Christopher Brant
C19816588
Due on 10/11/2017

ECE 3270 Microcontroller Interfacing Lab
Lab 5: Interrupts

Abstract

This lab was performed to physically demonstrate how an interrupt can be programmed and implemented using the PIC32MX150F128D microcontroller. As well as to give hands on experience with implementing basic interrupts for embedded programming.

Introduction

This experiment included the building of a circuit using the PIC32MX150F128D microcontroller to control a series of LEDs that will continuously count from 0 to 15 in binary, except when an interrupt is triggered bya toggle switch/debouncer being pressed. When that interrupt is triggered, the count will pause, the LEDs will display a given value that is set in the interrupt handler, and the interrupt handler includes a delay to show that the interrupt is what is changing the LEDs. After the interrupt's delay, the main code picks up its functionality continuously from where it had left off.

Experimental Procedures

- This procedure is relatively short and includes two basic steps.
- Step one is to wire the 4 LEDs to output pins RB0-RB3, wire the debouncer output to the INT0 pin, and ensure the microcontroller is powered and grounded.
- Step two is to write and compile the code in Figure 2, which includes setting the interrupt control register, interrupt enable register, interrupt priority control bits, and the interrupt flag status register.
- The main function will function exactly as Lab 1, however in this lab, an interrupt handler is included so that whenever the specified interrupt is triggered, the LEDs will output a value of 15, hold that value momentarily, and then return to counting from 0 to 15 as it was doing prior to the interrupt.

Results

No tabulated values or calculations were necessary in this lab. However there are a few results that can be seen, and those are that the interrupt code/function is automatically handled by the microcontroller and does not have to be called and that it is important to enable your interrupts using the function INTEnableSystemMultiVectoredInt();. Another basic observation is that the interrupt does exactly as it is supposed to and pauses its continuously running function, runs the interrupt code, and then returns directly back to where the main function had left off.

Christopher Brant
C19816588
Due on 10/11/2017

Discussion

Final conclusions include that interrupts can happen at any time and are unnecessary to account for other than setting the __ISR() parameters in the interrupt function definition. One last conclusion is also that a debouncer allows for the state that is being input into the interrupt to not fluctuate highly during its being pressed.

Conclusions

Summarizing the conclusions listed above gives us a statement that mimics what was expected of this experiment, which was to show that interrupts are automatically handled by the PIC32MX150F128D microcontroller as long as a function definition is present, and that the debouncer allows for the interrupt input to not rapidly fluctuate.

Figures and Tables

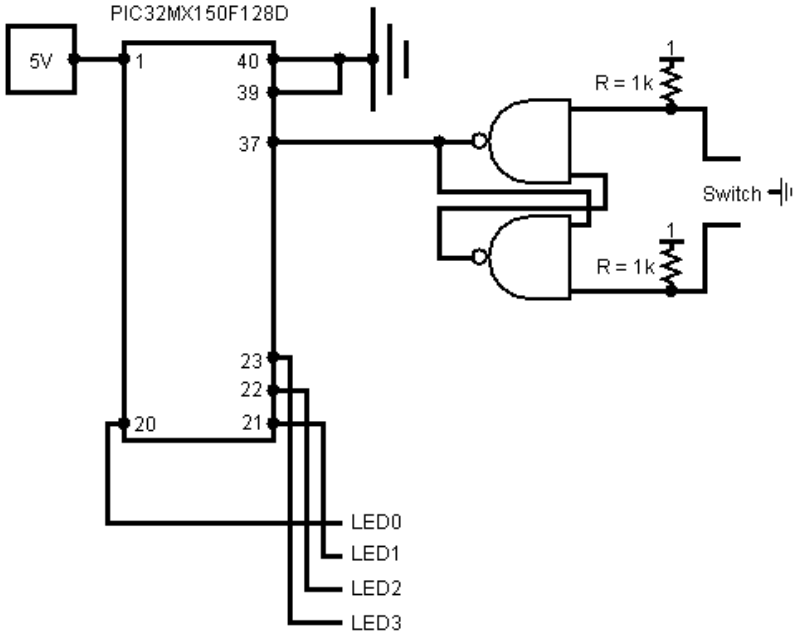

Figure 1: Wiring/Circuit Diagram for Lab 5:Interrupts

```
/* Christopher Brant
 * C19816588
 * Lab 5: Interrupts
 * 10/5/2017
 */

#include <plib.h>

delay()
{
    int i, j;
    for(i = 0; i < 500; i++)
        for(j = 0; j < 500; j++);
}

int main(void)
{
    // Set important registers and enable necessary functions
    TRISB = 0x80;
    INTEnableSystemMultiVectoredInt();

    // Setting interrupt control register
    INTCONbits.INT0EP = 1;

    // Setting the interrupt enable register
    IEC0bits.INT0IE = 1;

    // Setting the interrupt priority control bits
    IPC0bits.INT0IP = 0x1;

    // Setting the interrupt flag status register
    IFS0bits.INT0IF = 0;

    // Declare and initialize count variable
    int count = 0;

    while(1)
    {
        LATB = count; //Output count to LEDs
        count++;

        if(count > 15)//Restrict count to 0-15, needing only 4 bits
            count = 0;

        delay();
    }

    return 0;
}

void __ISR(3) toggled(void)
{
    // Turn on LEDs
    LATB = 15;

    delay();

    // Turn off interrupt flag
    IFS0bits.INT0IF = 0;
}
```

Figure 2: Full Code for Lab 5:Interrupts