Application Note

74-0042-170418

Using the ThinkRF R5500 Real-Time Spectrum Analyzer with LabVIEW

Easily and quickly integrate your ThinkRF R5500 Real-Time Spectrum Analyzer into your existing or new NI LabVIEW based acquisition, measurement, automated test and validation systems.

The ThinkRF R5500 real-time spectrum analyzer 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 data acquisition, instrument control, and industrial automation on a variety of platforms including Microsoft Windows.

thinkRF

Contents

Required Software

To follow the steps described in this Application Note, the following software is required:

- Windows 7/8/10 32-bit/64-bit operating system;
- LabVIEW 2014 or later 32-bit/64-bit;
- R5500 Software and Firmware Release Package. The Release Package may be download from http://www.thinkrf.com/firmware-updates/

The R5500 real-time spectrum Analyzer

This Application Note also assumes you have access and are connected to a R5500 real-time spectrum Analyzer with firmware version of 4.0 or later.

You can connect via the Internet to ThinkRF's Evaluation R5500s. The R5500 is designed for distributed or remote deployment and as such you can use and/or evaluate any of our third-party software applications by connecting to one of ThinkRF's R5500s which we have deployed on the internet. You may access to one of our evaluation units on the internet from thinkrf.com/demo

Running the R5500 LabVIEW Example VIs

Several R5500 example VIs (Virtual Instruments) can be found within the *"\APIs\Labview\Examples"* directory of the R5500 Software and Firmware Release Package. A list as well as the description of all the examples can be found on page 15.

For the purpose of this application note, we will use the Simple Spectral Plot as the example. The Simple Spectral Plot Example *(\APIs\Labview\Examples\SimpleSpectralPlot.vi)* demonstrates how a user can use the R5500 Labview API to configure, acquire time-domain data, and calculate spectral data. The example uses a subset of the provided VIs which are described in the following sections.

Please follow the instructions below to launch and use the Simple Spectral Plot:

1. Launch the Simple Spectral Plot Example by double clicking on *SimpleSpectralPlot.vi*
2. Launching the *SimpleSpectralPlot.vi* will launch the front panel (see figure 1). Type *'Ctrl+t'* to display the block diagram beside the front panel (see figure 2).
    - At this point it is important to note a few key features:
    - All ThinkRF Labview VIs have this icon:  .
    - All the ThinkRF Labview VIs are documented on Page 5.
    - The R5500 is connected using the wsaConnect VI 

.
    - The R5500 is configured using the configureWSA VI 

.
    - The wsaRead Vi 

and computeFFT VI a 

are continuously being called inside the *while* loop.

- The wsaRead VI reads the time IQ domain data, as well as context data.
- the computeFFT VI uses the time domain data and context data to compute the spectral data.

3. You can Run the *SimpleSpectralPlot.vi* VI by typing your IP address under the 'WSA IP Address' field, and clicking on the Run command (see figure 3).

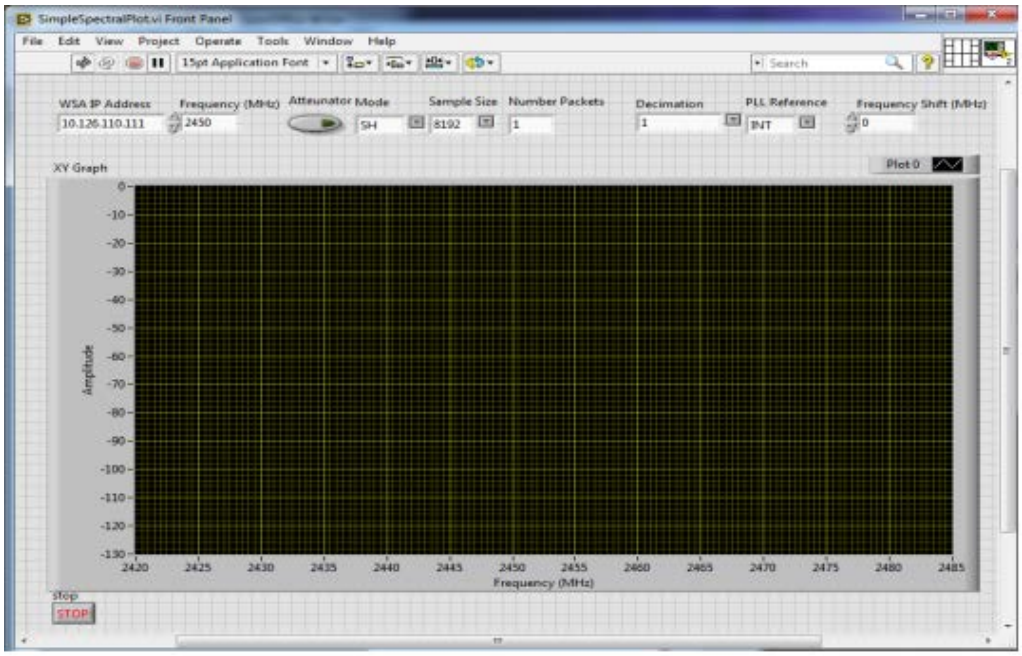

Figure 1: SimpleSpectralPlot.vi front panel

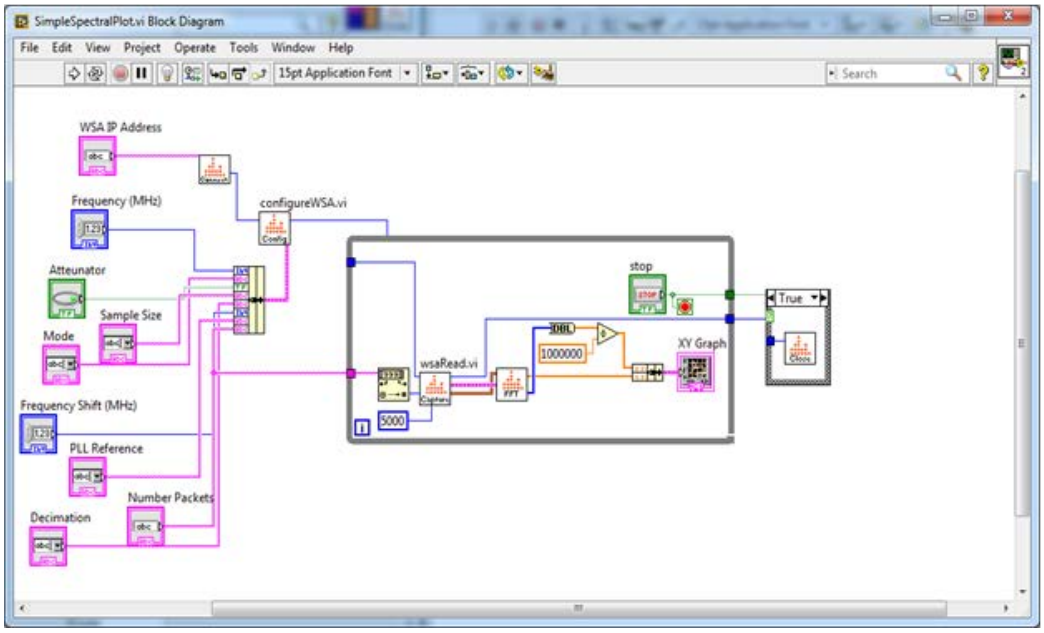

Figure 2: SimpleSpectralPlot.vi block diagram

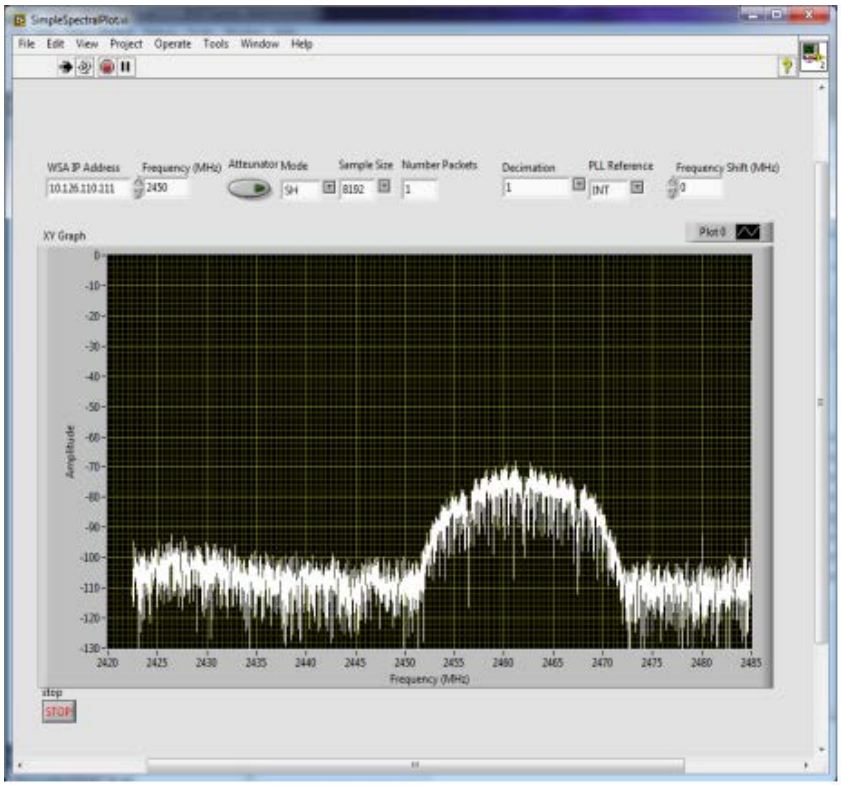

Figure 3: SimpleSpectralPlot.vi Running

The R5500 LabVIEW VIs

The following is list of the Labview VIs that can be used to connect, configure, and acquire data from the R5500. The files can be found within the "*\APIs\Labview\API*" directory of the R5500 Software and Firmware Release Package.

wsaConnect.vi

Establish a connection to the R5500.

Inputs:

1. WSA IP Address (string): The IP Address of R5500.
2. error in (Labview error cluster): A Labview error cluster to prevent the VI from executing if a previous VI had an error.

Outputs:

3. wsa_handle (64-bit integer): Represents the socket connection to the R5500.
4. return type (16-bit integer): Holds the return value of the connect function.

5. error out (Labview error cluster): Indicates whether an error has occurred.

wsaDisconnect.vi

Close a connection to the R5500.

Inputs:

1. WSA IP Address (string): The IP Address of the R5500 connection to be closed.
2. error in (Labview error cluster): A Labview error cluster to prevent the VI from executing if a previous VI had an error.

Outputs:

1. error out (Labview error cluster): Indicates whether an error has occurred.

wsaSendSCPI.vi

Send a SCPI command to a R5500.

Inputs:

1. wsa_handle (64-bit integer): The R5500 Session.
2. command (string): The SCPI command to be sent to the R5500
3. error in (Labview error cluster): A Labview error cluster to prevent the VI from executing if a previous VI had an error.

Outputs:

1. wsa_handle (64-bit integer): The R5500 Session returned.
2. error out (Labview error cluster): Indicates whether an error has occurred.

wsaQuerySCPI.vi

Query a SCPI command to the R5500.

Inputs:

1. wsa_handle (64-bit integer): The R5500 Session.
2. command (string): The SCPI command to be queried to the R5500
3. error in (Labview error cluster): A Labview error cluster to prevent the VI from executing if a previous VI had an error.

Outputs:

1. wsa_handle (64-bit integer): The R5500 Session returned.
2. return_string (string): The returned value of the SCPI query
3. error out (Labview error cluster): Indicates whether an error has occurred.

wsaGetErrorMessage.vi

Query the R5500 for an error message

Inputs:

1. wsa_handle (64-bit integer): The R5500 Session.
2. error in (Labview error cluster): A Labview error cluster to prevent the VI from executing if a previous VI had an error.

Outputs:

1. wsa_handle (64-bit integer): The R5500 Session returned.
2. error_message (string): The returned error message
3. error out (Labview error cluster): Indicates whether an error has occurred.

wsaReadRaw.vi

Read one IF data packet, with the relevant context packets

Inputs:

1. wsa_handle (64-bit integer): The R5500 Session.
2. timeout (unsigned 32-bit integer): The time delay before the socket stops trying to read data from the network socket
3. error in (Labview error cluster): A Labview error cluster to prevent the VI from executing if a previous VI had an error.

Outputs:

1. wsa_handle (64-bit integer): The R5500 Session returned.
2. Data (Labview data cluster): The IF data stored in a cluster. Note the cluster will have different values depending on the RFE mode of the R5500. 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
3. Context (Labview data cluster): The context data of the IF packet. The cluster contains:
   - stream_id (unsigned 32-bit integer): An id that identifies the data format
   - spec_inv (unsigned 8-bit integer): An integer indicating whether spectral inversion is present (0 - no inversion, 1 - there is inversion)
   - samples_per_packet (32-bit integer): The name of samples in the data packet
   - timestamp_sec (unsigned 32-bit_integer): The UTC timestamp value in seconds indicating when the data was captured (seconds)
   - reference_level (unsigned 64-bit integer)The UTC timestamp value in seconds indicating when the data was captured (pico-seconds)
   - bandwidth (64_bit integer): The full bandwidth of the IF data (Hz)
   - center_freq (64_bit integer): The center frequency of the IF data (Hz)
4. error out (Labview error cluster): Indicates whether an error has occurred.
5. return type (16-bit integer): Holds the return value of the readVRT function.

wsaRead.vi

Capture, read multiple IF data packets, and concatenate all the data packets values into one data cluster.

Inputs:

1. wsa_handle (64-bit integer): The R5500 Session.
2. number_of_packets (32-bit integer): The number of packets to capture (note you must send the :TRACE:BLOCK:PACKETS SCPI command to set the number of packets before wsaRead is executed).
3. timeout (unsigned 32-bit integer): The time delay before the socket stops trying to read data from the network socket
4. error in (Labview error cluster): A Labview error cluster to prevent the VI from executing if a previous VI had an error.

Outputs:

1. wsa_handle (64-bit integer): The R5500 Session returned.
2. Data (Labview data cluster): The IF data stored in a cluster. Note multiple packets all concatenated. Please see wsaReadRaw.vi for format
3. Context (Labview data cluster): The context data of the IF packets. Please see wsaReadRaw.vi for format.
4. error out (Labview error cluster): Indicates whether an error has occurred.

configureWSA.vi

Reset the R5500 to the default settings, and then configure the R5500 to the provided settings.

Inputs:

1. wsa_handle (64-bit integer): The R5500 Session.
2. configuration cluster (Labview data cluster): A cluster with the desired R5500 configuration. The configuration cluster's types are chosen with Labview controls in mind. The cluster contains:
   - o Frequency (64-bit integer): The center frequency (MHz)
   - Mode (String): The RFE mode
   - Atteunator (Boolean): The 20dB attenuation value (only available on R5500-408 models.
   - Sample Size (String): The sample size of each packet
   - Decimation (String): The decimation value
   - Frequency Shift (64-bit): The frequency shift value (MHz)
   - PLL Reference (String): The PLL reference source (INT or EXT)
   - Number of packets (String): The number of packets to be captured
3. error in (Labview error cluster): A Labview error cluster to prevent the VI from executing if a previous VI had an error.

Outputs:

1. wsa_handle (64-bit integer): The R5500 Session returned.
2. error out (Labview error cluster): Indicates whether an error has occurred.

wsaGetFFTSize.vi

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

Inputs:

1. Context (data cluster): The context data of the IF packets. Please see wsaReadRaw.vi for format.

Outputs:

1. memory_size (32-bit integer) The size required for the FFT buffer

computeFFT.vi

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

Inputs:

1. Context (data cluster): The context data of the IF packets. Please see wsaReadRaw.vi for format.
2. Data (data cluster): The time domain data. Please see wsaReadRaw.vi for format.

Outputs:

1. fft_buffer (Double Precision Float Array) Stores the computed spectral data
2. frequency (64-bit integer array) Stores frequency values which correspond to the spectral data.

wsaSweepAlloc.vi

Configure the R5500 to the desired sweep configuration and retrieve the size of the buffer required to store the sweep data obtained from wsaSweep.vi.

Inputs:

1. sweep_config (data cluster): The configuration of the sweep. Below is the structure of the cluster:
2. wsa_handle (64-bit integer): The R5500 Session
3. start_freq (64-bit integer): The start frequency of the sweep (Hz)
4. stop_freq (64-bit integer): The stop frequency of the sweep (Hz)
5. rbw (unsigned 64-bit integer): The RBW of the sweep (Hz)
6. attenuator (boolean): The attenuator state (only available on the R5500-408 models)
7. mode (string): The RFE mode of the sweep
8.

Outputs:

1. buff_size (Unsigned 32-bit integer): The size of the buffer required to store the spectral data.

wsaSweep.vi

Read swept spectral data from the R5500

Inputs:

1. sweep_config (data cluster): The configuration of the sweep. See wsaSweepAlloc.vi for the cluster's content.
2. buff_size (Unsigned 32-bit integer): The expected size of the spectral data

Outputs:

1. fft_buffer (Double Precision Float Array) Stores the swept spectral data

R5500 LabVIEW Utility VIs

The following are several utility VIs that can be used to do simple operations that are useful subset of the files within the *"\APIs\Labview\Util"* directory of the R5500 Software and Firmware Release Package.

average.vi

Compute the average of a given 16-bit array

Inputs:

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

Outputs:

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

centerSpanToStartStop.vi

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

Inputs:

1. Context (data cluster): The context data of the IF packets. Please see wsaReadRaw.vi for format.

Outputs:

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

findPeak.vi

Find the peak spectral and frequency valuse of given spectral and frequency arrays.

Inputs:

1. Spectral Data (Double Precision Float Array): Array containing spectral data (dBm).
2. Frequency Data (64-bit integer Array): Array containing frequency data (Hz).

Outputs:

1. Max Power Level (Double Precision Float): Power level of peak (dBm)
2. Frequency Of Max Power (64-bit integer): Frequency of peak (Hz)

linspace.vi

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

Inputs:

1. start (Double Precision Float): First value of array
2. stop (Double Precision Float): Last value of array
3. size (32-bit integer): The size of the array

Outputs:

1. array (Double Precision Float Array): An array with start/stop/size matching the input values

removeDC.vi

Remove the DC offset from a given data cluster.

Inputs:

1. Data (data cluster): The time domain data. Please see wsaReadRaw.vi for format.

Outputs:

1. i_data (Double Precision Float Array): I data that without a DC offset
2. q_data (Double Precision Float Array): Q data without a DC offset

Contact us for more information

Please contact us via email at *sales@thinkrf.com* or via phone at +1.613.369.5104.