<u>Driver Project: ADC Data Capture using Python programing (Group Project)</u>

Due Date: as per D2L

Assignment:

In this project, you will acquire and plot data from the Microcontroller's ADC on a computer using Python. The ADC will record voltage generated by the potentiometer on IO pin 8 (RA3) of the Microcontroller. Your project should perform the following tasks.

1. Perform Analog to Digital conversion of the analog voltage that the potentiometer induces on pin 8 (RA3) of the Microcontroller
2. Acquire and store the value of ADC's buffer register (ADCBUF) and the resultant voltage on IO pin 8 of the Microcontroller in a data frame on a computer with the help of Python programing over a 10 sec interval.
3. Plot the value of the ADC buffer and ADC's analog voltage in volts vs time over a 10 sec interval on a computer using Python programing. During this 10 sec interval, turn the potentiometer to its lowest and highest resistance value.

Additional info:

- Implement the above controller using the hardware kit and your code, which will be designed using basic ANSI C commands; Microcontroller's ADC, prior driver functions and Python programing.
- Function names: ADC function should be named uint16_t do_ADC(void) and placed in source files ADC.c and ADC.h
- Display instructions: The plots of ADC's butter and voltage vs time should be plotted on 2 separate graph windows.

Note:

Port RA2 is one of those exceptional ports that is also multiplexed to the input for an external oscillator and an analog input port. To be able to use it as a digital input with a pushbutton, it's multiplexed analog input has to be disabled by including the following line of code in your IOinit() function. However, do ensure that pin 8 (RA3) used in this project is properly configured as an analog input for the data acquisition.

```
AD1PCFG = 0xFFFF; // Turn all analog pins as digital
```

Deliverables:

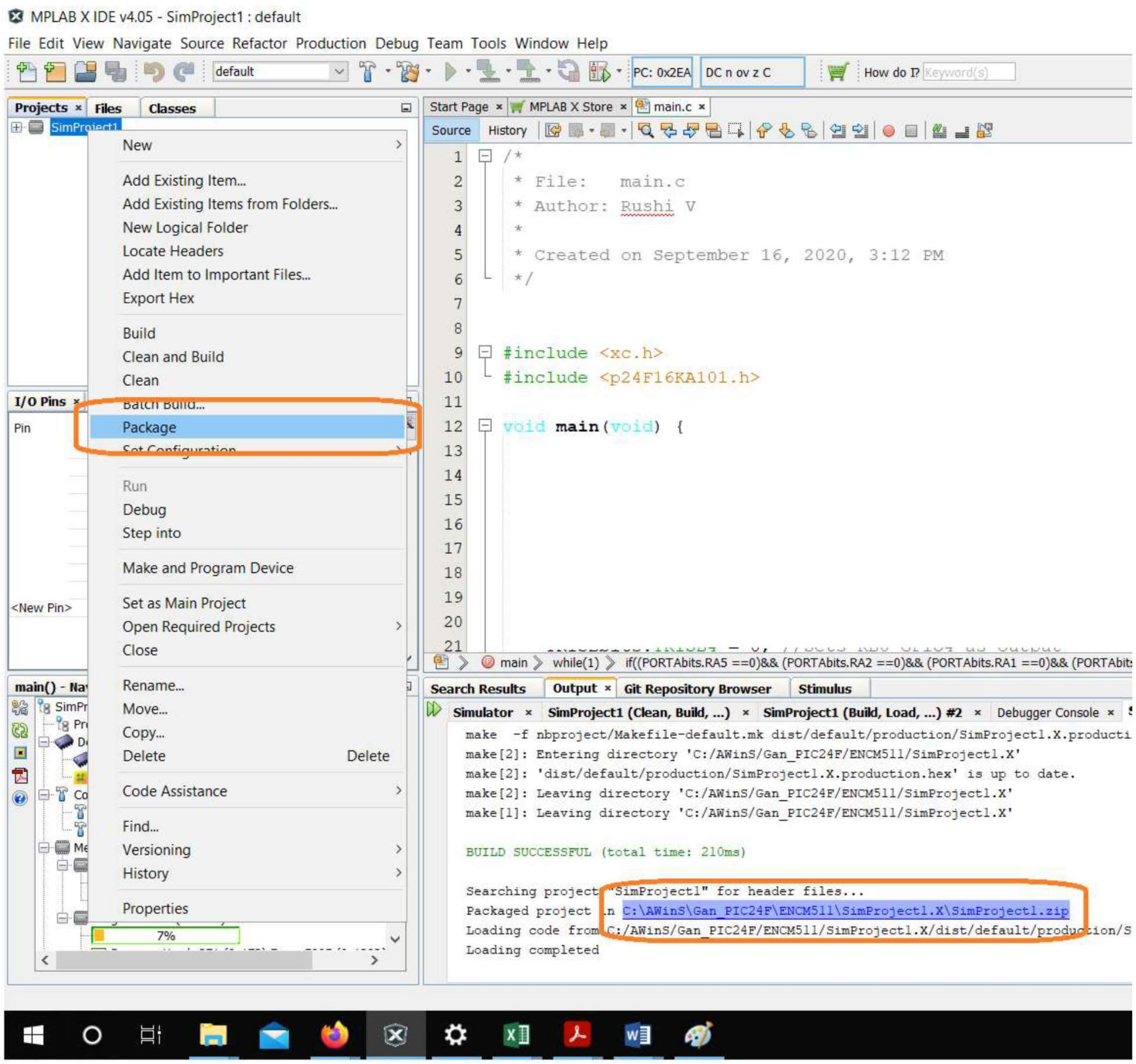
This is a group project. Each group should upload the following onto their respective group D2LDropbox folder created:

1. **Zipped up file of the MPLAB project. MPLAB projects can be zipped up by right clicking on the project and selecting package (See screenshot below). The zipped project is saved in the same project folder created by user. <u>Make sure your driver code is commented properly especially any mathematical computations used</u>.**

2. **Properly commented Python code used for Data acquisition**
3. **A CSV or Excel file containing the data frame of the acquired data. The Data Frame should have properly labeled column names and contain columns for time (sec), ADC Buffer Value and ADC input voltage value.**
4. **Link to your video demo uploaded on youtube, Vimeo or similar video hosting website along with the zipped up project. Dropbox or Google or OneDrive links are allowed as well but ensure that videos are in .mp4 or .mov format. Videos uploaded in any other format will lose points. Video demo should be as follows:**
    a. **Single recording no more than 2 mins long**
    b. **Show <u>UCID or government issued ID cards of all 3 group members</u> placed in front of the computer with MPLAB and/or hardware running**
    c. **Demo of the code and hardware operation showing the following:**
        i. **The potentiometer being turned from its minimal to its maximal resistance value a few times during the 10 sec measurement interval**
        ii. **On the computer, your Python code should generate a data frame containing the time of data capture in seconds, the digital ADC buffer value and ADC's input voltage in volts. You should display the Data frame in Python's terminal window and in its CSV or EXCEL generated using Python commands during the video demo.**
        iii. **On the computer, your Python code should generate 2 plots showing the digital ADC buffer value vs. time (sec) and ADC's input voltage (volts) vs time(sec).**
    d. **Explanation of the code organization in MPLAB and Python including any special power or time saving features (i.e. interrupts, clock switching, sleep/idle) used, and respective contribution of each group member towards code development and hardware/software testing.**

Grading rubric: (Total = 10 points)

- Correct setup and use of timers, IOs, UART, ADCs, interrupts and clock modules = 1
- Correct Python code showing correct data frame and plot generation = 6 (2 pts for data frame and 4 pts for 2 plots)
- Proper video and code upload format including commenting of all lines of code = 2
- Group participation = 1



```
/*
 * File:   main.c
 * Author: Rushi V
 *
 * Created on September 16, 2020, 3:12 PM
 */


#include <xc.h>
#include <p24F16KA101.h>

void main(void) {
```

```
main > while(1) > if((PORTAbits.RA5 ==0)&& (PORTAbits.RA2 ==0)&& (PORTAbits.RA1 ==0)&& (PORTAbit
```

```
make  -f nbproject/Makefile-default.mk dist/default/production/SimProject1.X.producti
make[2]: Entering directory 'C:/AWinS/Gan_PIC24F/ENCM511/SimProject1.X'
make[2]: 'dist/default/production/SimProject1.X.production.hex' is up to date.
make[2]: Leaving directory 'C:/AWinS/Gan_PIC24F/ENCM511/SimProject1.X'
make[1]: Leaving directory 'C:/AWinS/Gan_PIC24F/ENCM511/SimProject1.X'

BUILD SUCCESSFUL (total time: 210ms)

Searching project "SimProject1" for header files...
Packaged project in C:\AWinS\Gan_PIC24F\ENCM511\SimProject1.X\SimProject1.zip
Loading code from C:/AWinS/Gan_PIC24F/ENCM511/SimProject1.X/dist/default/production/S
Loading completed
```