MILESTONE 4B - Interfacing

March 4, 2024

MECH 458 - B01 Group 1

Mckinlay, Samantha (V00954147)

Berezowska, Kiara (V00937549)

Documented Codes

```
/*
        Course          : UVic Mechatronics 458
        Milestone       : 4B
        Title           : INTERFACING

        Name 1:     Mckinlay, Samantha          Student ID: V00954147
        Name 2:     Berezowska, Kiara           Student ID: V00937549

        Description: Program for running a simple brushed DC motor. Motor can operate in both CW and
CCW at different speeds.

        Direction change is controlled by a switch and a potentiometer circuit has been implemented for
the speed.

        There is also a kill switch programmed to remove any signal that will cause the motor to move,
when pressed.
*/


#include <avr/io.h>
#include <avr/interrupt.h>
#include "lcd.h"
#define disable 0 //sets ENA and ENB to low disables driver
```

```
#define brake  0b01111000 //set INA (PB3),INB(PB4),DIAGA/ENA(PB5),and DIAGB/ENB(PB6) to high
#define CW 0b01101000
#define CCW  0b01110000



// define the global variables that can be used in every function ==========
volatile unsigned char ADC_result;
volatile unsigned int ADC_result_flag;
void mTimer(int count);
void PWM();

int main(void)
{
	CLKPR = 0x80;
	CLKPR = 0x01;

	TCCR1B = _BV(CS11);
	DDRA = 0x00; //sets PORTA to input
	DDRF = 0x00; //sets PORTF to input
	DDRC=0xFF; //sets PORTC to output
	PWM();
	//Initialize LCD module
	InitLCD(LS_BLINK|LS_ULINE);

	//Clear the screen
	LCDClear();
	//LCDWriteString("Congrats ");
	cli(); // disable all of the interrupt ===========================
	// config the external interrupt ======================================
	EIMSK |= (_BV(INT2)); // enable INT2
	EICRA |= (_BV(ISC21)); // falling edge interrupt
```

```
	EIMSK |= (_BV(INT3)); // enable INT3
	EICRA |= (_BV(ISC31) | _BV(ISC30)); //rising edge interrupt
	// config ADC ===========================================================
	// by default, the ADC input (analog input is set to be ADC0 / PORTF0
	ADCSRA |= _BV(ADEN); // enable ADC
	ADCSRA |= _BV(ADIE); // enable interrupt of ADC
	ADMUX |= _BV(ADLAR) | _BV(REFS0); //REFS0 to 1 selects voltage for ADC (AVCC with external capacitor at AREF pin)
	//ADLAR to 1 left adjusts the result - to be able to read 8 bit mode (shoves it to the high bite)
	//set the PORTC as output to display the ADC result ==================
	//LCDWriteString("Congrats ");
	DDRC = 0xff;

	//set the PORTB as output to display the ADC result ==================
	DDRB = 0xff;
	PORTB = CW;

	// sets the Global Enable for all interrupts ==========================
	sei(); //enables all of the interrupts
	// initialize the ADC, start one conversion at the beginning ==========
	ADCSRA |= _BV(ADSC); //starts conversion and grabs value
	//set up driver to turn the motor (still needs to be done)

  while (1)
  {
		mTimer(20);
		if (ADC_result_flag)
		{
			//PORTC = ADC_result; //output data on LED
			ADC_result_flag = 0x00; //flag is rest to 0 therefore, it fails the if statement
			//write to LCD - display the ADC value 1st line forward or revers, 2nd percentage of speed, 3rd ADC result
```

```
                    if(PORTB == CW){
                        LCDClear();
                        LCDWriteStringXY(0,0,"FORWARD");
                        LCDWriteIntXY(0,1,ADC_result*100/255,3);
                        LCDWriteStringXY(3,1,"%");
                        LCDWriteIntXY(13,1,ADC_result,3);
                        }else{
                        LCDClear();
                        LCDWriteString("REVERSE");
                        LCDWriteIntXY(0,1,ADC_result*100/255,3);
                        LCDWriteStringXY(3,1,"%");
                        LCDWriteIntXY(13,1,ADC_result,3);
                        }
                }
                ADCSRA |= _BV(ADSC);//starts conversion and grabs value

    }
}// end main

// change direction
ISR(INT3_vect){
    mTimer(20); //debounce
    int prev = PORTB;
    //brake DC motor to Vcc; need to set INA (PB3), INB(PB4), DIAGA/ENA(PB5), and
DIAGB/ENB(PB6) to high
    PORTB = brake; //set INA (PB3),INB(PB4),DIAGA/ENA(PB5),and DIAGB/ENB(PB6) to high
    mTimer(5);
    //change moving direction;
    if(prev == CW){
        PORTB = CCW; //set INB (PB4) to low
        }else{
        PORTB = CW; //set INB (PB4) to high
```

```
        }
}

// sensor switch: Active HIGH starts AD converstion =======
// kill switch
ISR(INT2_vect)
{
        mTimer(20); //debounce
        //brake DC motor to Vcc; need to set INA (PB3), INB(PB4), DIAGA/ENA(PB5), and
DIAGB/ENB(PB6) to high
        PORTB = brake; //set INA (PB3),INB(PB4),DIAGA/ENA(PB5),and DIAGB/ENB(PB6) to high
        mTimer(5);
        //disable drive by setting EA and EB make zeros use #define
        PORTB = disable;
        LCDClear();
        LCDWriteString("KILL ACTIVATED");//write to display and flash kill switch has been activated
        while(1){
                //killing program
        }
}
// the interrupt will be trigured if the ADC is done ========================
ISR(ADC_vect)
{
        ADC_result = ADCH; //global variable for high bite (where the data is)
        OCR0A = ADC_result;//duty cycle
        ADC_result_flag = 1;
}

void mTimer (int count){
  /***
    Setup Timer1 as a ms timer
          Using polling method not Interrupt Driven
```

```
***/

int i;

i = 0;

//TCCR1B |= _BV (CS11);  // Set prescaler (/8) clock 16MHz/8 -> 2MHz
/* Set the Waveform gen. mode bit description to clear
  on compare mode only */
TCCR1B |= _BV(WGM12);

/* Set output compare register for 1000 cycles, 1ms */
OCR1A = 0x03E8;

/* Initialize Timer1 to zero */
TCNT1 = 0x0000;

/* Enable the output compare interrupt */
//TIMSK1 = TIMSK1 | 0b00000010;

/* Clear the Timer1 interrupt flag and begin timing */
TIFR1 |= _BV(OCF1A);

/* Poll the timer to determine when the timer has reached 1ms */
while (i < count){
  if((TIFR1 & 0x02) == 0x02){

        /* Clear the interrupt flag by WRITING a ONE to the bit */
        TIFR1 |= _BV(OCF1A);
        i++;
        }
```

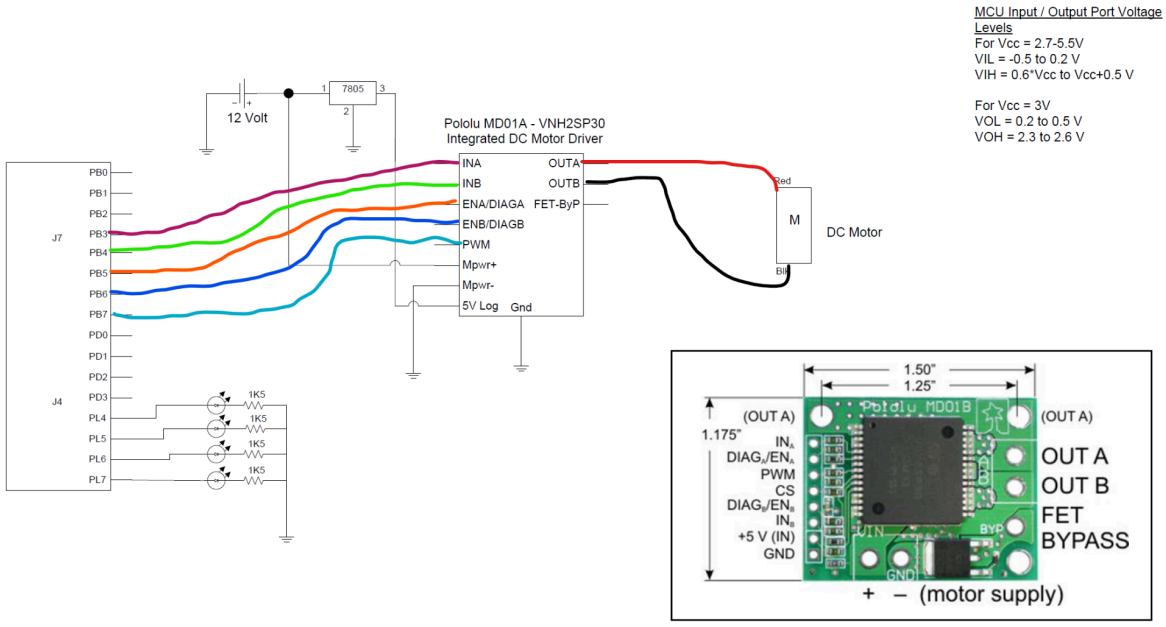
```
        }
  return;
}  /* mTimer */


void PWM(){
        TCCR0A |= _BV(WGM01)|_BV(WGM00); //selecting Fast PWN mode 3
        //TIMSK0 |= _BV(OCIE0A); //enable output compare interrupt for timer0
        TCCR0A |= _BV(COM0A1);//set compare match output mode to clear and set output compare A when timer reaches TOP
        TCCR0B |= _BV(CS01);//sets prescale factor to 8
        OCR0A = 0x80;//default duty cycle
}
```

Lab Assignment Solutions

Worksheet 2:

Appendix A: UVic Mechatronics Milestone 4 – Interfacing Worksheet 2



IN_A	IN_B	$DIAG_A/EN_A$	$DIAG_B/EN_B$	OUT_A	OUT_B	CS	Operating mode
1	1	1	1	H	H	High Imp.	Brake to V_{CC}
1	0	1	1	H	L	$I_{SENSE}=I_{OUT}/K$	Clockwise (CW)
0	1	1	1	L	H	$I_{SENSE}=I_{OUT}/K$	Counterclockwise (CCW)
0	0	1	1	L	L	High Imp.	Brake to GND

Supplementary Information

No supplementary information.