Application Note

74-0042-190503

Using the ThinkRF Real-Time Spectrum Analyzer with LabVIEW API v2.1.0

thinkRF

monitor. detect. analyze

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 for controlling an RTSA and performing data capture and analysis. This Application Note will help you to easily and quickly integrate your ThinkRF RTSA into your 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 supported RTSA (see Supported RTSA Product List). 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
- 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 & APIs may be downloaded from https://www.thinkrf.com/download-updates/.

Supported RTSA Product List

LabVIEW API v2.1.0 currently supports RTSA R5xx0 product and its associated models (408, 418, 427) .

Note that the API does work with WSA5000 excepts for where there are SCPI differences, especially with the attenuation setting included in the API and examples (set attenuation to 0 if the examples are used, and manually change attenuation through *RTSASendSCPI.vi* function or *SimpleSCPISend.vi* Example). See for R5500 Programmer's Guide for the complete list of differences between R5500 vs WSA5000.

Installation Instructions

Getting The Right DLL

In the API folder, there are 32-bit and 64-bit DLLs provided. By default, *rtsaInterface.dll* is of 32-bit. If your LabVIEW system is of 64-bit version, you might want to rename *rtsaInterfcex64.dll* to *rtsaInterface.dll*.

Install API Using VI Package Manager

Note: This section applied to LabVIEW version 2017 or higher only as *.vip file is compiled using LabVIEW 2017 version. It won't work on versions earlier than 2017.

Perform the following steps to install the provided API's vip package to your PC:

1. Double click on the included ***.vip** file or open the ***.vip** file in VI Package Manager.
2. Follow the instructions to install the package on your PC.
3. After the installation, the API VI functions could be found in the function palette under ThinkRF RTSA API.
4. The Examples included with the API could be found in the Example Finder (**Help** >> **Find Examples**). Then in **NI Example Finder** window, under **Browse**, select **Directory Structure**. Look for **ThinkRF** >> **ThinkRF RTSA API** >> **Examples**.

Using API Without Installation

Alternatively, the API VIs could be incorporate into your project directly or open as a project in LabVIEW by selecting **File** >> **Open Project...** and browse to the API folder to select *ThinkRF RTSA API.lvproj*.

RTSA LabVIEW API VIs

All ThinkRF's RTSA LabVIEW API VIs have this icon  . The API files are organized as shown in Figures 1 and 2.

Note: The API VIs does not cover all available settings for the RTSA. Additional settings or any direct SCPI commands mentioned in a RTSA's Programmer's Guide could be done through *RTSASendSCPI.vi* & *RTSAQuerySCPI.vi* functions, or *SimpleSCPISend.vi* & *SimpleSCPIQuery.vi* included in the Examples folder.

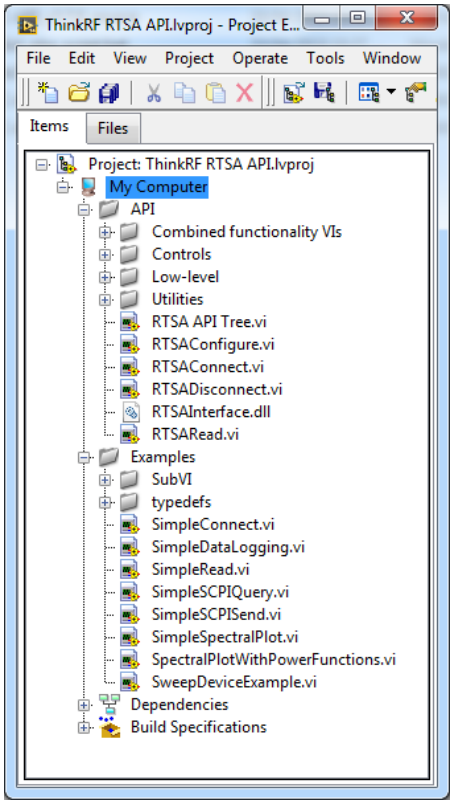

Figure 1: A tree view of the API files

RTSA API Tree.vi

RTSA API Tree.vi is an illustrative VI (see Figure 2) 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.

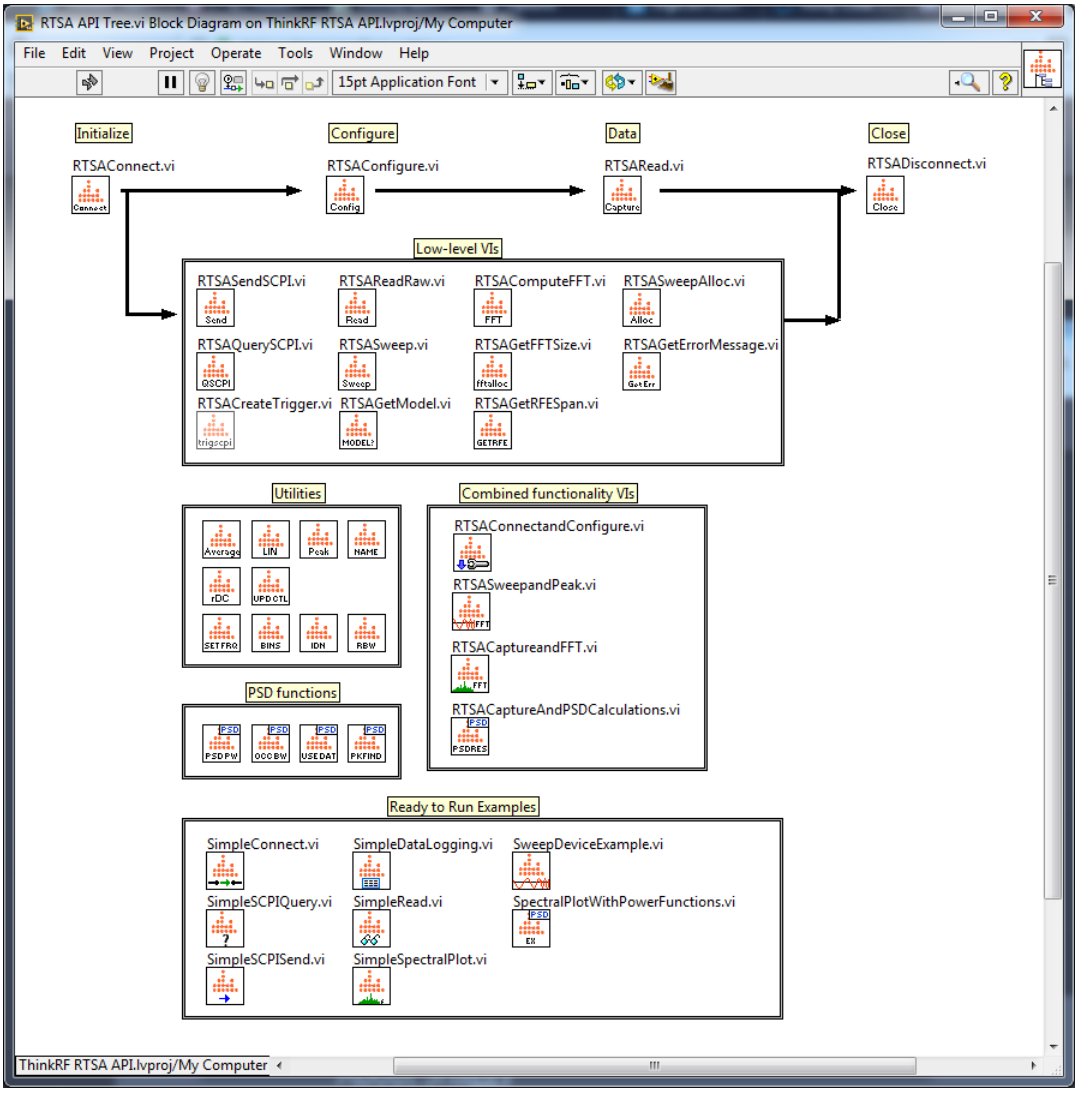

Figure 2: Content of *RTSA API Tree.vi*, illustrating the API Tree in blocks view

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. *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.

3. *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 Per Packet (string): The sample size of each VRT packet
  - Number of Packets (string): The number of packets to be captured per block
  - PLL Reference (string): The RTSA's PLL reference source, INT(ernal) or EXT(ernal)
  - 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.

4. *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.

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. They are grouped by similar functionality.

Commands Related VIs

1. *RTSASendSCPI.vi*

Sends SCPI command to the RTSA. See the 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*

Sends a SCPI query command to the RTSA and receive the response. See the 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. *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.

4. *RTSAGetModel.vi*

Gets 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.

5. *RTSACreateTrigger.vi*

Creates and sends the SCPI commands to set up the Frequency-Domain Level 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 or to disable.
  - Start Frequency (64-bit integer): The trigger's frequency start (MHz).
  - Stop Frequency (64-bit integer): The trigger's frequency stop (MHz).
  - Amplitude (64-bit integer): The trigger amplitude threshold (dBm).
- 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.

Data Capture and Processing Related VIs

6. *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 of the captured packet.
  - 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. If *RTSAComputeFFT.vi* is used to process the raw data, the reference level has been added to the output of this function.
  - 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.

7. RTSAGetRFESpan.vi

Gets the data span information of the read operation *RTSAReadRaw.vi*. The output bandwidth, start & stop frequencies are of the RFE 'full' bandwidth (ie. not of the useable/operating range). This function is useful (for plotting for example) as in some RFE modes, the IF center frequency is not at half the instantaneous bandwidth.

Inputs:

- RTSA_handle (64-bit integer): The RTSA session.
- Mode (string): The RFE mode.
- spec_inv (unsigned 8-bit integer): The spectral inversion indicator for the given frequency. Indicator is provided 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:

- RTSA_handle out (64-bit integer): The RTSA connection session returned.
- RFEspan (LabVIEW cluster): The RFEspan for the current acquisition operation
  - span_bw (unsigned 32-bit integer): bandwidth
  - span_center (64-bit integer): center frequency
  - span_fstart (64-bit integer): start frequency
  - span_fstop (64-bit integer): stop frequency
- error out (LabVIEW error cluster): Indicates whether an error has occurred.

8. *RTSAGetFFTSize.vi*

Retrieve the size of a buffer required to store the FFT data. This function is used with *RTSAComputeFFT.vi*.

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.

9. *RTSAComputeFFT.vi*

Compute the FFT of the given time domain data and return the Power Spectral Density (PSD) data. This VI does the following:

- normalized the data
- applied DC Offset correction on I & Q data
- performed Hann Windowing and FFT
- compute PSD with calibrated reference level adjustment
- perform spectral inversion on the spectral data if spectral inversion in the context is 1.

To get only usable data of the operating range within the whole spectrum after this function, see *RTSAComputePSDUsableData.vi*.

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.

Power Spectral Density (PSD) Data Related VIs

10. *RTSAComputePSDUsableData.vi*

Calculates and returns the usable PSD related data for the given info. This is useful if, for instances, data are to be plotted or to find power related information.
Note: This function is not applicable for sweep-device capture.

Inputs:

- RTSA_handle (64-bit integer): The RTSA connection session.
- mode (string): String containing the RFE Mode.
- spec_inv (unsigned 8-bit integer): The spectral inversion indicator for the given frequency.

- spectral data (single array): Single floating point array containing the full bandwidth of PSD data (in dBm) to be trimmed to usable 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.
- Usable Data (usable_data_context): the usable data information:
  - usable_bandwidth (unsigned 32-bit integer): the bandwidth of the usable data (in Hz)
  - usable_fstart (64-bit integer): the start frequency of the usable data (in Hz)
  - usable_fstop (64-bit integer): the stop frequency of the usable data (in Hz)
  - usable_data_size (unsigned 32-bit integer): The size of the usable data array
- Usable Frequency Spectrum (freq_spect): The usable frequency and spectrum array
  - Frequency (double array): The corresponding frequencies for the Amplitude array (in MHz)
  - Amplitude (single array): The amplitude values of the spectral data (in dBm)
- error out (LabVIEW error cluster): Indicates whether an error has occurred.

11. *RTSAComputePSDChannelPower.vi*

Calculates the channel power of a range in the given spectral (PSD) data. *spectral_data* could be derived from I/Q data by calling:

- *RTSACaptureandFFT.vi* for trace capture or manual sweep capture using SCPI commands.
- *RTSASweep.vi* for sweep-device capture.

Inputs:

- start bin (unsigned 32-bit integer): The first bin where the channel power calculation should take place.
- stop bin (unsigned 32-bit integer): The last bin where the channel power calculation should take place.
- spectral data (single array): A floating point array containing the spectral data (in dBm).
- error in (LabVIEW error cluster): A LabVIEW error cluster to prevent the VI from executing if a previous VI had an error.

Outputs:

- Channel Power (PSD_channelPower): The resulting start_bin, stop_bin and channel power (in dBm)
  - start_bin (unsigned 32-bit integer): the start bin that was used for the calculation.
  - stop_bin (unsigned 32-bit integer): the stop bin that was used for the calculation.
  - Channel Power (single): the resulting channel power (in dBm).
  - error out (LabVIEW error cluster): Indicates whether an error has occurred.

12. *RTSAComputePSDOccupiedBandwidth.vi*

Calculates the occupied power bandwidth for the specified occupied percentage of the given spectral (PSD) data.
Spectal data of I/Q data could be derived from:

- in trace block capture mode, call *RTSAComputeFFT.vi*, but recommend to apply *RTSAComputPSDUsableData.vi* on the spectral_data before calling this function
- with sweep-device mode, call *RTSASweep.vi*

Inputs:

- RBW (32-bit integer): The resolution BW (RBW) value of the data (in Hz).

- spectral data (single array): the spectral data (in dBm).
- occupied percentage (single array): The channel power percentage (in %) to be used for calculating the corresponding occupied bandwidth.
- error in (LabVIEW error cluster): A LabVIEW error cluster to prevent the VI from executing if a previous VI had an error.

Outputs:

- occupied bandwidth (unsigned 64-bit integer): The calculated bandwidth (in Hz).
- error out (LabVIEW error cluster): Indicates whether an error has occurred.

13. RTSAPSDPeakFind.vi

Find the peak within the "usable (operating) bandwidth" range of the given spectral (PSD) data of a non-sweep-device capture mode.
Recommend to use this function for peak finding as SH/SHN/HDR's center frequency is not at 0 IF (or center of a viewing bandwidth).

Inputs:

- RTSA_handle (64-bit integer): The RTSA connection session.
- mode (string): The RFE mode.
- spectral data (double array): A floating point array containing the spectral data (in dBm).
- spectral inversion (unsigned 8-bit integer): The setting of the spectral inversion.
- 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.
- Peak (PSD_peakData): The resulting peak frequency (in MHz) and power level (in dBm).
- error out (LabVIEW error cluster): Indicates whether an error has occurred.

Sweep-Device Related VIs

14. RTSASweepAlloc.vi

Configure the RTSA to the desired sweep configuration and retrieve the size of the sweep 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.

15. *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 (in dBm).
- 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.vi*


Connects to the RTSA unit, configures the acquisition and sets the frequency-domain level 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 through a frequency range and finds 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 (MHz).
- error out (LabVIEW error cluster): Indicates whether an error has occurred.

3. *RTSACaptureandFFT.vi*

Uses the RTSA to capture a block of I/Q data and calculates 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 (LabVIEW cluster) Stores amplitude (dBm) and frequency values (MHz) which correspond to the spectral data.
- error out (LabVIEW error cluster): Indicates whether an error has occurred.

4. *RTSACaptureAndPSDCalculations.vi*

Reads a block of data and calculates the FFT prior to calling the PSD analysis functions on the data to calculate the usable data, occupied bandwidth, channel power and peaks. Use in *SpectralPlotWithPowerFunctions.vi* example.

Inputs:

- Settings in (spectralPlot_config): The cluster containing the settings for the spectral measurement.
- Occupied Percentage (single): The percentage to be used for the occupied bandwidth calculation.
- Channel Power Settings (PSD_channelPowerSettings): User's start and stop frequency settings (in MHz) for the Channel Power calculation.
- References (freqReferences): The references for the frequency setting controls on the front panel. (todo: verify)
- 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 (spectralPlot_config): The cluster containing the settings for the spectral measurement.
- Usable Frequency Spectrum (Cluster of 2 single arrays): The usable frequency (MHz) and spectrum amplitude (dBm) arrays.
- PSD result (PSD_results): The results of all the PSD calculations.
- 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. *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.

2. *average.vi*

Computes 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.

3. *calculateBins.vi*

Calculates the Bins used in *RTSAComputePSDChannelPower.vi*.

Inputs:

- Usable Data context in (usable_data_context): The context for the usable data as returned by the *RTSAComputePSDUsableData.vi*
- usable_bandwidth (unsigned 32-bit integer): the bandwidth of the usable data (in Hz).
- usable_fstart (64-bit integer): the start frequency of the usable data (in Hz).
- usable_fstop (64-bit integer): the stop frequency of the usable data (in Hz).
- usable_data_size (unsigned 32-bit integer): The size of the usable data array
- User Fstart Fstop (user_start_stop): The user-set start and stop frequency
- user_fstart (64-bit integer): Start frequency (in Hz).
- user_fstop (64-bit integer): Stop frequency (in Hz).
- rbw in (double): the resolution bandwidth.

Output:

- start bin (unsigned 32-Bit Integer): the resulting start bin.
- stop bin (unsigned 32-Bit Integer): the resulting stop bin.

4. *calculateRBW.vi*

Calculates the resolution bandwidth (RBW) used when analysing PSD parameters from FFT data.

Inputs:

- Settings in (spectralPlot_config): the settings for the capture and FFT operations.

Output:

- rbw out (unsigned 32-Bit Integer): the resulting resolution bandwidth (in Hz).

5. *findPeak.vi*

Finds 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).

6. *linspace.vi*

Computes 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.

7. *parseIDNstring.vi*

Parses the IDN string returned by the *IDN? query.

Inputs:
- IDN string (string): The string returned by the query.

Output:
- Device Info (RTSA_device_info): The parsed information (all strings)
  - Manufacturer
  - Product Name
  - Model#
  - Sub Model IDN
  - HW Version
  - Serial Number
  - FW Version.

8. *removeDC.vi*

Remove the DC offset of the time-domain data from a given LabVIEW cluster.

Input:
- Data (LabVIEW cluster): The time0domain 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.

9. *updateCtrls.vi*

Updates some front panel controls depending on the selected configuration. Use in *SimpleSpectralPlot.vi*. Recommend using *updateCtrlsPSD.vi* instead.

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.

10. updateCtrlsPSD.vi

Updates some front panel controls depending on the selected configuration. Use in *SpectralPlotWithPowerFunctions.vi*.

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 Figures 1 and 2).

For this application note, we will use the *SpectralPlotWithPowerFunctions.vi* as an example. The two sub-folders *\SubVI* and *\typedefs* are created for this VI example.

The example *(..\Examples\SpectralPlotWithPowerFunctions .vi)* demonstrates how a user can use the RTSA LabVIEW API to configure, acquire a block of time-domain data, calculate and analyze spectral data, and plot and display the computed 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 *SpectralPlotWithPowerFunctions.vi*.

1. Double clicking on *SpectralPlotWithPowerFunctions.vi* to launch the example
2. Launching the *SpectralPlotWithPowerFunctions.vi* will launch the control panel and the display (see Figure 3). 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.
        - The PSD vi functions to plot only usable spectral data and to compute power related outputs (peak, power channel and occupied bandwidth).

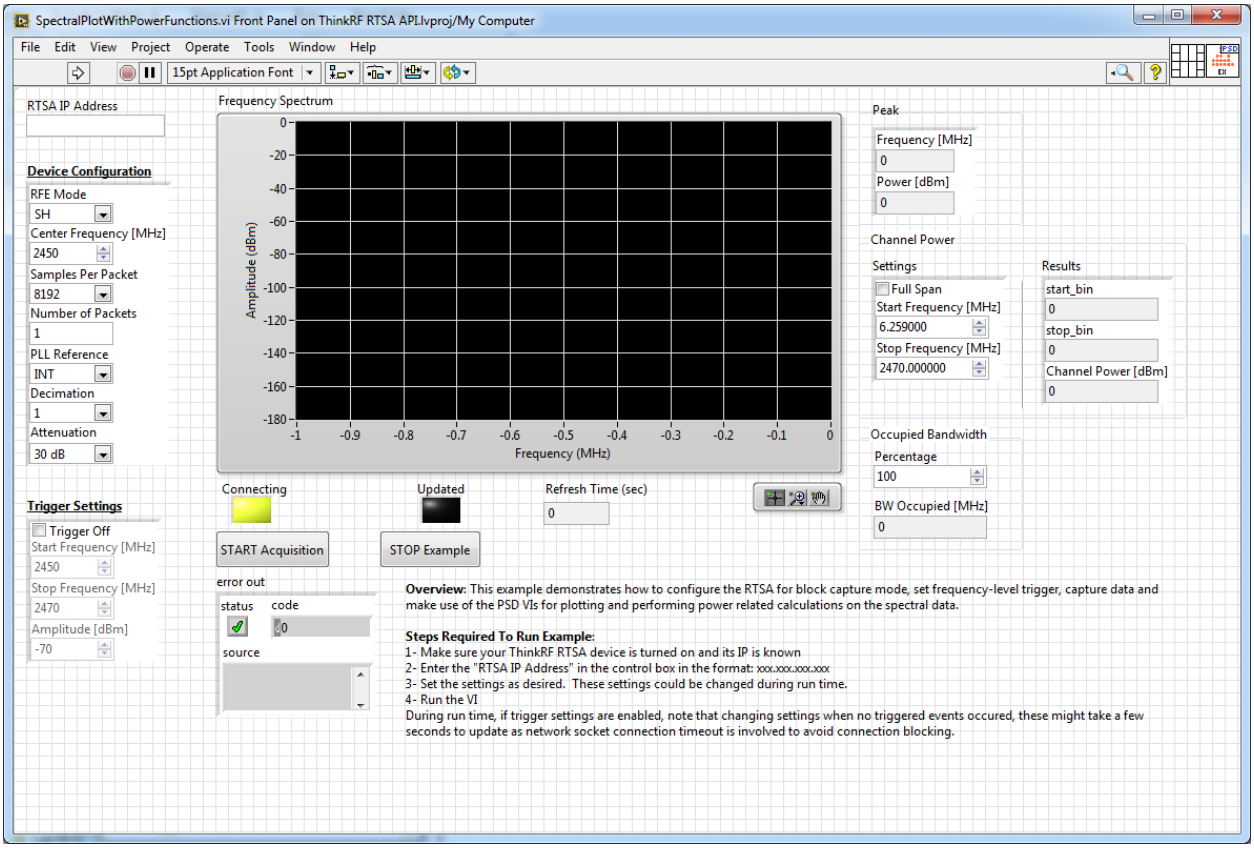

Figure 3: An example of the input fields before running

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 as needed, which are also modifiable during run-time. ***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.
5. Run the *SpectralPlotWithPowerFunctions.vi* by clicking on the Run button. Figure 4 shows an example of the VI running.

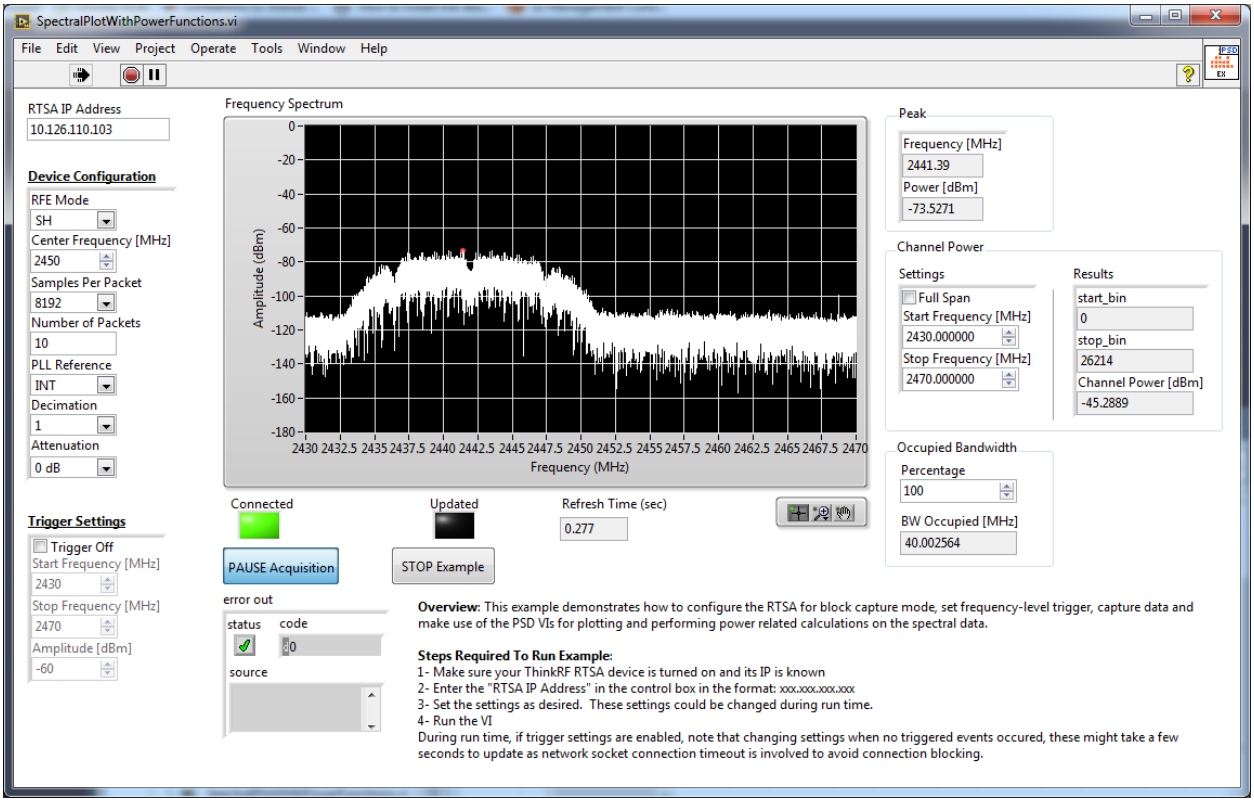

Figure 4: An example of the *SpectralPlotWithPowerFunctions.vi* running

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
v1.1	Sept 25, 2018	Fixed missing cross references to Figures
v1.2	May 03, 2019	+ Added Supported RTSA Product List and Installation Instructions sections + Added new VIs, available with LabVIEW API v2.1.0: - RTSAGetModel.vi - RTSAGetRFESpan.vi - RTSAComputePSDUsableData.vi - RTSAComputePSDChannelPower.vi - RTSAComputePSDOccupiedBandwidth.vi - RTSAPSDPeakFind.vi - RTSACaptureAndPSDCalculations.vi - calculateBins.vi - calculateRBW.vi - parseIDNstring.vi - updateCtrlsPSD.vi + Reorganized or grouped VIs for easier navigation + Updated some functions' comment + Change the example section to use the new example *SpectralPlotWithPowerFunctions.vi* + Removed centerSpanToStartStop.vi and replaced with RTSAGetRFESpan.vi

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 https://www.thinkrf.com/support/ or by calling **+1.613.369.5104**.