Computer Science 230
Computer Architecture and Assembly Language
Summer 2023

Assignment 4

Due: Friday July 28, 11:55 pm by Brightspace submission
(Late submissions are **not** accepted as this is the last week of classes)

Programming environment

For this assignment you must ensure your work executes correctly on Arduino boards in ECS 249. If you have installed Microchip Studio on your own computer then you are welcome to do some of the programming work on your machine. However, please plan to spend a significant amount of time in the lab.

Individual work

This assignment is to be completed by each individual student (i.e., no group work). Naturally you will want to discuss aspects of the problem with fellow students, and such discussion is encouraged. **However, sharing of code fragments is strictly forbidden without the express written permission of the course instructor.** If you are still unsure regarding what is permitted or have other questions about what constitutes appropriate collaboration, please contact me as soon as possible. (Code-similarity analysis tools will be used to examine submitted work.) The URLs of significant code fragments you have found and used in your solution must be cited in comments just before where such code has been used.

Objectives of this assignment

- Use AVR timers in C
- Use LCD Display in C
- Use Buttons in C
- Gain confidence with C programming.

Using Interrupts and LCD panel in C

In this assignment you will again use two features of the Arduino: interrupts and the use of the 2x16 LCD display. The handlers required for the assignment are not particularly complex, but they must work precisely.

Mirroring the assembly language files you have used, the LCD driver in C uses similar functionality such as *lcd_init()* to initialize the LCD, *lcd_puts()* to display a string, *lcd_xy(x,y)* to position the cursor at (x,y), *lcd_blank(len)* blanks "len" number of characters, *lcd_putchar()* to display a character etc. Please see *lcd_drv.h* file for appropriate function templates and main.c shows how to use the LCD functions.

Likewise, *timer_interrupt.c* is an example file that shows of how to use timers and *button.c* is a C example to use buttons on the LCD shield.

The timer interrupts can be used for example as:

```
// timer1 overflow
ISR(TIMER1_OVF_vect) {
   // process the timer1 overflow here
}
```

You can reuse the code from *main.c , button.c* and *timer_interrupt.c* and the LCD driver files provided for this purpose. Your program should do the following:

1. Use a timer to generate interrupts. Choose the timer initial value and scaling factors appropriately.
2. The Interrupt service routine will maintain four counters, hours, minutes, seconds and sub-seconds. Sub-second counter is incremented every 1/100 second.
3. These counters are updated appropriately based on the interrupt.
4. Display the time as "hr:min:sec.sub_sec" format on Line 1 of display, two digits for each.
5. Display the same time on Line 2 of the display.
6. When a button is pressed (any button), Line 2 is paused while the time in Line 1 continues to increment.
7. When a button is pressed again (any button), the time displayed in Line 2 resumes and syncs with the same time as Line1. Line 2 acts as a stopwatch function.
8. Note that the display program takes ASCII characters for display. So if you need to display a number you need to convert that into equivalent ASCII string. For example to display the number 59, you cannot pass the value of 59 directly to the LCD. Instead, it should be converted to two ASCII characters, as 53 (for 5) and 57 (for 9). You can use divide (/) and modulo (%) operators and add a 48. For example use 59/10=5 and 59%10=9 to get the digits and adding a 48 to each of these numbers will convert to ASCII equivalent of 53 and 57. and 59/10=
9. **Submit your "main.c" program on Brighspace**

What you must submit

- **Submit only one main.c file on Brightspace.**

Evaluation

- 6 marks for displaying the time
- 4 marks for stopwatch function

Some of the evaluation above will also take into account whether or not submitted code is properly formatted (i.e., indenting and commenting are suitably used), and the file correctly named. Please appropriately comment your code so that the evaluator can understand your design.