Assignment 3 - Interrupts

EGC 443 Embedded Systems

George Dagis (CE), NAME OMITTED (CE), NAME OMITTED (CE)

March 13th, 2018

Instructor: Michael Otis

LED Frequency Interrupt

```
#include "TM4C123GH6PM.h"
void delayMs(int n);
int main(void)
{
   SYSCTL->RCGCGPIO |= 0x03;   /* enable clock*/
   GPIOA->DIR &= ~0x20;       /* PORTA5 switch */
   GPIOB->DIR |= 0x20;        /* PORTB5 LEDs */
   GPIOA->DEN |= 0x20;        /* PORTA5 dig*/
   GPIOB->DEN |= 0x20;        /* PORTB5 dig*/
   GPIOA->PUR |= 0x20;        /* enable pull up for PORTF4, 0 */

   GPIOA->IS  &= ~0x20;       /* make bit 6 edge sensitive */
   GPIOA->IBE &= ~0x20;       /* trigger is controlled by IEV */
   GPIOA->IEV &= ~0x20;       /* falling edge trigger */
   GPIOA->ICR |= 0x20;        /* clear any prior interrupt */
   GPIOA->IM  |= 0x20;        /* unmask interrupt */

   /* enable interrupt in NVIC and set priority to 3 */
   NVIC->IP[0] = 3 << 5;     /* set interrupt priority to 3 */
   NVIC->ISER[0] |= 0x1;     /* enable IRQ0 (D0 of ISER[0]) */

   __enable_irq(); /* global enable IRQs */
```

```
    /* toggle the red LED continuously */
    while(1)
    {
        GPIOB->DATA |= 0x20;
        delayMs(500);
        GPIOB->DATA &= ~0x20;
        delayMs(500);
    }
}

/* Switch is connected to PORTA and triggers the interrupt */
/* Slows down the frequency of the LED for a little bit*/
void GPIOA_Handler(void)
{
    int i;
    volatile int readback;
    /* toggle slowed LED five times */
    for (i = 0; i < 5; i++)
    {
        GPIOB->DATA |= 0x020;
        delayMs(100);
        GPIOF->DATA &= ~0x20;
        delayMs(100);
    }
    GPIOA->ICR |= 0x20; /* clear the interrupt flag before return */
    readback = GPIOA->ICR;     /* a read to force clearing of interrupt flag */
```

```
}


/* delay n milliseconds (16 MHz CPU clock) */
void delayMs(int n)
{
   int i, j;
   for(i = 0 ; i < n; i++)
      for(j = 0; j < 3180; j++)
         {}  /* do nothing for 1 ms */
}


/* This function is called by the startup assembly code to perform system specific initialization tasks. */
void SystemInit(void)
{
   __disable_irq();    /* disable all IRQs */


   /* Grant coprocessor access */
   /* This is required since TM4C123G has a floating point coprocessor */
   SCB->CPACR |= 0x00F00000;
}
```

The external LED system can be seen in the figure below

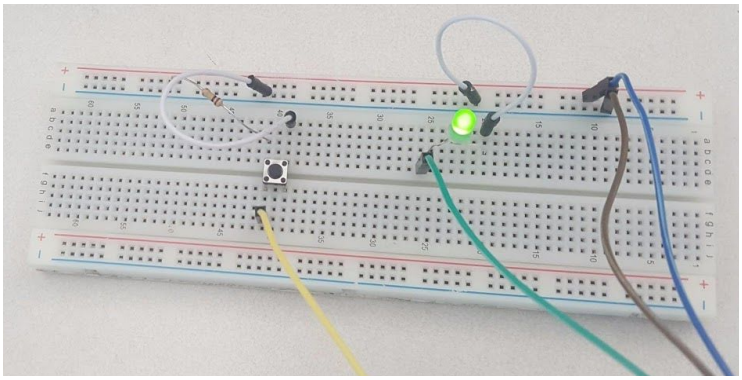

Figure 1: External LED system

Summary

In this lab, it was first necessary to ensure that the ports were configured for use. This is because one of our IO pins was an input coming from a switch and the other was an output going to an LED. In our diagram above, it is shown that a resistor is in place, preventing high current and burnout. The LED toggle function is set up in the main thread, while a secondary thread controls the interrupts. When the code intercepts an interrupt, the LED's frequency changes (the LED fade becomes slower).