the spectral data based on the given time domain data and context data.

Inputs:

- Context (LabVIEW Cluster): The context data of the IF packets. See *RTSAReadRaw.vi* for format.
- IQ Data (LabVIEW Cluster): The time domain data. See *RTSAReadRaw.vi* for format.
- error in (LabVIEW error cluster): A LabVIEW error cluster to prevent the VI from executing if a previous VI had an error.

Outputs:

- Amplitude (Double precision float array): Stores the computed spectral data (dBm).
- frequency spectrum (64-bit integer array): Stores frequency values which correspond to the spectral data.
- error out (LabVIEW error cluster): Indicates whether an error has occurred.

8. *RTSASweepAlloc.vi*

Configure the RTSA to the desired sweep configuration and retrieve the size of the buffer required basing on the given settings to store the sweep data obtained from *RTSASweep.vi*.

Inputs:

- RTSA_handle (64-bit integer): The RTSA session.
- sweep_config (LabVIEW Cluster): The configuration of the sweep. Below is the structure of the cluster:
  - start_freq (64-bit integer): The start frequency of the sweep (MHz)
  - stop_freq (64-bit integer): The stop frequency of the sweep (MHz)
  - rbw (unsigned 64-bit integer): The RBW of the sweep (kHz)
  - mode (string): The RFE mode of the sweep
  - Attenuator (string): The 0, 10, 20 and 30 dB attenuation value
- error in (LabVIEW error cluster): A LabVIEW error cluster to prevent the VI from executing if a previous VI had an error.

Outputs:

- RTSA_handle out (64-bit integer): The RTSA session connection returned.
- buff_size (Unsigned 32-bit integer): The size of the buffer required to store the spectral data.
- sweep_config out (LabVIEW Cluster): The configuration of the sweep returned.
- error out (LabVIEW error cluster): Indicates whether an error has occurred.

9. *RTSASweep.vi*

Read swept spectral data from the RTSA.

Inputs:

- sweep_config (LabVIEW Cluster): The configuration of the sweep. See *RTSASweepAlloc.vi* for the cluster's content.
- buff_size (Unsigned 32-bit integer): The expected size of the spectral data. See the buff_size from *RTSASweepAlloc.vi*.
- error in (LabVIEW error cluster): A LabVIEW error cluster to prevent the VI from executing if a previous VI had an error.

Outputs:

- fft_buffer (Double precision float array): Stores the swept spectral data.
- error out (LabVIEW error cluster): Indicates whether an error has occurred.

10. RTSACreateTrigger.vi

Creates and sends the SCPI for the Trigger functionality.

Inputs:

- RTSA_handle (64-bit integer): The RTSA connection session .
- Trigger settings (LabVIEW Cluster): the front panel trigger settings to be sent to the RTSA unit. The cluster consists of the following parameters:
  - Trigger (Boolean): Set the trigger to active to disabled
  - Start Frequency (MHz) (64-bit integer): The trigger frequency start threshold
  - Stop Frequency (MHz) (64-bit integer): The trigger frequency stop threshold
  - Amplitude (dBm) (64-bit integer): The trigger dBm amplitude threshold
- error in (LabVIEW error cluster): A LabVIEW error cluster to prevent the VI from executing if a previous VI had an error.

Outputs:

- RTSA_handle out (64-bit integer): The RTSA connection session returned.
- error out (LabVIEW error cluster): Indicates whether an error has occurred.

Combined Functionality VIs

The following VIs are used in the Example VIs as they are combination of the previously mentioned VIs.

1. RTSAConnectandConfigure


Connects to the RTSA unit, configures the acquisition and set trigger. Used in the *SimpleSpectralPlot.vi* example.

Inputs:

- settings_in (LabVIEW Cluster): All the controls on the example front panel.
- error in (LabVIEW error cluster): A LabVIEW error cluster to prevent the VI from executing if a previous VI had an error.

Outputs:

- settings_out (LabVIEW Cluster): All the controls on the example front panel returned.
- error out (LabVIEW error cluster): Indicates whether an error has occurred.

2. RTSASweepandPeak.vi


Uses the RTSA to sweep a frequency range and find the peak. Used in *SweepDeviceExample.vi* example.

Inputs:

- RTSA_handle (64-bit integer): The RTSA connection session.
- sweep_config (LabVIEW Cluster): The configuration of the sweep. See *RTSASweepAlloc.vi* for the cluster's content.
- buff_size (Unsigned 32-bit integer): The size of the buffer required to store the spectral data.
- error in (LabVIEW error cluster): A LabVIEW error cluster to prevent the VI from executing if a previous VI had an error.

Outputs:

- RTSA_handle out (64-bit integer): The RTSA connection session returned.
- Spectral Plot (LabVIEW Cluster): Stores amplitude and frequency values which correspond to the spectral data.
- Max Power Level (Double precision float): Power level of peak (dBm).
- Frequency of Max Power (64-bit integer): Frequency of peak (Hz).
- error out (LabVIEW error cluster): Indicates whether an error has occurred.

3. *RTSACaptureandFFT.vi*

Uses the RTSA to capture I/Q data and calculate the FFT spectrum. Used in the *SimpleSpectralPlot.vi* example.

Inputs:

- RTSA_handle (64-bit integer): The RTSA connection session.
- number_of_packets (32-bit integer): The number of packets to capture.
- Mode (string): The RFE mode.
- error in (LabVIEW error cluster): A LabVIEW error cluster to prevent the VI from executing if a previous VI had an error.

Outputs:

- RTSA_handle out (64-bit integer): The RTSA connection session returned.
- frequency spectrum (c) Stores amplitude and frequency values which correspond to the spectral data.
- error out (LabVIEW error cluster): Indicates whether an error has occurred.

Utility VIs

The following are utility VIs that can be used to do simple operations that are useful subset to the Utility VIs.

1. *average.vi*

Compute the average of a given 16-bit array.

Input:

- array (16-bit integer array): Input array.

Output:

- average (Double Precision Float) Value containing the average.

2. *centerSpanToStartStop.vi*

Use a given context cluster to compute the start/stop frequency of a given IF data packet.

Input:

- Context (LabVIEW Cluster): The context data of the IF packets. Please see *RTSAReadRaw.vi* for the format.

Outputs:

- Frequency Start (Double Precision Float) Start frequency of IF data (Hz).
- Frequency Stop (Double Precision Float) Stop frequency of IF data (Hz).

3. *findPeak.vi*

Find the peak spectral power and frequency values of given spectral data and frequency arrays.

Inputs:

- Spectral Data (Double precision float array): Array containing spectral data, in dBm.
- Frequency Data (64-bit integer array): Array containing frequency points, in Hz.

Outputs:

- Max Power Level (Double precision float): Peak power level, in dBm.
- Frequency Of Max Power (64-bit integer): Frequency of the peak power, in Hz.

4. *linspace.vi*

Compute a linear space operation on a start, stop, and size values which results in an array.

Inputs:

- start (Double precision float): First value of array.
- stop (Double precision float): Last value of array.
- size (32-bit integer): The size of the array.

Output:

- array (Double precision float array): An array with start/stop/size matching the input values.

5. *removeDC.vi*

Remove the DC offset from a given LabVIEW Cluster.

Input:

- Data (LabVIEW Cluster): The time domain data. Please see *RTSAReadRaw.vi* for the data format.

Outputs:

- i_data (Double precision float array): I data that with DC offset corrected.
- q_data (Double precision float array): Q data with DC offset corrected.

6. *automatedName.vi*

Creates a path for ASCII file name with the current date and time in the VI directory.

Output:

- New_path (Path): Absolute path to the new ASCII file.

7. *updateCtrls.vi*

Update some front panel controls depending on the selected configuration.

Inputs:

- Cen_freq (LabVIEW reference): Reference to the Center Frequency numeric control.
- Decim (LabVIEW reference): Reference to the Decimation combobox control.
- Trig_conf (LabVIEW reference): Reference to the Trigger Configuration cluster control.
- Freq_spect (LabVIEW reference): Reference to the Frequency Spectrum graph plot.
- Mode (string): The RFE mode.
- error in (LabVIEW error cluster): A LabVIEW error cluster to prevent the VI from executing if a previous VI had an error.

Output:

- error out (LabVIEW error cluster): Indicates whether an error has occurred.

Running the RTSA Example VIs

Several RTSA example VIs can be found within the *"..\Examples"* directory of the API Package (as shown in Figure Illustration and Illustration).

For this application note, we will use the *SimpleSpectralPlot.vi* as an example. The *SimpleSpectralPlot.vi* example *(..\Examples\SimpleSpectralPlot.vi)* demonstrates how a user can use the RTSA LabVIEW API to configure, acquire time-domain data, calculate spectral data and plot the data. The example uses a subset of the provided VIs in state-machine architecture which are described in the previous sections.

Please follow the instructions below to launch and use the *SimpleSpectralPlot.vi*.

1. Double clicking on *SimpleSpectralPlot.vi* to launch the example
2. Launching the *SimpleSpectralPlot.vi* will launch the control panel and the display (see Figure Illustration). At this point, it is important to note a few key features:
    - The VI resets the front panel controls to their initial values
    - The RTSA is connected to the specified IP address and configured with the specified settings using the *RTSAConnectandConfigure.vi*.
    - The VI uses *RTSACaptureandFFT.vi* to read the IQ data and context then compute the FFT:
        - The *RTSARead.vi* reads the time IQ data, as well as context data.
        - The *computeFFT.vi* provides the spectral data to the Frequency Spectrum plot.

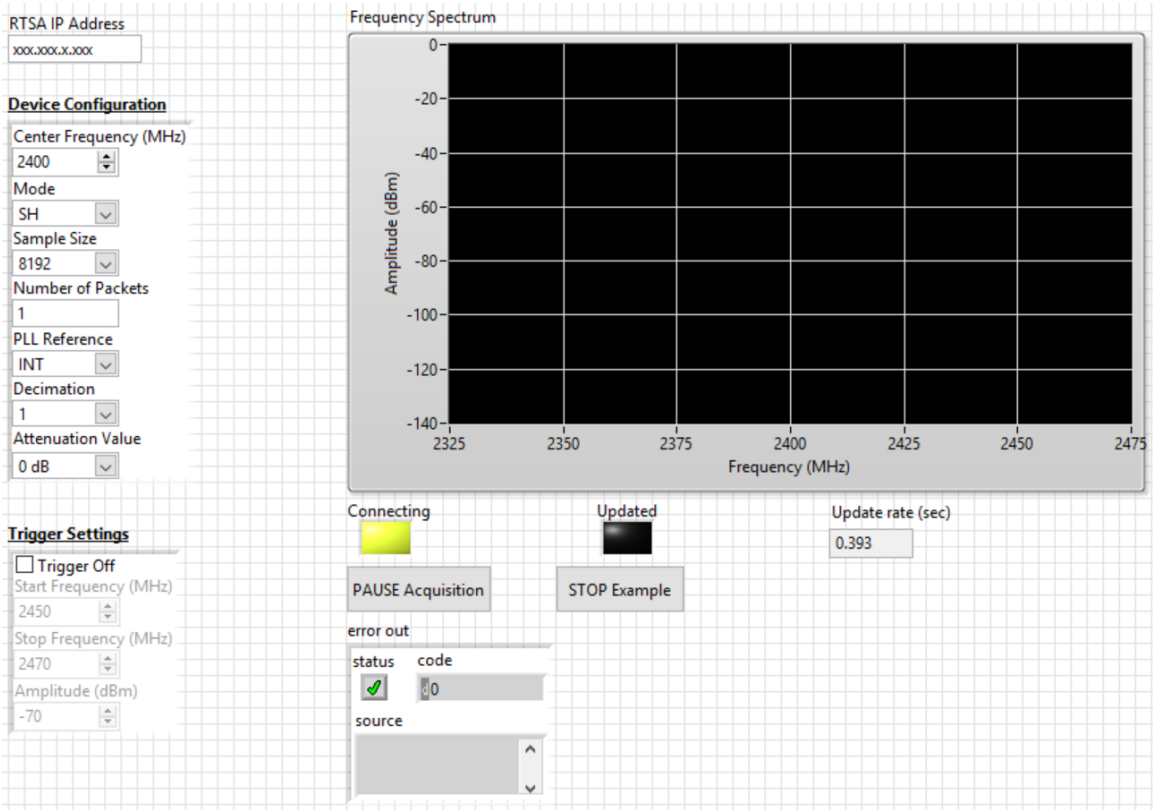

Figure 3: An example of the input fields before running

3. The VI waits for user interaction or update in the “Idle” state, if no change is requested the *RTSACaptureandFFT.vi* is called again. Otherwise, the old connection is closed, and a new connection is started by the *RTSAConnectandConfigure.vi* again
4. Modify the input fields in Figure Illustration as needed. *If Trigger Settings are used, make sure the Start and Stop Frequencies are within range with the Center Frequency.* See Programmer’s Guide as needed.

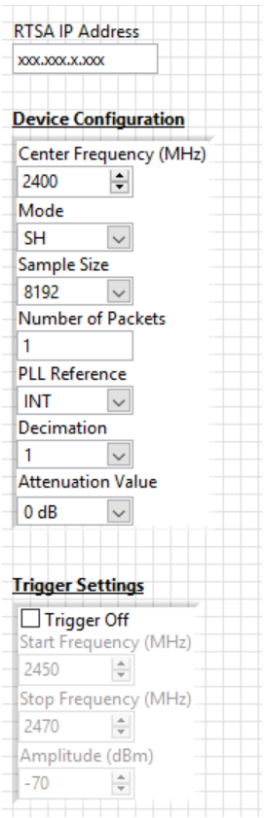

Figure 4: *SimpleSpectralPlot.vi* front panel input fields

5. Run the *SimpleSpectralPlot.vi* by clicking on the Run button.

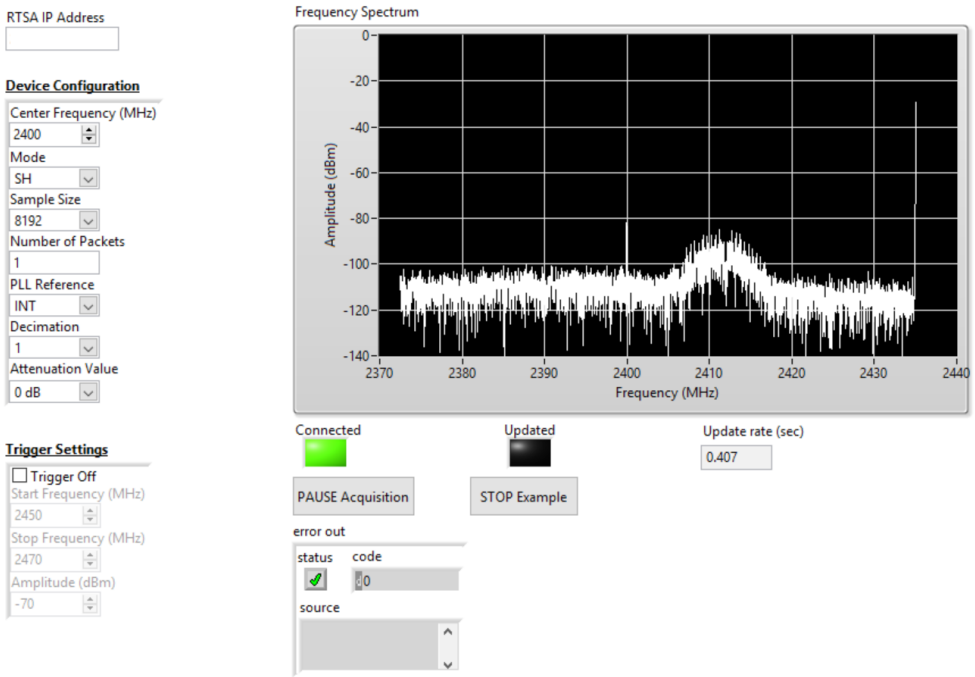

Figure 5: An example of the *SimpleSpectralPlot.vi* running

Document Revision History

This section summarizes document revision history.

Document Version	Release Date	Revisions and Notes
v1.0	Mar 05, 2018	First document release for RTSA LabVIEW API v2.0.0

Contact us for more information

ThinkRF Support website provides online documents for resolving technical issues with ThinkRF products at http://www.thinkrf.com/resources.

For all customers who hold a valid end-user license, ThinkRF provides technical assistance 9 AM to 5 PM Eastern Time, Monday to Friday. Contact us at support@thinkrf.com, sales@thinkrf.com or by calling **+1.613.369.5104**.