1 Linux

1.1 Helpful Commands

**sudo** - Run commands with administrative privileges. Note: For security reasons, you cannot access your network-mapped home directory when you run commands as root on the machines in AK 227. If you use “sudo” to run a single command as root, you might see error messages. To avoid these error messages, simply use “sudo -i” or “sudo su” to log in to run multiple commands as root. Or, use “sudo -H” to force sudo to use root’s home directory instead.

**ifconfig** - Configure network interfaces.
- Use the “-a” option to list all interfaces
- Use “up”/“down” to enable/disable interfaces

1.2 Modify the Iptables

Iptables is a firewall, installed by default on all official Ubuntu distributions (Ubuntu, Kubuntu, Xubuntu). When you install Ubuntu, iptables is there. However, in order to use the USRP2 Transmitter and Receiver Blocks, you need to modify the Iptables to allow all traffic.

You can access the Iptables on your computer by typing the following commands on the terminal:
- sudo bash
- cd /root/scripts/
- emacs iptables.sh

Now, you should have the Iptables open in front of you. You only need to modify two lines concerning the default firewall policies:

- IPTABLES.COM -P INPUT ACCEPT
- IPTABLES.COM -P FORWARD ACCEPT

When this modification is done, you should save the Iptables and execute it by typing the following command on the terminal:
- ./iptables.sh

2 MATLAB

2.1 How to Start MATLAB
- Windows: Choose the submenu “Programs” from the “Start” menu. From the “Programs” menu, open the “MATLAB” submenu. From the “MATLAB” submenu, choose “MATLAB”.
- Linux: At the prompt, type `matlab`
- Mac: Double-click on the icon for MATLAB.

You can quit MATLAB by typing `exit` in the command window.

2.2 The MATLAB Environment

The MATLAB environment consists of menus, buttons and a writing area similar to an ordinary word processor.

2.2.1 Command Window

The writing area that you will see when you start MATLAB, is called the command window. In this window, you give the commands to MATLAB. It is also useful if you just want to use MATLAB as a scientific calculator or as a graphing tool.

Once you have typed the command you wish MATLAB to perform, press <enter>. If you want to interrupt a command that MATLAB is running, type `<ctrl> + <c>`.

2.2.2 Command History Window

The commands you type in the command window are stored by MATLAB and can be viewed in the command history window. To repeat a command you have already used, you can simply double-click on the command in the history window, or use the `<up arrow>` at the command prompt to iterate through the commands you have used until you reach the command you desire to repeat.
2.3 Obtaining Help

To obtain help on any of the MATLAB commands, you simply need to type

\[ \text{help <command>} \]

at the command prompt.

For example, to obtain help on the gamma function, we type at the command prompt:

\[ \text{help gamma} \]

You may also get help about commands using the “Help Desk”, which can be accessed by selecting the MATLAB Help option under the Help menu.

2.4 Variables in MATLAB

You can define your own variables in MATLAB. Suppose you need to use the value of \( 3.5 \sin(2.9) \) repeatedly. Instead of typing \( 3.5 \sin(2.9) \) over and over again, you can denote this variable as \( x \) by typing the following:

\[ x = 3.5 \sin(2.9) \]

If you do not want to have the result of a calculation printed-out to the command window, you can put a semi-colon at the end of the command; MATLAB still performs the command in “the background”.

- To know what variables have been declared, type \texttt{whos}. Alternatively, you can view the values by opening the workspace window. This is done by selecting the Workspace option from the View menu.
- To erase all variables from the MATLAB memory, type \texttt{clear}.
- To erase a specific variable, say \( x \), type \texttt{clear x}.
- To clear two specific variables, say \( x \) and \( y \), type \texttt{clear x y}, that is separate the different variables with a space. Variables can also be cleared by selecting them in the Workspace window and selecting the delete option.

2.5 Vectors and Matrices in MATLAB

2.5.1 Vectors

Create a vector by putting the elements within \[ \] brackets.

Example: \[ x = [1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10] \]

We can also create this vector by typing \texttt{x=1:10}.

The vector \( (1.1 \ 1.2 \ 1.3 \ 1.4 \ 1.5) \) can be created by typing \texttt{x=[1.1 \ 1.2 \ 1.3 \ 1.4 \ 1.5]} or by typing \texttt{x=1:0.1:1.5}.

2.5.2 Matrices

Matrices can be created according to the following example. The matrix \( A \) is created by typing

\[ A = [1 \ 2 \ 3 \ 4 \ 5 \ 6 ; 7 \ 8 \ 9] \]

\( i.e., \) rows are separated with semi-colons.

2.5.3 Additional Information

- In MATLAB, the first index of a vector or matrix starts at 1, not 0 as is common with other programming languages.
- If the matrices (or vectors which are special cases of a matrices) are of the same dimensions then matrix addition, matrix subtraction and scalar multiplication works just like we are used to.
- If you want to apply an operation such as squaring each element in a matrix, you have to use a dot \( . \) before the operation you wish to apply. The dot allows you to do operations element-wise. All built-in functions such as \texttt{sin}, \texttt{cos}, \texttt{exp} and so on automatically act elementwise on a matrix.

3 USRP2 Hardware

3.1 XCVR2450 Daughtercard

- The XCVR2450 daughtercard supports two frequency bands: 2.4 to 2.5 Ghz and 4.9 to 5.9 GHz
- There are two cables inside the USRP2 that connect the the RF1 and RF2 SMA connectors on the front of the USRP2 case to connectors J2 and J1 on the daughtercard. We have set up all of the USRP2s such that RF1 connects to J2 and RF2 connects to J1.
- The XCVR2450 always uses J1 for transmit and J2 for receive. (This behavior appears to hard-coded into the FPGA firmware and is not software-controllable at present time.)
- To summarize:
  Receive: RF1 (J2)
  Transmit: RF2 (J1)

3.2 Sampling

3.2.1 Decimation

The sampling frequency of the USRP2 is:

\[ f_s = (100 MSps)/(\text{decimation}) \]
The *decimation* value can be any multiple of 4 from 4 through 512. (Non-multiples of 4 yield different types of cascaded integrator-comb (CIC) filters, but this probably isn’t what you want.)

The decimation parameter affects the bandwidth of the received signal. This is because it affects the sampling rate. Decimation essentially involves lowpass-filtering and then downsampling by the decimation factor. Recall from ECE 3311 that downsampling by a factor of X is equivalent to sampling the original signal at a factor of \( f_s/X \) instead of \( f_s \). Therefore, larger decimation factors result in smaller sampling rates, and thus smaller bandwidths. (After all, the bandwidth depends on the Nyquist frequency, or \( f_s/2 \).)

The frequency resolution of the FFT is determined by the sampling frequency \( f_s \) and the number of points (N) in the FFT:

\[
\text{frequency resolution} = \frac{f_s}{N}
\]

For example, an FFT with 1024-point and a sampling frequency of 128 kHz has a resolution of

\[
\frac{(128 \ \text{kHz})}{(1024 \ \text{points})} = 125 \ \text{Hz}.
\]

In other words, each bin of the FFT is 125 Hz wide. (Spectrum analyzers commonly refer to this value as “resolution bandwidth.”)

Recall the tradeoffs between resolution in time and resolution in frequency. FFTs with more points have finer-grain frequency resolution, but require more time to collect the samples (since there are more points). FFTs with fewer points have coarser resolution in frequency. Also note that larger FFTs are more computationally-expensive, which is important to keep in mind because you will be running the FFTs in real time.

Larger decimation values result in a smaller sampling rate, thus, if you keep your FFT at 1024 points, larger decimation values will yield finer-grain frequency resolution because \( \text{BW}/(\text{N points}) \) will be a smaller resolution bandwidth value.

The punchline: You should use a high decimation value (e.g., 400 or 512) and a sufficient number of FFT points (e.g., 1024) when taking fine measurements, such as your oscillator frequency offset error measurements. This will afford the best possible measurement precision.

Finally, here are a few additional useful links:

- [http://zone.ni.com/devzone/cda/tut/p/id/3983](http://zone.ni.com/devzone/cda/tut/p/id/3983)

### 3.2.2 Interpolation

The interpolation parameter is used for transmitting (DAC) whereas the decimation parameter is used for receiving (ADC).

Interpolation has the same range of values (any multiple of 4 from 4 to 512) as decimation.

According to [http://gnuradio.org/redmine/wiki/gnuradio/USRP2GenFAQ](http://gnuradio.org/redmine/wiki/gnuradio/USRP2GenFAQ), “The FPGA talks to the DAC at 100 MS/s just like it talks to the ADC at 100 MS/s. The interpolation from 100 MS/s to 400 MS/s happens inside the DAC chip itself. Unless you are doing something fancy, you can think of the DAC as operating at 100 MS/s.”

### 3.3 Clocking

The clock rate of the Spartan-3 FPGA in the USRP2 is 100 MHz.

### 3.4 DDC and DUC

The USRP2 FPGA uses a digital down converter (DDC) and a digital up converter (DUC) to convert between baseband and intermediate frequency (IF).

#### 3.4.1 Digital Down Converter

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 DDC basically does two things. 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 DDCs in USRP2 are very similar to the USRP1 with a 4 stage CIC but with 2 HBF filters. The CIC can decimate in the range \([1..128]\). The high rate HBF has 7 taps and it decimates by 2. The low rate HBF has 31 taps and also decimates by 2.

#### 3.4.2 Digital Up Converter

The digital up converter (DUC) will interpolate the signal, up convert it to the IF band and finally send it through the DAC. There us a 4 stage CIC and 2 HBF filters each working in interpolation mode.
3.4.3 More Information

For more information on these converters, see the following pages:

- http://gnuradio.org/redmine/wiki/gnuradio/USRP2GenFAQ

4 Differential Phase-Shift Keying (DPSK)

Differential phase shift keying (DPSK) is a form of phase modulation that conveys data by changing the phase of the carrier wave. Referring to both BPSK and QPSK modulation schemes, there is phase ambiguity whenever the constellation is rotated by some amount due to the presence of phase distortion within the communications channel through which the signal passes. This problem can be resolved by using the data to change rather than set the phase.

For example, in differentially-encoded BPSK (DBPSK) system is capable of transmitting a binary “1” by adding 180° to the current phase, while a binary “0” can be transmitted by adding 0° to the current phase.

Since a physical channel is present between the transmitter and receiver within a communication system, this channel will often introduce an unknown phase-shift to the PSK signal. Under these circumstances, the differential schemes can yield a better error-rate than the ordinary schemes which rely on precise phase information.