SYSC2003 Winter 2017

Assignment 5

The intent is to use as many devices and as many interrupt sources as possible while simulating an automated greenhouse management system. **This assignment cannot be done in one sitting**.

The greenhouse management system (GMS) comprises:

- A thermostat to monitor the current temperature in the greenhouse
- A fan to run if the temperature is too hot
- A heater to run if the temperature is too cold
- An automated blind that rolls itself up or down upon command.
- A water pump that distributes water at variable flow rates.
- A keypad allows the GMS operator to increase/decrease the pump rate and raise/lower the blinds.
- A LCD shows the current time (in seconds), current temperature and pump rate.

You are to write the code for the GMS according to the following specifications:

- The temperature is to remain at a (nearly) constant room temperature of degrees Celsius.
- The temperature is to be sampled at a regular rate of 2 Hz and displayed on the LCD, with the display being updated only when the temperature has changed.
- The heater and fan are to be driven so as to ensure that the temperature remains constant +/- 2 degrees Celsius.
- The blind is driven by the stepper motor which runs autonomously once started because it is driven by PWM. The time to open or close the blind is 5 seconds. Output capture must be used to capture this 5 second delay between starting and stopping the stepper motor to open/close the blind.
- The water pump is to be driven by a DC motor, with its actual pump rate (in RPS) captured by the Pulse Accumulator and displayed on the LCD, the display being updated only when the actual rate has changed.
- The keypad will use five buttons:
    - ‘1’ to increase the desired water pump rate
    - ‘2’ to decrease the desired water pump rate
    - ‘4’ to raise the blind
    - ‘5’ to lower the blind
    - ‘F’ to quit the GMS.
    - The keypad can be polled to avoid any trouble with keypad interrupts.
- The LCD display shall look as follows

```
GMS Time: xxxx s

T: xxC P: xxxRPS
```

Ideally, you should be able to develop your GMS using these requirements only. Give it a try on paper before peeking at the next page where the development is broken down into smaller steps called milestones. One strategy is to think about the variables that will connect the various tasks together.

Development Plan

1. Write an initial empty version of void initHW(). Add to this function as needed, as you progress through the steps.
2. Write the main loop first. The main loop shall take the responsibility for polling the keypad and for maintaining the display.
    - Initialize the LCD and then write the static portion of the LCD display (shown in red).
    - Write the initial values of the variables for *time*, *temperature* and *actual pump rate*.
    - Add in the polling for the keypad, starting by polling only for the Quit key.
3. Add in the (initial framework for the) periodic tasks by adding an RTI ISR.
    - Initialize the RTI appropriately.
    - In your first version of the RTIISR, simply maintain an increasing *time*
    - You should see your LCD clock increment at approximate 1 Hz
    - Important: The RTI ISR must not write to the LCD.
4. Add in the temperature monitoring.
    - The initiation of the temperature A/D conversion shall be triggered at a regular rate (i.e. by the RTI ISR)
    - The completion of the temperature A/D conversion shall be interrupt-driven. The ADC ISR must convert the digital reading into a Celsius value and save this *temperature*.
    - You should see the *temperature* update in your LCD display.
5. Add in the temperature adjustment
    - Based on the actual *temperature* reading that you get (i.e. your typical room temperature), set the *desired temperature* to 2 degrees above.
    - Your ADC ISR shall turn on the heater and/or the fan whenever this *desired temperature* have not been reached.
    - Once you get this going, you should see a practical example of hysteresis which you will have studied in electronics or chemistry.
6. Add in the blind control.
    - When either a key is pressed for raising or for lowering the blind, the stepper motor shall be started (in the appropriate direction).
    - Because the blind takes 5 seconds to complete its movement, set a 5 second delay using *Output Compare*.
    - The Output Compare delay shall be interrupt-driven, shutting off the stepper motor.
    - Further key presses to raise/lower the blind shall be ignored while the blind is in motion.
7. Finally add in the water pump.
    - The water pump shall be continuously pumping, initially at a moderately low rate.
    - The speed of the pump shall be displayed on the LCD, in RPS.
    - The speed of the pump shall be adjusted in small-but-noticeable increments when the GMS operator pushes the increase/decrease key.
    - The increase/decrease keys shall be ignored if they push the motor outside its safe operating range [minimum, maximum].

Marking Scheme (22 marks)

- Submit assign5.c and assign5vectors.c (will be compiled with common version of basicLCD and util libraries)
- The marks are doubled in the SUBMIT program to allow ½ marks)
- An assignment will receive zero if it is late OR any of the directions are not followed.

4 marks: Proper structure of the program

- Correct number of ISRs and good use of functions
- Good (efficient, minimal) definition and use of global variables to synchronize activities across program components.
- Allocation of activities to proper program components (who turns on the LCD, who reads the temperature…)

15 marks: Code Inspection

- Initialize Hardware: 7 marks (1 per device)
- Main Loop – 4 marks
  - Keypad monitoring – all 3 keys with appropriate reaction
  - LCD updates – selective access of static and dynamic portions
- RTI ISR: 3 marks
  - Time keeping, temperature conversions.
- Temperature : 3 marks
  - Interrupt driven.
  - Logic for raising/lowering temperature
  - Use of heater/fan at appropriate times, and turned off when not needed.
- Blind Control: 3 marks
  - Interrupt Driven Output Compare
  - Independent movement of Stepper Motor using PWM
- Water Pump: 3 marks
  - Interrupt Driven monitoring of speed
  - Logic for raising/lower speed
  - Enforcement of limits on speed

3 marks: Overall Coding Quality

- Good use of loops/ifs and subroutines as well as symbols to avoid hard-coded numbers
- Good documentation and comments
- Good formatting (tabs) and consistency (naming conventions of variables, use of capital letters)