App Project 1: Count-down timer (Group Project)

Due Date: Sun 28 Nov 2021 at 11:59pm on D2L Dropbox

Assignment:

Using the Microcontroller and the driver functions developed so far, you will design a count-down timer app to be displayed on the Computer terminal as shown below. The App will use the push buttons (PBs) connected to the input ports RA2, RA4 and RB4 as shown in the schematic in the lecture slide 'Week 3 HW Control.pdf'. PB1, PB2 and PB3 represent push buttons connected to ports RA2, RB4 and RA4 respectively. The count down timer should operate as follows

User input(s)	**Output(s)**
While PB1 is pressed	Sets the minutes to be counted down from. 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. The 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 from previous countdowns
While PB2 is pressed	Sets the seconds to be counted down from. 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 from previous countdowns
short presses (< 3 sec) on PB3	If the minutes and seconds are correctly set, a short PB3 press starts or pauses the count down
Long press (> 3sec) on PB3	A long press on PB3 stops the count-down and resets the timer to 0 on the PC terminal as shown below **00m : 00s**

During countdown	The PC terminal should be updated to display the decremented minutes and/or second every second for e.g. **05m : 08s** LED should blink at 1 sec intervals
When countdown is complete	The PC terminal should be updated to display the following: **00m : 00s -- ALARM** LED stays ON

Challenge Exercise

User Input	Output
While PB1 and PB2 are pressed together >3 seconds (long press)	Switch clock mode from countdown to timer or timer to countdown mode. Clear any alarms from the alarm clock The PC terminal should be updated to display the following in timer mode: **00m : 0000s**
When PB1 is pressed	Start or stop the timer. While running the LED will be ON. The PC terminal should be updated to display the current time count in timer mode: **00m : 0000s**
When PB2 is pressed	Reset the timer back to zero, this is a valid state transition even when the timer is running. The PC terminal should be updated to display the current time count in timer mode: **00m : 0000s**
When PB3 is pressed	Reaction game. Blink the LED 3 times at 1 second blink rates and then wait a random time period after the final LED blink between 0.5 and 3 seconds. After the wait period, turn on the LED and timer at the same time.

	Leave the LED on until PB1 is pressed to stop the timer. PB2 exits reaction game. PB3 will reset the game. PB1 will do nothing after a reaction time has been recorded.

Additional info:

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

Function names: Students can use any convention when naming functions or organizing code. A state diagram 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 debounce effects of the push buttons. A debounced switch will result in several hi to lo and lo to hi fluctuations at the Microcontroller input before stabilizing to a steady and fixed voltage when the switch is pressed. Your code should 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:

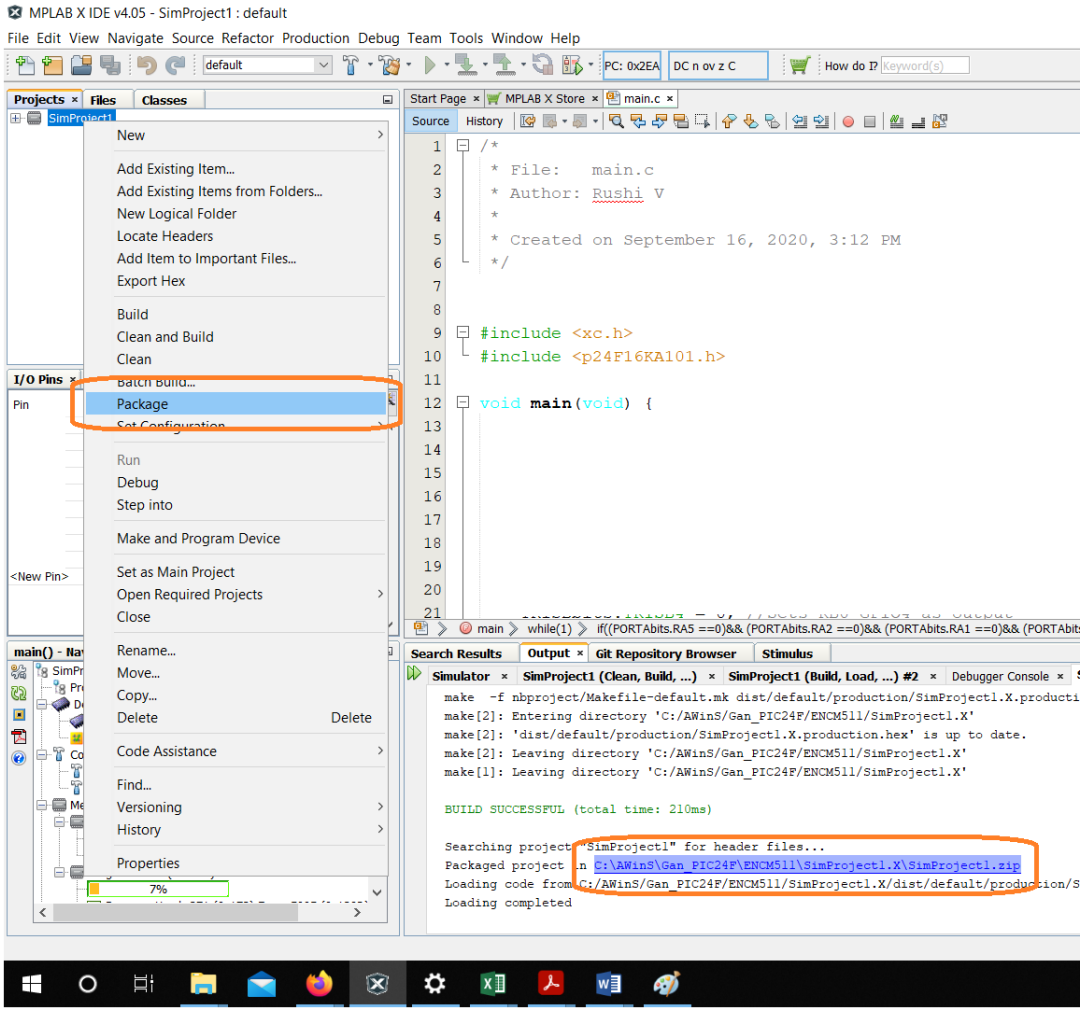
1. **Zipped up file of the MPLAB project**. MPLAB projects can be zipped up by right clicking on the project and selecting package (See screenshot below). The zipped project is saved in the same project folder created by user. Make sure your driver code is commented properly.
2. **A single pdf document** 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 wherever application in the state diagram.
    c. List of tasks performed by each group member
3. **Link to your video demo** uploaded on youtube, Vimeo or similar video hosting website along with the zipped up project. Include the link under description while uploading 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 all states. To verify the timing of your count down, have your microcontroller app running adjacent to your smart phone's count down timer app. Both should be set up to count down from the same minutes and seconds. Your submitted video should show this side by side operation.

Grading rubric: (Total = 25 points)

- Correct setup and use of timer/IO interrupts, display functions and clock modules – 3 point for each of the states above = 18 points
- 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 = 3 point
- Proper video and code upload format including commenting of all driver lines of code = 2 points
- Group participation = 2
- Challenge exercise demo = 5 bonus points (demo only, no code review required for challenge exercise)


MPLAB X IDE v4.05 - SimProject1 : default
File Edit View Navigate Source Refactor Production Debug Team Tools Window Help
Projects
Files
Classes
SimProject1
New
Add Existing Item...
Add Existing Items from Folders...
New Logical Folder
Locate Headers
Add Item to Important Files...
Export Hex
Build
Clean and Build
Clean
Package
Run
Debug
Step into
Make and Program Device
Set as Main Project
Open Required Projects
Close
Rename...
Move...
Copy...
Delete
Code Assistance
Find...
Versioning
History
Properties
I/O Pins
Pin
<New Pin>
Start Page
MPLAB X Store
main.c
Source
History
/*
* File: main.c
* Author: Rushi V
*
* Created on September 16, 2020, 3:12 PM
*/
#include <xc.h>
#include <p24F16KA101.h>
void main(void) {
Search Results
Output
Git Repository Browser
Stimulus
Simulator
SimProject1 (Clean, Build, ...)
SimProject1 (Build, Load, ...) #2
Debugger Console
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 to 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