CPE301 – SPRING 2019

Design Assignment Midterm 2

Student Name: Tanner Tindall

Student #: 8000733733

Student Email: tindat1@unlv.nevada.edu

Primary Github address: <https://github.com/TannerTindall51>

Directory:

<https://github.com/TannerTindall51/tindalltannerm_submission/tree/master/Midterm>2

You’ll use the ADC, and PWM/CCP Module of the ATmega328/p to set and determine the speed of the DC Motor.

1. Use the motor driver, program the ATMega to drive the geared DC motor in CW and CCW direction for a given PWM.

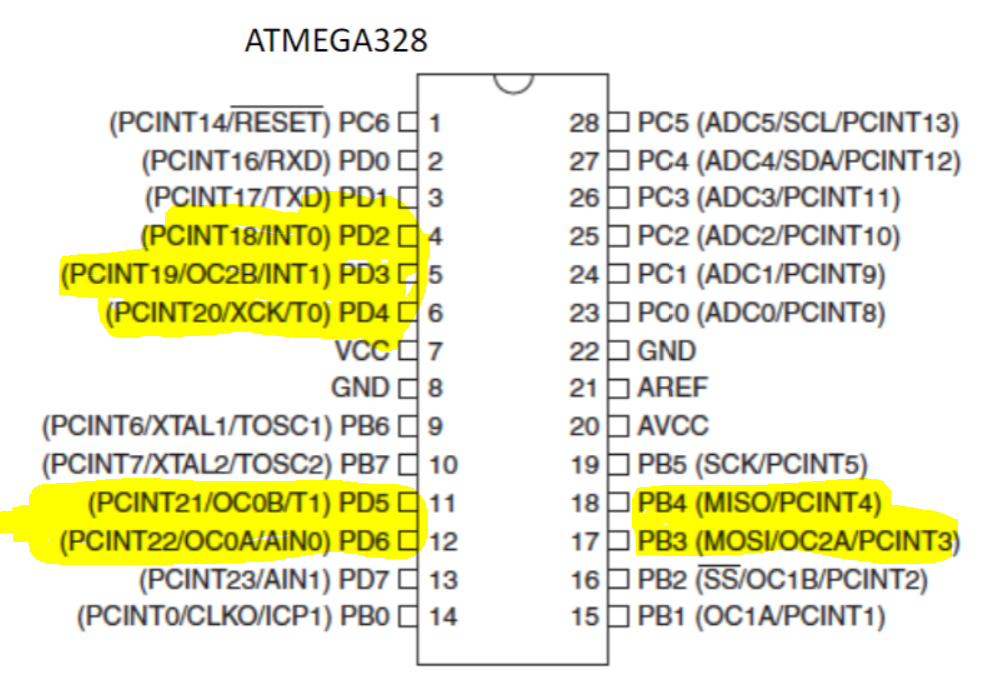
2. Using the Potentiometer connected to ADC0, translate the ADC value to PWM value/ speed of the motor. Verify the operation.

3. Using the CCP capture pin of PWM1, in mode 1x and 2x determine the speed of the DC Motor for a set ADC Pot value/position.

4. Using CCP capture and interrupt (mode 4x), determine the speed of the DC Motor for a set ADC Pot value/position.

1. **COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS**

* Atmel Studio 7.0 (Assembler, Simulator, & Debugger)
* Atmega328PB-Xmini
* Micro USB
* DC Motor
* Md80a Motor Driver
* Potentiometer



1. **INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1/A**

|  |
| --- |
|  |

//

//CPE301 - Midterm 2

//Tanner Tindall

//

#define *F\_CPU* 16000000UL

#define BAUD 9600

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/setbaud.h>

#include <stdlib.h>

#include <stdio.h>

#include <util/delay.h>

#include <stdbool.h>

#include <util/delay.h>

volatile *uint16\_t* RPMx1\_Value;

volatile *uint16\_t* RPMx4\_Value;

static char array[20];

volatile bool delay1;

volatile bool delay2;

void initializeUART()

{

DDRD |= 0x01; //set transmit port direction

UBRR0 = 103; //BAUD rate

UCSR0C |= (0<<UPM01) | (0<<UPM00) | (1<<UCSZ01) | (1<<UCSZ00); //disable parity and set to asynchronous

}

void startADC()

{

DDRD &= ~(1<<0); //set PD0 to input

ADMUX = (1<<REFS0); //set Vref and set ADC0 as input

ADCSRA = (1<<ADEN) | (1<<ADATE) | (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0); //enable ADC and set prescalar to 128

ADCSRB = (1<<ADTS0); //set analog compare match

}

*uint16\_t* adcRead()

{

ADCSRA |= (1<<6); //enable ADC conversion

while (!(ADCSRA & (1<<4))); //waits for conversion to complete by ADIF flag

ADCSRA |= (1<<4); //reset when complete

return ADC; //return value once complete

}

void USART\_tx\_transmit()

{

UBRR0 = 103;

UCSR0B |= (1 << TXCIE0) | (1<<TXEN0); //enable data transmission

DDRD |= 0x01; //transmit through PD1

}

void USART\_tx(char\*data) //outputs data to terminal

{

while((\*data != '\0')) //loop until all data is emptied

{

while(!(UCSR0A & (1 << UDRE0))); //wait until data register in emptied

UDR0 = \*data; //once emptied, import into data register UDR0

data++; //increments pointer position for data

}

}

void adcValue()

{

*uint16\_t* x = adcRead();

float adcVal = ((x)\*(110.0/1023.0));

*snprintf*(array,sizeof(array), "%f\r\n", adcVal);

USART\_tx(array);

return;

}

void setPorts()

{

DDRB |= (1<<3) | (1<<4);//set output ports used for motor direction

PORTB &= ~(1<<3);

PORTB |= (1<<4);

DDRD &= ~(1<<2); //set INT0

PORTD &= ~(1<<2);

DDRD &= ~(1<<3); //set INT1

PORTD &= ~(1<<3);

DDRD |= (1<<4); //input capture 1

PORTD &= ~(1<<4);

DDRD |= (1<<5); //input capture 2

PORTD &= ~(1<<5);

}

int main()

{

startADC(); //initialize functions above

USART\_tx\_transmit();

initializeUART();

initializeTimer0();

initializeTimer4();

setPorts();

sei();

while(1) //display RPMs to UART

{

USART\_tx("RPM:");

RPMx1();

USART\_tx("\n");

USART\_tx("RPM:");

RPMx2();

USART\_tx("\n");

USART\_tx("RPM:");

RPMx4();

USART\_tx("\n");

*\_delay\_ms*(10);

}

}

void RPMx1()

{

initializeTimer3(); //run counter

initializeTimer4(); //run delay (100ms)

delay1 = true; //set flag for while loop below

while(delay1); //collect data until delay is finished

PORTD &= ~(1<<4); //reset interrupts

PORTD &= ~(1<<5);

return (float)(RPMx4\_Value \* (60.0/96.0)); //convert given value to rpm

}

void RPMx2()

{

initializeTimer3(); //run counter

initializeTimer4(); //run delay (100ms)

EIMSK |= (1<<INT1); //set INT1 to handle interrupt for data collection

EICRA |= (1<<ISC01) ; //enable external interrupt on falling edge

delay1 = true; //set flag for while loop below

while(delay1); //collect data until delay is finished

disableTrigger(); //disable external interrupt

PORTD &= ~(1<<4); //reset interrupts

PORTD &= ~(1<<5);

return (float)(RPMx4\_Value \* (60.0)/(192.0)); //convert given value to rpm

}

void RPMx4()

{

initializeTimer1(); //run counter 1

initializeTimer3(); //run counter 2

initializeTimer4(); //run delay (100ms)

EIMSK |= (1<<INT1) | (1<<INT0); //set INT0/1 to handle interrupt for data collection

EICRA |= (1<<ISC11) | (1<<ISC00); //enable external interrupt on falling edge

delay1 = true; //set flags

delay2 = true;

while(delay1 || delay2); //collect data until delay is finished

disableTrigger(); //disable external interrupt

PORTD &= ~(1<<4); //reset interrupts

PORTD &= ~(1<<5);

return (float)((RPMx1\_Value + RPMx4\_Value) \* (60.0)/(384.0)); //convert given value to rpm

}

void initializeTimer0() //PWM timer

{

TCCR0A |= (1<<COM0A1) | (1<<WGM00); //non-inverting, Fast PWM

TCCR0B |= (1<<WGM02) | (1<<CS02);

TIMSK0 |= (1<<TOIE0); //enable interrupts

TCNT0 = 0x00;

OCR0A = 0x00;

}

void initializeTimer1() //RPMx4

{

DDRD &= ~(1<<7);

TCNT1 = 0x00;

TCCR1B |= (1<<FOC4B) | (1<<CS32) | (1<<CS31) | (1<<CS30); //set CTC mode and prescaler of 64

TIMSK1 |= (1<<ICIE1);

}

void initializeTimer3() //RPMx1 & RPMx2 & RPMx4

{

DDRD &= ~(1<<6);

TCNT3 = 0x00; //initialize timer3

TCCR3B |= (1<<FOC4B) | (1<<CS32) | (1<<CS31) | (1<<CS30); //set to rising edge trigger

TIMSK3 |= (1<<ICIE3); //set input compare interrupt

}

void initializeTimer4() //ADC sampling delay timer

{

TCNT4 = 0x00; //initialize timer

TCCR4B |= (1<<WGM42) | (1<<CS41) | (1<<CS40); //set CTC mode and prescaler of 64

TIMSK4 |= (1<<OCIE4A); //set output compare interrupt

OCR4A = 24999; //100ms in cycles

}

ISR(INT0\_vect)

{

motorDirection(); //motor changes direction on interrupt

if(delay1) //if collecting data

{

TCCR3B ^= (CS31); //switch edge trigger

}

}

ISR(INT1\_vect)

{

if(delay2) //if collecting data

{

TCCR1B ^= (CS11); //switch edge trigger

}

}

ISR(TIMER0\_OVF\_vect) //PWM interrupt subroutine

{

ADCSRA |= (1<<ADIE); //set ADC interrupt

}

ISR(TIMER1\_CAPT\_vect) //RPMx4 interrupt subroutine

{

TIMSK1 &= ~(1U<<ICIE1); //reset interrupt

delay2 = false; //change delay condition to enable calculations

RPMx1\_Value = (float) ICR1; //store value from input data

delay1 = false; //change flag for calculation enable

}

ISR(TIMER3\_CAPT\_vect) //RPMx1 & RPMx2 & RPMx4 interrupt subroutine

{

TIMSK3 &= ~(1<<ICIE3); //reset interrupt

delay1 = false; //change delay condition to enable calculations

RPMx4\_Value = (float) ICR3; //store value from input data

delay2 = false; //change flag for calculation enable

}

ISR(TIMER4\_COMPA\_vect)

{

TIMSK3 &= ~(TOIE0);

PORTD |= (1<<4); //set interrupt

if(delay2)

{

PORTD |= (1<<5); //set interrupt for additional timer

}

}

void disableTrigger()

{

EIMSK = 0x00; //disable INT1 interrupt

EICRA = 0x00; //disable external interrupt

}

void motorDirection() //invert current direction of motor

{ //by reversing H-Bridge config

PORTB ^= (1<<3);

PORTB ^= (1<<4);

}

1. **DEVELOPED MODIFIED CODE OF TASK 2/A from TASK 1/A**

N/A

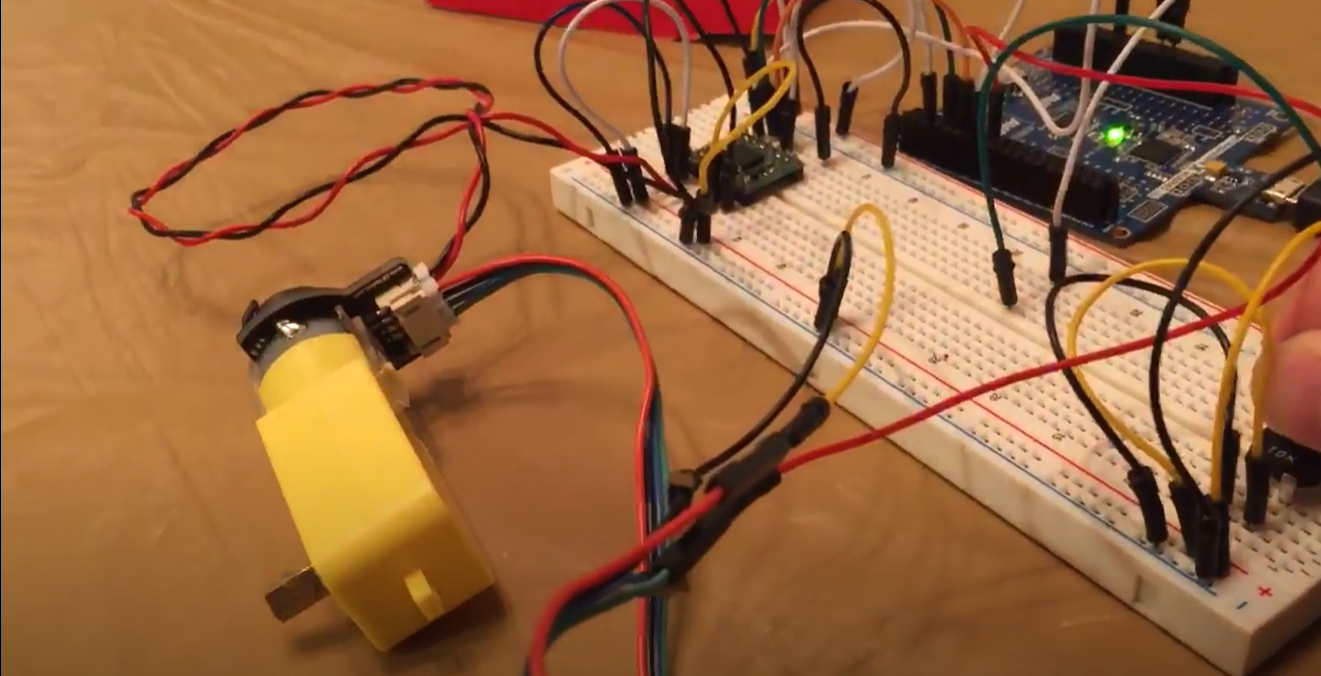
1. **SCHEMATICS**

N/A

1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

N/A

1. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**



1. **VIDEO LINKS OF EACH DEMO**

Demo Video: https://youtu.be/6lWBPpY6V\_Y

1. **GITHUB LINK OF THIS DA**

<https://github.com/TannerTindall51/tindalltannerm_submission/tree/master/Midterm>2

**Student Academic Misconduct Policy**

<http://studentconduct.unlv.edu/misconduct/policy.html>

“This assignment submission is my own, original work”.

Tanner Tindall