ECE 3720

Microcomputer Interfacing Laboratory

Section 002

Daniel Waldner

Date Performed: 2/18/2020

Lab 5

ABSTRACT:

This lab was designed to show how to use the hardwired interrupt on the PIC32 microcontroller to stop the code and perform another task, then go back to where it left off. The lab also showed how to use a debouncing circuit to keep a switch from bouncing during press.

This study resource was shared via CourseHero.com

https://www.coursehero.com/file/56497270/Lab-5-Postlabpdf/

Introduction:

For this lab the circuit was setup so that four of the B register pins were outputting to LEDs to continuously count from 0 to 15 in binary. Then 5V was connected to a debouncing circuit that would go into the hardwired interrupt pin on the microcontroller. The circuit would send a logic 1 into the interrupt pin when the button was pressed and a logic 0 otherwise. When the button was pressed and a logic 1 was sent to the interrupt pin, the LEDS would stop counting, light up all four LEDs, turn off all four LEDs, then continue counting where it left off.

Experimental Procedure:

The layout of the circuit can be seen in **FIGURE 1** below. 5 volts was put into the power supply of the microcontroller (pin 1) and the ground was connected to the ground (pins 39 and 40). 5 volts was then connected to the two pull up resistors in a debouncing circuits. The resistors were connected to two of the inputs on the NAND gates and one to each side of a switch. The other pin on the switch was connected to ground and the two outputs of the NAND gates were connected to the inputs of the other NAND gates. This circuit would cause on of the NAND gates to out put a logic 0 at the start and change to a logic 1 when the switch is flipped. The other NAND gate would output the opposite. The output of one of the outputs was then connected to pin 37 of the microcontroller. This pin was chosen because it is the only hardwired interrupt pin on the microcontroller. Pins 21, 21, 22, and 23 were then used as outputs to four different LEDs. These pins were chosen for us because we reused the code from lab 1.

CODE 1 below is the code from lab 1 that counted from 0 to 15 on four LEDs. **CODE 2** below is the modified code that adds an interrupt to the program. The first change in the code was to add the line INTEnableSystemMultiVectoredInt(). This enables multi vectored interrupts for the program. Next, the line TRISB = 0x00 was changed to TRISB = 0xF0. The

This study resource was shared via CourseHero.com

original code set all the pins of register B to outputs and the modified code set the first four pins of register B to outputs and the second four pins of register B to outputs. This was needed because our interrupt input is in B7. Next, IPC0bits.INT0IP = 0x1; was used to set the priority control register for interrupt 0, this value just needed to be greater than 0 because it is the only interrupt use, INTCONbits.INT0EP = 1; was used to set the interrupt edge trigger direction, this was set to 1 to interrupt on the rising edge of the input, and IEC0bits.INT0IE = 1; was used to enable the register of interrupt 0. After the interrupt registers were set a function for the interrupt was created. There is no return value or input to the function, so these values were set to void, and __ISR(3) was used to select interrupt 0 (vector 3). This function will set the output to 15, set the output to 0, use the line IFS0bits.INT0IF = 0; to set the interrupt flag of interrupt 0 back to 0, then continue counting from where it left off.

RESULTS and DISCUSSION:

The final observed behavior of the circuit was that the LEDs would continuously count from 0 to 15 in binary, then, when the switched was pressed, light up 15, light up 0, then continue counting where it left off. The only problem I had with this lab was that, when I was trying to write my code to the microcontroller, I did not have my lab 5 code set as the main project. Because of this, it was writing a different labs code to the microcontroller. This was easily fixed by setting my lab 5 code as the main project. Interrupts are very useful in programs when you need to switch to doing something else in your program, then continue where you left off. The debouncing circuit is also commonly used in switching to remove the bouncing effect of switches.

CONCLUSION:

This study resource was shared via CourseHero.com

This lab helped build an understanding of how the interrupts on the PIC 32 microcontroller are used and the theory and implementation of a debouncing circuit.

REFERENCES:

Clemson University ECE 3720 Lab 2 PowerPoint.

PIC 32 Family Data Sheet

chipKIT Cmod Reference Manual

This study resource was shared via CourseHero.com

FIGURES and TABLES:

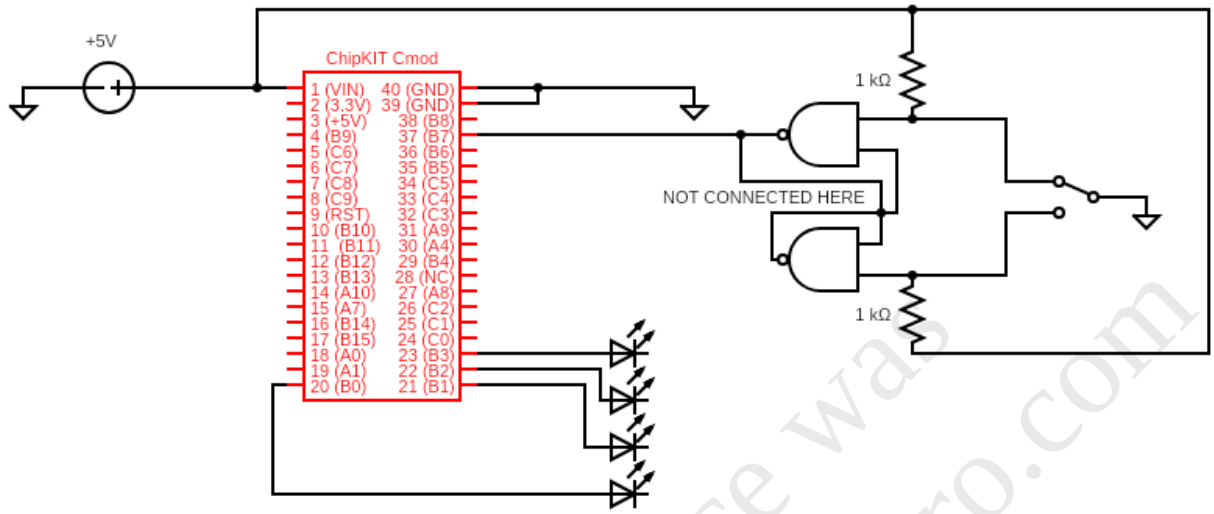

FIGURE 1: Wiring for lab 5 (Pin Connections described in Experimental Procedures)

CODE:

CODE 1

```
#include <plib.h>

delay(){
   int i, j;
   for(i = 0; i < 500; i++)
      for(j = 0; j < 500; j++);
}
main(){
   int count = 0;
   TRISB = 0x00; //Set 0-3 to output and 4-7 as input\

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

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

This study resource was shared via CourseHero.com

CODE 2

```
#include <plib.h>
void __ISR(3)interupt(void){
    LATB = 15;
    delay();
    LATB = 0;
    delay();

    IFS0bits.INT0IF = 0;
}
delay(){
    int i, j;
    for(i = 0; i < 500; i++)
        for(j = 0; j < 500; j++);
}
main(){
    INTEnableSystemMultiVectoredInt();
    int count = 0;
    TRISB = 0xF0; //Set 0-3 to output and 4-7 as input


    IPC0bits.INT0IP = 0x1;
    INTCONbits.INT0EP = 1;
    IEC0bits.INT0IE = 1;


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

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

This study resource was shared via CourseHero.com