**PUSLE WIDTH MODULATION**

**Aayahna M. Herbert**

**November 7, 2017**

**ECE 3720 Section 007**

**Microcontroller Interfacing Laboratory**

**Abstract:** A circuit was designed that receives an interrupt from a push button sends it as an output to the L293DNE Motor Driver IC, then going to the motor. Within the microcontroller, an interrupt was set up so that every time the push button is pressed, the spin speed of the motor will change.

**INTRODUCTION**

The purpose of this lab was to set up an Output Compare Module in PWM mode to drive a motor using the Motor Driver IC and use an external interrupt to modify the Duty Cycle. Within the microcontroller, an interrupt was set up through the push button that changes the speed the motor is spinning.

**EXPERIMENTAL PROCEDURES**

The equipment used include one PIC32 MC, NI-ELVIS II board, a push button, an L293DNE motor chip, and a motor. The first thing to be done, to avoid forgetting it later, is to set pins B7 and B8 as the input from the push button to the microcontroller and the output from the microcontroller to the motor gate chip L293DNE. Because we are dealing with one interrupt signal, the register INT0 should be enabled, set to high direction, be interrupted on a high value, and have a priority of one by using INTCON, IECx, IFSx, and IPCx. In order to use PWM as an output from the microcontroller, the PPSOutput() function is used with RPB8 being the output and OC2 being the input. Because PWM does not have its own register, it must be accessed through the Output Compare module and Timer 2; the settings for Timer 2 should all be set to zero using TGATE, TCS, T32, and TCKPS. In order for the mode of the OC to be in PWM, the mode must be set to either 6 or 7 using OCM; in terms of the Duty Cycle, the timer and OCxRS should be set to zero while PRx is set to 100. To let the microcontroller know that both the timer and the Output Compare to be used and their settings have been modified, the last bit that should be added to the main function is turning Timer 2 and Output Compare Module 2 on by using TxCON and OCxCON. In addition to the interrupt, it must also be globally enabled by adding INTEnableSystemMultiVectoredInt() and INTEnableInterrupts() to the main function. With the main function comes the ISR function for the specific register number in use, void \_\_ISR(3) interruption(void). Inside of these functions goes the commands for what should happen to the motor each time the push button is used. For this function, we want the Duty Cycle of the motor to increment by 25% while staying in the range of 0% to 100%. In terms of wiring, make sure the power the chip is receiving is +5V, a resistor of 220Ω is connected to the push button’s power, and the motor chip is using a diode.

**RESULTS**

After the circuit was hooked up and ready to be tested, the program detected no compiling errors when ran so the board could then be tested. Each time the button was pushed, the motor’s spin speed increased by a factor of 25; when the speed was at 100, the motor cycled back to 0 restarting the speed.

**DISCUSSION**

Problems arose initially when the speed of the motor did not change every time the push button was pressed. The code and wiring were second-checked by the TA and the wires all came from a new pack recently purchased to avoid the possibility of dead wires being used and affecting the lab results. The source of the error was the incorrect flag was being used in the interruption function. After it was changed to the correct flag, the motor began to run as anticipated.

**CONCLUSIONS**

In general, the take-away of this lab was to learn how to set up Output Compare in PWM mode and use the motor driver chip.

**FIGURES AND TABLES**

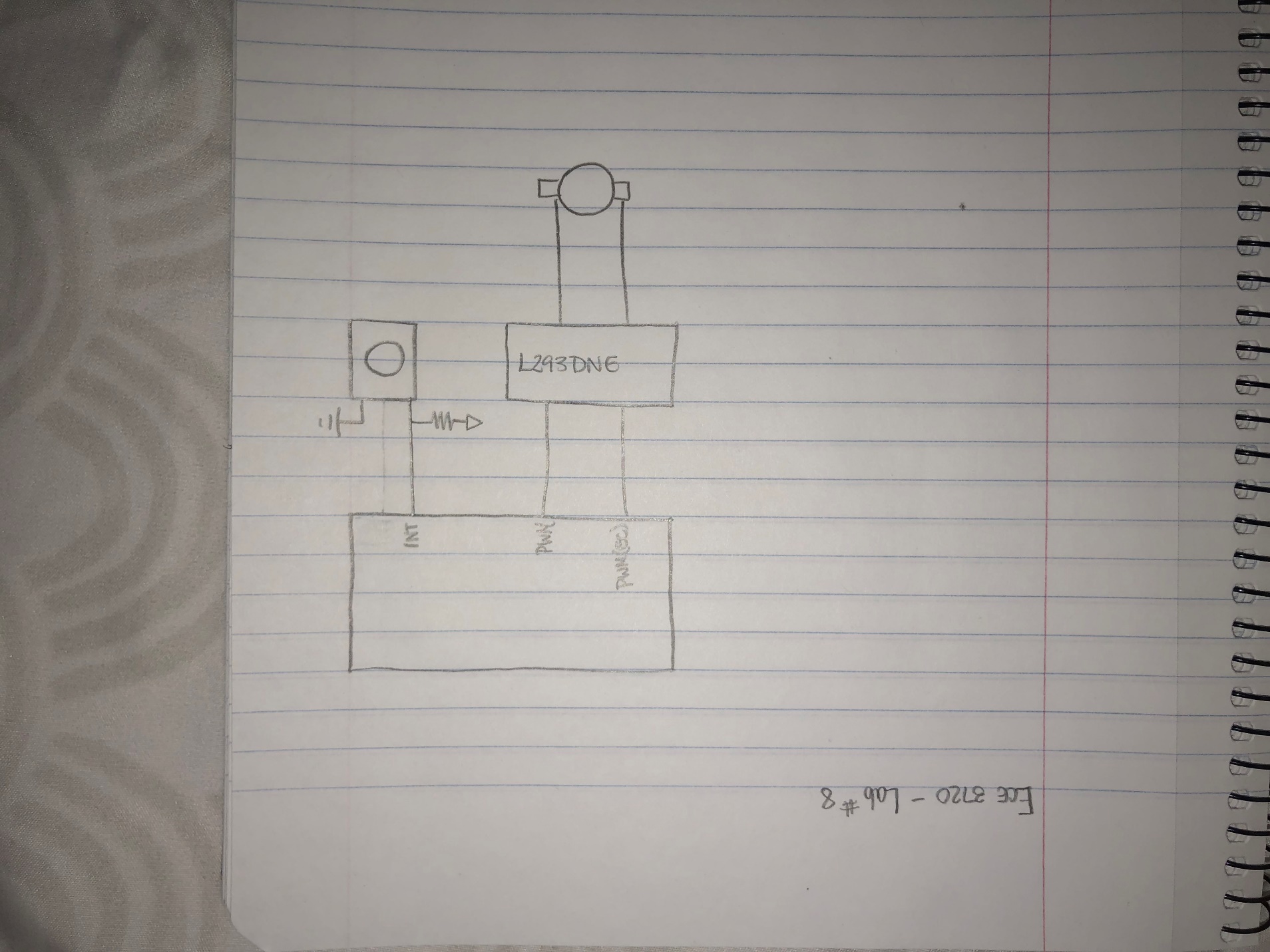


Figure 1: Circuit Schematic

**CODE**

#include<plib.h>

//Interrupt Function

void \_\_ISR(3) Interruption(void)

{

if(OC2RS > 100)

OC2RS = 0;

else if(OC2RS <= 100 && OC2RS >= 0)

OC2RS += 25;

IFS0bits.INT0IF = 0;

}

main()

{

INTEnableSystemMultiVectoredInt();

INTEnableInterrupts();

/\* Interruption 0 for Push Button \*/

TRISBbits.TRISB7 = 1;

INTCONbits.INT0EP = 1;

IEC0bits.INT0IE = 1;

IFS0bits.INT0IF = 1;

IPC2bits.INT0IP = 1;

/\* For PWM/OC2 Output \*/

TRISBbits.TRISB8 = 0;

PPSOutput(2,RPB8,OC2);

T2CON = 0;

T2CONbits.TGATE = 0;

T2CONbits.TCS = 0;

T2CONbits.T32 = 0;

T2CONbits.TCKPS = 0;

TMR2 = 0;

PR2 = 100;

OC2RS = 0;

OC2CONbits.OCM = 6;

OC2CONbits.OCTSEL = 0;

T2CONbits.ON = 1;

OC2CONbits.ON = 1;

while(1) //Run Continuously

{

}

}