OpenCPI
Rx App Guide
(E310 Supplement)

Version 1.4

Revision History

Revision	Description of Change	Date
v1.3.1-E3XX	Updated for E310 support	3/2018
v1.4	Updated with simplications and references to assets' document	9/2018

Table of Contents

1 Document Scope

This document describes the ANGRYVIPER Receive demo application or "Rx App". It includes a description of the RX App application and instructions on how to setup the various supported hardware platforms, build and execution of the application.

2 Description

For more information on this application, see `ocpi.assets`'s more in-depth version of the *RX_app* document.

3 Hardware Portability

This application is specific to the `e3xx` platform.

4 Building the Application

4.1 Common Application Worker Dependencies

The following application workers, sorted by component library name, must be built prior to building the RX application assembly. See Appendix A for the parameter configurations used in the application, and see the individual component datasheets for more information.

- ocpi.core

  file_write.rcc

- ocpi.assets.util_comps

  timestamper.hdl

- ocpi.assets.dsp_comps

  cic_dec.hdl

  complex_mixer.hdl

  iq_imbalance_fixer.hdl

  dc_offset_filter.hdl

4.2 Hardware-Specific Worker Dependencies

The following workers, sorted by component library name, must be built prior to building RX app. See Appendix A for the parameter configurations used in the application, and see the individual component datasheets for more information and build instructions.

E310

- ocpi.bsp.e310.cards

  e3xx_mimo_xcvr_filter_proxy.rcc

  e3xx_mimo_xcvr_filter.hdl

  e3xx_i2c.hdl

  e3xx_rx.rcc

  e3xx_mimo_xcvr_ad5662.hdl

4.3 HDL Assembly and HDL Container

For more information on this application, see `ocpi.assets`'s more in-depth version of the *RX_app* document.

4.4 Performance and Resource Utilization

Table 1: Resource Utilization Table for hdl-assembly: dc_offset_iq_imbalance_mixer_cic_dec_timestamper

Container	OCPI Platform	OCPI Target	Tool	Version	Device	Registers (Typ)	LUTs (Typ)	Fmax (MHz) (Typ)	Memory/Special Functions
cnt_1rx_0tx_thruasm_mode_2_cmos_e3xx	e3xx	zynq	Vivado	2017.1	xc7z020clg484-1	9571	9370	100.0	DSP48E1: 15 BUFGCTRL: 2 RAMB36E1: 18 ODDR: 1 RAMB18E1: 1 BUFG: 2

4.5 Executable

To build for the E310 (which runs the xilinx13_4 PetaLinux operating system), run the following command from the *rx_app* directory:

```
ocpidev build --rcc-platform xilinx13_4
```

For more information on this application, see `ocpi.assets`'s more in-depth version of the *RX_app* document.

5 Testing the Application

5.1 Sample Test Setup

For more information on this application, see `ocpi.assets`'s more in-depth version of the *RX_app* document.

5.2 Artifacts

For more information on this application, see `ocpi.assets`'s more in-depth version of the *RX_app* document. Appendix B includes a list of the artifacts required for each platform and mode.

5.3 Arguments to executable

For more information on this application, see `ocpi.assets`'s more in-depth version of the *RX_app* document.

5.4 Library Path Requirements

Prior to running the application, the environment variable OCPI_LIBRARY_PATH must be configure, such that, all of the Rx application's run-time artifacts can be located. OpenCPI conveniently provides access to a project's run-time artifacts at the top-level of each project in a directory called artifacts. Reference the OpenCPI Application Development Guide for more about OCPI_LIBRARY_PATH.

Run "`make show`" in the application directory for a recommended `OCPI_LIBRARY_PATH`.

5.5 Expected results

A python script is included with the application for plotting the received data in both the time and frequency domain. Using the test setup shown above and the default settings for the GNUradio block diagram, run the application with the following arguments:

E310

```
# Usage is:
# ./target-xilinx13_4/rx_app rf_tune_freq data_bw rf_bw rf_gain bb_bw bb_gain if_tune_freq runtime enable_timestamps frontend
  ./target-xilinx13_4/rx_app 1000         0.512   -1    24      1     -1      0.256        1       1                 e3xx
```

The output file can then be plotted with the python script with the following syntax and the output can be seen below:

```
python ./scripts/plotAndFftAndTime.py odata/rx_app_raw.out complex 18000 256000 16352
```

Alternatively, the shortened file can be plotted which will ignore optionally unwanted startup data:

```
python ./scripts/plotAndFftAndTime.py odata/rx_app_shortened.out complex 18000 256000 16352
```

The default sample rate for the GNUradio block diagram is 512 kS/s. It is recommended that when using RX app with this input signal that a sample rate close to 512 kS/s be used. Higher sample rates are still valid, but may produce plots that look drastically different than those shown here.

Timestamps are embedded, optionally, in the output file, and in addition to plotting, the script parses out and prints the timestamps. An example output gathered using the syntax above:

```
Timestamp at index:  000000000 :  1.0728292 Seconds:  0x1 Fraction:  0x12a4eec4
Timestamp at index:  000008180 :  1.0887978 Seconds:  0x1 Fraction:  0x16bb73ba ('Delta:  0.0159686', 'Expected:, 0.0159688')
Timestamp at index:  000016360 :  1.1047664 Seconds:  0x1 Fraction:  0x1ad1f906 ('Delta:  0.0159686', 'Expected:, 0.0159688')
```

A small discrepancy (+/- 10) between Delta and Expected is typical. The difference is an artifact of the resolution of the fractional part of the timestamp applied in the timestamper HDL component. More information can be found in the timestamper component datasheet.

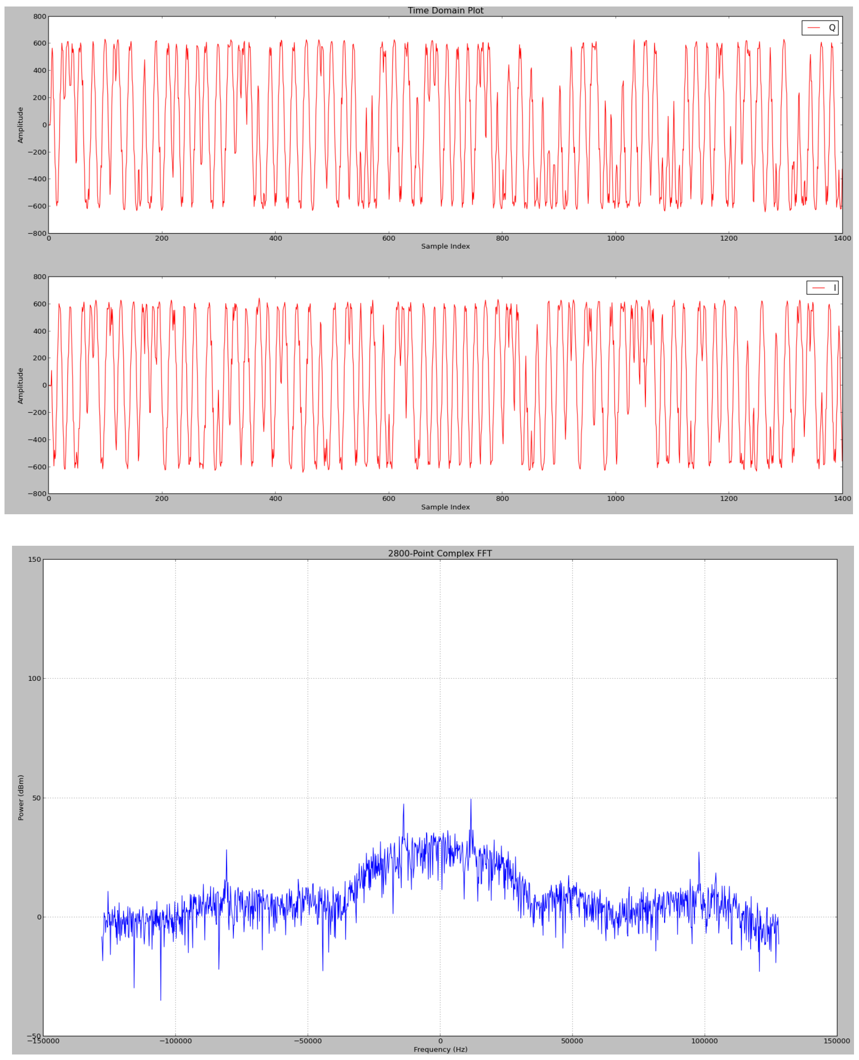


Figure 1: Output of RX app

5.6 Using a RF Signal Generator

An arbitrary RF signal generator can be used with RX app instead of the Ettus N210.

Hardware Setup

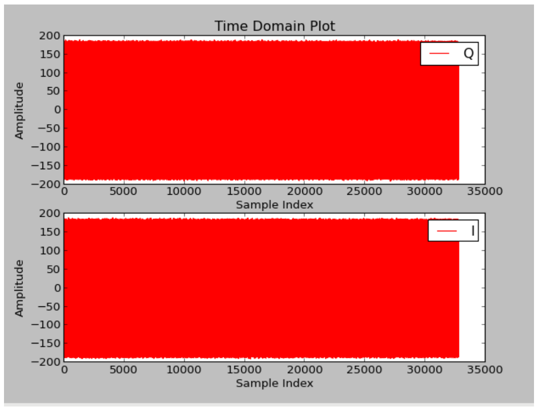
- Signal generator is set to 2400.8 MHz with an amplitude of -40 dBm.
- Signal generator is connected to the E310 TRXA SMA.

Example 1 - Execution (remote system)
An example run with an IF tune freq of 0.1 MHz would be as follows.

```
SAMP_RATE_MHZ=2.5
RF_TUNE_FREQ_MHZ=2400
IF_TUNE_FREQ_MHZ=0.1
./<target>/rx_app $RF_TUNE_FREQ_MHZ $SAMP_RATE_MHZ -1 24 2.5 -1 $IF_TUNE_FREQ_MHZ 1 1 e3xx TRXA
```

Example 1 - Verification (development system)
Verify that the signal generator tone exists at 700,000 Hz in the plotted baseband signal FFT (calculated as signal generator frequency - RF tune frequency - IF tune frequency = 2400.8 MHz - 2400 MHz - 0.1 MHz = 700,000 Hz).

```
SAMP_RATE_HZ=2500000
python ./scripts/plotAndFftAndTime.py odata/rx_app_shortened.out complex 65536 $SAMP_RATE_HZ 16352 &
```



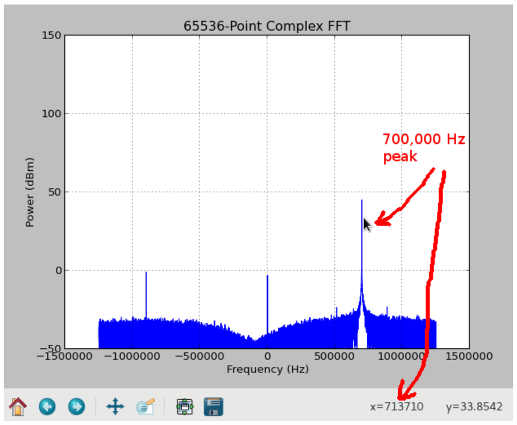


Figure 2: Output of RX app

Example 2 - Execution (remote system)
An example run with an IF tune freq of 0 MHz would be as follows.

```
SAMP_RATE_MHZ=2.5
RF_TUNE_FREQ_MHZ=2400
IF_TUNE_FREQ_MHZ=0
./<target>/rx_app $RF_TUNE_FREQ_MHZ $SAMP_RATE_MHZ -1 24 2.5 -1 $IF_TUNE_FREQ_MHZ 1 1 e3xx TRXA
```

Example 2 - Verification (development system)
Verify that the signal generator tone exists at 800,000 Hz in the plotted baseband signal FFT (calculated as signal generator frequency - RF tune frequency - IF tune frequency = 2400.8 MHz - 2400 MHz - 0 MHz = 800,000 Hz).

```
SAMP_RATE_HZ=2500000
python ./scripts/plotAndFftAndTime.py odata/rx_app_shortened.out complex 65536 $SAMP_RATE_HZ 16352 &
```

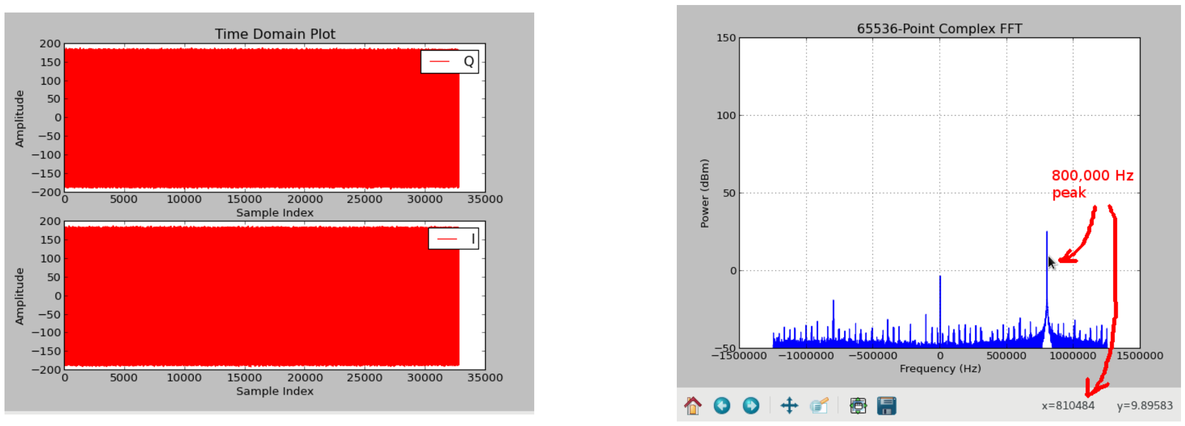


Figure 3: Output of RX app

5.7 Known Issues

- If the path `/var/volatile` does not exist or requires root permission to write to, you will need to modify the ACI and the application XML to use a different directory for writing data. This involves simply finding and replacing `/var/volatile` with a different directory in the `.cxx` and `.xml` files. Failing to make this change when necessary may result in a segmentation fault error at application runtime.

6 Appendix A: Worker Parameters

E310

- cic_dec.hdl
  N = 3
  M = 1
  R = 8
  DIN_WIDTH = 16
  ACC_WIDTH = 25
  DOUT_WIDTH = 16
- complex_mixer.hdl
  NCO_DATA_WIDTH_p = 12
  INPUT_DATA_WIDTH_p = 12
  CORDIC_STAGES_p = 16
  PEAK_MONITOR_p = true
- iq_imbalance_fixer.hdl
  DATA_WIDTH_p = 16
  ACC_PREC_p = 34
  PEAK_MONITOR_p = true
- dc_offset_filter.hdl
  DATA_WIDTH_p = 16
  PEAK_MONITOR_p = true
- e3xx_i2c.hdl
  CP_CLK_FREQ_p = 100e6
  FMC_GA1 = 0
  FMC_GA0 = 0
- ad9361_spi.hdl
  CP_CLK_FREQ_HZ_p = 100e6
- ad9361_data_sub.hdl
  LVDS_p = false
  HALF_DUPLEX_p = false
  SINGLE_PORT_p = true
  SWAP_PORTS_p = false
  DATA_CLK_Delay = 11
  RX_Data_Delay = 0
  FB_CLK_DELAY = 12
  TX_DATA_DELAY = 0
- ad9361_adc_sub
  LVDS_p = false
  HALF_DUPLEX_p = false
  SINGLE_PORT_p = true
- ad9361_dac_sub
  LVDS_p = false
  HALF_DUPLEX_p = false
  SINGLE_PORT_p = true

7 Appendix B: Artifacts

7.1 E310

- dc_offset_iq_imbalance_mixer_cic_dec_timestamper_e3xx_cfg_1rx_0tx_mode_2_cmos_cnt_1rx_0tx_thruasm_mode_2_cmos_e3xx.bitz
- target-xilinx13_4/file_write_s.so
- target-xilinx13_4/e3xx_rx_s.so
- target-xilinx13_4/e3xx_mimo_xcvr_filter_proxy_s.so
- target-xilinx13_4/ad9361_config_proxy_s.so