App Project 1: Timer App (Group Project)

Due Date: As per D2L Dropbox

Assignment: Using the Microcontroller and the driver functions developed so far, you will design a *Timer App* to be displayed on a Computer terminal (e.g., RealTerm) as shown below. The App will use the push buttons (PBs) connected to the input ports RA2, RA4 and RB4. PB1, PB2 and PB3 represent push buttons connected to ports RA2, RA4 and RB4 respectively. The Timer App should operate as follows. You must use UART and Timer peripheral(s).

User input(s)	**Output(s)**
While PB1 is pressed	Sets the minutes to be counted to. When PB1 is pressed, the minute count increments by 1 minute from 0 to a maximum of 59. The incrementing minute count should be displayed on the PC terminal as below. While PB1 is continued to be pressed, the timer should continue to increment. Increments should occur at a pace slow enough for a user to stop at a desired minute count for e.g., 52nd minute shown below. `52m : 00s` Clears any ALARM message
While PB2 is pressed	Sets the seconds to be counted to. When PB2 is pressed, the seconds count increments by 1 sec from 0 to a maximum of 59. The incrementing second count should be displayed on the PC terminal as below. The increments should occur at a pace slow enough for a user to stop at a desired minute count for e.g., 45th second shown below. `52m : 45s` Clears any ALARM message
short presses (< 3 sec) on PB3	If the minutes and seconds are correctly set, a short PB3 press starts or pauses the Timer App count. While the Timer App is counting, the PC terminal should be updated to display the decremented minutes and/or second every second for example: `05m : 08s`

	While counting, the LED should blink at 1 sec intervals When the desired time is reached, the PC terminal should be updated to display the following: **`00m : 00s -- ALARM`** The LED should remain ON
Long press (> 3sec) on PB3	A long press on PB3 stops the count and resets the timer to 0 on the PC terminal as shown below **`00m : 00s`**
While no buttons are pressed	The program should be in a low-power mode, with no new updates on the display.

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 of PORTx or TMRx directly, 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 datasheet by yourselves, accordingly. Your implementation should be as power efficient as 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 and bit names wherever applicable in the state diagram.

Display instructions: All displays on the PC terminal window should be on a single line. 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 may need to filter out any such effects.

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 Monday Nov. 6th, 2pm (i.e., before the demos)**.

2. **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 and bit names, and code-specific function names/labels wherever application in the state diagram.
    c. List of tasks performed by each group member

3. **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 **12 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 code and hardware operation showing all states, preferably with reference to your state machine. To verify the timing of your count, have your microcontroller app running adjacent to your smart phone's count down timer app. 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-efficient and seamless operation i.e., optimal use of clock switching, interrupts and idle state; ease for the user to increment minutes and seconds = 20 points
- Code upload format including commenting of all driver lines of code, plus evidence of group participation, PDF/design documentation = 5 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. **0-1 points**	Some evidence of code comments, but infrequent or incomplete. Several states missing or incorrect. Some evidence of group work. **2 points**	Evidence that most elements of the code are commented. Some states missing or incorrectly drawn. Strong evidence of group participation. **3-4 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. **5 points**	
				Total (25):	