EE 357 Lab 4–Stopwatch

Prof. Annavaram

DUE: 5 PM., Monday, 11/19/2012

1 Introduction

You will work in teams to write a C code for your Coldfire board to implement the functionality of a stopwatch. You will plug your microcontroller board into the Tower system along with an LCD add-on board. This project, though well-defined, will not provide a large amount of pre-written code. It will be up to your creative and design abilities to complete the project.

2 What you will learn

This lab will help you integrate the embedded coding concepts and different I/O functionality into a single project. You will need to integrate the concepts of timers, polling and GP I/O to arrive at a working solution. You may pull in code from the basic demo lab/projects that we performed earlier.

3 Background Information and Notes

Stopwatch Application

This project will implement a stopwatch application that counts upwards in increments of tenths of seconds from 00.0 to 59.9 seconds. It will provide the ability to start, stop, and reset (back to 00.0) the stopwatch. It should also implement a "lap" feature which freezes the displayed time at the instant the "lap" button is pressed while still keeping the internal watch time incrementing. When "lap" is toggled again (or "start" is pressed again) the internal watch time (which has been running) should be re-displayed and then continue as normal.

Tower Prototyping Board

This project requires more and different I/O than is present on the MCF52259 board (more than the 4 LED's, 2 push buttons, etc.). To provide the ability to prototype larger projects, we will utilize the Tower connectors. The 'elevators' on the Tower (the sides of the tower where your board plugs into) take signals from your MCU and distribute them to other boards that plug into the tower. You will develop your code using specific pins defined later in the documentation, plug your individual microcontroller board into the development board, and test/debug your program.

Other I/O

As input to your stopwatch you will use the two push-buttons on your microcontroller board (i.e. the same as those used in CF Lab 1). SW1 will serve as your start/stop button. When pressed it shall simply toggle the state unless in lap mode (more to come…). SW3 will serve as the lap/reset button. When the

stopwatch is started/running, a press of SW3 will serve as the 'lap' feature. Any subsequent press of either SW1 or SW3 will toggle the lap feature to return to running display of the current time. When the stopwatch is stopped, a press of SW3 will reset the time to 00.0.

Suggested Approach

It will likely be beneficial to keep a "state" or mode variable tracking what mode your stopwatch is in. Presses of the button will cause updates/transitions between modes and states as shown in Figure 1. What should be done in each state (whether the time should be incremented and what should be displayed should be fairly intuitive). You may also find it beneficial to keep an internal representation of the time (separate variables for each digit) and then only update the display I/O every tenth of a second (using some kind of timer interrupt).

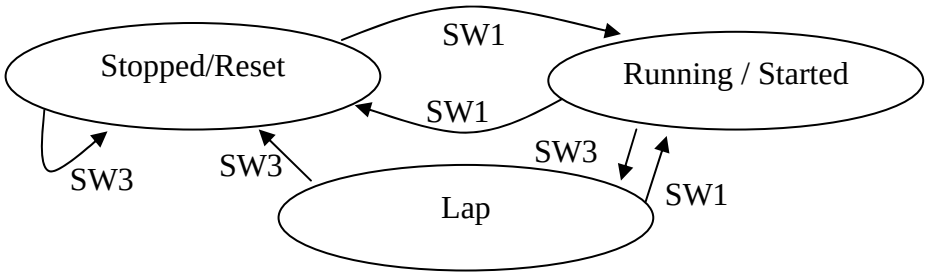

Figure 1 - State Transition Diagram for the Stopwatch

In terms of overall structure, there are many ways to organize your program. One suggested approach is to sit in a loop to poll the button inputs and update a state variable. Then inside your timer interrupt handler you can examine the state variable to determine if an update of the internal and/or displayed time is necessary.

4 Prelab

You may re-use/modify code from the GPIO, timers, and any other training labs. You will want to download and review pertinent sections of the MCF52259 Reference Manual for specific I/O control register addresses and bit definitions.

5 Procedure

1. Download the skeleton source code from Blackboard.
2. Complete the `init_gpio()` function provided to set up the appropriate output ports for the digit displays and sets their direction to 'output' as well as the input ports for the pushbuttons (SW1 and SW3).
3. Your program should meet the following **requirements**:
    a. Initialize the count to 00:0 on startup.
    b. Correctly display all times on the LCD display.
    c. Starting counting in tenths of seconds when SW1 is pressed and the timer is stopped.
    d. Stop counting in tenths of seconds when SW1 is pressed and the timer is counting.
    e. Hold the displayed time while continuing internal time updates when SW3 is pressed and the timer is counting.
    f. Update the display with the current internal time and continue counting when SW1 is pressed and the display time is being held (i.e. LAP state).
    g. Reset the time to 00:0 when SW3 is pressed and the timer is stopped or the display time is being held (the LAP state).
4. When you are satisfied with your code and it compiles, you may test it on one of the development boards we will provide in lab/class. However, without the development board you can still get an idea if the internal time is being kept correctly by setting a breakpoint in whatever handler updates the time and displays it. View the PORTTH and PORTTG register values via the debugger as well as your time variables/array. This is because the switches are on your individual board and thus all input stimulus can be exercised; just the output displays are missing.
5. Comment your code with enough information to convey your approach and intentions. Try to organize your code in a coherent fashion.
6. **Submit your source file per team, on Blackboard, attaching only the 'main.c'. Make sure you click "Submit" on Blackboard and not just "Save".**

Student Name: _____

Item	Outcome	Score	Max.
Demo Requirements: • Requirement 3. a • Requirement 3. b • Requirement 3. c • Requirement 3. d • Requirement 3. e • Requirement 3. g (1) SW3 pressed when stopped. • Requirement 3. g (2) SW3 pressed during LAP.	 Yes / No Yes / No Yes / No Yes / No Yes / No Yes / No Yes / No		 1 1 2 1 1 1 1
Code comments • (2 = Mostly complete, 1 = Partially correct, 0 = Missing)			2
SubTotal			10
Late Deductions (30% per day)			
Total			
Open Ended Comments:			