BIEN4220: Laboratory 2 - Stop Light State Machine on the MSP430F2013

Introduction

This laboratory exercise introduces you to reverse engineering existing code, state machine concepts, and Timer_A interrupts on the MSP430.

The first part of this lab explores the use of a hardware timer on the MSP430. This timer runs in the background while the CPU sleeps in Low Power Mode. The timer can be configured to control digital pins without the CPU.

The second part of this lab focuses on understanding a stop light state machine in assembly. You will first want to attempt to draw out a flow diagram of the system to better navigate it – this requires reading through each line of code and looking for jump statements and key operations or variables. You will then identify a problem with the state machine an have to fix it.

Preparation:

1. Read this document.
2. Read Davies Chapter 8.11
3. Download from D2L the stop-light circuit diagram that you will interface to the MSP430.
4. Download the source files for the lab.
5. For stop_light.asm, identify the events that trigger each state transition. Create a state transition table or flow diagram identifying each state, each event that causes entry into that state, each event that causes exit from that state, and the actions taken upon those state transitions.
6. Bring these documents to lab to help you perform the required activities.

Procedure:

Part I: Exploring Timer A interrupts

Before you go crazy with the stop-light lab you should familiarize yourself with timer interrupts (since the stop light code uses them).

1. Create a project for timerA_interrupts.asm
2. Load and run this program on the MSP430. What is the timing (frequency, duty cycle) of the P1.1 output?
3. Modify your code to set the timer in *continuous* mode. How does this change the timing of the interrupt?
4. Modify your code to yield each of these three different periods: 400ms, 1.5 seconds, and 16 seconds. You may want to adjust the resolution and period of Timer_A in the various modes with different clock sources and clock dividers (See Davies, Table 8.1). Show your timer configurations and resulting output signals to an evaluator.

   To get this signature you must:

   - Show what timer modifications you made to yield each of the three periods above.

   Evaluator Signature: _____

Part II: Manipulating the state machine

1. Using a breadboard, prototype the stop_light circuit as described in the circuit diagram.
2. Create a project for stop_light.asm
3. Load and run this program on the MSP430. What is the code size? What is the Data size?
4. **What is the timing (period) of the interrupt (not when the LEDs change)?** You may want to use P2 to act as a debug pin that you control to measure timing. P2 is not set to GPIO by default! You must modify P2SEL to use them as GPIO.
5. **What is the time-related limitation of the state machine as implemented** (Hint, what is the bit-size of the timeout parameters)**? What are two ways of getting around this limitation?**
6. Approximately what *percentage* of time is spent servicing the state machine outside of the main loop? In other words, "How long does it take for the MSP430 to execute a longer branch of your state machine relative to the period between two long branches?" Answer: _____%. This is the CPU load of your state machine.
7. Hit the little button in your circuit. How much does the service time increase the CPU load? Why?
8. In answering the previous question, what is the obvious benefit of using an event-driven state machine such as the one implemented here?
9. Changing just one line of code or variable, make the state machine operate twice as fast. Describe which line you modified, and what you changed it to.
10. Press *and hold* the button. **Describe what happens and describe why the system behaves this way.** Add some code managing the button to eliminate erratic behavior due to "operator mis-use."
11. Have an evaluator review your working state machine and sign here:

    To get this signature you must:
    - Articulate the problem of the polling button system.
    - Modify the state machine system to prevent the overpolling button issue.

    Evaluator Signature: _____

    Submit on D2L:
    - Scan this signed form
    - Your stop_light.asm file