

Lab 2: Debugging a deployed design

In this lab we will examine ways of debugging our deployed HDL design through IIO and GNU Radio. The methods covered will be with respect to AXI-Memory Mapped interfaces and libIIO. These techniques are very useful for accessing portions of a design once deployed to hardware for purposes of data inspection, algorithmic tuning, verification,...etc.

The board

For this lab we will be utilizing the AD9361-Z7035 (SOM) which is shown in Figure 1. The HDL design has been compiled for this target specifically. Synthesis has been done ahead of time and the design has been loaded onto the bootable SD card to save time. This is the same model discussed by the instructor.

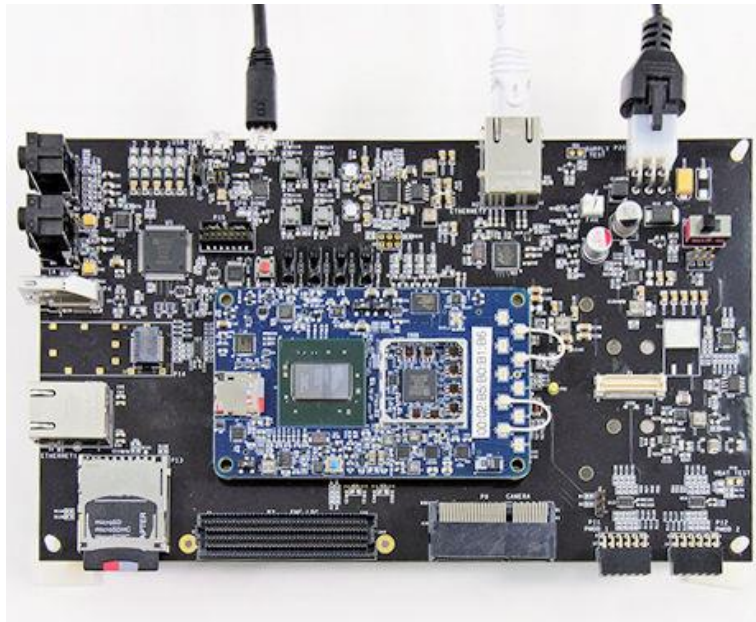


Figure 1

Connect the ethernet cable from the “Ethernet 1” port on the SOM to the ethernet adaptor on your PC. Insert the provided SD card and connect the power cable. Power on the board with the toggle switch located on the carrier.

Check your boards connectivity by pinging the IP address “192.168.3.2” as shown in Figure 2 from a command prompt. If you cannot reach the board check cables and ask instructors for help.

```
Select Command Prompt
C:\Users\tcollins>ping 192.168.3.2

Pinging 192.168.3.2 with 32 bytes of data:
Reply from 192.168.3.2: bytes=32 time=17ms TTL=64
Reply from 192.168.3.2: bytes=32 time<1ms TTL=64
Reply from 192.168.3.2: bytes=32 time<1ms TTL=64
Reply from 192.168.3.2: bytes=32 time<1ms TTL=64
```

Figure 2

Running the Deployed Design

Now that we have the board running and is connectable we can begin to inspect the IIO devices. From the command line again, use the “iio_attr” tool to list the available IIO devices on the board as show in Figure 3.

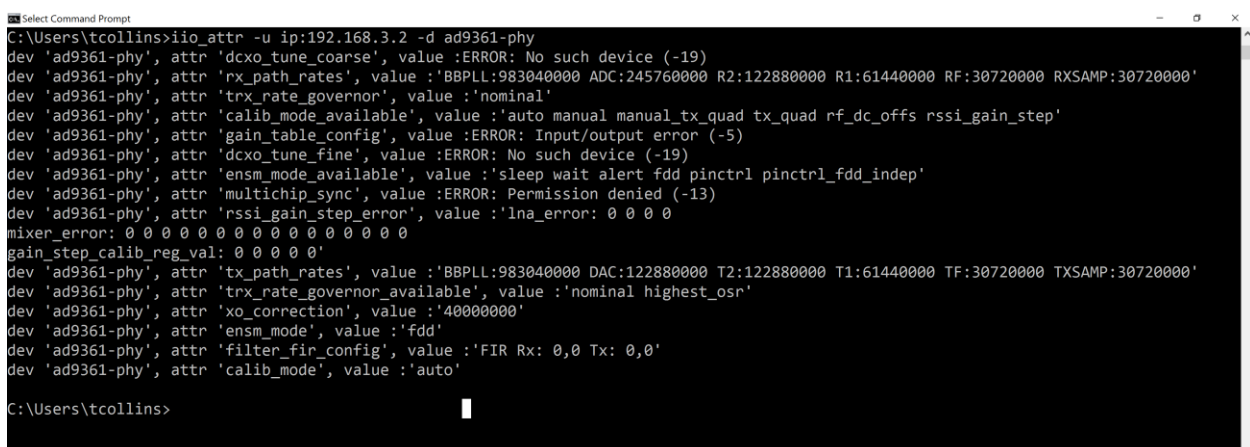


```
Command Prompt
C:\Users\tcollins>iio_attr -u ip:192.168.3.2 -d
IIO context has 8 devices:
    iio:device3: ad9517-3, found 0 device attributes
    iio:device1: ad9361-phy, found 15 device attributes
    iio:device6: mwipcore0:mmrd0, found 2 device attributes
    iio:device4: cf-ad9361-dds-core-lpc, found 0 device attributes
    iio:device2: xadc, found 1 device attributes
    iio:device0: ad7291, found 0 device attributes
    iio:device7: mwipcore0:mmwr0, found 2 device attributes
    iio:device5: cf-ad9361-lpc, found 0 device attributes

C:\Users\tcollins>
```

Figure 3

As in Figure 3, you should observe several devices but we are only interested in a few. The transceiver devices are associated with “ad9361-phy”, “cf-ad9361-lpc” (receiver driver), and “cf-ad9361-dd-core-lpc” (transmitter driver). If you are interested, you can inspect some of the transceiver attributes through a similar ‘iio_attr’ command shown in Figure 4.



```
Select Command Prompt
C:\Users\tcollins>iio_attr -u ip:192.168.3.2 -d ad9361-phy
dev 'ad9361-phy', attr 'dcxo_tune_coarse', value :ERROR: No such device (-19)
dev 'ad9361-phy', attr 'rx_path_rates', value :BBPLL:983040000 ADC:245760000 R2:122880000 R1:61440000 RF:30720000 RXSAMP:30720000'
dev 'ad9361-phy', attr 'trx_rate_governor', value :nominal'
dev 'ad9361-phy', attr 'calib_mode_available', value :auto manual manual_tx_quad tx_quad rf_dc_offs rssi_gain_step'
dev 'ad9361-phy', attr 'gain_table_config', value :ERROR: Input/output error (-5)
dev 'ad9361-phy', attr 'dcxo_tune_fine', value :ERROR: No such device (-19)
dev 'ad9361-phy', attr 'ensm_mode_available', value :sleep wait alert fdd pinctrl pinctrl_fdd_indep'
dev 'ad9361-phy', attr 'multichip_sync', value :ERROR: Permission denied (-13)
dev 'ad9361-phy', attr 'rssi_gain_step_error', value :lma_error: 0 0 0 0
mixer_error: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
gain_step_calib_reg_val: 0 0 0 0 0
dev 'ad9361-phy', attr 'tx_path_rates', value :BBPLL:983040000 DAC:122880000 T2:122880000 T1:61440000 TF:30720000 TXSAMP:30720000'
dev 'ad9361-phy', attr 'trx_rate_governor_available', value :nominal highest_osr'
dev 'ad9361-phy', attr 'xo_correction', value :40000000'
dev 'ad9361-phy', attr 'ensm_mode', value :fdd'
dev 'ad9361-phy', attr 'filter_fir_config', value :FIR Rx: 0,0 Tx: 0,0'
dev 'ad9361-phy', attr 'calib_mode', value :auto'

C:\Users\tcollins>
```

Figure 4

However, we are primarily interested in the ‘mwipcore0:mmrd0’ and ‘mwipcore0:mmwr0’ devices, which are associated with the IP which was designed from HDL-Coder. These are integrated into the

device tree by the HDL-Coder reference designs and allow direct register access to our IP. One device is for reading and one is for writing registers. Inspect these devices by again running the 'iio_attr' command as in Figure 5.

```
Command Prompt
C:\Users\tcollins>iio_attr -u ip:192.168.3.2 -d mwipcore0:mmwr0
dev 'mwipcore0:mmwr0', attr 'reg_access', value : 'disabled'
dev 'mwipcore0:mmwr0', attr 'reg_access_available', value : 'disabled enabled'
C:\Users\tcollins>
```

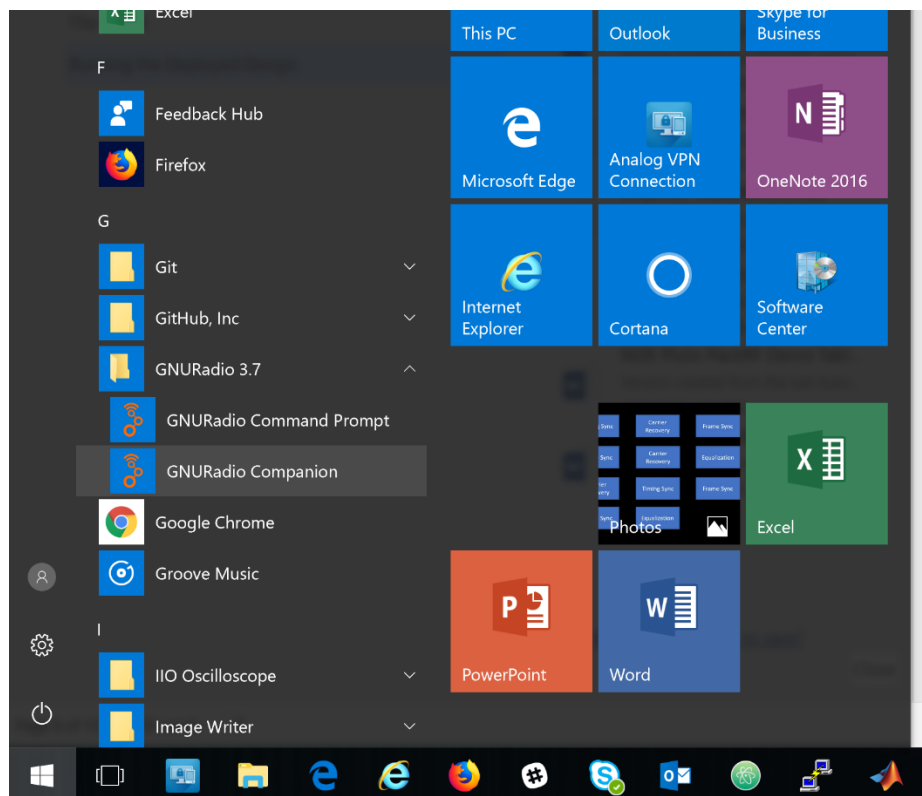
Figure 5

On the output, as in Figure 5, you should observe 2 attributes. 'reg_access' is protecting the register access is needs to be set to 'enabled' before using trying to write registers to this device's memory space. Fortunately, this enable writing, register writing and read, can all be done through GNU Radio itself. 'reg_access_available' just lists possible options for the related attribute.

GNU Radio

For this design, we have several input control registers, one debug status register which is selectable, and selectable IQ data from different parts of the design muxed into the IQ data path. We will be using GNU Radio to visualize the IQ constellations as well as to read and write from the AXI-lite registers we have exposed.

Open GNU Radio Companion from the start menu:



Once GNU Radio Companion is running, open the flowgraph “packrf.grc” located in **C:\GRCon\Workshop\Lab2**. It should look similar Figure 6 to once open.

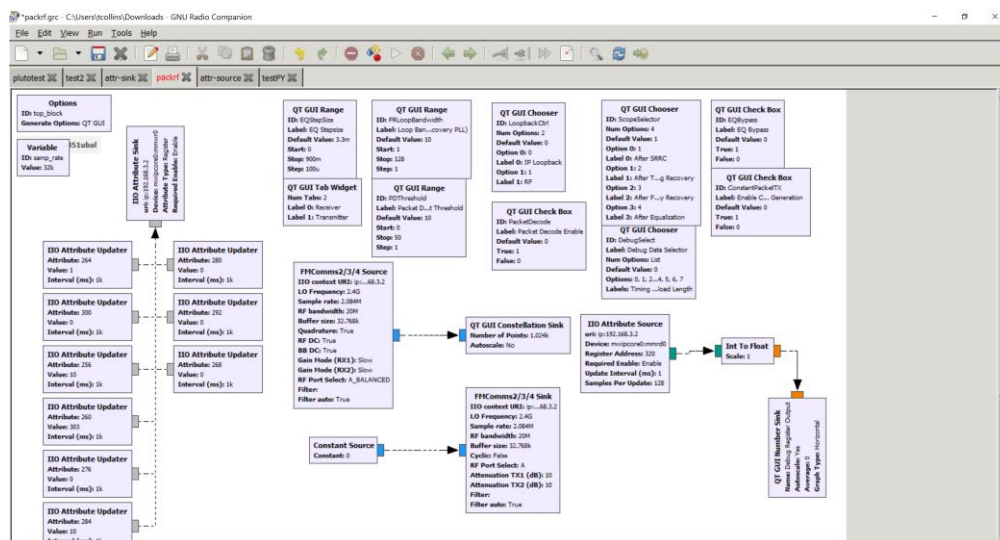


Figure 6

There are three main sections to the flowgraph. First are the transceiver blocks, which will both configuration the transceiver and collect data from the IQ data paths. These blocks are shown in Figure 7, where the “FMComms2/3/4 Source” is connected to a constellation plot and we are sending zeros to the related sink block for the SOM. The deployed design has internal packet generation, so we do not have to supply a useful signal, we are simply configuring the TX path.

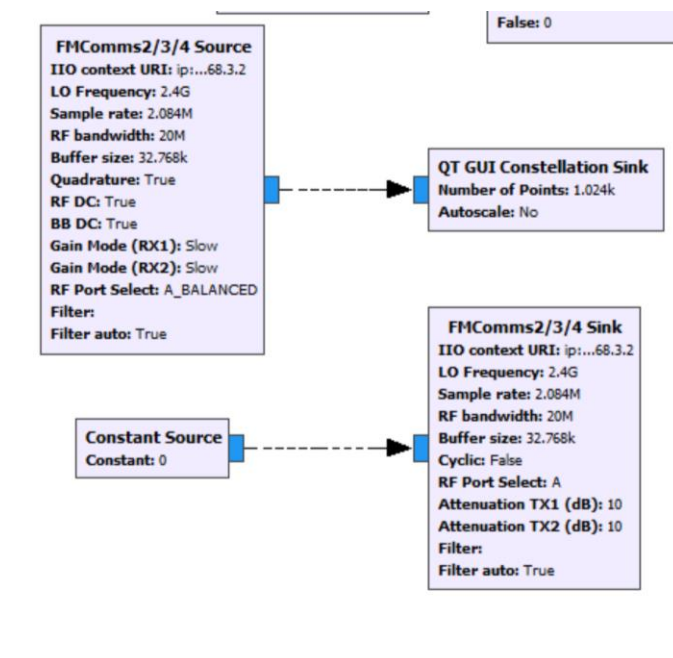
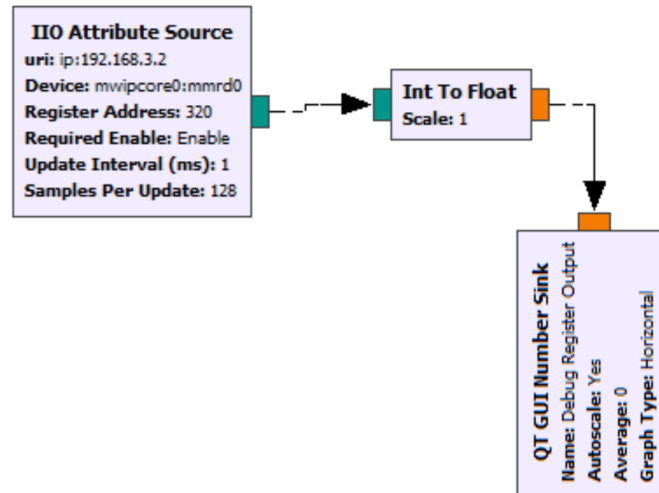


Figure 7

To read registers we are using the “IIO Attribute Source”, which is pointed to the register address provided by HDL-Coder during the build process. The output of this block will be dependent on an input register that selects this debug register’s output. The device name of this block is set to that ‘mwipcore0:mmrd0’ device we inspected before.



On the register writing side we will be using the “IIO Attribute Sink” block as shown Figure 8. Since this block is message based, we can use a single block to write to several different attributes for a given device for a give attribute type. We will also use “IIO Attribute Updater” blocks, which are essentially helpers for creating messages that can be changed at runtime. This block creates PMT’s who key is the attribute name and value is the value to be written.

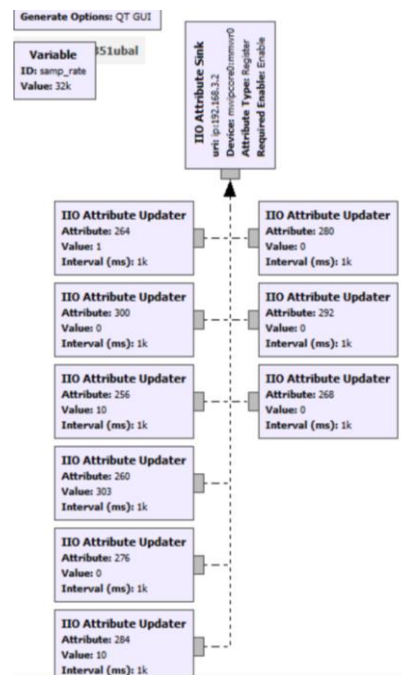



Figure 8

Next, start the flowgraph by hitting the play button  on the top bar. A GUI panel will launch looking similar to Figure 9. The constellation shown on the bottom panel is routed out of the receiver running on the FPGA, and selected by the “ScopeSelector” parameter on the “Receiver” tab.

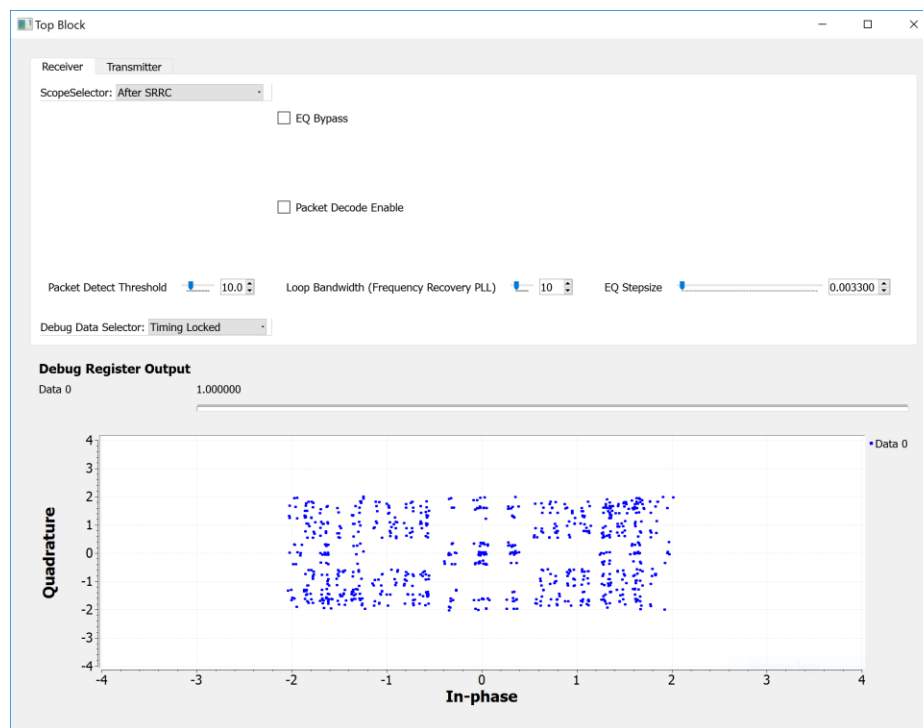
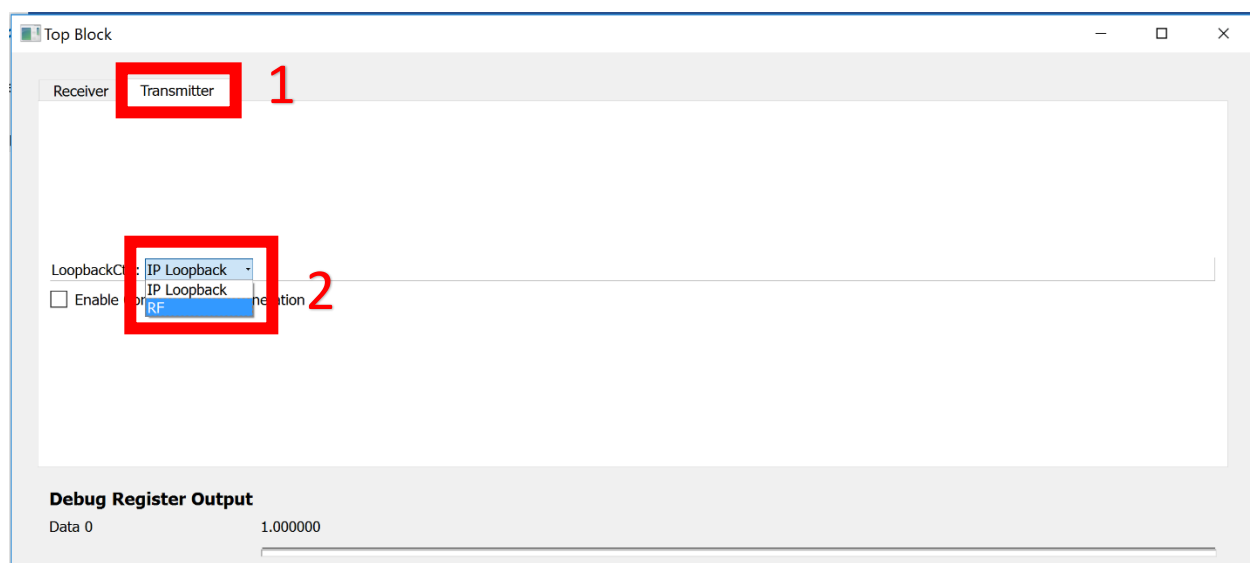
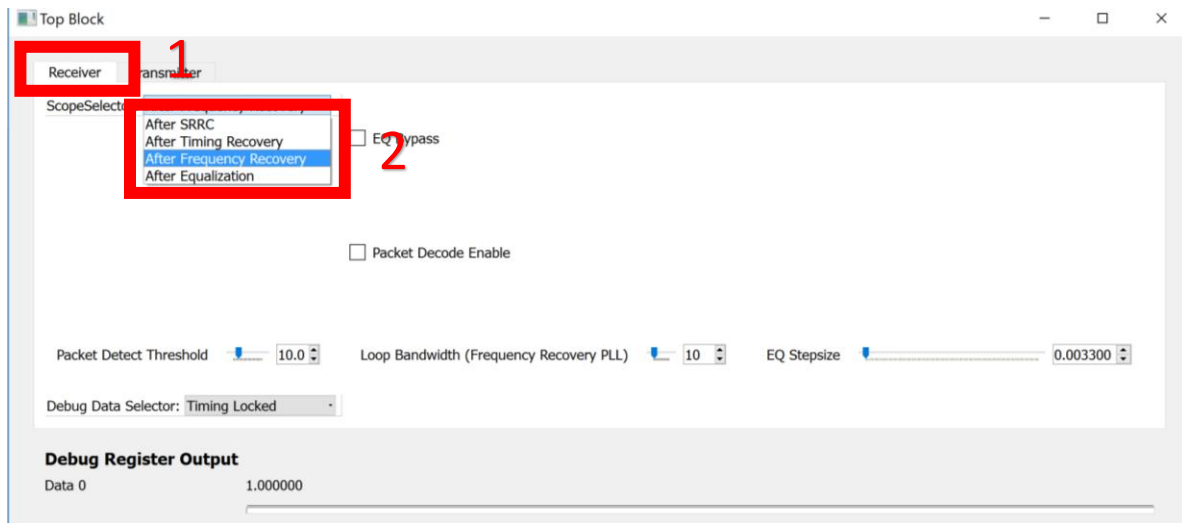


Figure 9

By default, the system is brought up in IP loopback, meaning the RF transceiver is not in the data path. To switch over to this data source select the “Transmitter” tab, then select “RF” from the “LoopbackCtrl” dropdown menu. Also, enable the “Constant Packet Generation” checkbox. This should update the constellation being plotted and appear more dynamic.



Next, go back to the “Receiver” tab, and select each of the “ScopeSelector” options and observe the constellation at each stage. By changing the “ScopeSelector” you are changing a mux which controls the source of data in the receive path which is passed back from libIIO. This is useful for debugging to understanding link quality. Feel free to adjust the other knobs in the receiver panel and inspect the effect on the constellation. Note there is a 400 Hz offset between TX and RX.



The remaining feature is the “Debug Data Selector”, which control the debug status register. Selecting different values will update the “Debug Register Output” number sink as shown in Figure 10. Make sure “Packet Decode Enable” is selected in the “Receiver” panel or no packets will be processed and many of the debug outputs will remain at zero.

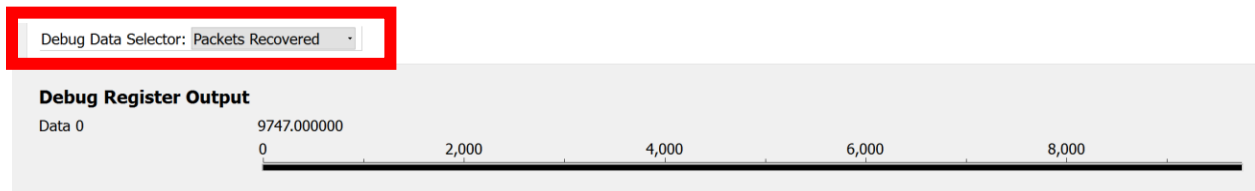


Figure 10

Try out different settings and examine their output. You can even offset the “Loop Bandwidth” so the packets will begin to fail CRC checks.