ECE 375 Lab 2

C -> Assembler -> Machine Code -> TekBot

**Lab Time: Thursday 10-12**

*Daniel Barnes*

*Jeffery Noe*

# Introduction

This second lab focuses on the C workflow for the ATMega128 and recreates the Lab 1 program in C for comparison against a direct assembly implementation. It begins with a premade “DanceBot” routine written in C to ensure understanding of the basic structure of a C program for the microcontroller as well as how to compile and load a program written in C onto the board for operation. From there, the BumpBot routine is replicated in C.

# Program Overview

The BumpBot program provides the basic behavior that allows the TekBot to react to whisker input. The TekBot has two forward facing buttons, or whiskers, a left and a right whisker. By default the TekBot will be moving forward until one of the whiskers are triggered. If the left whisker is hit, then the TekBot will backup and then turn right for a bit, while a right whisker hit will backup and turn left. After the either whisker routine completes, the TekBot resumes its forward motion.

As compared to the assembly implementation, the C implementation of the BumpBot routine looks quite simple with everything including initialization being done within the main function.

## Main Routine

The GPIO ports being used are set up for input and output as necessary, with pullup resistors being used on the input. An infinite loop contains a conditional branch which checks for input and then performs the correct sequence of output to the motors. A built in delay function is used to time the motor control signals being output.

# Additional Questions

Almost all of the labs will have additional questions. Use this section to both restate and then answer the questions. Failure to provide this section when there are additional questions will result in no points for the questions. Note that if there are no Additional Questions, this section can be eliminated. Since the original lab does not have any questions, I will make some up to illustrate the proper formatting.

*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.*

Writing in C has a number of advantages. Two that stand out are the ease with which a developer can create a solution in C vs the same in assembly, and the portability across multiple different devices without having to rewrite the program manually for each. Additionally, C is still very fast in operation with little overhead. On the other hand, unlike assembly, you can not as precisely control what the processor is doing or guarantee an exact execution time when writing in C.

*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?*

The assembly hex file from lab 1 came to 485 bytes and the C hex file from lab 2 to 848 bytes. Although the programs were written to perform the same behavior, the manner in which they go about it must be different at some level.

# Conclusion

In this lab, we explored embedded C programming by recreating the first lab’s BumpBot routine in C. By doing this we gained an appreciation for both C and assembly from their respective strengths.

# Source Code

Provide a copy of the source code. Here you should use a mono-spaced font and can go down to 8-pt in order to make it fit. Sometimes the conversion from standard ASCII to a word document may mess up the formatting. Make sure to reformat the code so it looks nice and is readable.

/\*  
BumpBot  
  
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  
\*/  
#define F\_CPU 16000000  
#include <avr/io.h>  
#include <util/delay.h>  
#include <stdio.h>  
  
int main(void)  
{  
 DDRB = 0b11110000; // configure Port B pins for input/output  
 DDRD = 0b00000000; // Configure Port D's pins as an inputs  
 PORTD = 0b00000011; // Pull-up resistors on used pins  
 //PORTB = 0b11110000; // set initial value for Port B outputs  
 // (initially, disable both motors)  
  
 PORTB = 0b01100000; // Initiate the tekbot by move forward  
 while (1) { // loop forever  
  
 if((~PIND & 0b00000010) || (~PIND & 0b00000011) ){ //Left whisker or both whiskers has been depressed  
 PORTB = 0b00000000; // move backward  
 \_delay\_ms(500); // wait for 500 ms  
 PORTB = 0b01000000; // turn right  
 \_delay\_ms(2000); // wait for 2 s  
 PORTB = 0b01100000; // move forward  
 }  
 else if(~PIND & 0b00000001){ //Right whisker has been depressed  
 PORTB = 0b00000000; // move backward  
 \_delay\_ms(500); // wait for 500 ms  
 PORTB = 0b00100000; // turn left  
 \_delay\_ms(2000); // wait for 2 s  
 PORTB = 0b01100000; // move forward  
 }  
 }  
  
 /\*  
 PORTB = 0b01100000; // make TekBot move forward  
 \_delay\_ms(500); // wait for 500 ms  
 PORTB = 0b00000000; // move backward  
 \_delay\_ms(500); // wait for 500 ms  
 PORTB = 0b00100000; // turn left  
 \_delay\_ms(1000); // wait for 1 s  
 PORTB = 0b01000000; // turn right  
 \_delay\_ms(2000); // wait for 2 s  
 PORTB = 0b00100000; // turn left  
 \_delay\_ms(1000); // wait for 1 s  
 \*/  
}

# Challenge

The challenge of this lab modifies the program above so that the tekbot tries to push objects that it hits instead of turning away from them. A couple of simple modification to the above code can make it appropriate for this task.

1. Add a delay at the beginning of each conditional branch so that the TekBot continues to push for a bit after hitting an object.
2. Change the direction of the turn so that the tekbot turns back toward the object instead of turning away.

The following has been produced to complete this task.

# Challenge Source Code

In this lab, we explored embedded C programming by recreating the first lab’s BumpBot routine in C. By doing this we gained an appreciation for both C and assembly and their respective strengths.

/\*  
Challenge Push BumpBot  
  
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  
\*/  
#define F\_CPU 16000000  
#include <avr/io.h>  
#include <util/delay.h>  
#include <stdio.h>  
  
int main(void)  
{  
 DDRB = 0b11110000; // configure Port B pins for input/output  
 DDRD = 0b00000000; // Configure Port D's pins as an inputs  
 PORTD = 0b00000011; // Pull-up resistors on used pins  
 //PORTB = 0b11110000; // set initial value for Port B outputs  
 // (initially, disable both motors)  
  
 PORTB = 0b01100000; // Initiate the tekbot by move forward  
 while (1) { // loop forever  
  
 if((~PIND & 0b00000010) || (~PIND & 0b00000011) ){ //Left whisker or both whiskers has been depressed

\_delay\_ms(500); // wait for 500 ms  
 PORTB = 0b00000000; // move backward  
 \_delay\_ms(500); // wait for 500 ms  
 PORTB = 0b00100000; // turn left  
 \_delay\_ms(300); // wait for 300 ms  
 PORTB = 0b01100000; // move forward  
 }  
 else if(~PIND & 0b00000001){ //Right whisker has been depressed

\_delay\_ms(500); // wait for 500 ms  
 PORTB = 0b00000000; // move backward  
 \_delay\_ms(500); // wait for 500 ms  
 PORTB = 0b01000000; // turn left  
 \_delay\_ms(300); // wait for 300 ms  
 PORTB = 0b01100000; // move forward  
 }  
 }  
  
 /\*  
 PORTB = 0b01100000; // make TekBot move forward  
 \_delay\_ms(500); // wait for 500 ms  
 PORTB = 0b00000000; // move backward  
 \_delay\_ms(500); // wait for 500 ms  
 PORTB = 0b00100000; // turn left  
 \_delay\_ms(1000); // wait for 1 s  
 PORTB = 0b01000000; // turn right  
 \_delay\_ms(2000); // wait for 2 s  
 PORTB = 0b00100000; // turn left  
 \_delay\_ms(1000); // wait for 1 s  
 \*/  
}