Driver Project 2: 16 bit Timers with Interrupts (Group Project) Due Date 17 Oct 2021

Assignment:

Using the PIC 24F16KA101 and Timer interrupts, you will design a simple IO controller to test out Timers and Interrupts. Design a state machine to turn on, turn off and blink a LED connected to port RB8 based on the push buttons (PBs) connected to the input ports RA2, RA4 and RB4 as shown in the schematic in the lecture. PB1, PB2 and PB3 represent push buttons connected to ports RA2, RA4 and RB4 respectively. The state machine should operate as follows:

User input(s)	Output(s)
While PB1 is pressed	LED blinks at approx. 1 sec intervals (1 sec on and 1 sec off)
While PB2 is pressed	LED blinks at approx. 2 sec intervals (2 sec on and 2 sec off)
While PB3 is pressed	LED blinks at approx. 3 sec intervals (3 sec on and 3 sec off)
While 2 or more PBs are pressed together	LED stays on without blinking
No PBs pressed	LED stays off

Additional info:

Implement the above controller using the hardware kit and your code, which will be designed using basic ANSI C commands and Timer interrupts. **Use of polling instead of timers and interrupts will lose points.**

IOinit() – initializes the IO ports and placed in IOs.c

IOcheck() – implements the IO checks and LED blinking functions and placed in IOs.c

Delay_ms(time_ms) – implements the delay functions used to time the LED blinks. Time_ms is the user specified time delay in milliseconds. Place timer related functions in source file TimeDelay.c

main() – Used to call IOinit() and IOcheck() and placed in main.c

Hint on generating delay cycles: Use timers and interrupts for the delay function.

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. We will revisit this multiplexing when we look at ADC converters in a couple of weeks.

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

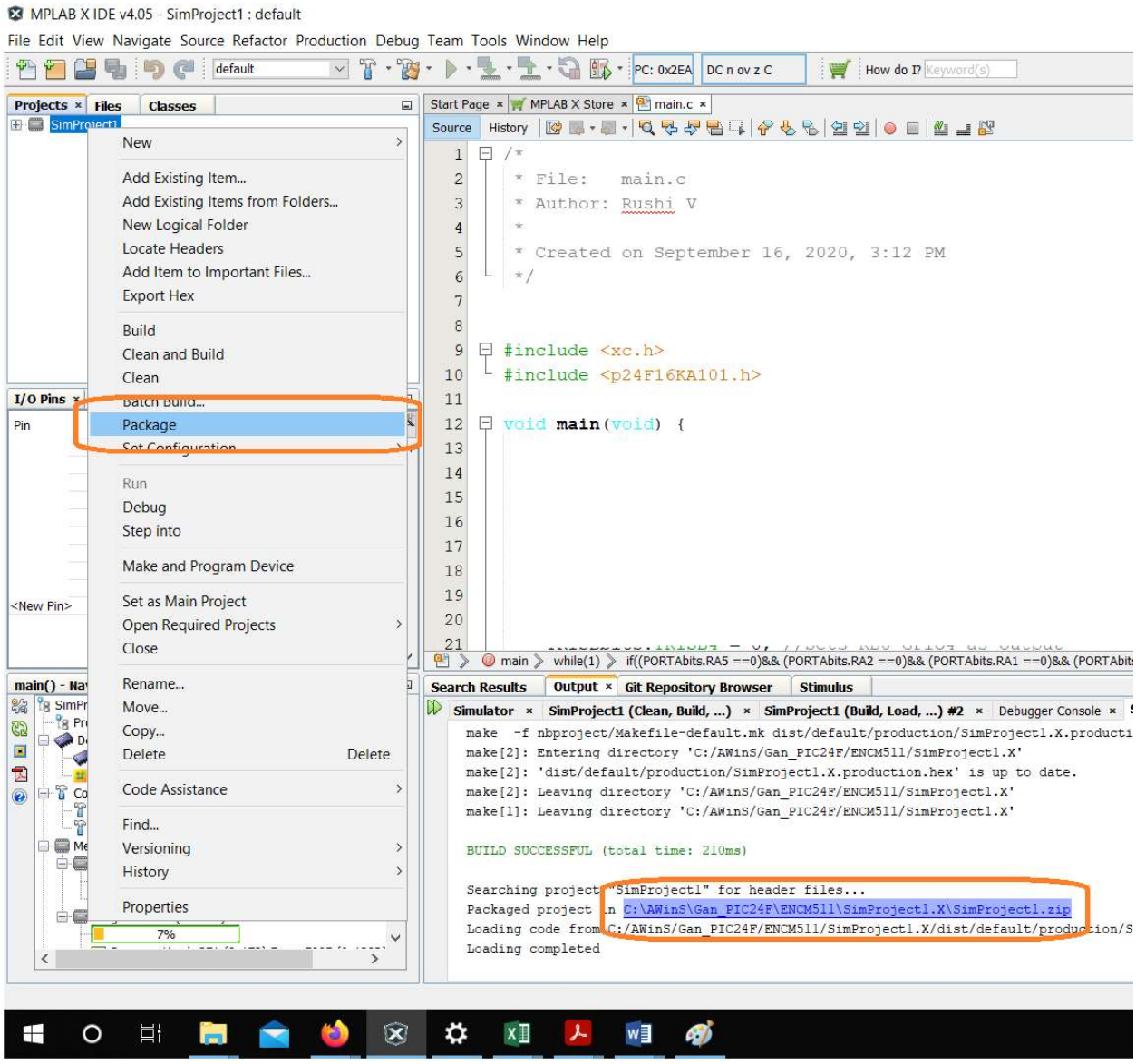
Deliverables:

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 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.**
2. **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 a single recording and show the following

a. **UCID card of one group member placed in front of the computer with MPLAB and/or hardware running**
b. **Demo of the code and hardware operation showing the following:**
    i. **Each of the PBs pressed individually**
    ii. **2 or more PBs pressed simultaneously**
    iii. **No PBs pressed**



Grading rubric: (Total = 10 points)
Correct setup and use of timers, interrupts and clock modules = 7 points
Proper video and code upload format including commenting of all driver lines of code = 2 points
Group participation = 1
Bonus 1: Submit your code using a clean structure of header files and C files that include config.h, timers.h, clocks.h, gpio.h and the associated .c files at a minimum. (1 point)
Challenge 1: When your microprocessor is in a waiting state for inputs, keep your microcontroller configured in a state that uses the minimum power. Show this value in the final video (or in the lab) using a multimeter or similar. (1 point)