ECE 3720

PWM (Pulse Width Modulation)

Section 2

Brandon Wider

Date Performed: March / 10 / 2020

Lab 8

**ABSTRACT:**

This lab discusses how to use the PIC32’s comparator in PWM mode to produce a signal to power a DC motor at different speeds.

**INTRODUCTION:**

The purpose of this lab is to program the microcontroller to power a DC motor with different voltage levels, causing the motor to operate at different speeds. The PIC32 doesn’t have a PWM, so we use the comparator to act like a PWM. The speed is controlled by changing the duty cycle of the PWM, which changes the average voltage seen by the motor. A debouncer is used to cycle through different duty cycles values.

**EXPERIMENTAL PROCEDURES:**

* Wire circuit using the crude block diagram
* For the interrupt part of the code, cycle through to the next speed by writing the new duty cycle to the OSxRS register.

**RESULTS and DISCUSSION:**

When done correctly, motor will initially be off, and then each time the debouncer switch is pressed the duty cycle is cycled through 0%, 25%, 50%, 75%, and 100%. The duty cycle determines the average voltage received by the DC motor, which determines the motors speed. Even though the PIC32 doesn’t have a PWM, the comparator can be used to act just like it. This is done by switching the signal on and off whenever the clock counter is equal to the duty cycle.

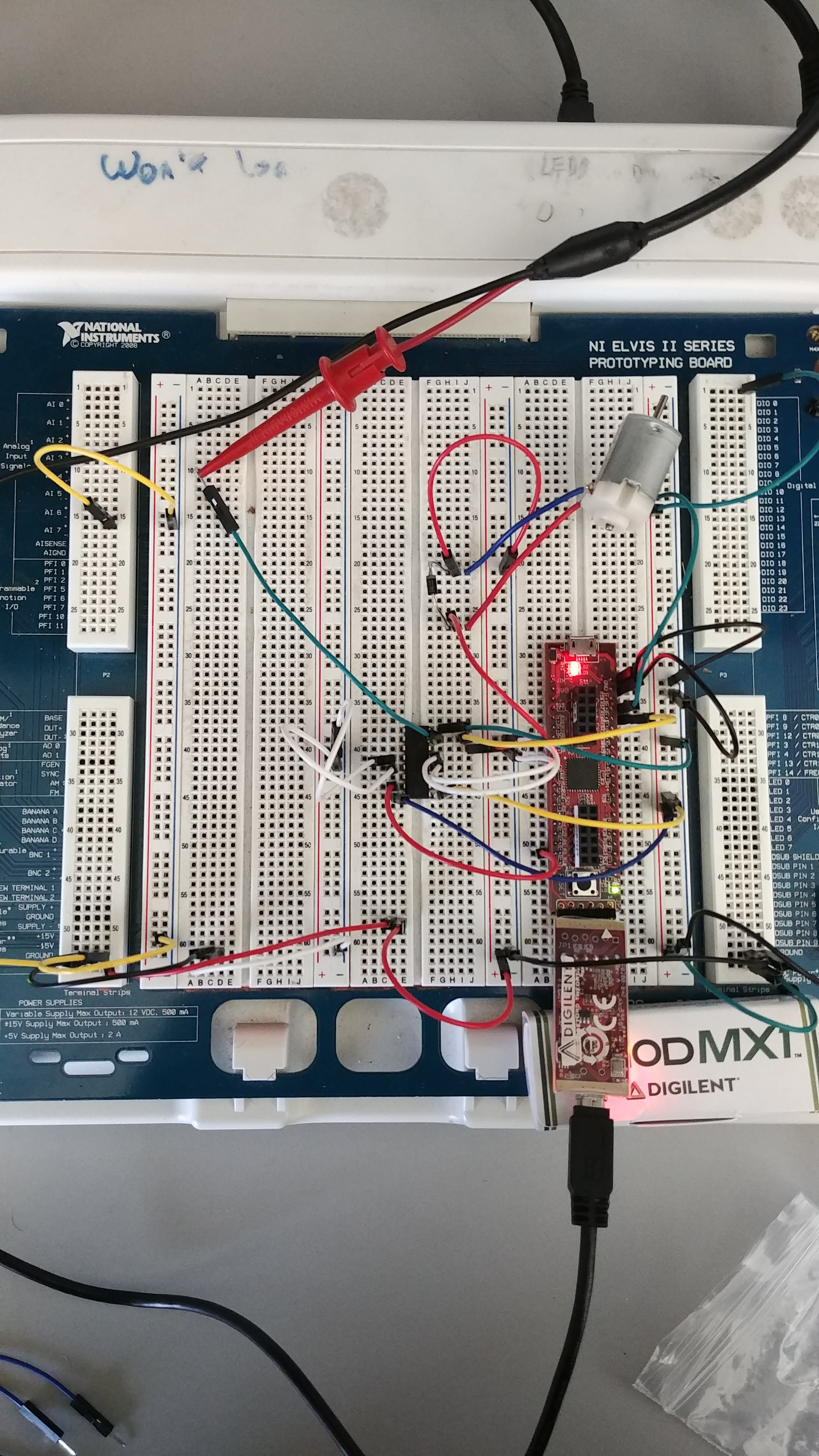
**CONCLUSSION:**

This lab shows how to implement a PWM in the PIC32 by using the comparator, and how to change the duty cycle of the PWM while the program’s running.

**REFERENCES:**

Clemson University’s ECE 372 Lab 8 Powerpoint.

**FIGURES AND TABLES:**



**CODE:**

#include <plib.h>  
  
int count;  
void \_\_ISR(3)debuncer() {  
 count += 25;  
 if (count > 100)  
 count = 0;  
 OC4RS = count;  
 IFS0bits.INT0IF = 0;  
}  
  
main() {  
 PPSOutput(3, RPB6, OC4);  
 INTEnableSystemMultiVectoredInt();  
 INTCONbits.INT0EP = 1; //Enable INT0 to happen  
 IPC0bits.INT0IP = 1; //Set priority control  
 IEC0bits.INT0IE = 1; //Enable interrupt 0  
 IFS0bits.INT0IF = 0; //Sets up interrupt flag status  
  
 int count = 0;  
 T2CON = 0;  
 T2CONbits.TCS = 0;  
 T2CONbits.TCKPS = 1;  
 T2CONbits.TGATE = 0;  
 TMR2 = 0;  
 PR2 = 100;  
 T2CONbits.ON = 1;  
   
 TRISBbits.TRISB0 = 0; //LED 1  
 TRISBbits.TRISB1 = 0; //LED 2  
 TRISBbits.TRISB2 = 0; //LED 3  
 TRISBbits.TRISB3 = 0; //LED 4  
   
 TRISBbits.TRISB7 = 1;  
 TRISBbits.TRISB6 = 0;  
   
 OC4RS = 50;  
 OC4CONbits.ON = 0;  
 OC4CONbits.OC32 = 0;  
 OC4CONbits.OCM = 7;  
 OC4CONbits.OCTSEL = 0;  
 OC4CONbits.ON = 1;  
 while (1){  
 }  
}