ECE-320 Lab 7: Introduction to the dsPIC30F6015.

Overview: *In this lab we will get software set up to utilize the dsPIC30F6015 in the next lab to control a simple wheel. We will also review some of the basic features we will need to utilize in our next lab. Finally, you will be given the opportunity to have the joy of reviewing how to program in C. Most of the code will be given to you, but you will have to make some modifications.*

The dsPIC30F6015 has been mounted on a carrier board that allows us to communicate with a terminal (your laptop) via a USB cable. In what follows you will need to make reference to the pin out of the dsPIC30F6015 (shown in Figure 1) and the corresponding pins on the carrier (shown in Figure 2).

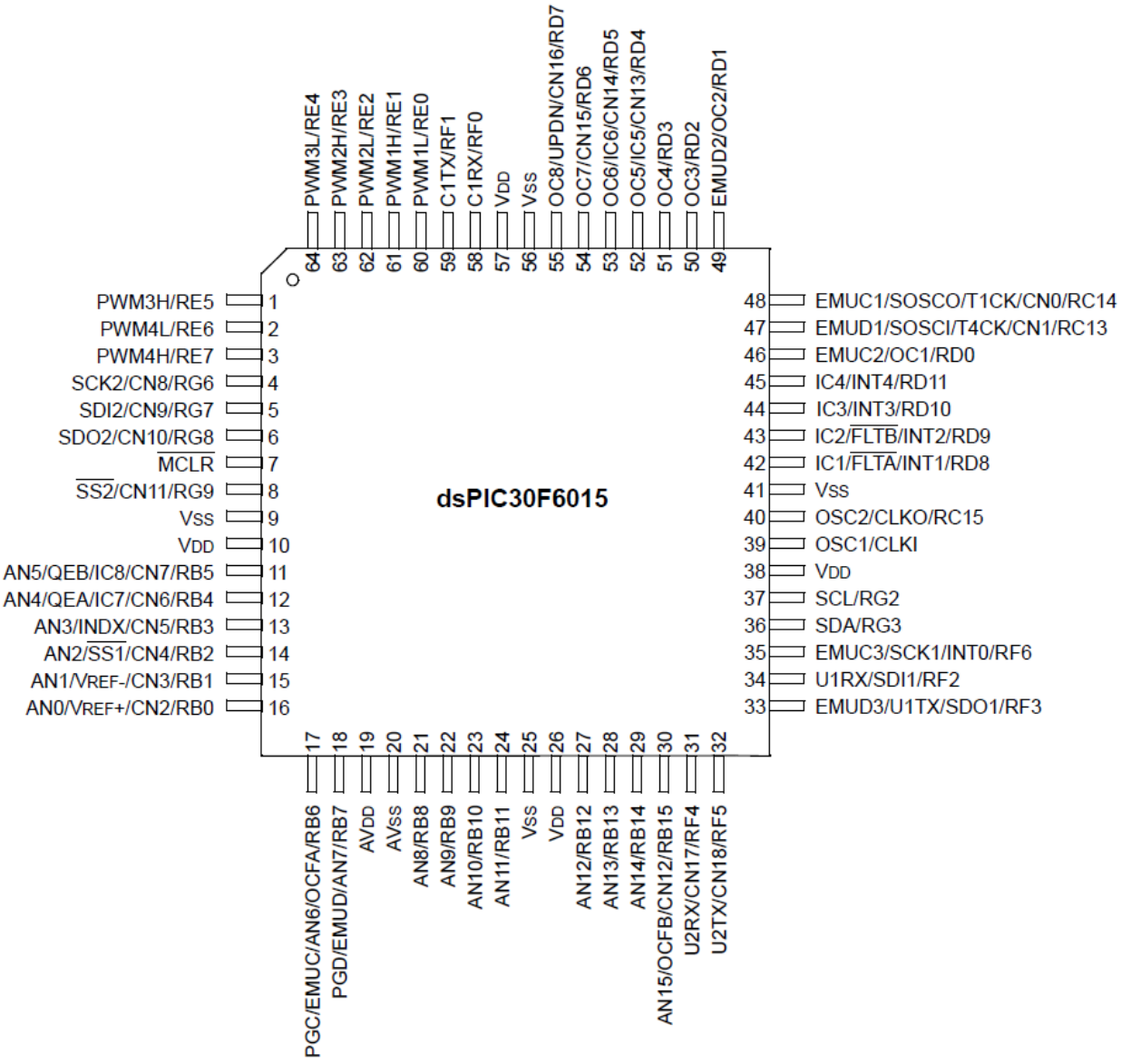


Figure 1. dsPIC30F6015 64-PIN pinout.

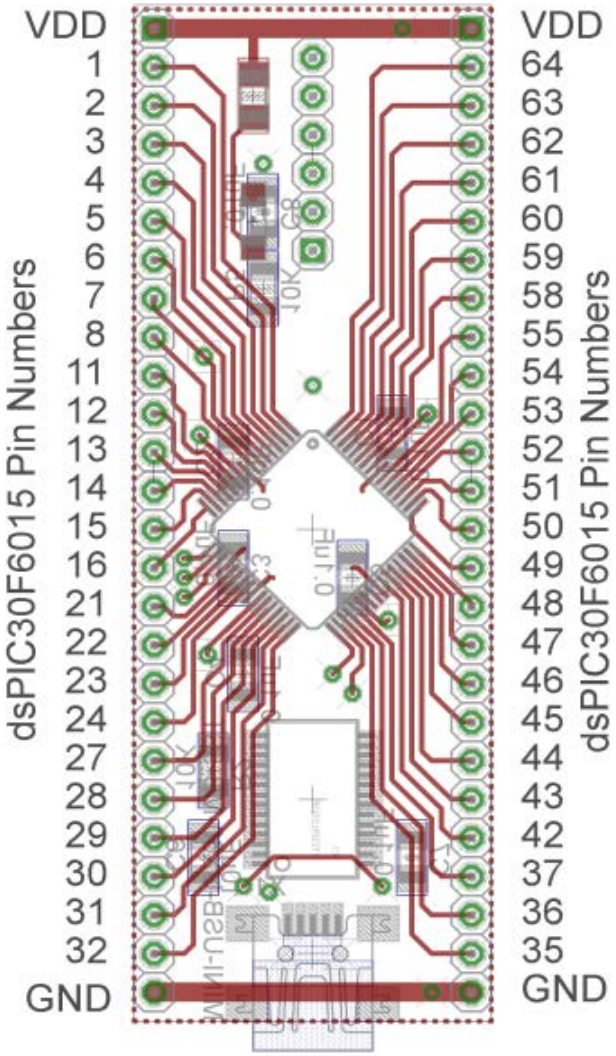


Figure 2. dsPIC30F6015 carrier. Note that the pin numbers are not consecutive.

Part A (Copying the initial files)

Set up a new folder for this lab, and copy the files for this Lab from the class website to this folder.

Part B (Downloading and installing MPLAB)

You will need to go to the website

http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469

and download the newest version of MPLAB (MPLAB IDE v8.83+). If you have an older of MPLAB it will probably work (mine is v8.76) and you may be able to skip this step. It is probably easiest to download a zipped version and the unzip it. Once the file has been unzipped, run **setup.exe**. Accept all of the defaults. Do not intall any C compilers yet.

Part C (Downloading and installing MPLAB C30 compiler)

You will need to go to the website

http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en536656

and download the MPLAB C compiler for the dsPIC. This is the academic version. You will need to register your use (click on **register now**). Once the file has been downloaded, you need to run **mplab30_v3_30c_windows.exe** to install it. I used the **Versioned Directory Name.** Select the **Evaluation Compiler.** The devices we want are the **dsPIC DCSs**

Part D (Setting up Communications with Secure CRT)

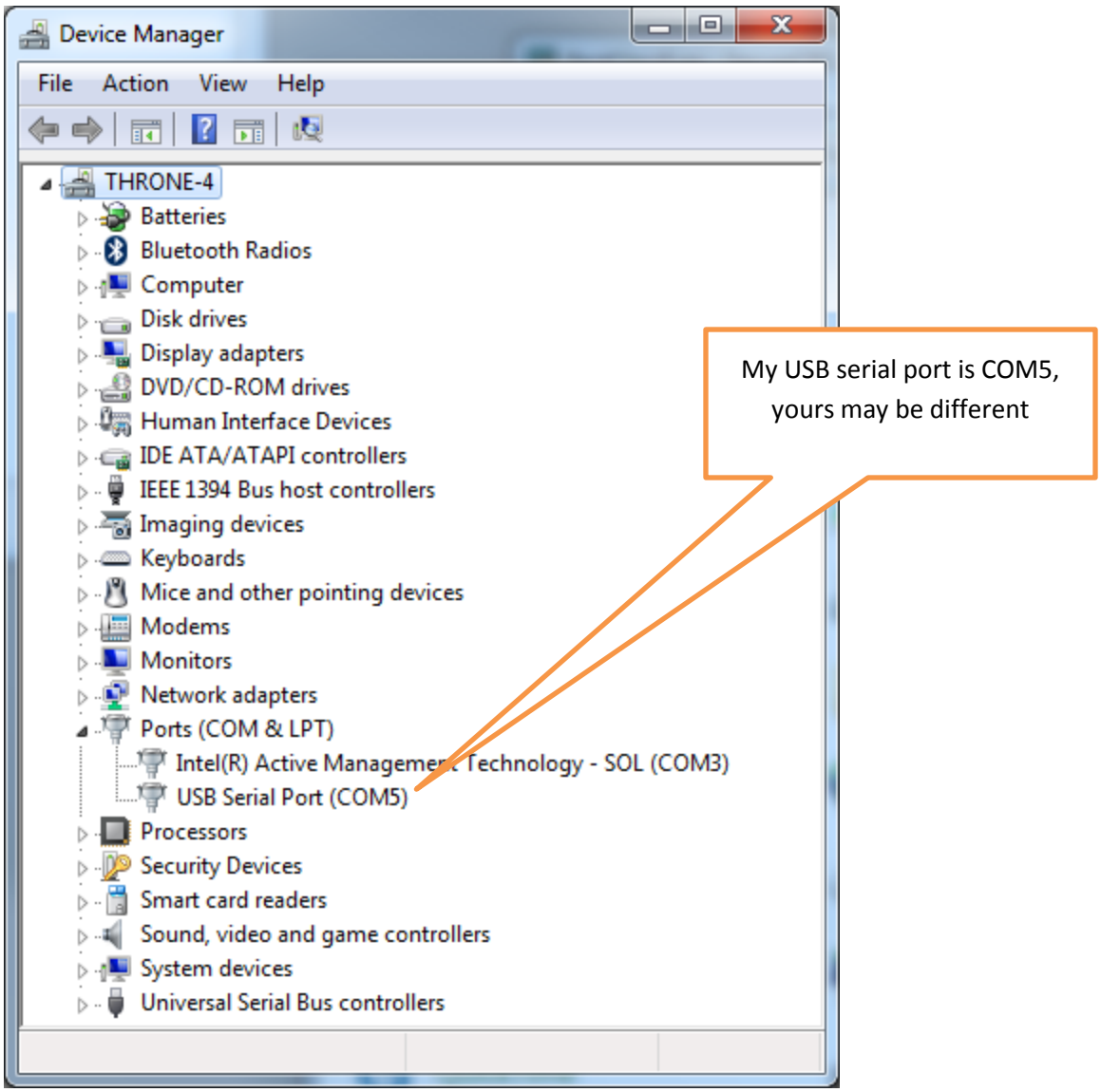
In this step we will make sure all of our systems are working before we go on.

Connect the PICkit3 to the carrier board and your computer (the white arrow goes near the white spot on the carrier board.)

Connect the separate communication cable to the carrieer board and your computer.

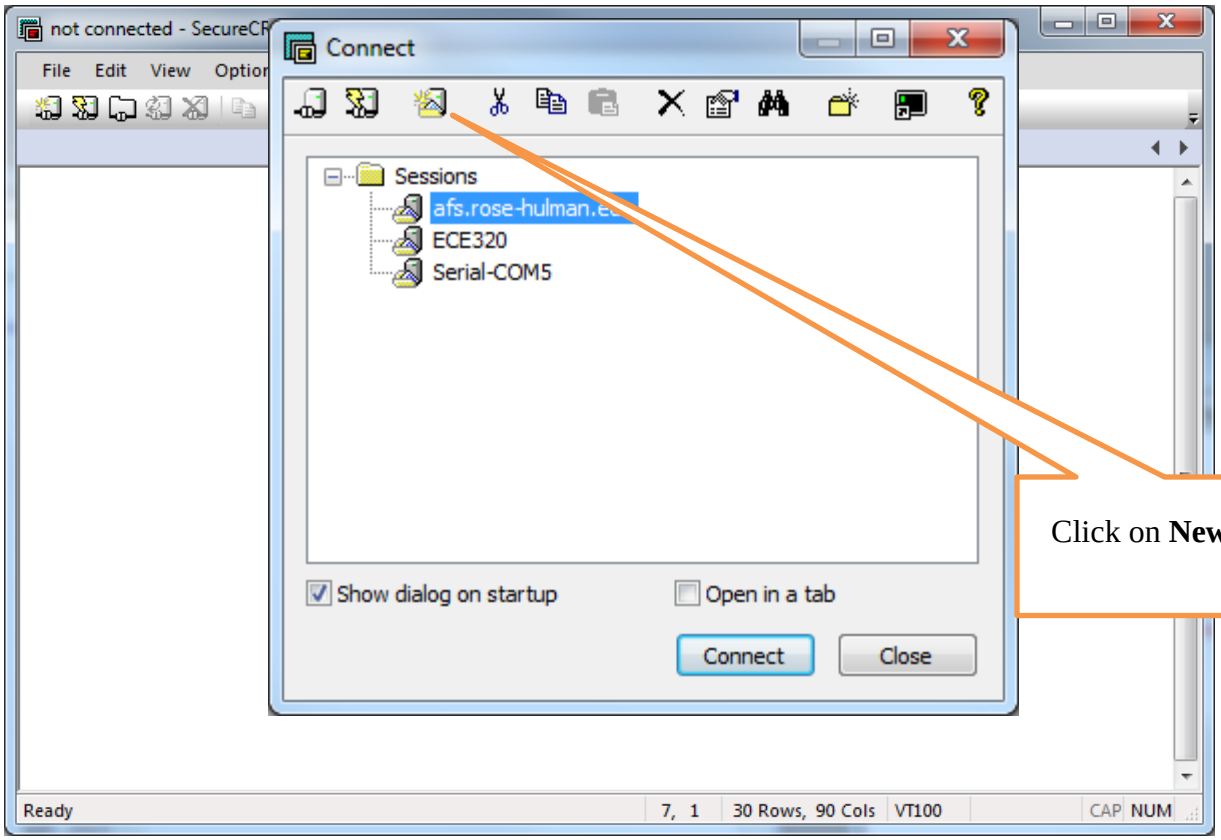
Connect power to the power board and toggle to on switch (the green LED should then be on)

Wait a few minutes (see if your system is trying to find any devices). The go to **Programs->Control Panel** and select **Dervice Manager**. Expand the **Ports** option as shown below. The following figure shows that the USB is connected to my COM5 port. You will need this information shortly.

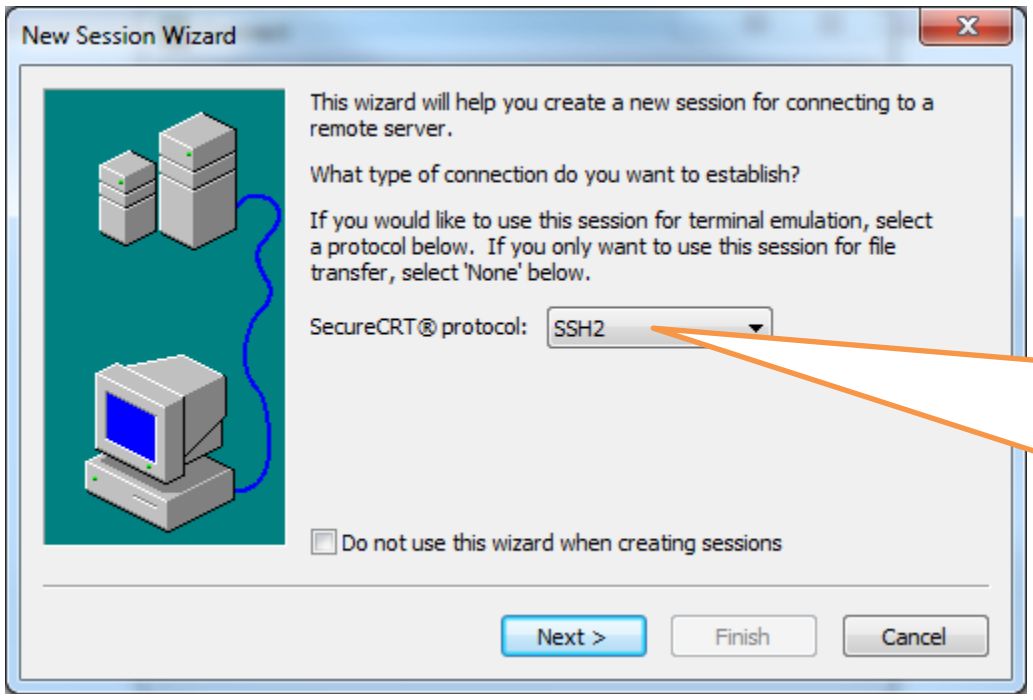


We are going to want to be able to record and plot data, so we will be writing to the screen. In order to do this we will using Secure CRT because it is installed on your computer by default.

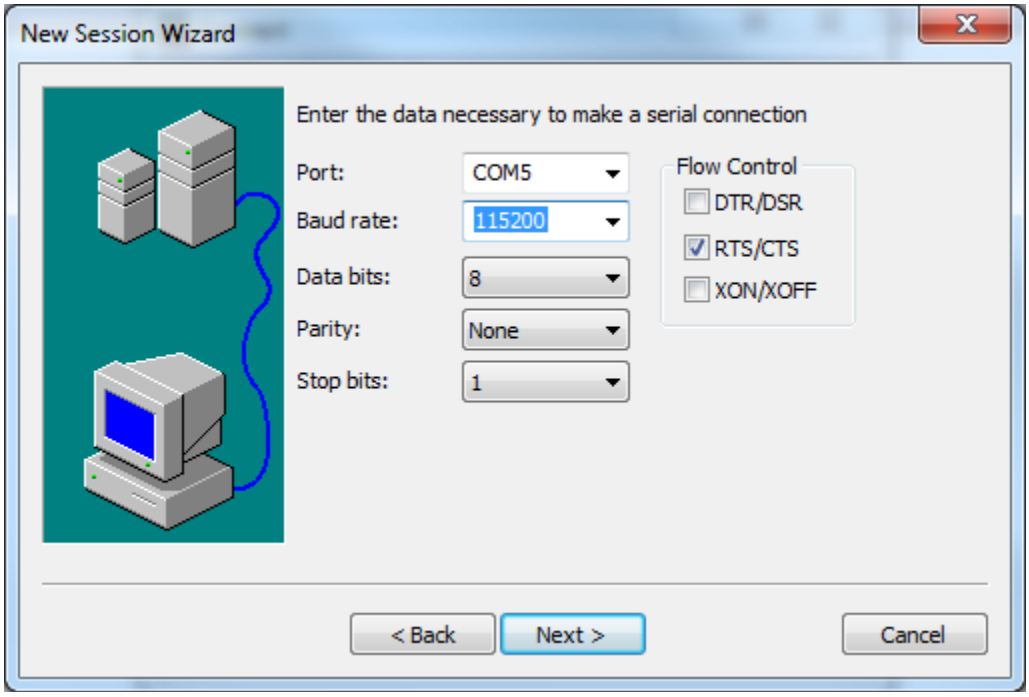
Go to **Programs→All Programs→SecureCRT→SecureCRT6.2** (it's ok if you don't have 6.2). You should get a screen like this (although the list of Sessions will probably be shorter for you):



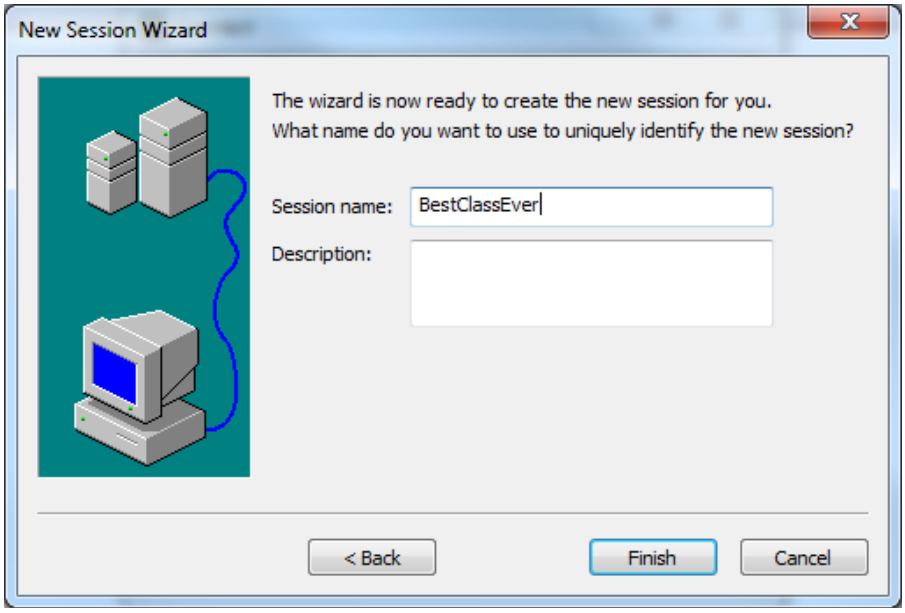
You should then get a screen that looks like this



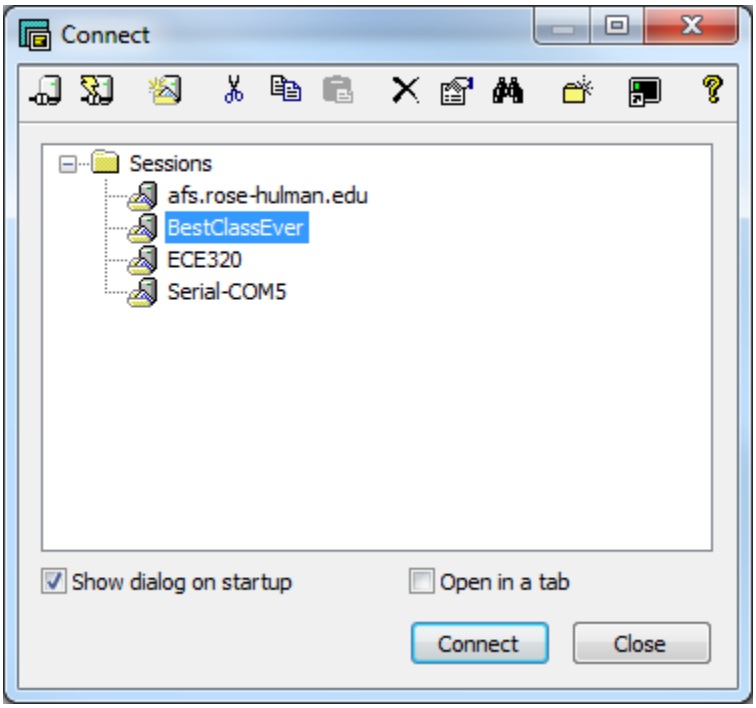
You will next get the following screen. You should already know which port to use (it may not be COM5 on your computer). Select a Baud rate of 115200 and be sure everything is filled in as below. The click on **Next**.



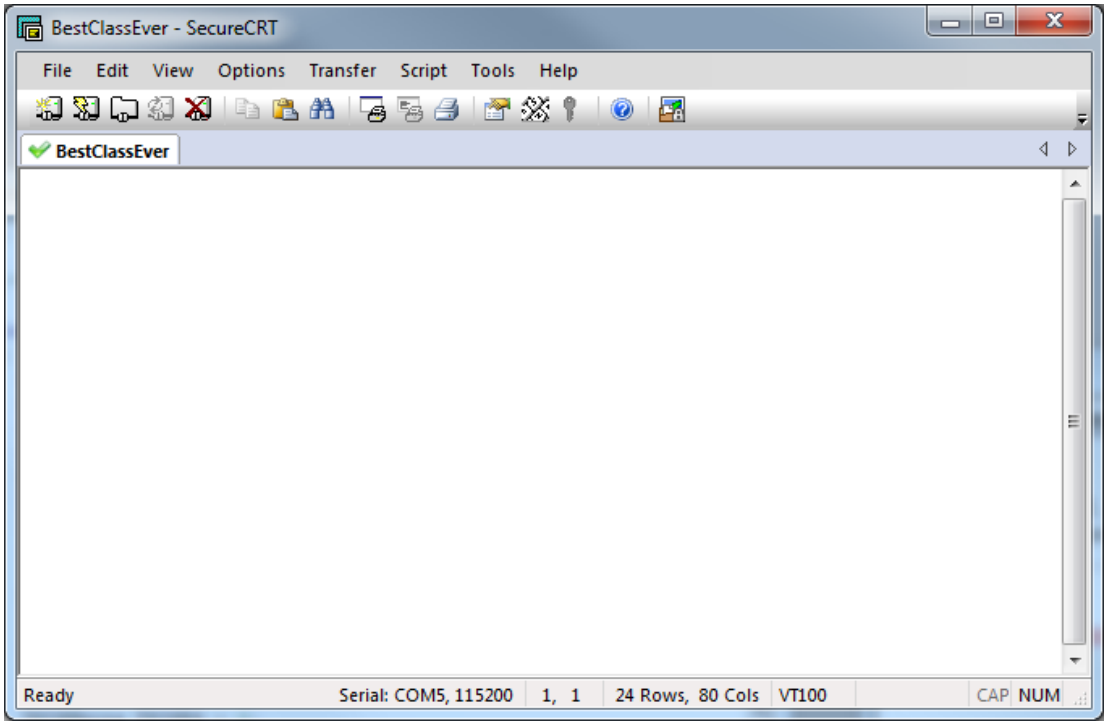
Finally, you need to create a session name so you do not have to go through these steps again. My choice in names was pretty intuitive. Once you have named your session select **Finish**.



Finally, you will get a screen like that below, and you need to select **Connect**.



Now you should have a terminal like that shown below



From this screen, the option we will use most is **File-> Log Session.** This will allow us to write data collected to a file to be processed later.

Part E (Putting it all together)

Now we are ready to start. Note that at some point in the following process you may need to let the system download firmware. Be sure you have selected the correct device (or it will do this twice).

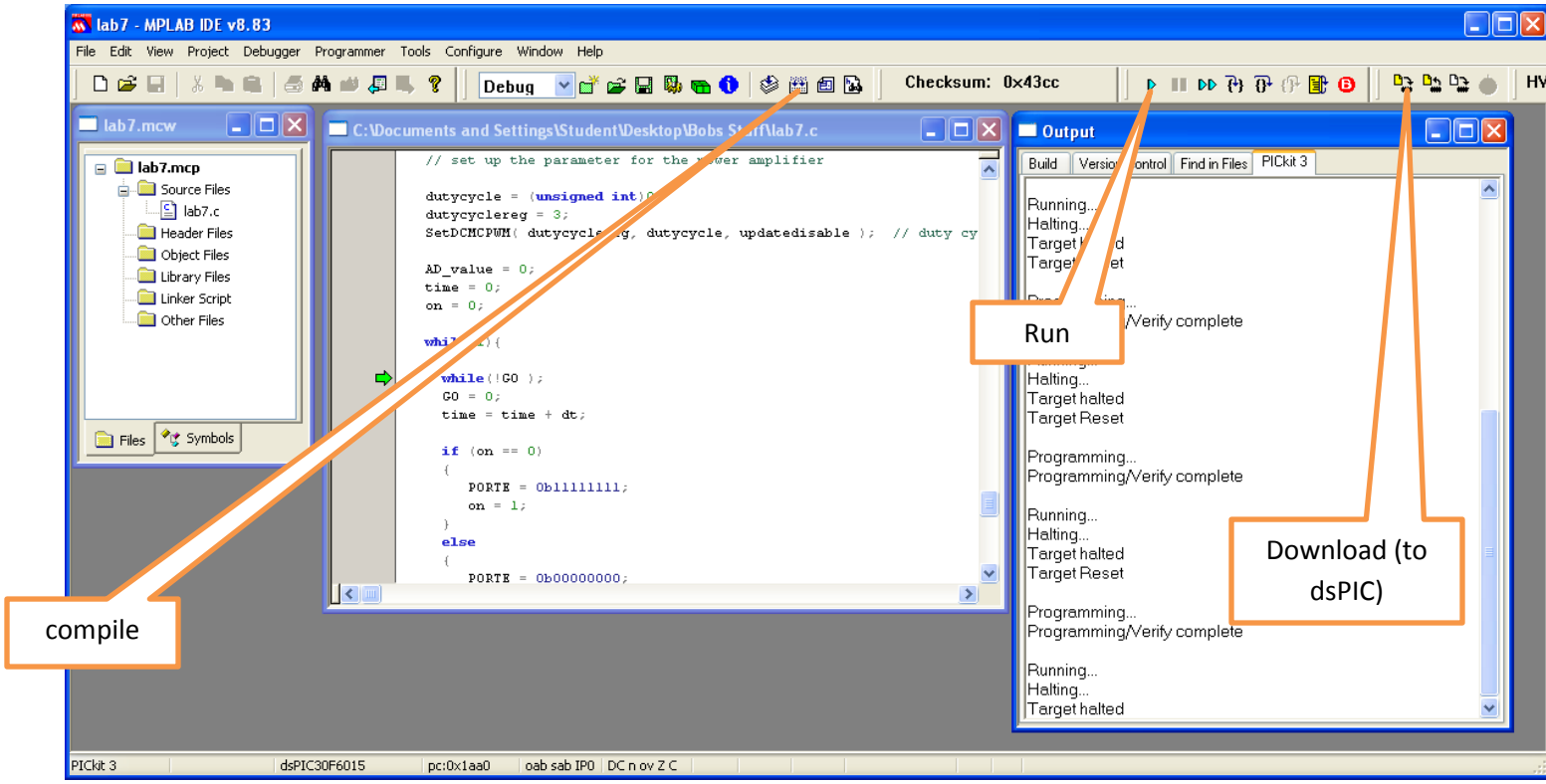
Start **MPLAB**

Select **Project** and open the project **lab7.mcp**

Select **Configure,** then **Select Device** and choose **dsPIC30F6015**

Select **Debugger,** then **Select Too**l and choose **PICkit 3**

You may need to click on **lab7.c (**under **Source Files)** to see the C code. You should get a screen (more or less) like the following:

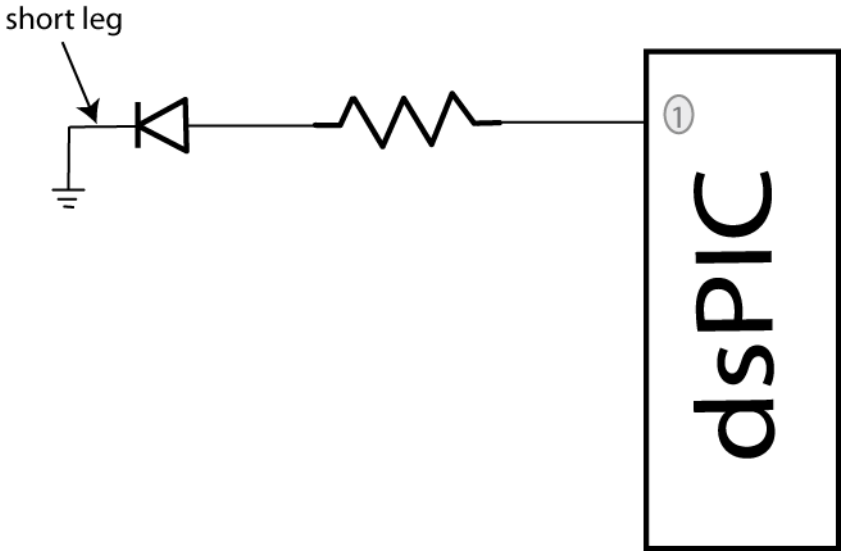


You should **compile** the existing program**, download** it to the dsPIC**,** and **run** it**.** You should see a sequence of numbers on your screen (the secure CRT screen). These correspond to sample times.

Part F (Turning one LED on and off)

In this step we will make one LED turn on and off.

Locate pin 1 (RE5). Connect a resistor and an LED to this pin as shown below:



We need to get this ready for output, so in the file **lab7.c**, insert the lines (before the main loop)

TRISEbits.TRISE5 = 0;

You will see in the main loop we are writing all 1's or all zeros to all of port E every other time interval.

Compile your code and run it. Your LED should turn on and off regularly.

Part G (Running Two LEDs)

Connect your other LED to pin2 (RE6). Modify you code so the LED's alternate (one goes on and the other off). Once this has been checked off, comment out the code that turns the LEDs on and off in the main code (do not delete it).

Have this verified by your instructor.

Part H (Utilizing PWM to run your LEDs)

We will now utilize the PWM function to drive the LED. Note that RE6 corresponds to PWM4L. In the routine **pwm_init**, change PWM_PDIS4L to PWM_PEN4L. This allows us to use this pin as a PWM signal.

Set the variable AD_value = 512.

Uncomment the code:

*dutycycle = (unsigned int) ((double) AD_value)*scale;*

dutycyclereg = 4;

SetDCMCPWM(dutycyclereg, dutycycle, updatedisable);

This code determines the duty cycle value and writes it to dutycycle register 4.

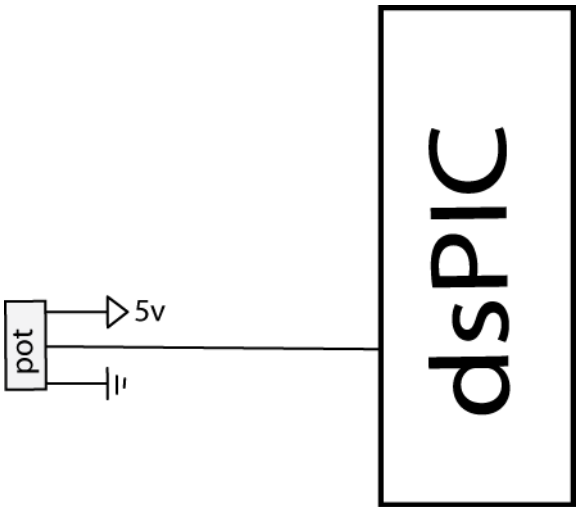
Compile and run your code. A single LED should light (constantly).

Modify the values in AD_value from 0 to 512 to verify that the brightness of the LEDs changes as the dutycycle changes. Note that at some point the LED stops getting brighter, so don't worry about that.

Part I (Reading in an A/D value)

Now we want to be able to connect a potentiometer so we can have a variable reference. The software is currently set up for A/D input on AN3/RB3. You will need to do the following things:

1) Connect the potentiometer as shown below
2) Set the appropriate TRIS bits for an input signal
3) Comment out the part of the code that sets the AD_value = 0;
4) Modify the printf statement to allow you to also print out the A/D value from the pot



Part J (Changing the dutycycle value by reading from the potentiometer)

Modify your code so the value of the dutycycle in the PWM routine is set by reading in a value from the potentiometer. As you turn the potentiometer the LED should change intensity.

Have this verified by your instructor.

Instructor Verification

Part G (2 LEDs) _____

Part J (Pot controls LED) _____

Part J (Pot values written to terminal) _____