ECE 3720

Microcomputer Interfacing Laboratory

Section 002

Daniel Waldner

Date Performed: 3/9/2020

Lab 8

ABSTRACT:

This lab was designed to show how to use PWM on the PIC32 microcontroller and how to use the PWM to create drive a motor in two different directions.

Introduction:

For this lab the circuit was setup so that two output compare modules were connected to the inputs of the 1293dne. The motor was then connected as an H-bridge so that it could run in two directions. A button was then set connected to an interrupt on the PIC32 to change the duty cycles of the PWMs, so that the motor could run at different speeds and directions.

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). A low active button was then connected to interrupt 0 (pin 37), so that the duty cycle would change when the button was pressed. Output compare 2 (pin 35) and output compare 1 (pin 34) were then connected to input 1 and 2 of the 1296dne motor driver (pin 2 and 7). 5 volts was connected to Vcc1 (pin 16), 1,2 enable (pin 1), and Vcc2 (pin 8). This was to power the integrated circuit, have driver channels 1 and 2 enabled, and supply power for the drivers. Driver outputs 1 and 2 (pin 2 and 6) were then connected to the motor in an H-bridge, so that it could run in both directions. The heat sink and ground pins (pin 4 and 5) were connected to ground.

From **CODE 1** below, the code starts by declaring two global counters i and j, enabling multi vector interrupts, and set bit 5 of register C to output, bit 5 of register B to output, and bit 7 of register B to input. Bit 5 of register C was used for output compare 2, bit 5 of register B was used for output compare 1, and bit 7 of register B was used for interrupt 0. This were chosen because they were the recommended pins for the output compares used and the only hardwired interrupt pin on the microcontroller. Then PPS was used to set output compare 1 and 2. Interrupt 0 was then set up with its polarity set to rising edge and its priority as 1. Next, the timer register was setup. Timer two was chosen because it worked with the output compare module. TCS was set to 0 because the internal clock was needed, TCKPS was set to 0 because the scaling was done in the code, TGATE was set to 0 because accumulation was not needed, and T32 was set to 0 because a 16 bit timer was needed not a 32 bit timer. TMR2 was set to 0 and PR2 was set to 100. The output compare registers then needed to be set. OC1CONbits.OCM=6 was used to turn PWM mode on with no fault pin, OC1CONbits.OC32=0 was used to select 16 bit timer source, OC1CONbits.OCTSEL=0 was used to select timer 2 as the source, and OC1CONbits.ON=1 was used to enable the module. OC1RS was set to 0 at the start to have the duty cycle start as 0 and the motor not spinning. Output compare 2 was setup the same way.

A while loop with nothing in it was then entered. The interrupt function was then setup. This function would increment count i by 25 every time the interrupt was called until the count was over 100. Then, when i was greater than 100 would set it back to 0 and start incrementing j by 25 every time the function was called. Then OC1RS was set to i, OC2RS was set to j, and the interrupt flag was cleared. The function was always setup this way because the duty cycle for one motor direction needed to be zero, because the motor can't spin in two directions, and the duty cycles needed to be 25%, 50%, 75%, and 100%. The duty cycle is the OC1RS value divided PR2 value, set to 100 for ease of computation. The duty cycle for compare module one would increment by 25% until it was over 100% while the duty cycle for compare module 2 was 0. Then, compare module 2 would increment by 25% until it was over 100% while the duty cycle for compare module 1 was 0. This would allow the motor to spin at different speed in two directions.

RESULTS and DISCUSSION:

The final observed behavior of the circuit was that the speed of the motor would increase as you pressed. After the button was pressed four times it would start spinning in the opposite direction. The only problem I had with this lab was that the original output compare module I tried to use was not working. This caused a lot of issues but was fixed by using a different compare module. Pulse width modulation (PWM) is used in a wide variety of applications and it is important to understand how it works and how to apply it in a circuit.

CONCLUSION:

This lab helped build an understanding of how to use pulse width modulation with the PIC 32 microcontroller and how to use an H-bridge, motor doriver, and PWM to run a motor at different speeds in different directions.

REFERENCES:

Clemson University ECE 3720 Lab 8 PowerPoint.

PIC 32 Family Data Sheet

chipKIT Cmod Reference Manual

129DNE Data Sheet

FIGURES and TABLES:

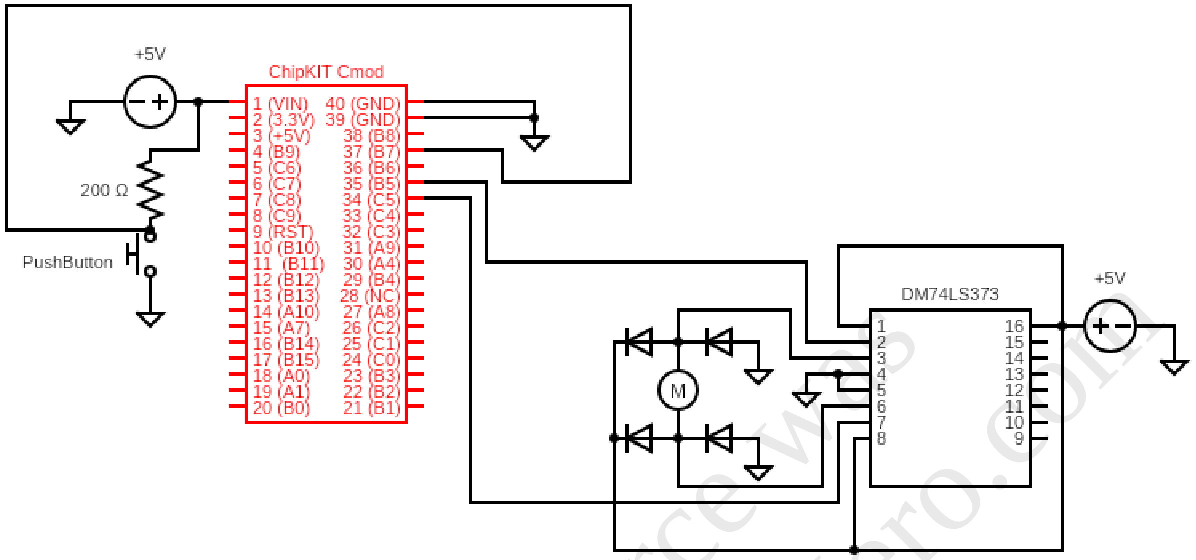

FIGURE 1: Wiring for lab 7 (Pin Connections described in Experimental Procedures)

CODE:

CODE 1

```
#include <plib.h>
int i,j;


void __ISR(3)pushbutton(void){

  if(j==0){
    i+=25;
    j=0;
    if(i>100){
      i=0;
      j+=25;
    }
  }
  else{
    j+=25;
    i=0;
    if(j>100){
      j=0;
      i+=25;
    }
  }

  OC1RS=i;
  OC2RS=j;

  IFSObits.INT01F=0;
}

void main(void){


        INTEnableSystemMultiVectoredInt();

        TRISCbits.TRISC5 = 0;
        TRISBbits.TRISB5 = 0;
        TRISBbits.TRISB7 = 1;

        PPSOutput(1,RPC5,OC1);
        PPSOutput(2,RPB5,OC2);

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

        T2CONbits.TCS=0;
        T2CONbits.TCKPS=0;
        T2CONbits.TGATE=0;
        T2CONbits.T32=0;
        TMR2=0;
        PR2=100;
        T2CONbits.ON=1;

        OC1CONbits.OCM=6;
        OC1CONbits.OC32=0;
        OC1CONbits.OCTSEL=0;
        OC1RS=0;
        OC1CONbits.ON=1;

        OC2CONbits.OCM=6;
        OC2CONbits.OC32=0;
        OC2CONbits.OCTSEL=0;
        OC2RS=0;
        OC2CONbits.ON=1;
```

```
	i=0;
	j=0;

	while(1){

	}

}
```