

NET 3001-F09 Lab 7

Date: Oct 27, 2006

Objectives: Interrupt code; text formatting, hardware tones

Interrupts

Start another project and include `<io.h>`, `<signal.h>` and `"lcd.h"`.

Make a timer interrupt that runs at 10msec. Use the settings from the last slide of the NET3001-8-RTI set. In the timer interrupt, simply increment a global variable.

Watch the global variable change in your `main()`. Keep track of seconds and minutes, and *every 10 seconds*, issue a `lcdSprintf()` statement that reports the passage of time. The output of your program should look like this:

```
"running 0 mins"  
"and 10 seconds"  
  
"running 0 mins"  
"and 20 seconds"  
  
"running 0 mins"  
"and 30 seconds"
```

Here is a summary of the `lcdSprintf()` statement.

The first argument is the address of a writable array of chars, the buffer that you are going to fill.

The 2nd argument of the `lcdSprintf()` is a "format string" which may contain % commands. The most common are

```
%d    decimal  
%x    hex
```

After the "format string" you can put the variable that you want to print. The `lcdSprintf()` command will run through the "format string" and substitute the % command with your variable. For example,

```
char someSpace[20];  
i=65;  
lcdSprintf(someSpace, "a demo: %d",i);  
lcdPuts(someSpace);    // and put it on the LCD screen
```

this results by filling the emptySpace array with

```
"a demo: 65"
```

Make sure the empty space that you provide is big enough to handle the string that `lcdSprintf()` will build for you.

Change your program so that it updates the display every second. Put the least significant digit of the “seconds” counter on the 7 segment display. On the LCD, use the first line to simply put the number of seconds the program has been running. On the second line puts something about the keyboard button. For example:

```
"running 132 secs"  
"keypress: yes"
```

```
"running 133 secs"  
"keypress: no"
```

Notice how the LCD display sometimes display extra (useless) information on the right hand side of what you intended to display. Think of how you might eliminate that text.

Save all this code; you will need it in assignment 5.

Hardware Tones

If you haven't already, build the code that plays a tone through the beeper using the hardware timer A. The instructions are included in assignment 4.

```
// run these next 3 lines of code once, at power up  
P2DIR = 4;           // P2.2 is an output  
P2SEL = 4;           // beeper pin P2.2 is run by the timer  
TACTL = TASSEL_2 + ID_0 + MC_1; // use SMCLK (same as main clock) 1usec  
  
// to make a tone using the half-period (in usec) (put this in subroutine)  
TACCR0 = period;     // timer counts up to this number  
  
// start the tone with  
TACCTL0 = OUTMOD_4;  
  
// and shut off the tone with  
TACCTL0 = OUTMOD_0;
```

Pulse Width Modulation

You can send an analog signal to the DC motor by using TimerA to send out a variable pulse width. To do this, you put the total pulse period in TACCR0, and the pulse width in TACCR1. Here's the complete formula:

```
// do this once
TACTL = TASSEL_2 + MC1;
TACCR0 = 1000; // or 100 if you wish
P1SEL |= 4; // turn one of the DC motor pins to special mode

// do this to turn on the motor at variable speed
TACCTL1 = OUTMOD_7;
TACCR1 = n; // where n is a number between 0 and 1000 (or between 0 and 100)
```