CALIFORNIA STATE POLYTECHNIC UNIVERSITY, POMONA
COLLEGE OF ENGINEERING

ECE 3301L Spring 2019 Microcontroller Lab Felix Pinai
Session 2

LAB 1: Getting Familiar with the MPLAB X software

To perform the lab below, you need to download the spec of the processor that we are using in the lab. Go to the following link:

http://ww1.microchip.com/downloads/en/DeviceDoc/39689f.pdf

Download the datasheet and save it somewhere that you can easily access it (Desktop). Don't use the datasheet provided in your textbook because not all PIC18F microcontrollers have the same definitions.

Part 1)

You need to get the exercise given on the Tutorial done (see attached 'MPLABX and PICKIT Tutorial.pdf'). Build the circuit given to you on the 'LAB1_schematics.pdf' file and run the program. If your program runs properly, the four LEDs will blink at different rate depending on the voltage V_Var supplied by the potentiometer to the pin RA0. The lower the voltage, the faster the LEDs will blink. When the voltage is increased, the blinking will slow down. Check with the instructor when this part is completed.

Note: On the schematics, the dotted block is the development board that I am going to give to your team on the first day of class. If you want to pre-wire the circuit, make the connections from the board to the external circuits.

Here is a copy of the program provided in the tutorial:

```
#include <p18f4321.h>
#pragma config OSC = INTIO2
#pragma config WDT=OFF
#pragma config LVP=OFF
#pragma config BOR =OFF
#define           delay  5

// Prototype Area
void Init_ADC(void);
unsigned  int Get_Full_ADC(void);
void Flash_LED(unsigned char);

void main(void)
{
 unsigned int ADC_Result;
     Init_ADC();
    TRISB =0x00;
```

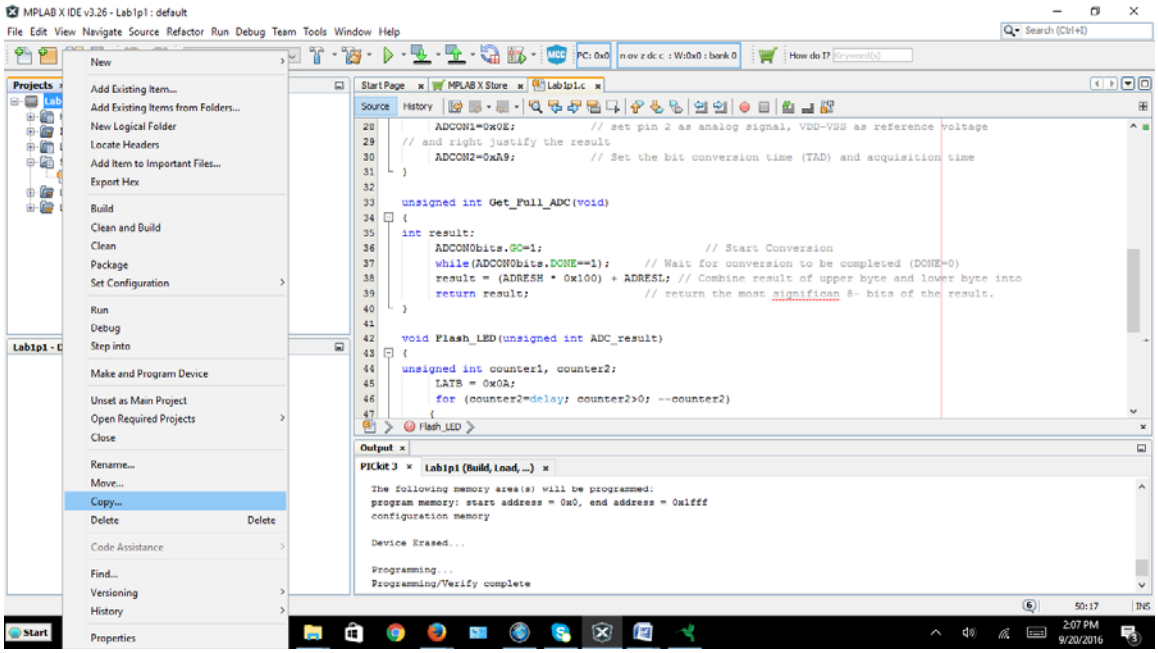
```
    while(1)
    {
        ADC_Result = Get_Full_ADC();
        Flash_LED(ADC_Result);
    }
}

void Init_ADC(void)
{
    ADCON0=0x01;                // select channel AN0, and turn on the ADDC subsystem
    ADCON1=0x0E;                // set pin 2 as analog signal, VDD-VSS as reference voltage
                                // and right justify the result
    ADCON2=0xA9;                // Set the bit conversion time (TAD) and acquisition time
}

unsigned int Get_Full_ADC(void)
{
int result;
    ADCON0bits.GO=1;                    // Start Conversion
    while(ADCON0bits.DONE==1);          // Wait for conversion to be completed (DONE=0)
    result = (ADRESH * 0x100) + ADRESL; // Combine result of upper byte and lower byte into
    return result;                      // return the most significant 8- bits of the result.
}

void Flash_LED(unsigned int ADC_result)
{
unsigned int counter1, counter2;
    LATB = 0x0A;
    for (counter2=delay; counter2>0; --counter2)
   {
       for (counter1=ADC_result ; counter1>0; -- counter1);
   }

   LATB = 0x05
   for (counter2=delay; counter2>0; --counter2)
   {
       for (counter1=ADC_result ; counter1>0; -- counter1);
   }
}
```

Part 2)

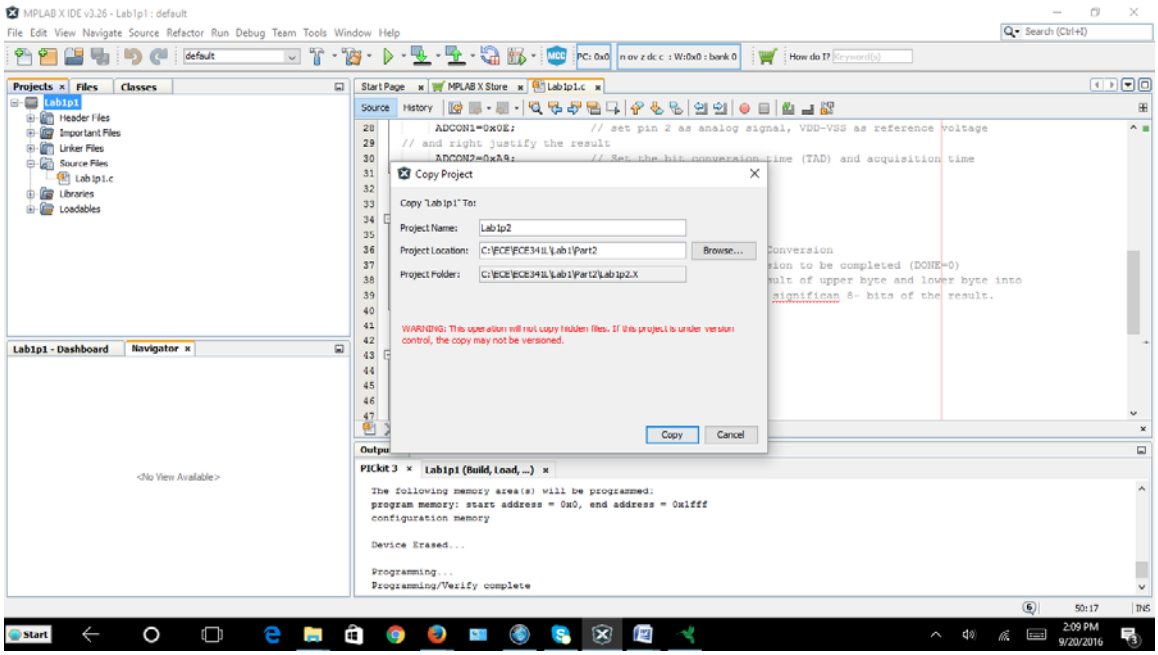
This section deals with the use of the serial port on the PIC development board that I have provided. It has a UART-to-USB translator chip designed to connect a serial port to an USB port available on your laptop. To be able to access to that USB port, you must perform Step 4) of the syllabus.

To avoid repeating the steps taken in the tutorial to create a new project, we can use an older project to quickly duplicate it into a new one. Follow the steps below:

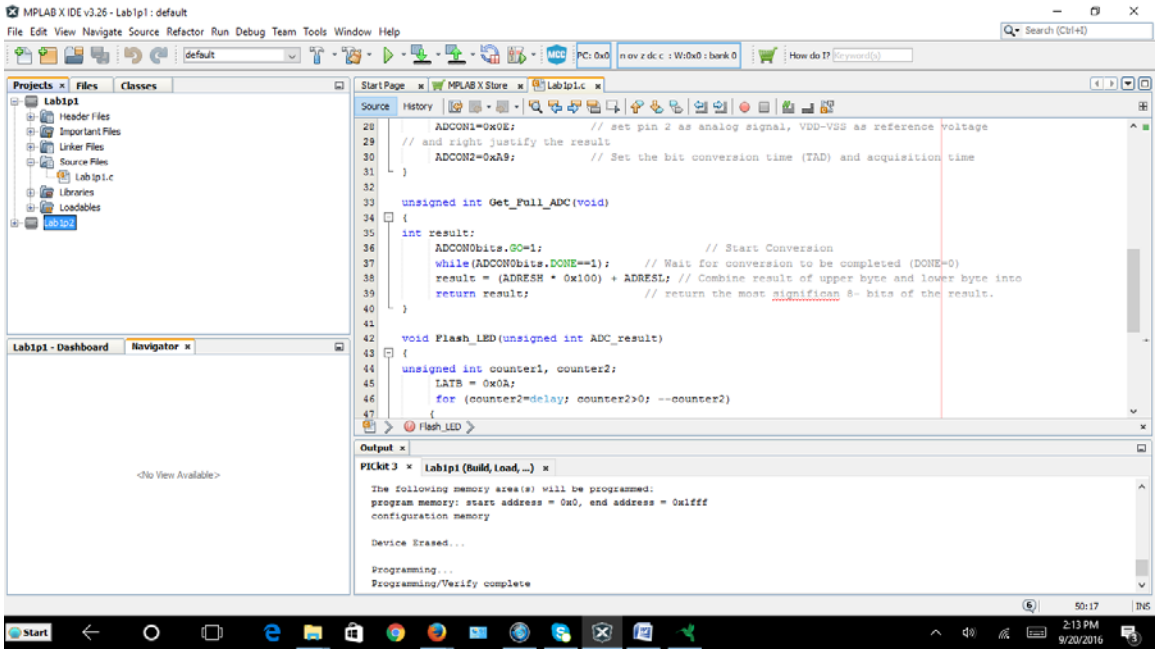
From the Lab1 Part1 project that you have done in the Tutorial, click the project name ‘Labp1’ and right click the mouse. Move the cursor down to ‘Copy’ field and click on it.



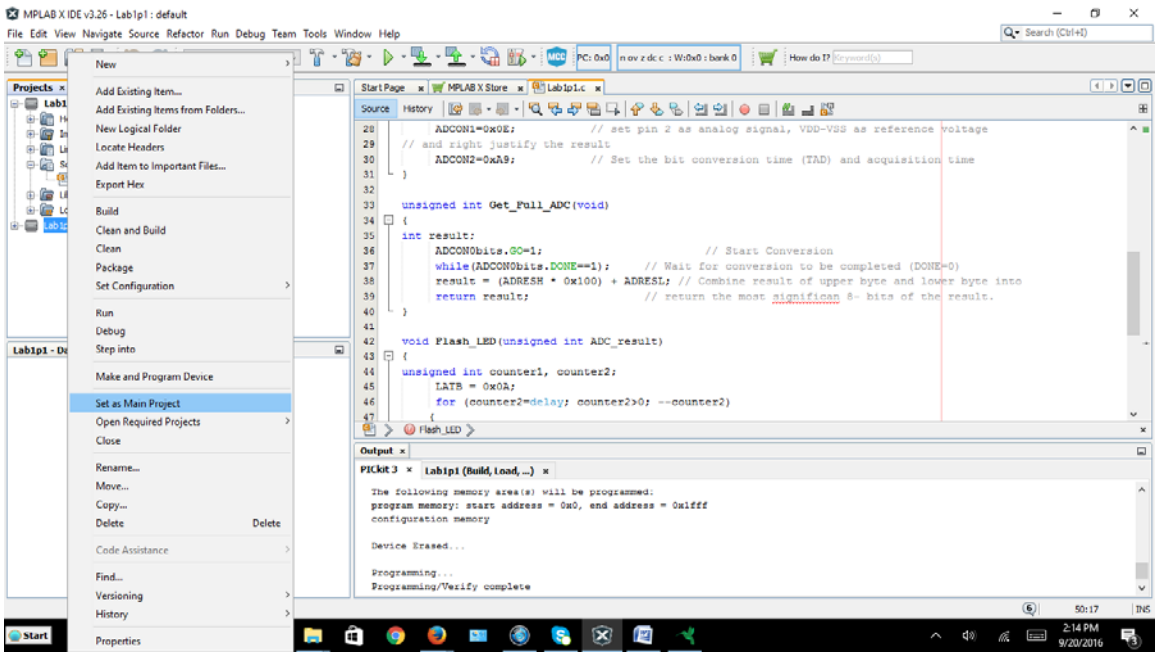
We will need to create the second part of the Lab1. We will call the project Lab1p2. Change the name of the Project Name to Lab1p2 and the Project Location should be changed from Part1 to Part2. Hit ‘Copy’ when done.



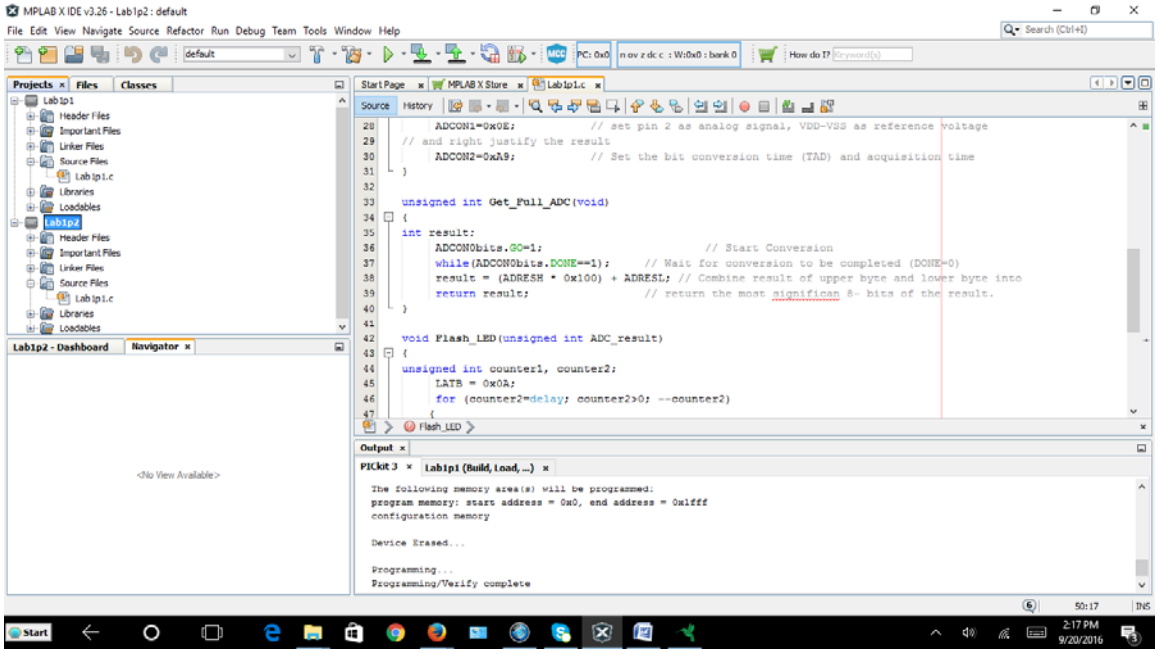
After the project is copied, a new project will appear on the left side of the screen. Go and select that new project:



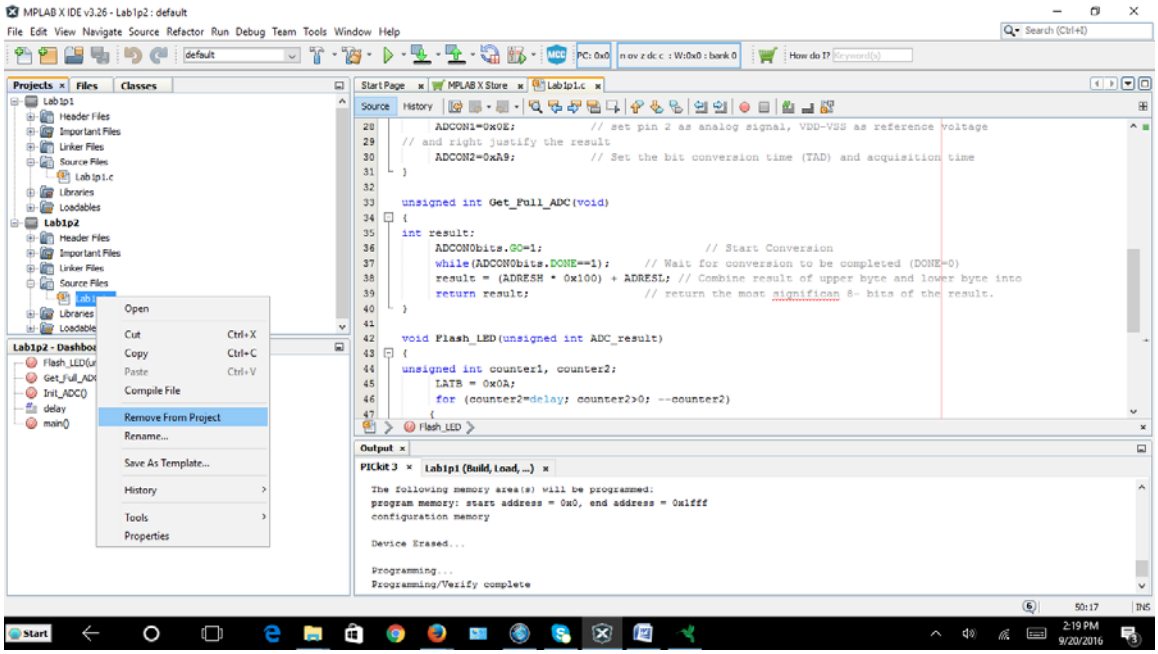
Right click the mouse and a new window will appear, Scroll down to ‘Set as Main Project’ and click on it.



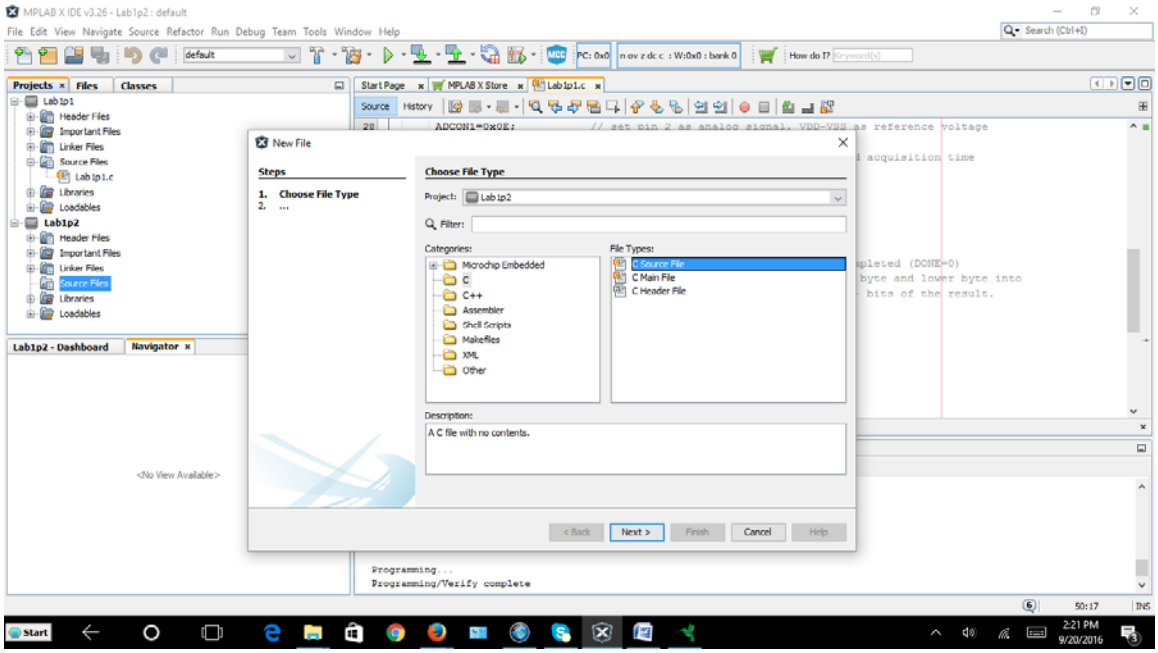
The new project will set in Bold indicating that this is the main project. Any compilation and programming will be done based on that project. Expand that project name by clicking on the '+' sign. Then expand the '+' sign of the 'Source File':



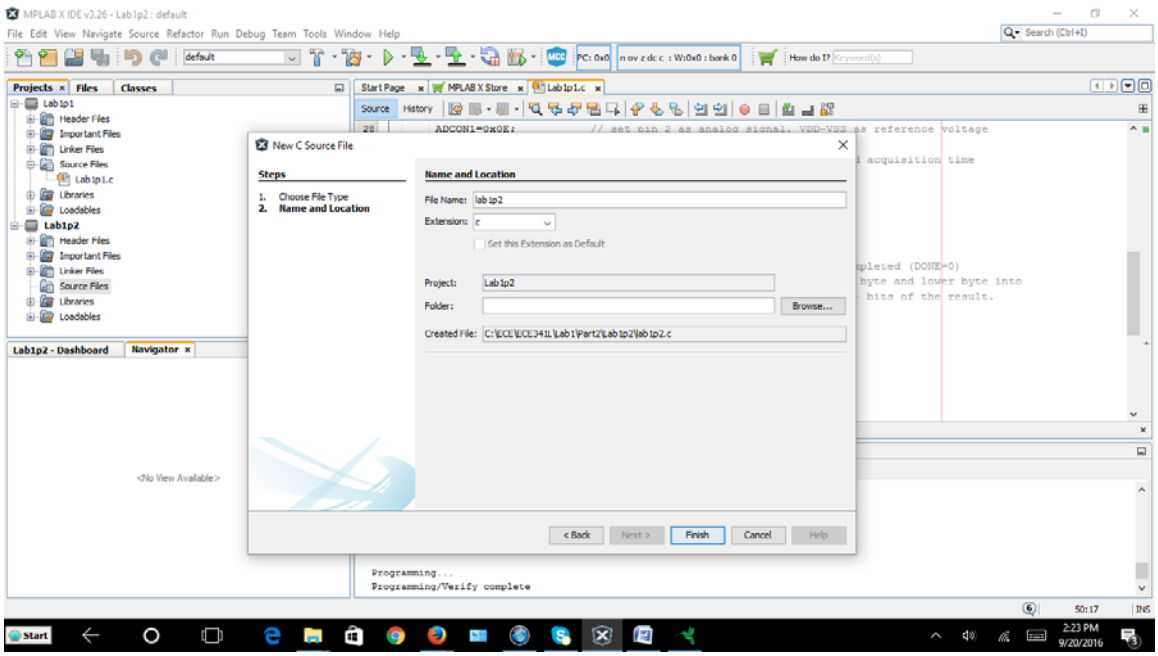
Next, go down to the 'Lab1p1.c' file under the 'Source Files' and right click on it. Scroll down to 'Remove From Project' and click on it. The 'Lab1p1' should disappear.



Next, go to File>New File like we did in the Tutorial by selecting ‘C’ option and C Source File:



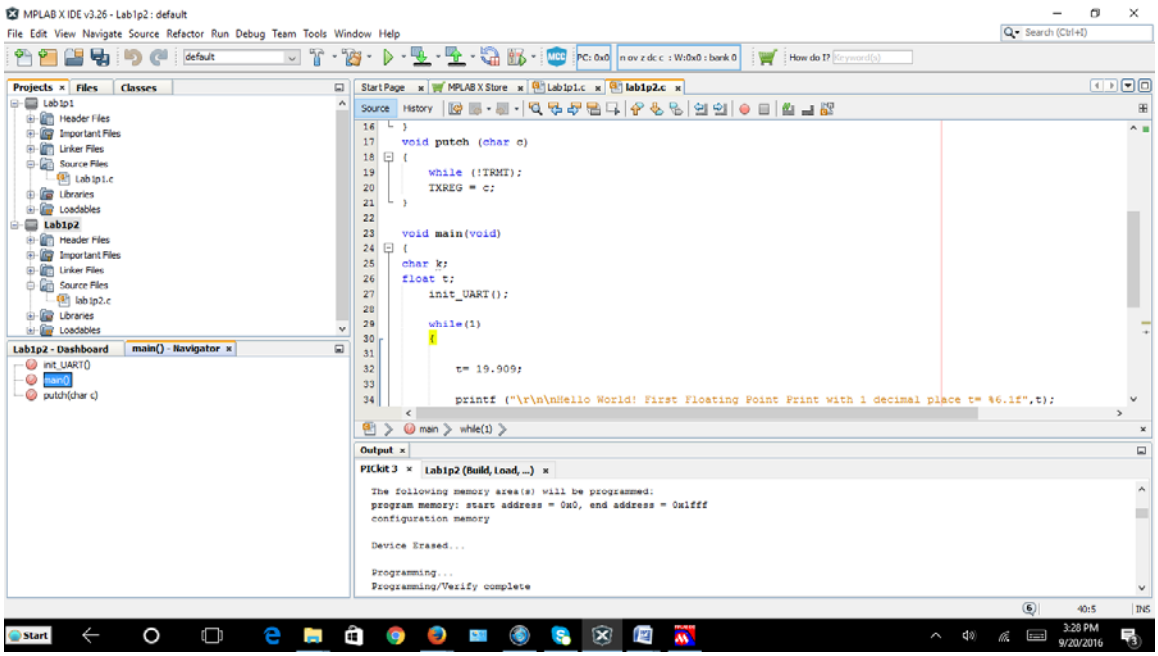
Enter ‘lab1p2’ in the ‘File Name’ field and hit Finish:



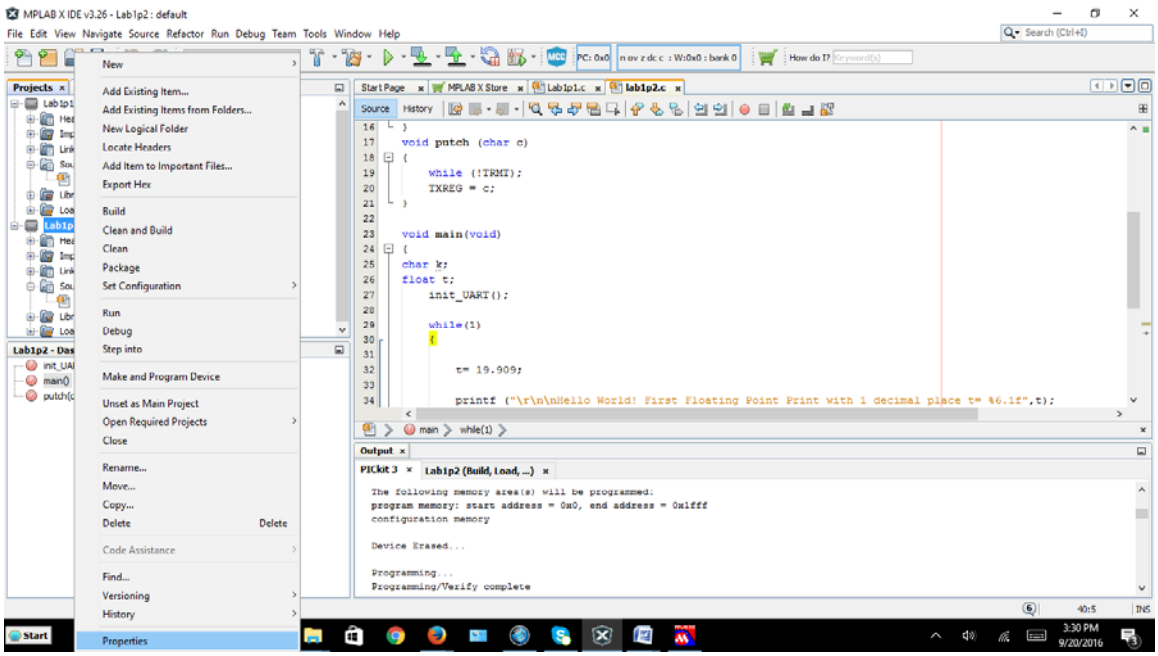
Copy and paste the following code into the blank document:

```
#include <p18f4321.h>
#include <stdio.h>
#include <math.h>
#include <usart.h>

#pragma config OSC = INTIO2
#pragma config WDT=OFF
#pragma config LVP=OFF
#pragma config BOR =OFF


void init_UART()
{
        OpenUSART (USART_TX_INT_OFF & USART_RX_INT_OFF &
USART_ASYNCH_MODE & USART_EIGHT_BIT & USART_CONT_RX &
USART_BRGH_HIGH, 25);
        OSCCON = 0x60;
}

void putch (char c)
{
   while (!TRMT);
   TXREG = c;
}

void main(void)
{
char k;
float t;
   init_UART();

   while(1)
   {

        t= 19.909;

        printf ("\r\n\nHello World! First Floating Point Print with 1 decimal place t= %6.1f",t);
        printf ("\r\nHello World! First Floating Point print with 2 decimal places t= %6.2f",t);

    }
}
```

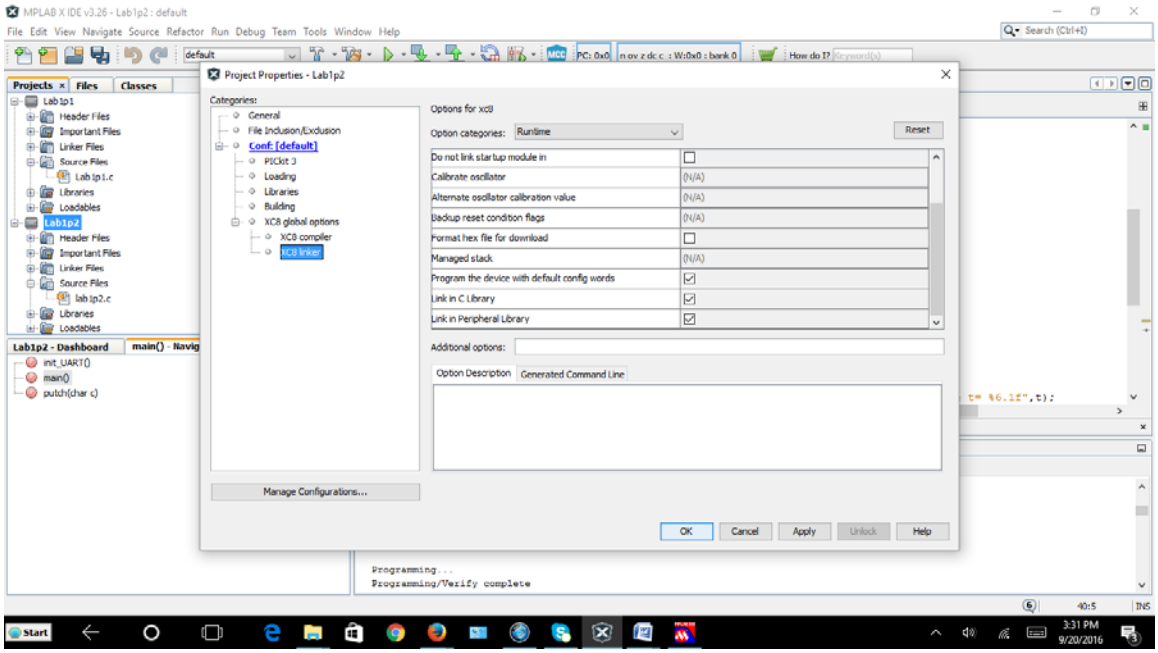
Save the file.



Before we compile the new piece of software, we need to setup a compilation option. Select the 'Lab1p2' project and right click on it. Scroll all the way down to Properties and click on it.



A new screen will appear. Select ‘XC8 linker’ and then go the right side and scroll until you see the selection ‘Link in Peripheral Library’. Check on that option. Hit OK afterwards.



We are now ready to compile.

Compile the above program and run it. This program does not have any error and it should be compiled cleanly. Program the prototype board with the new software.

Open the ‘TeraTerm’ software (see Syllabus) with the serial port detected (choose the one with USB Serial Port) and go to the ‘Setup’ tab and under ‘Serial Port’ make sure that you have the following:

Baud rate:	9600
Data:	8 bit
Parity:	none
Stop:	1 bit
Flow control:	none

If you have properly setup the port, you should see a continuous stream of data displayed on the screen when you run the program above. This indicates that you have established a good serial connection. If you have difficulties, you do need to get hold of me in class.

I will go over in class the meaning of this test code and the reason that you will need to use it as an example for a tool in debugging your code.

Part 3)

Repeat the steps in part 2 to create a new part 3. Make sure to have the sub-folder ‘Part3’ under the ‘Lab1’ sub-folder’.

```
#include <p18f4321.h>
#include <stdio.h>
#include <math.h>
#include <usart.h>

#pragma config OSC = INTIO2
#pragma config WDT=OFF
#pragma config LVP=OFF
#pragma config BOR =OFF

int ADC_Result;
float Volt;

void init_UART()
{
        OpenUSART (USART_TX_INT_OFF & USART_RX_INT_OFF &
USART_ASYNCH_MODE & USART_EIGHT_BIT & USART_CONT_RX &
USART_BRGH_HIGH, 25);
        OSCCON = 0x60;
}

void putch (char c)
{
   while (!TRMT);
   TXREG = c;
}

// Prototype Area
void Init_ADC(void);
unsigned  int Get_Full_ADC(void);

void Init_ADC(void)
{
    ADCON0=0x01;            // select channel AN0, and turn on the ADDC subsystem
    ADCON1=0x0a;            // set pins 2,3,4,5 & 7 as analog signal, VDD-VSS as ref voltage
// and right justify the result
    ADCON2=0xA9;            // Set the bit conversion time (TAD) and acquisition time
}

unsigned int Get_Full_ADC(void)
{
int result;
    ADCON0bits.GO=1;                // Start Conversion
    while(ADCON0bits.DONE==1);      // Wait for conversion to be completed (DONE=0)
    result = (ADRESH * 0x100) + ADRESL; // Combine result of upper byte and lower byte into
    return result;                  // return the most significant 8- bits of the result.
```

```
}


void main(void)
{
char k;
float t;
   init_UART();
   Init_ADC();

   while(1)
   {
         ADCON0=0x01;
         ADC_Result = Get_Full_ADC();
         Volt = ADC_Result / 1024.0 * 5.0;
         printf ("Volt at AN0 is %f\r\n", Volt);

         ADCON0=0x05;
         ADC_Result = Get_Full_ADC();
         Volt = ADC_Result / 1024.0 * 5.0;
         printf ("Volt at AN1 is %f\r\n", Volt);

         ADCON0=0x09;
         ADC_Result = Get_Full_ADC();
         Volt = ADC_Result / 1024.0 * 5.0;
         printf ("Volt at AN2 is %f\r\n", Volt);

         ADCON0=0x0d;
         ADC_Result = Get_Full_ADC();
         Volt = ADC_Result / 1024.0 * 5.0;
         printf ("Volt at AN3 is %f\r\n", Volt);

         ADCON0=0x11;
         ADC_Result = Get_Full_ADC();
         Volt = ADC_Result / 1024.0 * 5.0;
         printf ("Volt at AN4 is %f\r\n\n", Volt);

    }
}
```