ECE 3720

Microcomputer Interfacing Laboratory

Section 002

Daniel Waldner

Date Performed: 2/24/2020

Lab 6

ABSTRACT:

This lab was designed to show how to peripheral pin select (PPS) on the PIC32 microcontroller to use multiple interrupts at the same time. The lab also showed how to use the Grayhill 61C optical encoder.

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 Grayhill 61C optical encoder that would send its A output to pin the hardwired interrupt pin and its B output to the pin that is set as interrupt 1 using PPS. The optical encoder would send either a 1 or a 0 to each output depending on which of the four positions it was in. When the encoder was turned, one of the outputs would change from a 0 to a 1 or vise versa. This would cause the corresponding interrupt to occur that would then either increase or decrease a counter. The value of the counter was then displayed on 4 LEDs.

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 power supply (pin 6) of the optical encoder and two pull up resistors connected to the A and B outputs (pin 4 and 5). Pin 1 on the encoder was connected to ground and the other two pins were not used. Output A on the encoder was connected to the hardwired interrupt pin on the microcontroller (pin 37) and output B was connected to the pin 33. Pin 33 was chosen for output B because it was a 5V tolerant pin that could be used with the PPS.. Pins 21, 21, 22, and 23 were then used as outputs to four different LEDs.

From **CODE 1** below, the main function starts by enabling the multi-vector interrupts, setting the lower 4 bits of register B to outputs, the 7th bit to input, and the 4th bit of register C to input. The four output bits were set to send the value of the counter to four LEDs and the two input bits were set for the interrupts. Next the registers for interrupt 0 were set, the priority control bit was set to 1, the interrupt edge trigger was set to rising edge if the input to pin 37 was

0 and falling edge if the input to pin 37 was 1, and the interrupt enable bit was set to 1. The edge triggers were set this way because the interrupt needed to be used when the value of pin 37 changed from a 1 to a 0 or vise versa and the starting value of was not known. Next pin 33 needed to be set as interrupt 1 using peripheral pin select. To do this the command PPSInput(4, INT1, RPC4); was used. 4 was used because interrupt 1 is in group 4, INT1 was chosen because we wanted to use interrupt 1, and RPC4 was chosen because pin 33 is bit 4 in register C. Next the interrupt was set in the same way as interrupt 0, but with the edge trigger value being determined from the value at pin 33 instead of pin 37. A while loop was then entered that set the value of count to 0 if it was greater than 15 and 15 if it was less than 0 and the count was output to the lower four bits of register B.

To increment or decrement the counter based on the number of times the encoder was turned, interrupt 0 was called when the value of pin 37 changed and interrupt 1 was called when the value of pin 33 changed. When interrupt 0 was called the value of pin 33 was checked. If the value was a 1, The value of pin 37 was checked. If the value of pin 37 was a 1 the counter was decremented by 1 and incremented by 1 if it was a 0. If the value of pin 33 was a 0, the value of pin 37 was checked. If the value was a 0 the counter was decremented by 1 and incremented by 1 if it was a 1. The interrupt flag was then cleared the edge trigger was changed to the opposite of what is previously was. This logic was used because INT0 was called when the value of output A of the encoder was changed. If output A changed and was now a 1 and out put B was a 1, that meant that the position went from 4 to 3. This kind of logic was used for all the increments and decrements. Interrupt 1 was called when the value of output B of the encoder was changed. This logic for this is similar except that the value of output A was checked first and then the count was incremented or decremented based on the new value of output B.

RESULTS and DISCUSSION:

The final observed behavior of the circuit was that the LEDs would count up when the encoder was turned one way and count down when the encoder was turned the other way. The only problem I had with this lab was that I had the wrong vector selected for my interrupt 1 function. This caused a very strange output to be sent to the LEDs and the circuit not work. This was easily fixed by double checking the vector values from the pic32 datasheet. Having multiple interrupts in a circuit is very commonly used, so being able to use multiple can be important. Knowing how to use the PPS is also very useful, because it can be used to set a wide variety of function that can be used for many different programs and circuits.

CONCLUSION:

This lab helped build an understanding of how the peripheral pin select on the PIC 32 microcontroller are used and the theory and implementation of an optical encoder.

REFERENCES:

Clemson University ECE 3720 Lab 2 PowerPoint.

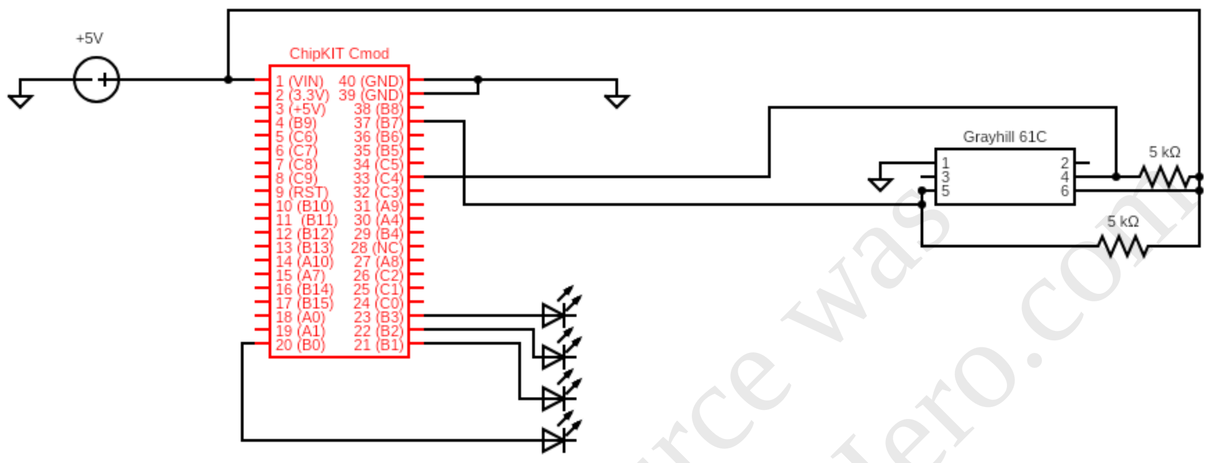
PIC 32 Family Data Sheet

chipKIT Cmod Reference Manual

Grayhill series 61C datasheet.

FIGURES and TABLES:

FIGURE 1: Wiring for lab 6



(Pin Connections described in Experimental Procedures)

CODE:

CODE 1

```
#include <plib.h>
int count = 0;
void __ISR(3)interupt1(void){

    if(PORTCbits.RC4 == 1){
        if(PORTBbits.RB7 == 1)
            count--;
        else
            count++; }
    else{
        if(PORTBbits.RB7 == 0)
            count--;
        else
            count++;}
    INTCONbits.INT0EP ^= 1;
    IFS0bits.INT0IF = 0;
}
```

```
void __ISR(7)interupt2(void){

    if(PORTBbits.RB7 == 1){
        if(PORTCbits.RC4 == 1)
            count++;
        else
            count--;
    }
    else{
        if(PORTCbits.RC4 == 0)
            count++;
        else
            count--;
    }
    INTCONbits.INT1EP ^= 1;
    IFS0bits.INT1IF = 0;
}
main(){
    INTEnableSystemMultiVectoredInt();
    TRISB = 0x80; //Set 0-3 to output and 4-7 as input
    TRISCbits.TRISC4 = 1;

    IPC0bits.INT0IP = 0x1;
    if(PORTBbits.RB7 == 0){
        INTCONbits.INT0EP = 1;}
    else{ INTCONbits.INT0EP = 0;}
    IEC0bits.INT0IE = 1;

    PPSInput(4,INT1, RPC4);

    IPC1bits.INT1IP = 0x2;
    if(PORTCbits.RC4 = 0){
        INTCONbits.INT1EP = 1;}
    else{ INTCONbits.INT1EP = 0;}
    IEC0bits.INT1IE = 1;


    while(1){
        if(count > 15)
            count = 0;
        if(count < 0)
            count = 15;
        LATB = count;
    }
}
```