EE/CS 120B: Introduction to Embedded Systems
University of California, Riverside
Winter 2016

Laboratory Exercise 5

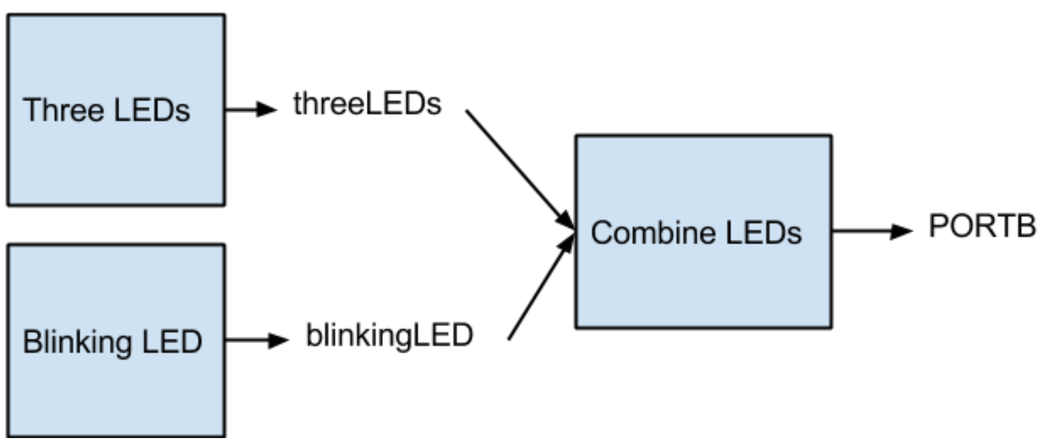
This laboratory exercise will introduce concurrent synchSMs. To keep things simple, all concurrent tasks will have the same period and inter-task communication will be limited. Use the timer abstraction and setup from Laboratory Exercise 3.

Pre-lab

Have your board fully wired and have your synchSMs and your complete C code for Exercise 1. *Be sure to use the clean timer abstraction and the structured method for converting synchSMs to C.* http://youtu.be/I_NU2ruzyc4

Exercises

1. Connect LEDs to PB0, PB1, PB2, and PB3. In one state machine (Three LEDs), output to a shared variable (threeLEDs) the following behavior: set only bit 0 to 1, then only bit 1, then only bit 2 in sequence for 1 second each. In a second state machine (Blinking LED), output to a shared variable (bilnkingLED) the following behavior: set bit 3 to 1 for 1 second, then 0 for 1 second. In a third state machine (Combine LEDs), combine both shared variables and output to the PORTB. Note: only one SM is writing to outputs. Do this for the rest of the quarter.



Concurrency with different period-tasks can be achieved by maintaining the elapsed time since the last tick for each task. A simple method ticks the timer at 1 ms and then counts X ticks to determine period X. Let's use that method here (**Do *not* tick the timer at the GCD of the tasks**). Refer to the first two pages of Chp8 of PES for an example.

Video Demonstration: http://youtu.be/Snmt0VFE_Zs

2. Modify the above example so the three LEDs light for 300 ms, while PB3's LED still blinks 1 second on and 1 second off.

   Video Demonstration: http://youtu.be/i8f5JSteH-U

   Generating Sound
   Sound can be generated by vibrating a membrane that creates sound waves in the air. A membrane vibrating at 261.62 Hz generates a "middle C" sound. A speaker has a membrane that moves when a voltage is applied (typically using a magnet that is moved by the electromagnetic wave of the changing electric current). Toggling a port from 0 to 1 at a frequency in the range of human hearing (around 20 Hz to 20,000 Hz) should generate sound if a speaker is connected to that port. The sound won't be pleasant because it's a square wave rather than a smoother sine wave, so it will sound more like a buzzer than a smooth tone.

   For more info, see Wikipedia: Audio frequency.

3. Start with the previous exercise's implementation. Connect your speaker's red wire to PB4 and black wire to ground. Add a third task that toggles PB4 on for 2 ms and off for 2 ms as long as a switch on PA2 is in the on position.

   Video Demonstration: http://youtu.be/Ufrlc6xyPyQ

Challenge Problems

4. Extend the previous exercise to allow a user to adjust the sound frequency up or down using buttons connected to PA0 (up) and PA1 (down). Using our 1 ms timer abstraction, the fastest you'll be able to pulse is 1 ms on and 1 ms off, meaning 500 Hz. You'll probably want to introduce another synchSM that polls the buttons and sets a global variable storing the current frequency that in turn is read by the frequency generator task.

   Video Demonstration: http://youtu.be/mt8eznAcp6o

5. Buttons are connected to PA0 and PA1. Output PORTB drives a bank of 4 LEDs. Pressing PA0 increments a binary number displayed on the bank of LEDs (stopping at 9). Pressing PA1 decrements the binary number (stopping at 0). If both buttons are depressed (even if not initially simultaneously), the display resets to 0. *If a button is held, then the display continues to increment (or decrement) at a rate of once per second.* However, if the button is held for 3 seconds, the incrementing/decrementing occurs once per 400 ms. Use synchronous state machines captured in C.

   Video Demonstration: http://youtu.be/D33pn3TcjpM

Each student must submit their .c source files according to instructions in the lab submission guidelines. Post any questions or problems you encounter to the wiki and discussion boards on iLearn.