Reliability and Responsiveness

CS122A Intermediate Embedded Systems

Prelab

You should have your board wired with the LCD connected and three buttons (`wakeDisplay`, `increment`, `decrement`). For this lab, you will be using the scheduler code from CS120B (and earlier in the quarter). You will want to update the task struct to contain an `unsigned char active` member which will be a 0x01 when the SM is active (by default) and 0x00 when inactive (waiting to be triggered). When initializing each task, don't forget to initialize this variable to 0x01 as well. The scheduler `for` loop will now first check if it is active, then check the `elapsedTime` vs. the `period`. If the task is inactive it shouldn't execute.

Introduction

You will be creating a system that will increment or decrement a counter (starting at 3) when the corresponding buttons are pressed (they should not continue to increment/decrement when the button is held down). If both buttons are pressed it should reset the count to 0. The counter should not go below 0 or above 9 (this is similar to a previous lab, but we have provided the code you will be using). The value of this counter should be displayed on the LCD screen.

Triggered SMs

When you make an SM triggered by either another SM or an interrupt, the triggering SM will need a pointer to the SM that it will trigger. Additionally, if the triggered SM will need to make itself inactive, it will also need a pointer to itself to be able to update it's `active` flag.

Exercises

1) The LCD screen will only display for 3 seconds and then will go to sleep (the display clears and the backlight turns off). Pressing the `wakeDisplay` button will wake it up for an additional 3 seconds. In this exercise, you will create a polling SM that will check the `wakeDisplay` button every 50 ms to see if it has been pushed. The LCD screen stays awake for 3 seconds from the *last* button press.
2) You will now update the polling SM to be an interrupt based on the wakeDisplay button being pressed. You will want to review Ch. 11. External Interrupts in the datasheet (p. 65) on how to use pin change interrupts.
   Note: PCINTs will trigger on any pin change. You only want to wake on the button being pressed, not released. I.e., if someone held the button down for 3 seconds, you don't want it to wake up again when the button is then released.
   Note: You have not been instructed on where to put the wakeDisplay button so you will need to make sure that you are setting the pin change interrupt correctly based on where it is wired.
3) There is a bug in the code for the increment/decrement SM that causes it to go into an infinite loop in a specific situation. This bug is not necessarily realistic and has been inserted intentionally. You do not need to fix the bug, but you will want to implement a

watchdog timer that will reset your system if it hits this situation. Use the datasheet (Ch. 9.3 WDT) to fill in the following functions. **Note:** The timing sequence is very precise for changing values on the WDTCSR.

```
void WDT_Init(void) {
      // Disable interrupts (critical timing section)
      // Reset watchdog (hint: use avr/wdt.h)
      // Set up WDT interrupt
      // Start watchdog with Xs prescalar
      // Re-enable global interrupts
}

ISR(WDT_vect) {
      /* Insert any code you need here */
}
```

4) (Challenge) Using the watchdog timer, you will save the current value of the counter and if your system is reset by the watchdog timer (only) you will reload the previous value. Otherwise, you will reset to the default value.

Submission

Each student must submit their source files (`.c, .h`) and test files (`.gdb`) according to instructions in the lab submission guidelines.

```
$ tar -czvf [cslogin]_lab1.tgz turnin/
```

Don't forget to commit and push to Github before you log out!