

**CALIFORNIA STATE POLYTECHNIC UNIVERSITY, POMONA
COLLEGE OF ENGINEERING**

**ECE 3301L Spring 2024 Microcontroller Lab
Session 1**

Felix Pinai

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) Running the tutorial

You need to get the exercise given on the Tutorial done (see attached '**MPLABX and PICKITx_Tutorial_2024.pdf**'). Build the circuit given to you on the 'LAB1_schematics.pdf' file. The box within the dotted line represents the development board that I have provided to your team. Connect only the wires that go from that box to the outside.

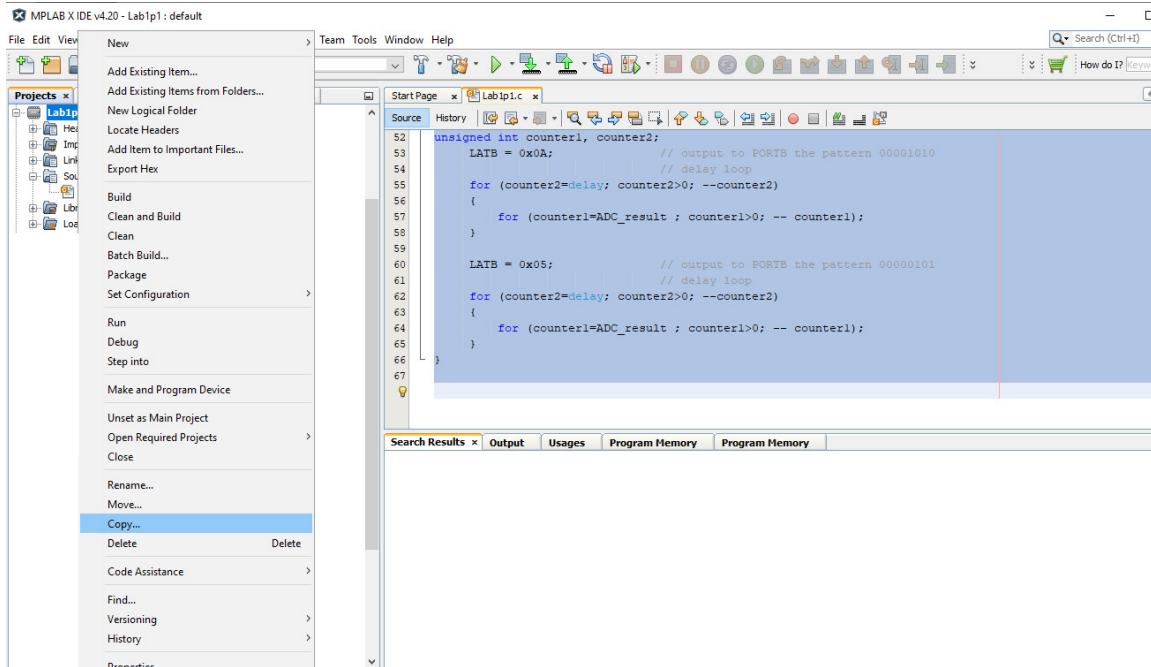
Run the program provided in the tutorial. If your program runs properly, the four LEDs will blink at different rate depending on the voltage V_Var supplied by the potentiometer to the pin RA0. The lower the voltage, the faster the LEDs will blink. When the voltage is increased, the blinking will slow down.

Part 2) Serial Port communications

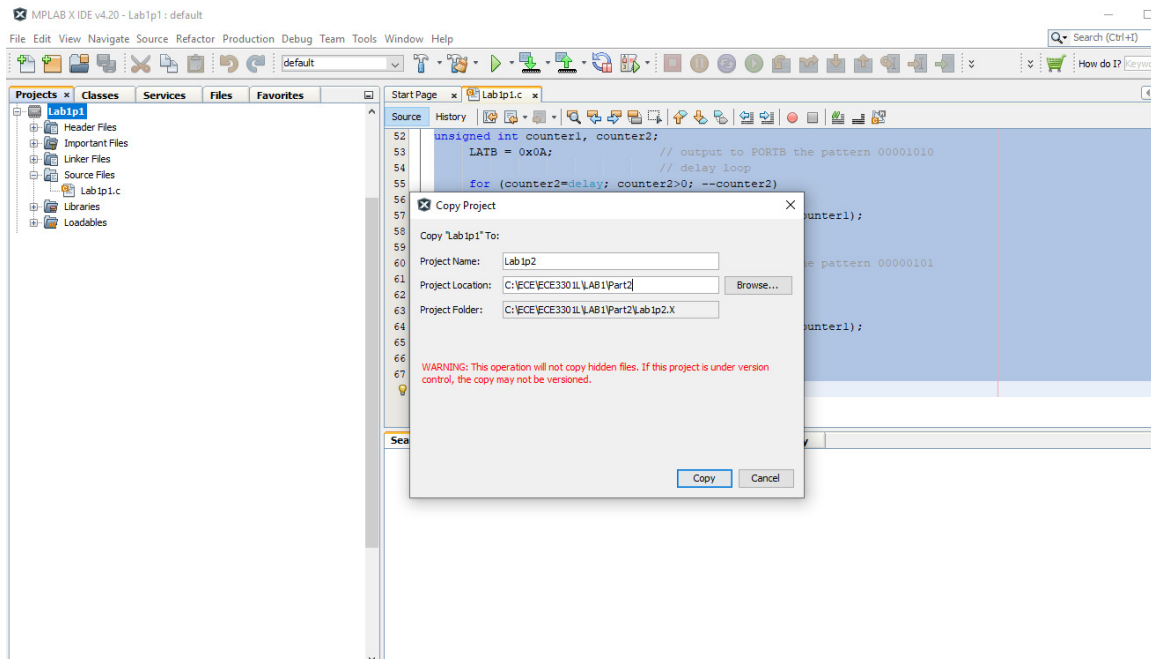
This section deals with the use of the serial port on the PIC development board. It has a UART-to-USB translator chip designed to connect a serial port to an USB port available on your laptop. To be able to access to that USB port, you may have to perform Step 4) of the syllabus.

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:

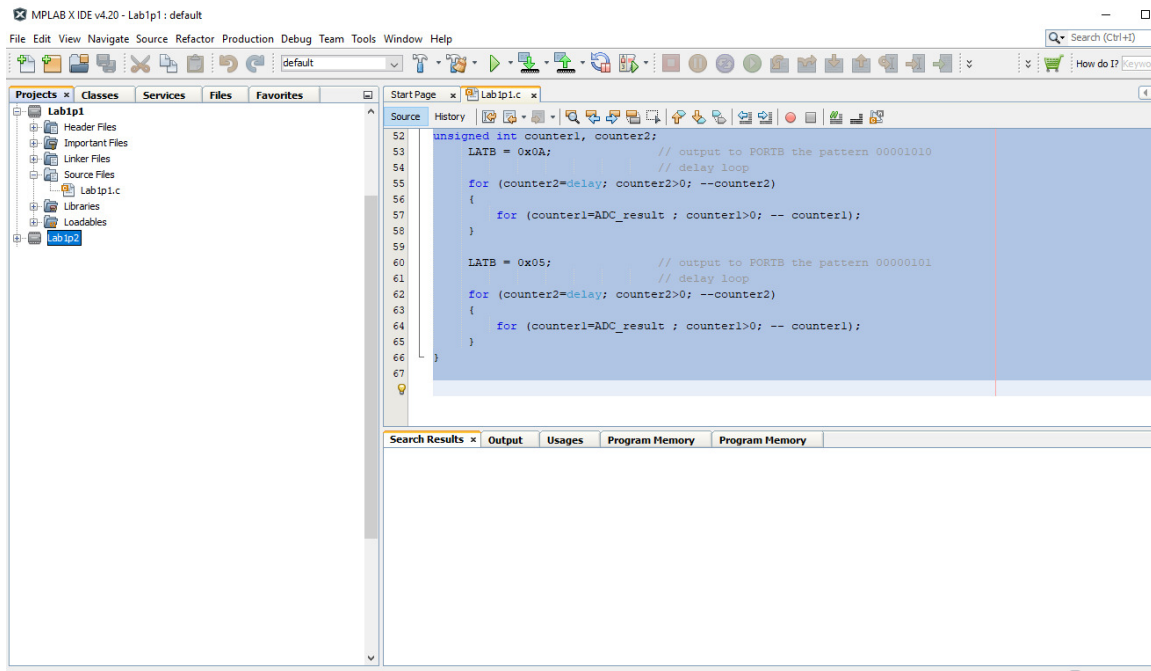
From the Lab1 Part1 project that you have done in the Tutorial, click the project name 'Labp1' and **right click the mouse**. Make sure that the label 'Lab1p1' is in bold mode. Move the cursor down to 'Copy' field and click on it.



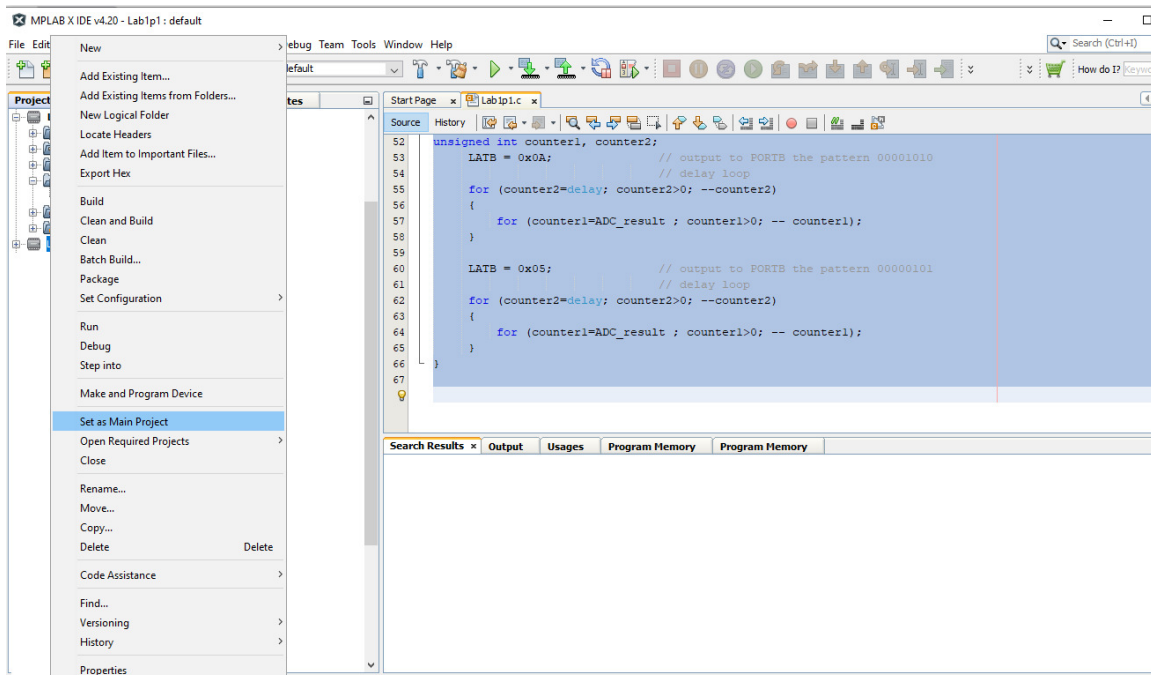
We will need to create the second part of the Lab1. We will call the project Lab1p2. Change the name of the Project Name to Lab1p2 and the Project Location should be changed from the subfolder **Part1** to a new subfolder **Part2**. Hit 'Copy' when done.



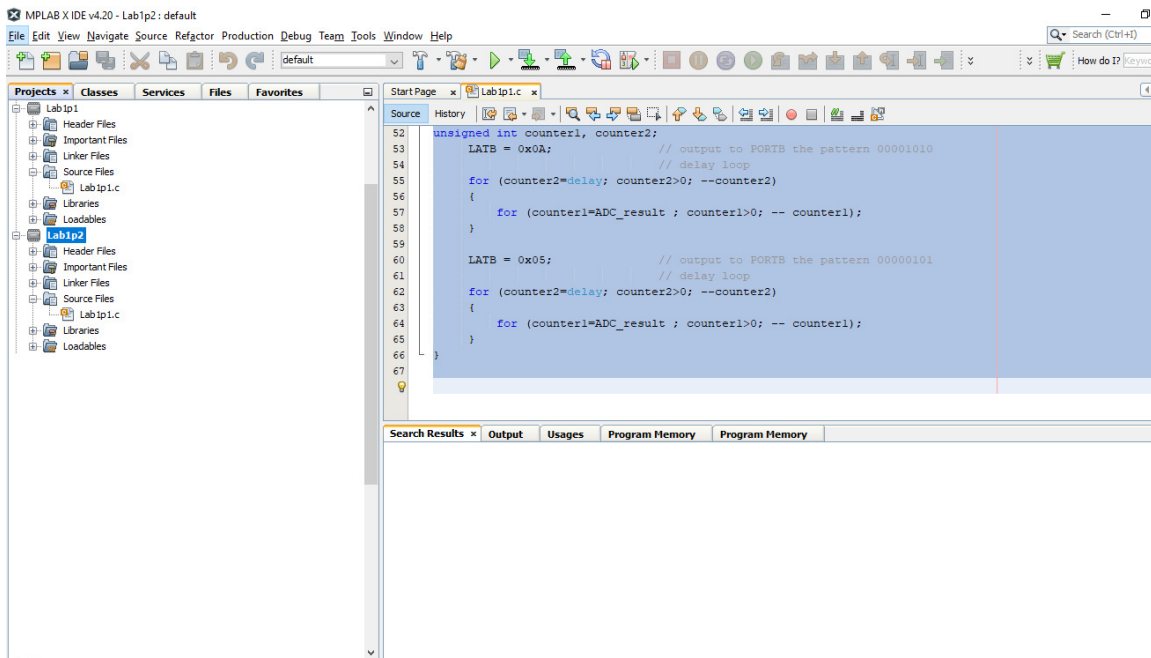
After the project is copied, a new project will appear on the left side of the screen. Go and select that new project:



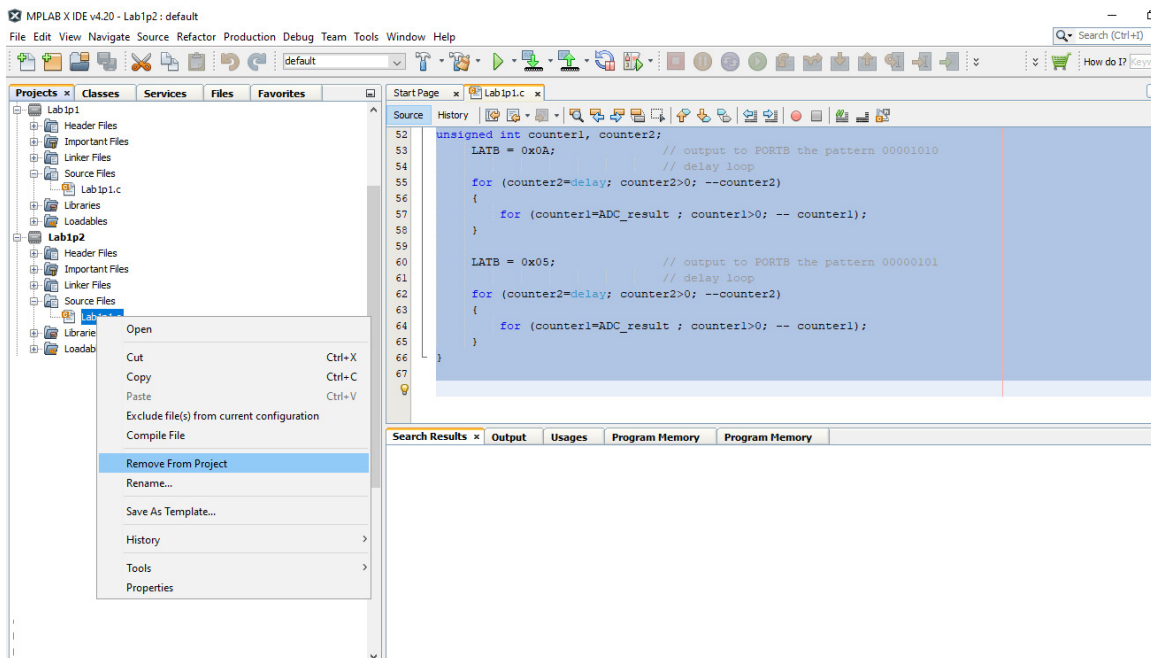
Right click the mouse and a new window will appear, Scroll down to ‘Set as Main Project’ and click on it.



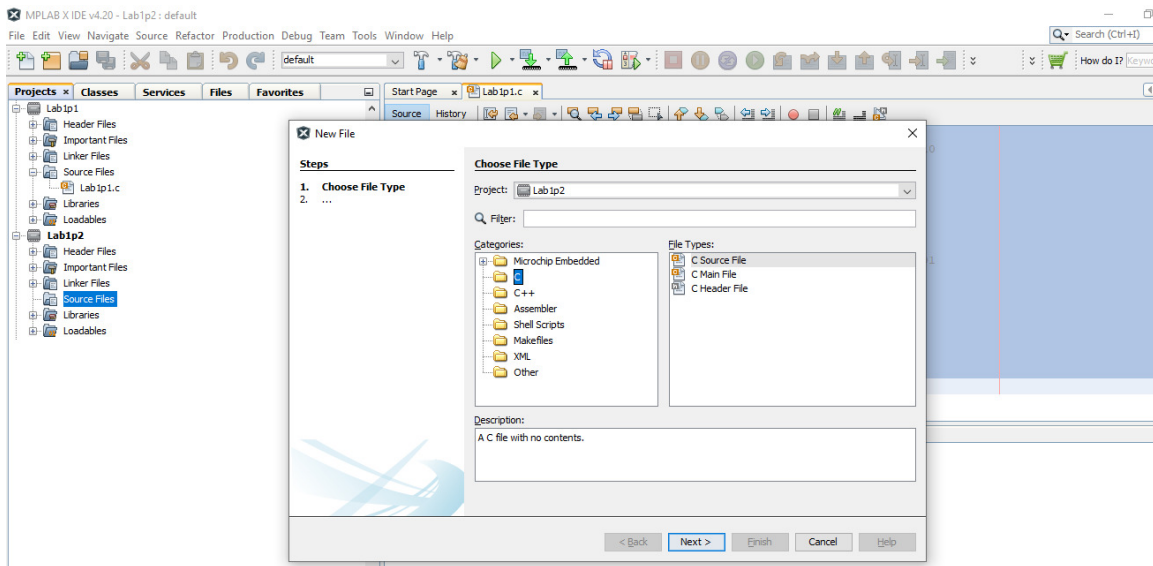
The new project will be set in Bold indicating that this is the main project. Any compilation and programming will be done based on that project. Expand that project name by clicking on the ‘+’ sign. Then expand the ‘+’ sign of the ‘Source File’:



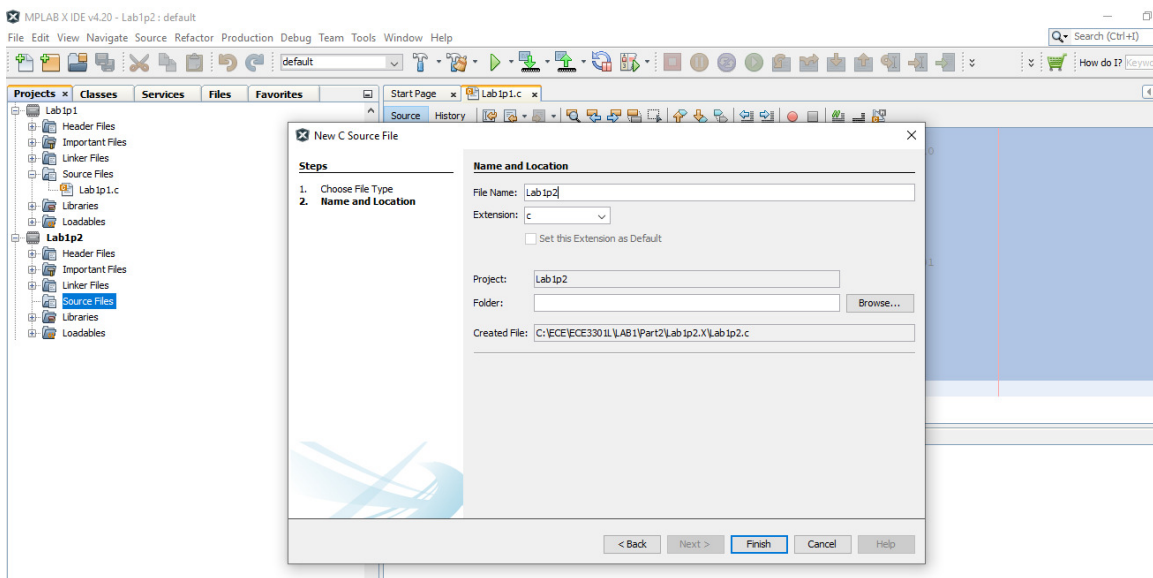
Next, go down to the ‘Lab1p1.c’ file under the ‘Source Files’ and right click on it. Scroll down to ‘Remove From Project’ and click on it. The ‘Lab1p1’ should disappear.



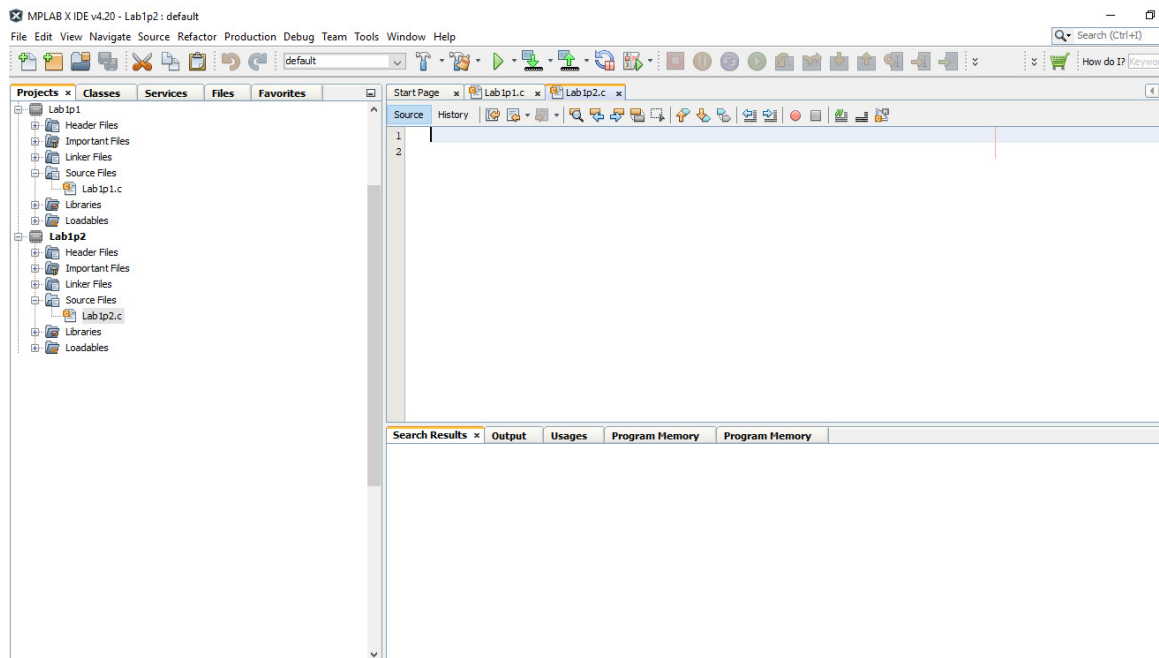
Next, go to File>New File like we did in the Tutorial by selecting ‘C’ option and C Source File. Hit ‘Next’



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



Here is the new screen. Notice the name 'Lab1p2.c' under the 'Source Files'.



Copy and paste the following code into the blank document:

```
#include <stdio.h>
#include <stdlib.h>
#include <xc.h>
#include <math.h>
#include <p18f4620.h>

#pragma config OSC = INTIO67
#pragma config WDT = OFF
#pragma config LVP = OFF
#pragma config BOREN = 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;

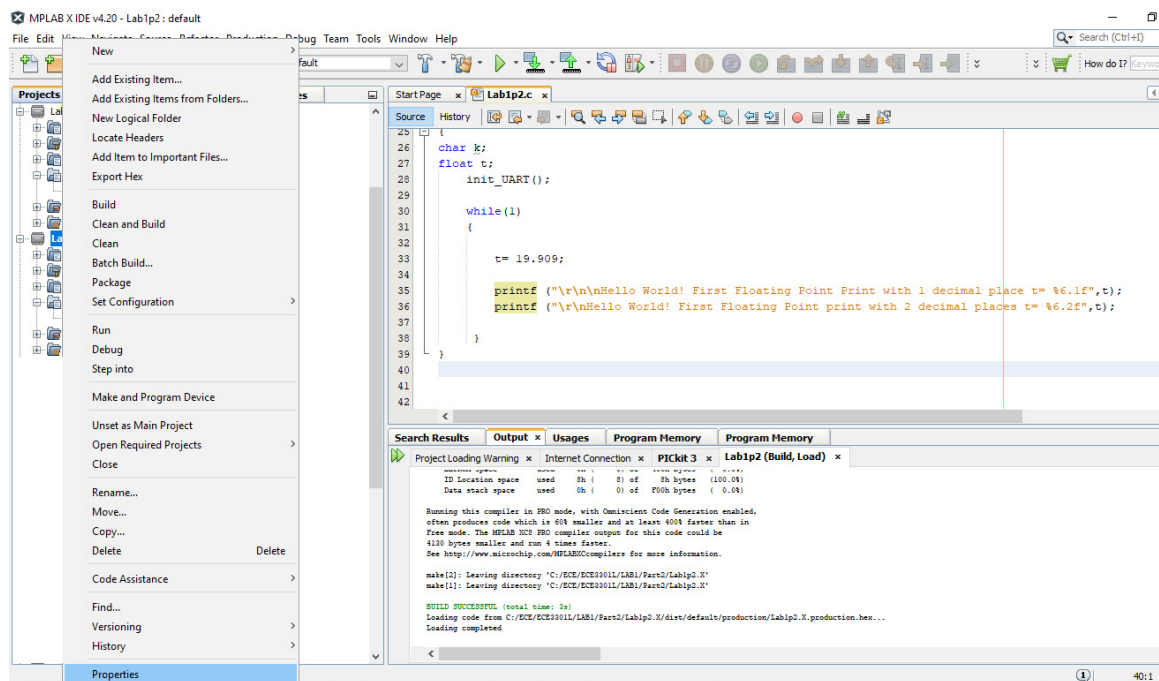
    printf ("\r\n\r\nHello World! Floating Point Print with 1 decimal place t= %6.1f",t);
    printf ("\r\nHello World! Floating Point print with 2 decimal places t= %6.2f",t);

}
}

```

Save the file.

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



The screenshot shows the MPLAB X IDE interface with the Project Properties dialog for Lab1p2. The dialog is divided into three main sections: Categories, Configuration, and Hardware Tools. The Configuration section is active, showing settings for the PIC18F4620 device. The Hardware Tools section lists various tools, including the PICkit3 and the PIC18F4620. The Project Properties dialog is open, showing the configuration for the Lab1p2 project. The Configuration section is active, showing settings for the PIC18F4620 device. The Hardware Tools section lists various tools, including the PICkit3 and the PIC18F4620.

The screenshot shows the MPLAB X IDE v4.20 interface. The main window displays the 'Project Properties - Lab1p2' dialog box. The 'Categories' tab is active, showing a tree view of project categories. The 'Conf: [default]' category is selected. The 'Options for xc8-cc (v1.34)' section is expanded, showing various options for the xc8-cc compiler. The 'Option categories' dropdown is set to 'Runtime'. The 'Additional options' field is empty. The 'Option Description' field shows the text: 'If you select an option its description will appear here.' The 'Build' button is highlighted in the bottom right corner of the dialog.

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. The LED marked D5 on

the schematics should be blinking continuously to indicate that there is a communications with the serial port.

If the program is run with the development board, run 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.

Part 3) Testing A/D channels

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

Create the new source file 'Lab1p3.c' and copy the code provided below. The new program will make sure that the A/D channels are working.

Compile the program and run it.

When the program is running, turn the potentiometer and make sure that voltage displayed on the channels AN0, AN1, AN2, and AN4 should be the same and does vary as the potentiometer is changed. In addition, make sure that the voltage on AN3 shows a constant value of **about 4.096V**.

```
#include <stdio.h>
#include <stdlib.h>
#include <xc.h>
#include <math.h>
#include <p18f4620.h>

#pragma config OSC = INTIO67
#pragma config WDT = OFF
#pragma config LVP = OFF
#pragma config BOREN = OFF

// Prototype Area
void Init_UART(void);
void putch (char c);
void Init_ADC(void);
unsigned int Get_Full_ADC(void);
float Read_Ch_Volt(char);
```

```

void Init_UART(void)                                // This routine is to initialize the 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)                                // This routine must always go with the
                                                    // 'Init_UART()'

{
    while (!TRMT);
    TXREG = c;
}

void Init_ADC(void)
{
    ADCON0=0x01;          // select channel AN0, and turn on the ADC subsystem
    ADCON1=0x0a;          // set pins 2,3,4,5 & 7 as analog signal, VDD-VSS as ref voltage
    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.
}

float Read_Ch_Volt(char ch_num)
{
    ADCON0 = ch_num * 0x4 + 1;
    int ADC_Result = Get_Full_ADC();
    float Volt = ADC_Result / 1024.0 * 5.0;
    return (Volt);
}

void main(void)
{
    float Volt;
    Init_UART();
    Init_ADC();

    while(1)
    {
        printf ("Testing A/D pins... \r\n\n");

        Volt = Read_Ch_Volt( 0);
        printf ("Volt at AN0 is %f\r\n", Volt);
    }
}

```

```
Volt = Read_Ch_Volt( 1);  
printf ("Volt at AN1 is %f\r\n", Volt);  
  
Volt = Read_Ch_Volt( 2);  
printf ("Volt at AN2 is %f\r\n", Volt);  
  
Volt = Read_Ch_Volt( 4);  
printf ("Volt at AN4 is %f\r\n\r\n", Volt);  
  
Volt = Read_Ch_Volt( 3);  
printf ("Reference voltage at AN3 is %f\r\n\r\n", Volt);  
}  
}
```