

## NET 3001-F09 Lab 4

**Date: Oct 6, 2009**

Objectives: I/O, Keyboard & timers

### ***I/O***

The MSP430 has several I/O ports, documented on the Resources web page. Write a short program which turns on the three LED's (red, yellow and green) and flashes them in a pattern.

Because this microcontroller runs so fast, you will need a delay() subroutine to pause between events, so that the LED's don't flash *too* fast.

This is basically a C version of what we did in Lab3.

### ***Timers***

The MSP430 has 2 timers on board which will allow us to measure the passage of time. In this lab we will use TimerA, and set it to count upward once every 64usec.

Counting at this rate, 16 counts will be approximately 1msec, and 16,384 counts will be approximately 1 second. (These counts are within 5%, and close enough for this lab).

To set TimerA to count up at this rate, you must set

```
BCSCTL2 = 6;      // sets the secondary internal clock to 1/8 MHz
TACTL = 0x02E0;   // sets timer A to use the secondary clock,
                  // to count up, and divide it by 8 again
```

Once you have done these two steps, you can read and write the TAR register, and watch the passage of time.

### **Target 1**

At the beginning of your program, you will need to set up the hardware: set the data direction registers, configure the timer and turn OFF the LED's.

For the first part of the lab, construct a program which waits a certain amount of time (you choose) and then lights up the LED's and resets timerA.

Modify the program to watch the keyboard for any activity; this is done by reading the bottom bit (bit 0) of P2. When activity is detected, turn on an LED, and check the timerA to see how much time has passed. Store the result in a variable and then put your program into an infinite loop.

You should probably debug this program using Eclipse and the debugger. Start the debugger, press the "run" button, and then hit a key on the MTS board. Then press the "pause" button in Eclipse to stop the cpu and you can check the variable. After you have checked the variable, change the program counter (r0) of your program to return to 0x8000 and try again.

## Target 2

Now modify the program so that after the key has been sensed, it lights up the green LED if the user was “fast” and the red LED if the user was “slow”. You are free to pick the decision point, based on the measurements you made above.

Report to the TA or the instructor the range of timings that you got in your experiments.

## Target 3 (optional)

Write a subroutine which takes the 2 byte timing measurement, and does the following:

- converts it to msec (divide by 16; shift right by 4)
- converts the hex number to decimal by this process:
  - create a global array of 4 bytes
  - subtract 1000 repeatedly until the remainder is less than 1000
  - store the number of times you had to do this in the array
  - subtract 100 repeatedly until the remainder is less than 100
  - store the number of times you had to do this in the next element of the array
  - subtract 10 repeatedly until the remainder is less than 10
  - store the number of times you had to do this in the next element of the array
  - store the remainder in the last element of the array

Add this subroutine to your reaction time test, so that your whole program:

- pauses
- lights the LED's
- waits for a keypress
- measures the time between LED & key
- converts this time to msec in decimal
- puts the “thousands” digit on the 7 segment display for .3 sec
- clears the 7 segment display for .2 sec
- puts the “hundreds” digit on the 7 segment display for .3 sec
- clears the 7 segment display for .2 sec
- puts the “tens” digit on the 7 segment display for .3 sec
- clears the 7 segment display for .2 sec
- puts the “units” digit on the 7 segment display for .3 sec
- clears the 7 segment display for .2 sec
  
- repeats

NOTE: to put a number on the 7 segment display, set the top bits of P2DIR to all 1's, and then send a 4-bit pattern into the top 4 bits of P2OUT. In addition, you will need the line:

```
P2SEL = 0;
```

in order to deal with a peculiarity of this chip

If you are concerned about accuracy, you can add these two lines to your hardware initialization:

```
BCSCTL1 = CALBC1_1MHZ;  
DCOCTL = CALDCO_1MHZ;
```

This sets the MSP chip to run at almost exactly 1MHz (within <1%).

### ***Beeper (optional)***

Modify the program above so that the stimulus for the user is 8 cycles of tone into the beeper. To do this, you can use a small subroutine to send alternate one's and zero's to the beeper. Make sure you don't go too fast, because human hearing only goes up to 100us/cycle.

Are the human reaction times faster or slower using a beep as the trigger?

The important thing to note here is that the timer is a helper circuit. It keeps track of time while your other code can go do something else.

Note: to enable the 7 segment display AND the beeper, you will need to set *all* the appropriate bits in P2DIR.