

Tutorial 4: The USRP Board

Dawei Shen*

August 8, 2005

Abstract

This article introduces the Universal Software Radio Peripheral (USRP) board, which is GNU Radio's hardware counterpart.

1 Introduction to the USRP board

These days, when we talk about the GNU Radio, the Universal Software Radio Peripheral (USRP) board has become an indispensable hardware component. It is developed by Matt wholly for the GNU Radio users. Basically, the USRP is an integrated board which incorporates AD/DA converters, some forms of RF front end, and an FPGA which does some important but computationally expensive pre-processing of the input signal. The USRP is low-cost and high speed, which is the best choice for a GNU Radio user to implement some real time applications. We could purchase the USRP boards from [Ettus](#). A USRP board consists of one mother board and up to four daughter boards. The price for the mother board is \$450 and basic daughterboards cost \$50 each. Figure 1 shows the picture of a USRP board equipped with four daughter boards (2 for RX and 2 for TX).

2 A 'data sheet' of the USRP board

This section introduces the hardware parts on the USRP board. It should be emphasized that the characteristics of those hardware parts are very important. They will influence your radio design and software programming extensively. You have to follow the constraints imposed by the hardware and memorizing some of them would be useful. So please read this section carefully.

A typical setup of the USRP board consists of one mother board and up to four daughter boards, as shown in Figure 2. On the mother board, we can see the DC power input and the USB 2.0 interface. At this stage, USB 1.x is not supported at all. So make sure your PC is equipped with USB 2.0.

2.1 AD / DA Converters

There are 4 high-speed 12-bit AD converters. The sampling rate is 64M samples per second. In principle, it could digitize a band as wide as 32MHz. The AD converters can bandpass-sample signals

*The author is affiliated with Networking Communications and Information Processing (NCIP) Laboratory, Department of Electrical Engineering, University of Notre Dame. Welcome to send me your comments or inquiries. Email: dshen@nd.edu

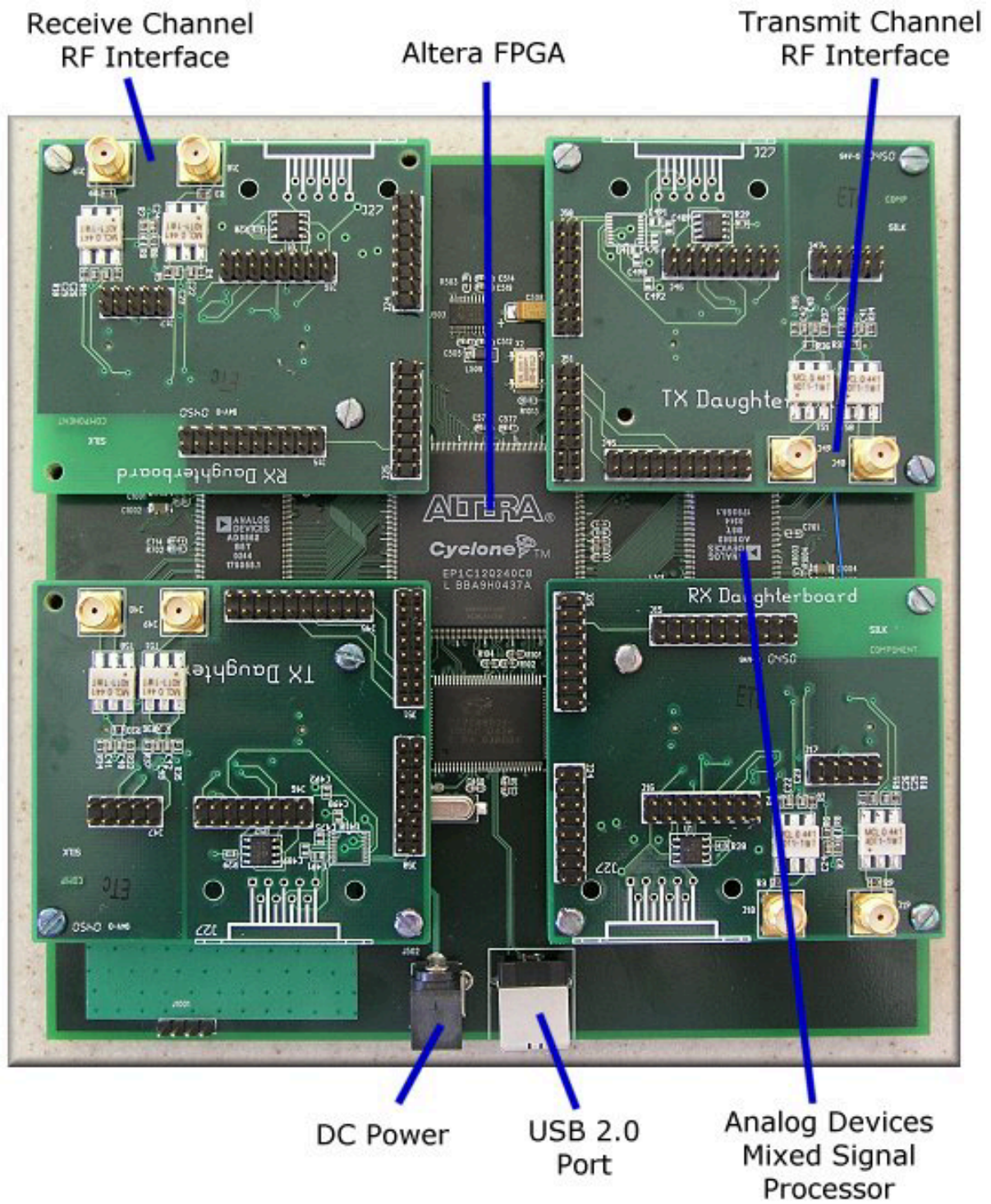


Figure 1: The USRP board: One Mother board, two RX / two TX daughter boards

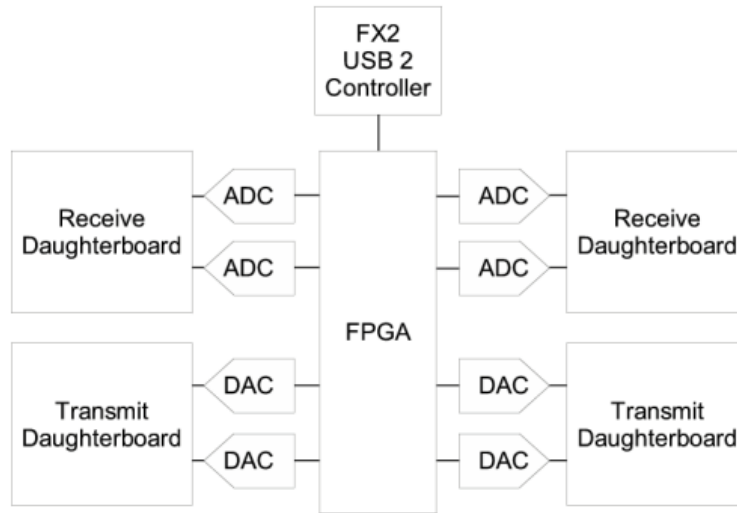


Figure 2: The USRP board

of up to about 150MHz, though. If we sample a signal with the IF larger than 32MHz, we introduce aliasing and actually the band of the signal of interest is mapped to some places between -32MHz and 32MHz. Sometimes this can be useful, for example, we could listen to the FM stations without any RF front end. The higher the frequency of the sampled signal, the more the SNR will be degraded by jitter. 100MHz is the recommended upper limit.

The full range on the ADCs is 2V peak to peak, and the input is 50 ohms differential. This is 40mW, or 16dBm. There is a programmable gain amplifier (PGA) before the ADCs to amplify the input signal to utilize the entire input range of the ADCs, in case the signal is weak. The PGA is up to 20dB. Note that we can use other sampling rates if desired. The available rates are all submultiples of 128MHz, such as 64 MS/s, 42.66 MS/s, 32 MS/s, 25.6 MS/s and 21.33 MS/s.

At the transmitting path, there are also 4 high-speed 14-bit DA converters. The DAC clock frequency is 128 MS/s, so Nyquist frequency is 64MHz. However, we will probably want to stay below about 50MHz or so to make filtering easier. So a useful output frequency range is DC to about 50MHz. The DACs can supply 1V peak to a 50 ohm differential load, or 10mW (10dBm). There is also PGA used after the DAC, providing up to 20dB gain. Note that the PGAs on both RX and TX paths are programmable.

So in principle, we have 4 input and 4 output channels if we use real samplings. However, we have more flexibility (and bandwidth) if we use complex (IQ) sampling. Then we have to pair them up, so we get 2 complex inputs and 2 complex outputs.

2.2 The daughter boards

On the mother board there are four slots, where you can plug in up to 2 RX daughter boards and 2 TX daughter boards. The daughter boards are used to hold the the RF receiver interface or tuner and the RF transmitter.

There are slots for 2 TX daughter boards, labeled TXA and TXB, and 2 corresponding RX daughter boards, RXA and RXB. Each daughter board slot has access to 2 of the 4 high-speed AD / DA converters (DAC outputs for TX, ADC inputs for RX). This allows each daughter board which uses real (not IQ) sampling to have 2 independent RF sections, and 2 antennas (4 total for the system). If complex IQ sampling is used, each board can support a single RF section, for a total of 2 for the whole system.

We can see there are two SMA connectors on each daughter board. We normally use them to connect the input or output signals.

There several kinds of daughter boards available now:

Basic daughter boards. Nothing fancy on it. Two SMA connectors are used to connect external tuners or signal generators. We can treat it as an entrance or an exit for the signal without affecting it. Some form of external RF front end is required.

TVRX daughter boards. With Microtune 4937 Cable Modem tuner equipped. This is a receive-only daughter board. The RF frequency ranges from 50MHz to 800MHz, with an IF bandwidth of 6MHz. What you need is just an antenna if your radio application is within this range, such as FM or TV detection.

DBSRX daughter boards. Similar with the TVRX boards. This is also receive-only. The RF frequency ranges from 800MHz to 2.4GHz.

Some new daughter boards are being developed, especially including the transceiver daughter boards. They will be on sale soon and will bring us great convenience.

2.3 The FPGA

Probably understanding what goes on the FPGA is the most important part for GNU Radio users. As shown in Figure 2, all the ADCs and DACs are connected to the FPGA. This piece of FPGA plays a key role in the GNU Radio system. Basically what it does is to perform high bandwidth math, and to reduce the data rates to something you can squirt over USB2.0. The FPGA connects to a USB2 interface chip, the Cypress FX2. Everything (FPGA circuitry and USB Microcontroller) is programmable over the USB2 bus.

Our standard FPGA configuration includes digital down converters (DDC) implemented with cascaded integrator-comb (CIC) filters. CIC filters are very high-performance filters using only adds and delays. The FPGA implements 4 digital down converters (DDC). This allows 1, 2 or 4 separate RX channels. At the RX path, we have 4 ADCs, and 4 DDCs. Each DDC has two inputs I and Q. Each of the 4 ADCs can be routed to either of I or the Q input of any of the 4 DDCs. This allows for having multiple channels selected out of the same ADC sample stream.

The digital up converters (DUCs) on the transmit side are actually contained in the AD9862 CODEC chips, not in the FPGA. The only transmit signal processing blocks in the FPGA are the interpolators. The interpolator outputs can be routed to any of the 4 CODEC inputs.

The multiple RX channels (1,2, or 4) must all be the same data rate (i.e. same decimation ratio). The same applies to the 1,2, or TX channels, which each must be at the same data rate (which may be different from the RX rate).

Figure 3 shows the block diagram of the USRP's receive path and the diagram of the digital down converter. The MUX is like a router or a circuit switcher. It determines which ADC (or constant zero) is connected to each DDC input. There are 4 DDCs. Each has two inputs. We can control the MUX using `usrp.set_mux()` method in Python. We will talk about the GNU Radio programming in later tutorials, but it's worth to take a look at it now.

Mux value:

```

      3           2           1
  1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
  +-----+-----+-----+-----+-----+-----+-----+-----+
  |  Q3  |  I3  |  Q2  |  I2  |  Q1  |  I1  |  Q0  |  I0  |
  +-----+-----+-----+-----+-----+-----+-----+

```

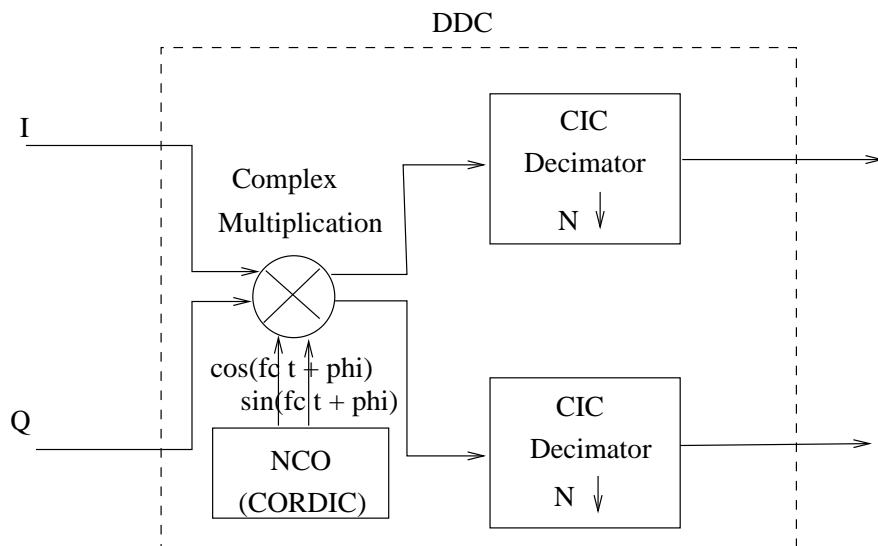
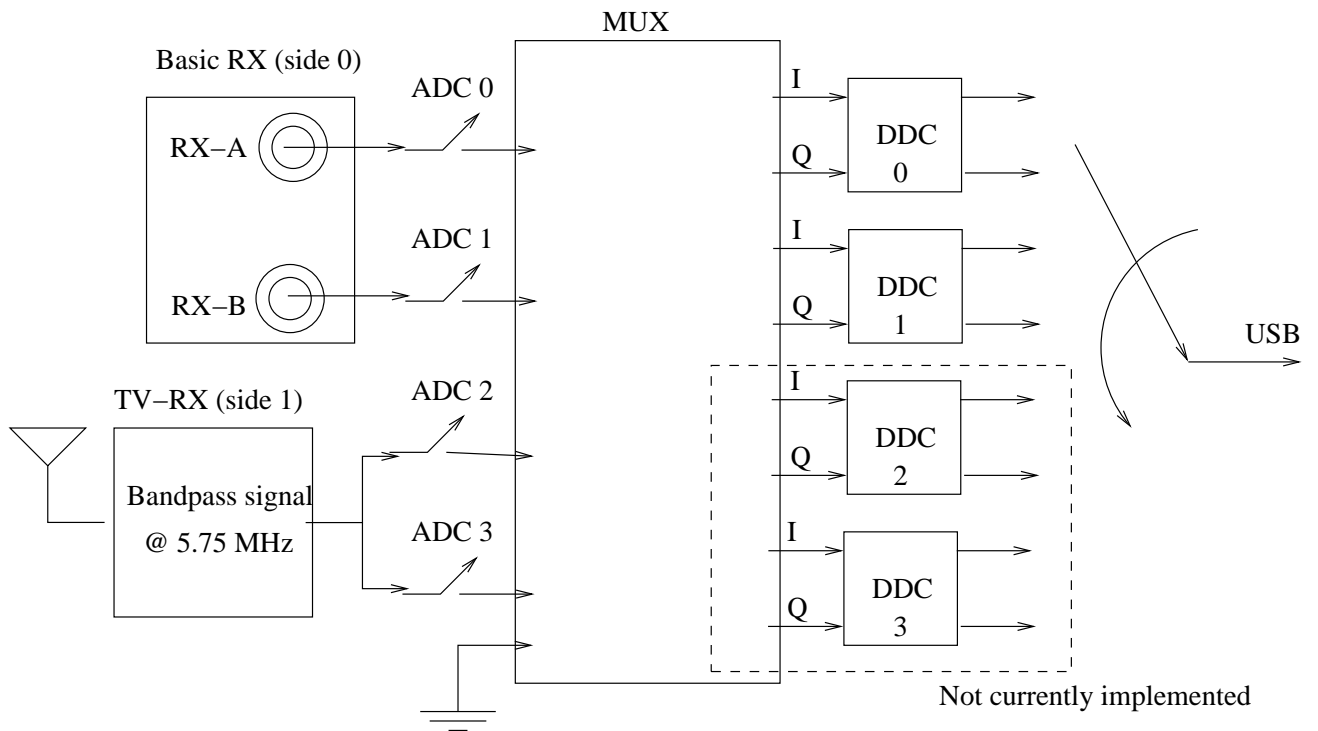


Figure 3: The block diagram of the USRP receive path

```
Each 4-bit I field is either 0,1,2,3
Each 4-bit Q field is either 0,1,2,3 or 0xf (input is const zero)
All Q's must be 0xf or none of them may be 0xf
```

We tell each input (I0, Q0, I1 ... I3, Q3) which ADC is connected to it by using 4 bits (0, 1, 2, 3 or 0xf). So a 32-bit integer would be enough for all 8 inputs to know which ADC is connected. Of course an integer in hexadecimal system will be more convenient if we want to use the `set_mux()` method. For most real sampling applications, the Q input of each DDC is constant zero. So quite often we don't need to modify the standard configuration of the FPGA. Actually it is anticipated that the majority of USRP users will never need to use anything other than the standard FPGA configuration.

Now let's see the digital down converter (DDC). What does it do? First, it down converts the signal from the IF band to the base band. Second, it decimates the signal so that the data rate can be adapted by the USB 2.0 and is reasonable for the computers' computing capability. The second part of Figure 3 shows the block diagram of the DDC. The complex input signal (IF) is multiplied by the constant frequency (usually also IF) exponential signal. The resulting signal is also complex and centered at 0. Then we decimate the signal with a factor N.

The decimator can be treated as a low pass filter followed by a downsampler. Suppose the decimation factor is D. If we look at the digital spectrum, the low pass filter selects out the band $[-\pi/D, \pi/D]$, and then the downsampler spread the spectrum in $[-\pi/D, \pi/D]$ to $[-\pi, \pi]$. So in fact, we have narrowed the bandwidth of the digital signal of interest by a factor of D. Regarding the bandwidth, we can sustain 32MB/sec across the USB. All samples sent over the USB interface are in 16-bit **signed integers** in IQ format, i.e. 16-bit I and 16-bit Q data (complex), resulting in 8M complex samples/sec across the USB. This provides a maximum effective total spectral bandwidth of about 8MHz by Nyquist criteria. Of course we can select much narrower ranges by changing the decimation rate. For example, suppose we want to design an FM receiver. The bandwidth of a FM station is generally 200kHz. So we can select the decimation factor to be 250. Then the data rate across the USB is $64\text{MHz} / 250 = 256\text{kHz}$, which is well suited for the 200kHz bandwidth without losing any spectral information. We can set the IF frequency of the DDC using `usrp.set_rx_freq()` method and set the decimation factor using `usrp.set_decim_rate()` method in Python. The decimation rate must be in [1, 256].

Note that when there are multiple channels (up to 4), the channels are interleaved. For example, with 4 channels, the sequence sent over the USB would be I0 Q0 I1 Q1 I2 Q2 I3 Q3 I0 Q0 I1 Q1, etc. The USRP can operate in full duplex mode. When in this mode, the transmit and receive sides are completely independent of one another. The only consideration is that the combined data rate over the bus must be 32 Megabytes per second or less.

OK! Finally the I/Q complex signal enters the computer via the USB. That's the software world!

At the TX path, the story is pretty much the same, except that it happens reversely. We need to send a baseband I/Q complex signal to the USRP board. The digital up converter (DUC) will interpolate the signal, up convert it to the IF band and finally send it through the DAC.

2.4 USRP Setup

There is almost nothing you need to do when you have received your new USRP boards. Probably the only thing you should do is to install all GNU Radio software packages correctly first, especially the `gnuradio-core`, `usrp` and `gr.usrp` packages. Please refer to tutorial 1 for the installation guide. Then you should be able to use the USRP board without any problem. Of course, you should make sure the DC supply, USB2.0 cable and daughter boards are properly connected.

When you are designing USRP related applications, you need USB device permissions to run your Python program. If you are the owner of your computer, maybe the simplest way is to put yourself in the `sudoers`.

Plug the Power cord into the usrp. You should see a green LED (hidden under the "RXA" daughter board, which is nearest to the USB connector) blinking at about 2Hz (twice a second). When a USRP application starts, the USRP will load two files from the directory `/usr/local/share/usrp/rev2`: `usrp_firmware.ihx` and `usrp_fpga.rbf`. The blinking LED should slow down to 1Hz (once a second) after the `usrp_firmware.ihx` file was loaded successfully. Ensure the file `usrp_fpga.rbf` is in that `/usr/local/share/usrp/rev2` directory. Note that if you install the `usrp` module from CVS, it won't be there. You'll have to get the tarball of `usrp-XXX.tar.gz` from [the GNU Radio download page](#), extract this one file, copy it to the right place.

```
cp usrp-*/fpga/rbf/usrp_fpga_rev2.rbf /usr/local/share/usrp/rev2/usrp_fpga.rbf
```

Finally let's try an example to test everything, including the software installation and the USRP. Find the `gnuradio-examples` directory, and go down into the `/python/usrp` subdirectory. Run

```
sudo ./usrp_oscope.py.
```

This should bring up something that looks like an oscilloscope, assuming you have installed `wxPython` and `gr-wxgui` correctly.

Then connect your speaker to the sound card and insert a piece of wire to the RX-A connector. Run the example:

```
sudo ./wfm_rcv_gui.py 101.5.
```

You should be able to hear strong FM stations.

If that works, then you have left the software installation and USRP setup stage behind you. Now you can use this exciting GNU Radio system to design your radio applications!

3 Conclusion

This article is actually a 'squeezed' version of the references [1][2][3], which covers almost all necessary information for most USRP users. Some details which may not be useful to most users are not introduced.

So far, we have completed the introduction to the software and hardware setup for GNU Radio. From next tutorial, we will focus on the GNU Radio programming.

References

- [1] **Universal Software Radio Peripheral on Wiki**,
<http://comsec.com/wiki?UniversalSoftwareRadioPeripheral>
- [2] **Rf Sections 4 USRP on Wiki**,
<http://comsec.com/wiki?RfSections4USRP>
- [3] Matt Ettus **USRP User's and Developer's Guide**,
http://home.ettus.com/usrp/usrp_guide.html
- [4] **USRP Install on Wiki**,
<http://comsec.com/wiki?UsrcInstall>