Christopher Brant
C19816588
Due on 10/18/2017

ECE 3270 Microcontroller Interfacing Lab
Lab 6: Peripheral Pin Select

**Abstract**

The purpose of this lab was to learn about the use of interrupts by assigning interrupts by peripheral pin selection. This will be done so that multiple interrupts can be utilized within one program, and specifically used so that there is no need for a main while loop segment other than to ensure that the program does not stop running. The 61C Optical Encoder was used as input for this lab as well, and is explained later.

**Introduction**

In this experiment, we learned how to utilize the peripheral pin selects to assign multiple interrupts to available pins and utilize those interrupts to then control LEDs which would count up or down between 0-15 and wrap around when necessary based on the input read in from a 61C Optical Encoder. When the optical encoder is turned clockwise, the overall count variable will increase until it hits a value of 15 and then upon the next clockwise turn, it will roll back over to 0. Likewise, when the optical encoder is turned counterclockwise, the overall count variable will decrease until it hits a value of 0 and upon the next counterclockwise turn, it will roll back over to 15. The inputs that indicate turning will be handled exclusively by interrupts INT0 and INT1.

**Experimental Procedures**

- The first thing that must be done is to wire the overall circuit to be just as it is shown in Figure 1.
- The next thing that is to be done is to decide on the logic that will be followed to determine whether the optical encoder has been turned clockwise or counterclockwise.
- Then, in your code you must set up the overall pin selections by setting the correct inputs and outputs on the tristate registers, and then using the PPSInput() function to set INT1 to pin RPC4.
- Then all of the overall interrupt options must be set, as well as the interrupt control register bits must be XORed with 1 to set the next value of the control bit to be opposite of its current value.
- Then the rest of the code should be written just as it is seen in Figure 2, including the logic for deciding whether or not the encoder was turned in what direction as well as including the correct interrupt vector values in the interrupt function headers. Then just run the program.

Christopher Brant
C19816588
Due on 10/18/2017

**Results**

There were no tabulated values and no values necessary for the record. Observations of this lab show that the difficulty in ensuring that all possible situations for turning direction of the optical encoder are slightly difficult to account for at first glance, but upon further work can become very simple. Overall, this lab was quite straightforward and moved on from the principles that were understood in the previous lab and built an understanding of the way that an optical encoder functions as well.

**Discussion**

Overall, the final conclusions of this experiment include the statements that the microcontroller can be programmed to determine the direction the encoder was turned, with only knowing the current state, and the current edge trigger. One other conclusion that can be drawn is that multiple interrupts can be configured for the microcontroller as well as multiple other peripherals can be configured at the same time as well.

**Conclusions**

The conclusions of this experiment include that the direction of the encoder's turning can be initially difficult to decipher, and that multiple interrupts and/or peripheral pins can be configured at a time. Lastly, this experiment propelled us to learn the configurations and use cases of multiple peripheral pins, which will be entirely useful later for our design project.

**Figures and Tables**

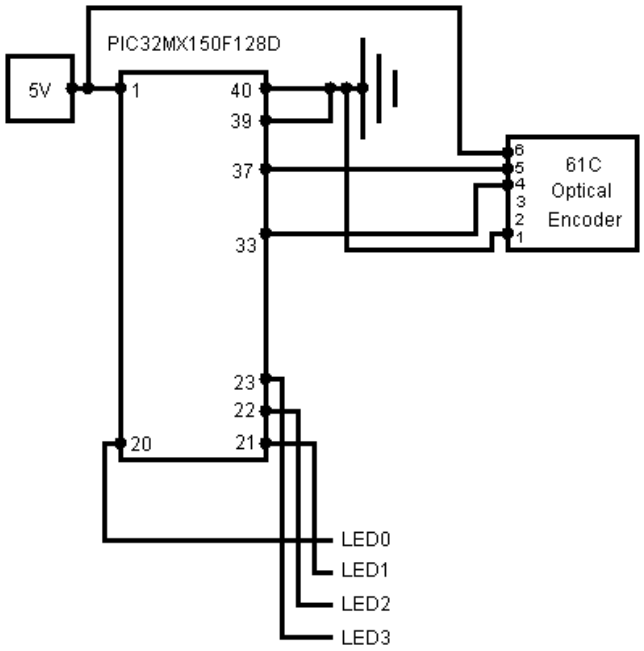


Figure 1: Wiring Diagram of the circuit for Lab 6: Peripheral Pin Select

Christopher Brant
C19816588
Due on 10/18/2017

```
/* Christopher Brant
 * C19816588
 * Lab 6: Peripheral Pin Select
 * 10/12/2017
 */

#include <plib.h>

// Declare global variable for count
int count = 0;

// Function to set the LEDs in the cases of cw or ccw
void setLEDs(int cw)
{
    if (cw == 1)
    {
        if (count + 1 > 15)
        {
            count = 0;
            LATB = count;
        }
        else
            LATB= ++count;
    }
    else
    {
        if (count - 1 < 0)
        {
            count = 15;
            LATB = count;
        }
        else
            LATB = --count;
    }
}
int main(void)
{
    // Set important registers and enable necessary functions
    TRISB = 0x80;
    TRISC = 0x10;
    INTEnableSystemMultiVectoredInt();

    // Set peripheral pin macro for INT1
    PPSInput(4, INT1, RPC4);

    // Setting interrupt control register bits
    INTCONbits.INT0EP = 1 ^ PORTBbits.RB7;
    INTCONbits.INT1EP = 1 ^ PORTCbits.RC4;

    // Setting the interrupt enable register bits
    IEC0bits.INT0IE = 1;
    IEC0bits.INT1IE = 1;

    // Setting the interrupt priority control bits
    IPC0bits.INT0IP = 0x1;
    IPC1bits.INT1IP = 0x1;

    // Setting the interrupt flag status register bits
    IFS0bits.INT0IF = 0;
    IFS0bits.INT1IF = 0;

    // Run forever and wait for interrupts
    while(1);

    return 0;
}
void __ISR(3) OutputA(void)
{
    // Declare cw variable
    int cw = 0;

    // Check to see what direction encoder was turned
    // Checks for cw options, otherwise returns 0 for ccw
    if (INTCONbits.INT0EP == 1)
    {
        if (PORTBbits.RB7 == 1 && PORTCbits.RC4 == 0)
            cw = 1;
    }
    else if (INTCONbits.INT0EP == 0)
    {
        if (PORTBbits.RB7 == 0 && PORTCbits.RC4 == 1)
            cw = 1;
    }

    // Set LEDs to new count value
    setLEDs(cw);

    // Flip the edge trigger bit for Output A
    INTCONbits.INT0EP ^= 1;

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

void __ISR(7) OutputB(void)
{
    // Declare cw variable
    int cw = 0;

    // Check to see what direction encoder was turned
    // Checks for cw options, otherwise returns 0 for ccw
    if (INTCONbits.INT1EP == 1)
    {
        if (PORTBbits.RB7 == 1 && PORTCbits.RC4 == 1)
            cw = 1;
    }
    else if (INTCONbits.INT1EP == 0)
    {
        if (PORTBbits.RB7 == 0 && PORTCbits.RC4 == 0)
            cw = 1;
    }

    // Set LEDs to new count value
    setLEDs(cw);

    // Flip the edge trigger bit for Output A
    INTCONbits.INT1EP ^= 1;

    // Turn off interrupt flag
    IFS0bits.INT1IF = 0;
}
```

Figure 2: Code for Lab 6: Peripheral Pin Select