Capturing Large Post Trigger Data in the RTSA7550

This document provides a brief explanation on how to capture a large amount of data, and condition it with a frequency level trigger. An example Python script using PyRF API is appended at the end of this document. The document targets two RTSA7550's modes of operation: the 100MHz-IBW mode and the 40MHz-IBW mode. The 100kHz-IBW mode, which is known as high dynamic range (HDR) is not included as it does not have a triggering capability.

Initializing the trigger

The RTSA7550's frequency level trigger can be enabled by using the '**TRIGGER:TYPE LEVEL'** SCPI command, and the configuration of the level trigger can be set using the **'TRIGGER:LEVEL'** SCPI command. Please refer to the Programmer's Guide for more details regarding each command. Note that the RTSA7550 will not capture any data unless the **'TRACE:BLOCK:DATA?'** SCPI command is issued.

Determining the number of data packets:

The RTSA7550 has a memory size M of 128 MBytes. This means that RTSA7550could store a large block of continuous and contiguous data up to that memory limit. The RTSA7550 sends the raw IQ data using VRT[1] packets. The sample size, which is the number of samples per packet (SPP) must be between 128 and 32768 and must be a multiple of 16 for the 100MHz-IBW mode, and between 256 and 65536 for the 40MHz-IBW mode (though for FFT analysis, a power of 2 is recommended). Multiple VRT packets (or packets per block (PPB)) can be requested to capture a large block of data, provided the maximum mentioned above. A simple formula to calculate the maximum PPB N_p for a required SPP is as follows: $N_p = (M/N_s)/(\mathrm{SPP} + 6)$, where N_s = 2 and 4 bytes/ sample for I14 and I14Q14 data format, respectively. I14Q14 refers to the 100MHz-instantaneous bandwidth (IBW) mode, while I14 refers to the 40MHz-IBW mode. Table 1 contains the maximum PPB that can be captured based on the format of the IQ data (I14Q14 vs I14), and SPP.

[1] VRT stands for the VITA-49 Radio Transport protocol for digitized data and its associated context information.

Berkeley Nucleonics Corporation - 2955 Kerner Blvd. - San Rafael, CA 94901
800-234-7858 or LiveChat @ www.berkeleynucleonics.com

It is worth noting that the RTSA7550 storage capacity, M, allows for a capture time of lossless data T_C up to

$$T_C = \frac{1}{R_s}\left(\frac{M}{N_s}\right) \cong \begin{cases} 268.4 & \text{for } 100\text{MHz} - \text{IBW} \\ 536.9 & \text{for } 40\text{MHz} - \text{IBW} \end{cases} \text{ ms},$$

where $R_s = 125e6$ samples/s is the sampling rate of the RTSA7550.

Table 1: Maximum PPB for a given SPP and IQ data format

SPP	Maximum PPB	
	I14	I14Q14
128	500812	250406
256	256140	128070
512	129553	64776
1024	65154	32577
2048	32672	16336
4196	15970	7985
8192	8186	4093
16384	4094	2047
32768	2047	1023

Capturing the data packets

SPP multiplied by PPB determines the amount of continuous and contiguous data per block to be captured. The SPP and PBB shall be set first using '**TRACE:SPP'** and **'TRACE:BLOCK:PACKETS'** SCPI commands, respectively. Then, **'TRACE:BLOCK:DATA?'** SCPI command is used to start the capture, after any other additional configuration has been set. Note that if the level trigger is enabled, the RTSA7550 will only capture data when the trigger event occurs.

Example Code

Below is a simple Python script using PyRF API, which demonstrates how a large block capture can be acquired after a level trigger criterion is met:

```
# import required libraries
import sys
from pyrf.devices.thinkrf import WSA
from pyrf.util import collect_data_and_context


# sample size with packets per block
SPP = 1024
PPB = 32577
RFE_MODE = 'ZIF'
TRIGGER_SET = {'type': 'LEVEL',
               'fstart': 2400 * 1e6,
               'fstop': 2500 * 1e6,
               'amplitude': -70}
# connect to WSA device
dut = WSA()
dut.connect(sys.argv[1])
# reset device to default settings
dut.request_read_perm()
dut.reset()
dut.flush()

# set RFE mode to ZIF, which yields I14Q14 data
dut.rfe_mode(RFE_MODE)

# configure the trigger setting
dut.trigger(TRIGGER_SET)

# capture the required data block
dut.capture(SPP, PPB)

# read the data from the WSA5000
for i in range(PPB):
    data, context = collect_data_and_context(dut)
```