App Project: LED Controller (Group Project)

Due Date: As per D2L Dropbox

Assignment: Using the Microcontroller and the driver functions developed so far, you will design a *LED Light control App* to control the LED brightness or intensity connected on pin 12 and display it on Computer using Python as shown below. The App will use the push buttons (PBs) connected to the input ports RA2, RA4 and RB4, and LED connected to pin 12 with a 1K resistor. PB1, PB2 and PB3 represent push buttons connected to ports RA2, RA4 and RB4 respectively. The LED Light control should operate as follows. You must use different Microcontroller Peripherals and Drivers namely UART and Timer peripheral(s) for designing this app.

User input(s)	**Output(s)**
If PB1 is pressed	If PB1 is pressed once, it should put the System in **ON MODE** i.e. turn on the LED (connected to Microcontroller pin 12) at 95-100% intensity (full brightness). At this point, turning the Potentiometer should adjust the LED intensity between 95-100% and 0% (LED off) using Pulse width modulation voltage (PWM) signals supplied to the LED as described below. **If PB1 is pressed again**, it should put the System in **OFF MODE** i.e. turn off the LED and put the system in low power Idle() mode.
If PB2 is pressed	If PB2 is pressed once, it should blink the LED at 100% intensity level at 500 msec intervals. **If PB2 is pressed again**, it should stop the LED blinking.
Python Script	When Python script is run on the computer, it should for a period of 1 min: 1. Capture and Store the intensity levels (0-100%) and average PWM voltage output (Vpwm) supplied by the Microcontroller to the LED vs time (in seconds) in a single CSV or Excel file with proper indexing and column names 2. Plot intensity levels (0-100%) and average voltage output supplied by the Microcontroller to the LED (Vpwm) vs time in seconds on 2 Separate graphs 3. When run in OFF mode, the CSV/Excel file and graphs should show only 0 for intensity and average voltage

	The 2 Graphs should display: Title (Intensity in % or PWM voltage Vpwm) X and Y axes labels Pulse Frequency used (see section on PWM) Group Number Group member names The CSV or Excel files should be named as per your group name and have proper index numbers and column names
PB 3	Not used in this project

Additional info:

Implement the above controller using the hardware kit and your code, which will be designed using basic ANSI C commands; IO(CN) and Timer interrupts; and Display driver functions provided. **Use of polling, instead of interrupts, will lose points.** Note, you may use multiple timer peripherals, or the 32-bit timer configuration, at your discretion. You will need to navigate the lecture material and datasheet by yourselves, accordingly. Your driver projects including delay_ms() may need to be tweaked depending on how you implement the app. Your implementation should be as power efficient as possible i.e. clock switching and sleep/idle modes should be used wherever possible.

Function names: Students can use any convention when naming functions or organizing code. A state diagram of your design is required as part of your submission. Use microcontroller-specific register, bit names and flags (to jump between states) wherever applicable in the state diagram.

Display instructions: All displays on the PC should be using Python. Note that display functions carried out at 32 kHz (300 Baud) can affect timer delays. Your code should account for such delays when producing delays specified in the table above.

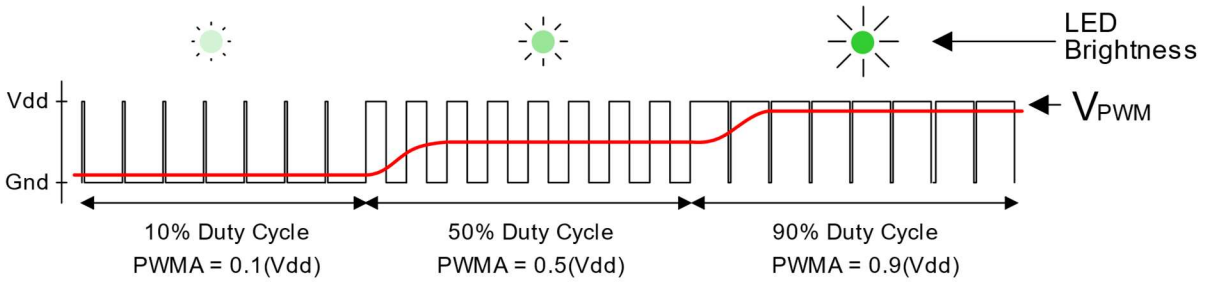
Interrupts: Interrupt ISR names are provided in the lecture slides. As specified in lecture, IO (CN interrupts) are triggered on rising and falling edges and due to any bounce effects of the push buttons that cause several high to low and low to high fluctuations at the Microcontroller input. Your code should filter out any such effects.

Pulse Width Modulated Signals (PWM):

PWM signals are commonly used in many mechatronic applications like light and motor control etc. and looks like Figure 1 and 2 below. You will program the microcontroller to generate PWM signal to come out of Pin 12 that is connected to LED via a 1K-ohm current limiting resistor. The duty cycle should be controllable based on the Potentiometer input.

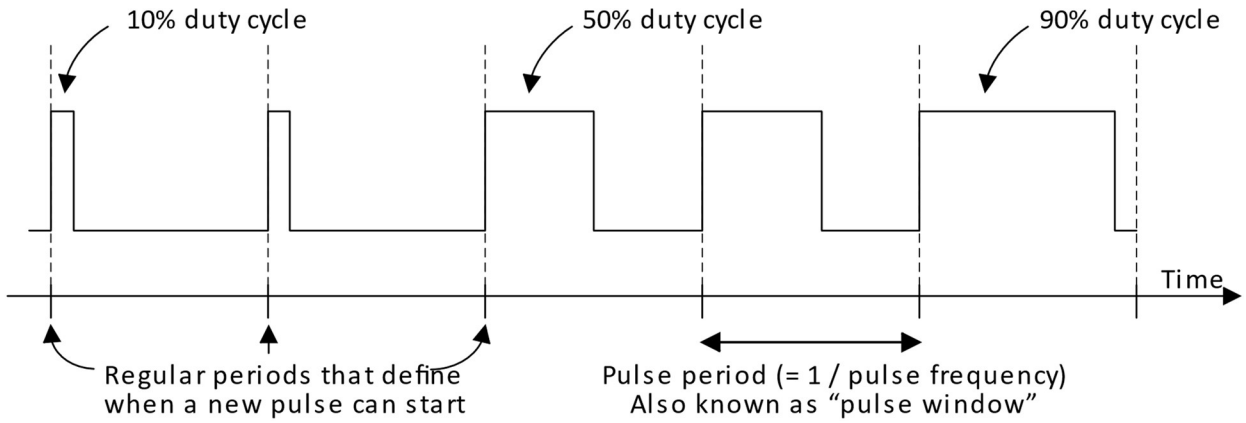
(Figure Sources: realdigital.org)

Figure 1: Relationship between PWM Duty cycle and LED brightness or intensity levels



Vpwm (red curve) = Duty Cycle/100 * VDD where
Vpwm: PWM voltage or average voltage of a PWM Signal
VDD: Voltage supplied on a Microcontroller pin

Figure 2: PWM Signal



Pulse Frequency: You can use any pulse frequency for your app. However, the pulse frequency should be high enough so that its resulting LED flicker is not visible to the human eye but at the same time not too high to consume excessive power. Use trial and error when coding your app to determine the optimum Pulse Frequency.

Deliverables:

This is a group project. Each group should upload the following onto their respective group D2L-Dropbox 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. Make sure your driver code is commented properly. **This must be submitted by due date on D2L (i.e., before the demos)**.

2. **Python Code in** .py format as a single text file.
3. **Single CSV or Excel file of the data capture showing time, intensity levels and Vpwm. File should be properly indexed and show column names for time, intensity level and Vpwm.**

4. **A single pdf document** showing the following:
    a. Names and UCIDs of all students in the group at the top of the document
    b. **A State diagram** showing the working of your code. Use microcontroller-specific register, bit names, flags and code-specific function names/labels wherever application in the state diagram.
    c. List of tasks performed by each group member

5. **Perform a live demo of your project to members of the teaching team in the lab during the lab times of the week starting November 6th.**
    a. To perform your demo, you will first download your submission from D2L in front of the assessor.
    b. You will have upto **10 minutes** to demonstrate your project, which includes the time to download and re-import your project to MPLAB X IDE to run on your microcontroller.
    c. Demo of the C and Python code and hardware operation showing all states, preferably with reference to your state machine. Each group member should be prepared to answer questions related to the working of their app project.
    d. Your group will be scheduled a time during your timetabled lab session or Tuesday lecture time.

Note: you are not allowed to share laptops/code/hardware between groups.

Grading rubric for the demo: (Total = 25 points)

- Correct setup and use of timer/IO interrupts, display functions and clock modules, power-efficiency i.e., optimal use of clock switching, interrupts and idle state; app working showing seamless operation of LED control and Python capture = 20 points
- Code upload format including commenting of all driver lines of code, plus evidence of group participation, PDF/design documentation = 2 points
- Proper Hardware kit disassembly and return = 3 points

	Fails to meet expectations	**Minimally meets expectations**	**Adequately Meets Expectations**	**Exceeds Expectations**	**Score awarded**
Peripheral Configuration and Use	None of the peripherals and states working correctly **0 to 3 points**	Some of the peripherals and states working correctly **4-5 points**	Most of the peripherals and states working correctly **6-7 points**	All peripherals and states correctly working **8 points**	
Program Logic	Does not provide evidence of appropriate program flow **0 to 3 points**	Provides evidence of attempting to use C control statements but has some errors or does not cover all scenarios **4-5 points**	Provides evidence of attempting to use C control statements to cover most but not all scenarios or has some errors **6-7 points**	Provides evidence of appropriate C control statements and implements all scenarios correctly **8 points**	
Efficiency and usability	Does not provide evidence of using power-efficient features (e.g., Idle) or usability **0 points**	Provides some limited evidence of using power-efficient features (e.g., Idle, clock switching) and usability **1-2 point**	Provides some evidence of using power-efficient features (e.g., Idle, clock switching) and usability (robustness of changing timer value, resetting the timer, etc.) **3 points**	Provides strong evidence of using power-efficient features (e.g., Idle, clock switching) and usability (robustness of changing timer value, resetting the timer, etc.) **4 points**	
Documentation/ Code Quality/	No evidence of code commenting or reasonable variable names. No state machine diagram uploaded. No evidence of group work. **0 points**	Some evidence of code comments, but infrequent or incomplete. Several states missing or incorrect. Some evidence of group work. **1 points**	Evidence that most elements of the code are commented. Some states missing or incorrectly drawn. Strong evidence of group participation. **1,5 points**	Evidence that all important elements of the code are commented meaningfully. Proper state machine showing all states and using Microcontroller and code specific register/flag/function names. Strong evidence of group participation. **2 points**	
Kit disassembly and retun	Missing parts and/or breadboard not completely disassembled 0 points			Kit fully dismantled and all parts returned **3 points**	

Total (25 points)	