ECE 375 Lab 2

Introduction to AVR Tools

Lab Time: Friday 4-6

Aaron Vaughan
Bradley Heenk

TA Signature

1 Introduction

The purpose of this lab is to compile a program written in c-language that will implement the bumpbot behavior. We will learn how to interact with the inputs and outputs on the ATmega128 board. I chose to write some subroutines to accomplish the turning commands and the bump command used for the challenge code. We started with the example code and cut and pasted much of the command procedures from that. We also had to look at the first lab that ws coded in avr to determine how to access the inputs (whiskers) from PORTD. It was unclear how this PORTD was connected to the physical buttons but Bradley found that by tracing out the wires on the printed board that they did indeed follow directly from the PORTD connector.

2 Internal Register Definitions and Constants

PORT MAP
Port B, Pin 4 -> Output -> Right Motor Enable
Port B, Pin 5 -> Output -> Right Motor Direction
Port B, Pin 7 -> Output -> Left Motor Enable
Port B, Pin 6 -> Output -> Left Motor Direction
Port D, Pin 1 -> Input -> Left Whisker
Port D, Pin 0 -> Input -> Right Whisker

3 Interrupt Vectors

Not used for this lab

4 Program Initialization

Text goes here

5 Main Program

```
int main(void)
{
        DDRB = 0b11110000;      // configure Port B pins for input/output
        PORTB = 0b11110000;     // set initial value for Port B outputs
        DDRD = 0b00000000;      // configure Port B pins for input/output
        PORTD = 0b11111111;     // set initial value for Port D inputs


        while (1) { // loop forever
```

```
                PORTB = 0b01100000;      // make TekBot move forward

                wskrL = PIND & 0x02;     // ignore all pins except pin 1
                wskrR = PIND & 0x01;     // ignore all pins except pin 0

                if((wskrL | wskrR) == 0x00) // if both wskr are depressed
                    turnR();

                else if (wskrR == 0x00) // if wskrR is depressed
                    turnR();      // execute the turn left command

                else if (wskrL == 0x00) // if wskrL is depressed
                    turnL();          // execute the turn right command
        }
}
```

6 A Subroutine

```
// a subroutine to back up for a short time
void bump()
{
        PORTB = 0b00000000;      // move backward
        _delay_ms(500);          // wait for 0.5 s
}

// a subroutine to turn left
void turnL()
{
        PORTB = 0b00000000;      // move backward
        _delay_ms(1000);         // wait for 1 s
        PORTB = 0b00100000;      // turn left
        _delay_ms(1000);         // wait for 1 s
};

// a sub routine to turn right
void turnR()
{
        PORTB = 0b00000000;      // move backward
        _delay_ms(1000);         // wait for 1 s
        PORTB = 0b01000000;      // turn right
        _delay_ms(1000);         // wait for 1 s
}
```

7 Stored Program Data

Text goes here

8 Additional Questions

1. 1. This lab required you to compile two C programs (one given as a sample, and another that you wrote) into a binary representation that allows them to run directly on your mega128 board. Explain some of the benefits of writing code in a language like C that can be "cross compiled". Also, explain some of the drawbacks of writing this way.

   The benefits of writing this in c-language is the clear advantages that a high-level language offers: we as the programmers do not need to worry about micro-operations.
   The disadvantages is that we no longer have as much control of what the CPU is actually doing. So if we cared how many clock cycles some operation requires, then using c-language to control the behavior of the device leaves us with uncertainty in this area.

2. 2. The C program you just wrote does basically the same thing as the sample assembly program you looked at in Lab 1. What is the size (in bytes) of your Lab 1 & Lab 2 output .hex files? Can you explain why there is a size difference between these two files, even though they both perform the same BumpBot behavior?

   My *Lab2.hex file is 958 Bytes while the *Lab1.hex file is 485 Bytes. When the c-language is translated into the machine language, there are some statements that translate into more micro-operations, even though the end result is the same. When translating from c-language to machine language there may be certain operations that just translate differently most likely due to a choice that the translation was coded so that it was easier to write.

9 Difficulties

I still do not fully understand the use of bitshifting that is demonstrated in the example code for week 1. I chose to use logical and with a constant value to force the input bus to use the desired pin location. This was the only problem my lab partner and I faced for Lab2.

10 Conclusion

Text goes here

11 Source Code

```
/*
 * Aaron_Vaughan_and_Bradley_Heenk_Lab2_sourcecode.c
 *
 * About: A program that will make the tekbot move like a roomba
 * Created: 10/12/2019 12:01:12 PM
```

```
 * Author : Aaron Vaughan and Bradley Heenk
 */


/*
This
PORT MAP
Port B, Pin 4 -> Output -> Right Motor Enable
Port B, Pin 5 -> Output -> Right Motor Direction
Port B, Pin 7 -> Output -> Left Motor Enable
Port B, Pin 6 -> Output -> Left Motor Direction
Port D, Pin 1 -> Input -> Left Whisker
Port D, Pin 0 -> Input -> Right Whisker
*/
#define F_CPU 16000000
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>

uint8_t wskrL = 0;
uint8_t wskrR = 0;

void bump()
{
        PORTB = 0b00000000;      // move backward
        _delay_ms(500);          // wait for 0.5 s
}
void turnL()
{
        PORTB = 0b00000000;      // move backward
        _delay_ms(1000);         // wait for 1 s
        PORTB = 0b00100000;      // turn left
        _delay_ms(1000);         // wait for 1 s
};

void turnR()
{
        PORTB = 0b00000000;      // move backward
        _delay_ms(1000);         // wait for 1 s
        PORTB = 0b01000000;      // turn right
        _delay_ms(1000);         // wait for 1 s

int main(void)
{
        DDRB = 0b11110000; // configure Port B pins for input/output
        PORTB = 0b11110000; // set initial value for Port B outputs
```

```
        DDRD = 0b00000000; // configure Port B pins for input/output
        PORTD = 0b11111111;      // set initial value for Port D inputs


        while (1) { // loop forever

                PORTB = 0b01100000;       // make TekBot move forward

                wskrL = PIND & 0x02;      // ignore all pins except pin 1
                wskrR = PIND & 0x01;      // ignore all pins except pin 0

                if((wskrL | wskrR) == 0x00) // if both wskr are depressed
                        turnR();     // turn right

                else if (wskrR == 0x00) // if wskrR is pressed
                        turnR(); // execute the turn left command

                 else if (wskrL == 0x00) // if wskrL is pressed
                        turnL();         // execute the turn right command
        }
}
```