CEEN 1060-003

Microprocessor Applications

Implementing Motor Functions

Austin Koch

Emily Helzer

Department of Computer and

Electronics Engineering

Peter Kiewit Institute

University of Nebraska

Due Date: 05/04/2015

**Introduction**

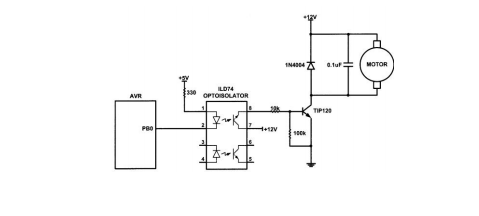
The purpose of this lab is to introduce the concept of motors as an auxiliary component to the AVR microcontroller. The lab demands code that will initialize and implement code for a motor circuit attached to the microcontroller. It serves as the primary step into seeing how the microcontroller can function as a brain for a real-world peripheral tool.

This report will consist of a brief background section to cover any necessary information followed by the procedure followed during the lab session. Results will come after followed by answers to the additional questions given. The report will then conclude and any addition information, such as the code, will be attached after in an appendix section. This lab took approximately 4 hours to complete, start to finish.

**Background**

This lab implements a simple motor device that operates through the transition of electrical energy to Magnetic, which in turn moves the coil motors within the device to generate motion. In this case, the blades of a fan are moved at varying speeds based upon the amount of electricity running through them – a function that is set in the program of the microcontroller.

The circuit was constructed implementing an ILD74 Optoisoloator as a buffer to manage the voltage inputs and outputs between the avr and the motor,



The AVR will input a signal to the optoisolator to change the pulse of electricity to the motor, which will in turn allow the AVR to control the speed of the fan on the motor.

This lab will serve as in introduction to using a network of components that function based on logic (code) as opposed to simple analogue connections seen in previous lab work.

**Procedure**

For Activity 1 we first needed to construct the circuit observed in the background section of this report. We imported the avr io files as well as the interrupt and delay files using the #include <avr/io.h>, #include <stdint.h>,

#include <util/delay.h> instructions. The materials were provided for us and assembly went smoothly. Meanwhile, we constructed the code necessary to make the motor rotate counterclockwise while PORT E pin 0 is held.

In our code, we began by setting port E as an input and Port B as a input. Then we set Port B to 0x80, which kept the motor off until it was set. Using an unsigned variable, masking all but the first bit, which takes on the value of PINE, we created a switch statement that poll the value of z. If z is set than the motor is turned on, if cleared, the motor is turned on. A while loop is used to continually poll that bit.

Activity 2 works much the same. We ended up using the polling method as opposed to the Timer interrupt method. We once again set PortE as inputs and PORTB as inputs and initialized an unsigned char that will poll PINE, mask all bits other than the first two, and will enter a switch statement that will change the duty cycle on the outputted wave signal depending on the switches being pressed. When none are being pressed, the device will output a 25% duty cycle wave, if PE0 is pressed, a 50% duty cycle will be outputted, if PE1 is pressed then a 75% duty cycle will be outputted, and if both the switches are pressed a duty cycle of 100% will be outputted. The program will then keep polling the bits on PORTE to determine the outputted duty cycle. Once we finished the process we had the TA sign us off and then cleaned up.

**Results**

The results of this lab were positive and accurate to our hypothesis. We observed the correct responses from out switches and viewed a real life application of previous coursework into a real working machine. The results of this lab demonstrate physically how the implementation of the motor and osoisolator work. Our results would only have been more impressive should we have completed the assignment with Timer interrupts.

**Additional Questions**

**a) The optoisolator serves as a buffer between the AVR and peripheral motor. It implements a phototransistor that operates depending on the light generated by the component. This device allows the AVR to effectively communicate with the motor by heavily reducing noise.**

**b) The DC motor is turned on by setting the last bit on Port B and off by clearing it.**

**c) The speed of the motor was adjusted by changing the driving signal’s duty cycle. When at 25% the device is working at much greater speeds than when the duty cycle becomes 50, 75, and 100%. Each case, determined by button presses, exhibits a different duty cycle.**

**d) Pulse Width Modulation is a technique of getting analogue results with digital means. It creates a square wave and can be manipulated to change the form and function of the wave.**

**e) Polling involves many more instructions and requires more time to constantly monitor. In addition, the use of delays isn’t as perfect as using the PWM’s inherent functions. Delays will result in rough estimations and thus serves as a disadvantage.**

**Conclusion**

This lab went smoothly. Most of the code had been generated before class time. Due to this we were able to finish the lab very quickly, however we decided to spend the remaining hour and a half on trying to make the timer interrupts working. This was not a successful attempt, however we kept the code used for the polling method and got signed off for what we’d accomplished earlier.

**Appendix**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Name: Emily Helzer and Austin Koch

//Lab: Bonus Lab

//Activity: 1

//Description: Run a DC motor clockwise continuously

// when the switch on PE0 is pressed

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <avr/io.h>

#include <stdint.h>

#include <util/delay.h>

//------------prototypes------------------------//

void delay\_us(unsigned int d);

//--------------functions------------------------//

void delay\_us(unsigned int d)

{

\_delay\_us(d);

}

//=========================Main=====================================

int main(void)

{

unsigned char z;

DDRE= 0x00; // PortE is input

DDRB= 0xFF; // PB7 is output

PORTB = 0x80; // Keeps motor off

while(1)

{

z = ~PINE; // Compliment data since

// switches are active low

z = z & 0b00000001; // Mask all bits except bit 0

// Turns motor on if switch is pressed

switch (z)

{

case(0):

{

PORTB = 0x80; // Keeps motor off

break;

}

case(1):

{

PORTB = 0x00 // Turns motor on

break;

}

}

}

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Name: Emily Helzer and Austin Koch

//Lab: Bonus Lab

//Activity: 2

//Description: Controls the speed of the DC motor

// depending on the switch on PE0 and

// the switch on PE1

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <avr/io.h> /\* Access I/0 ports, register name, etc. \*/

#include <stdint.h> /\* Access to 'uint8\_t' types, etc. \*/

#include <util/delay.h> /\* Access to '\_delay\_us()' \*/

//------------prototypes------------------------//

void delay\_us(unsigned int d);

//--------------functions------------------------//

void delay\_us(unsigned int d)

{

\_delay\_us(d);

}

//=========================Main=====================================

int main(void)

{

unsigned char z;

DDRE= 0x00; // PortE is input

DDRB= 0xFF; // PB7 is output

while(1)

{

z = ~PINE; // Compliments data since

// switches are active low

z = z & 0b00000011; // Masks all bits except bit 0 and 1

// Performs task based on what Pin 0 and Pin 1 are

switch (z)

{

case(0): // If both buttons aren't pressed send

{ // with 25 percent duty cycle

PORTB = 0x80; // Keep motor off

\_delay\_us(189); // Delay

PORTB = 0x00; // Turn motor on

\_delay\_us(63); // delay

break;

}

case(1): // 50 percent duty cycle

{

PORTB = 0x80; // Keep motor off

\_delay\_us(126); // delay

    PORTB = 0x00; // turn motor on

    \_delay\_us(126); // delay

break;

}

case(2):

{ // 75 percent duty cycle

PORTB = 0x80; // Keep motor off

\_delay\_us(63); // delay

PORTB = 0x00; // Turn motor on

\_delay\_us(189); // delay

break;

}

case(3):

{ // 100 percent duty cycle

PORTB = 0x00; // Turn motor on

\_delay\_us(255); // delay

break;

}

}

}

return 0;

}