Driver Project 3: 16 bit Timers with Interrupts (Group Project) – Fall 2022

Due Date: As per D2L

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 slide 'HW and IO Control.pdf'. 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 PB1 and PB2 are pressed	LED blinks for 1 msec or smallest delay that your delay function can produce. *LEDs blinking at such small time intervals are not visible to the human eye. Verify this delay by probing the RB8 pin connected to the LED with your ADALM oscilloscope.*
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 and built in __delay () functions 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. Your function is only expected to handle delays of whole numbers (e.g. 1 ms, 5 ms, 2504 ms) and not floating point numbers (e.g. 1.2 ms, 5.5 ms, 2504.8ms) . 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, interrupts and if applicable clock switching 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 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. **Each of the PBs pressed individually - showing LEDs blinking**
      ii. **2 or more PBs pressed simultaneously – showing ADALM screen shot of LED or RB8 output**
      iii. **No PBs pressed**
   d. **Explanation of the code organization in MPLAB 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.**
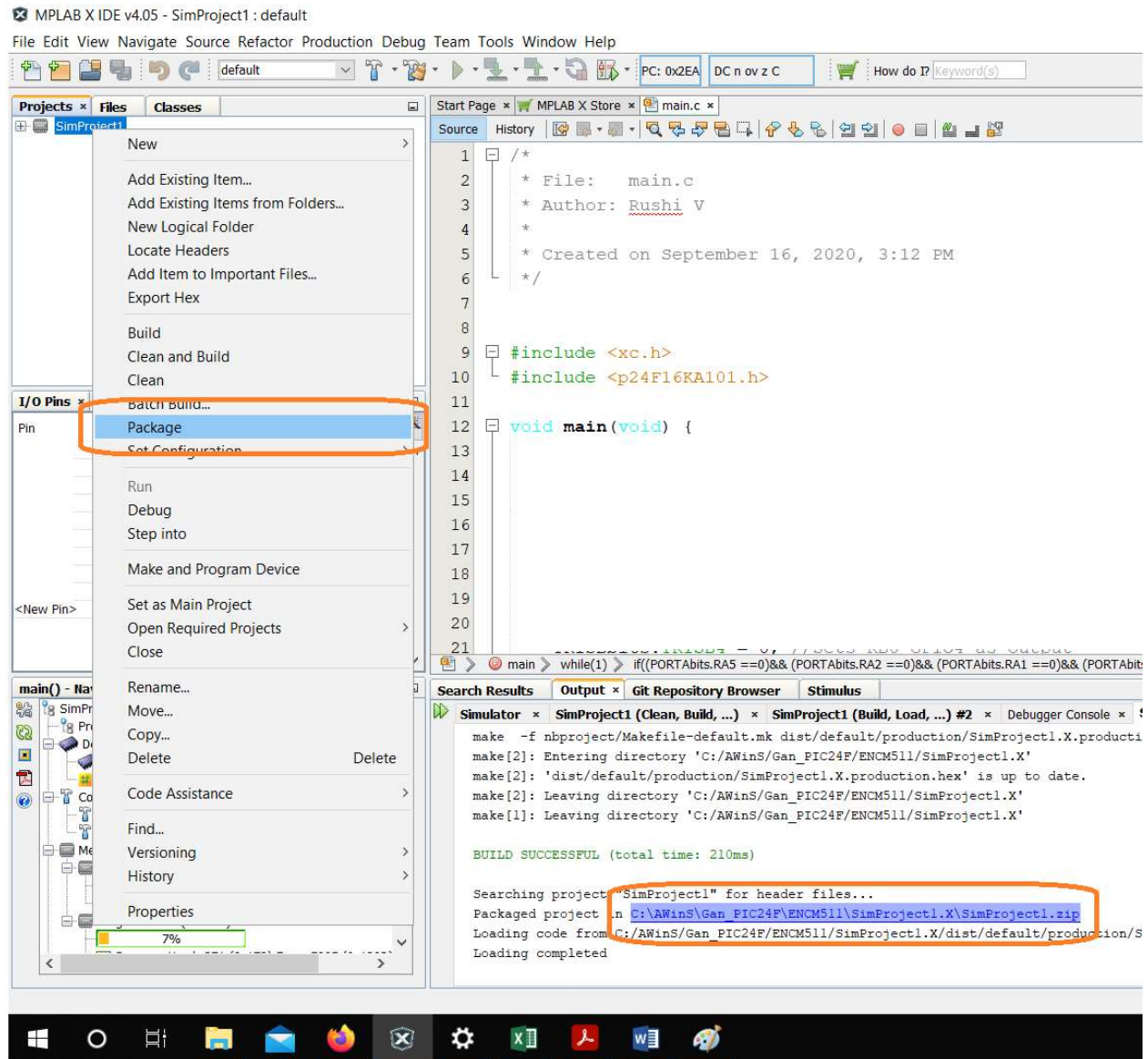
Grading rubric: (Total = 10 points)
Correct setup and use of timers, interrupts and clock modules = 5 points
Code optimization for better power savings and/or time resolution i.e. more use of clock switching, interrupts, sleep/idle modes = 2
Proper video and code upload format including commenting of all driver lines of code = 2 points
Group participation = 1