ECE 375 Lab 2

C->Assembly->Machine Code->TekBot

**Lab Time: Tuesday 8-10**

Zachary DeVita

# Introduction

The purpose of this lab is to become familiarized with using C programming to interact with the AVR microcontroller. A pre-made “DanceBot.c” program was provided as an example showing how to set up the ports and interact with the AVR.

Using the example code as well as the software tools provided, we wrote a C program which makes the TekBot travel forward until an object is encountered, i.e. one or both of the whiskers are bumped. When an object is encountered, the TekBot then backs up, turns away from the object, and then continues in the forward direction. After the program was written, it was converted to hex using AVRStudio, and then uploaded to the AVR board using the Universal Programmer.

# Program Overview

The program is essentially a C implementation of the BumpBot program from the previous lab. The program provides the basic behavior for the TekBot which allows it to react to whisker input. The TekBot has two whiskers, a left and a right whisker, which allow it to determine the location of an object that it comes into contact with. By default the TekBot will initially move forward until one of the whiskers are triggered. If the left whisker is triggered then the TekBot will back up for the specified amount of time, then turn slightly right, and continue in the forward direction. If either the right whisker is triggered or both whiskers are triggered at the same time, then the TekBot will similarly back up for the specified amount of time, then turn slightly left, and then continue in the forward direction.

Aside from the primary objective described above, an additional block of code was written to provide an alternate functionality for the lab ‘Challenge’. For the ‘Challenge’, the objective was to program the TekBot to push objects that it encounters for a short distance. For this implementation, the inputs remain the same and the TekBot begins with the same default, initial forward movement as in the previous, but the behavior is modified.

If, when an object is encountered, both bumpers are triggered, then the TekBot will continue forward for a short period of time, then back up for one second, and then continue forward. If the right bumper is triggered, then the TekBot will continue moving forward for a short period of time, then back up for one second, turn slightly toward the object, and continue forward again. Similarly, if the left bumper is triggered, then the TekBot will continue moving forward for a short period of time, then back up for one second, turn slightly toward the object, right in this case, and continue forward again.

Both implementations cannot be run simultaneously which is why the code for the primary objective is commented off in the source code below.

## Routine

Prior to entering main(), there is a constant ‘F\_CPU’ defined as well as a few libraries necessary for the avr functionality. When inside main(), there is a one-time initialization of key registers which configure ports B and D for input and output from the whiskers. The initial values for ports B and D are also set which allow the bot to initially move in the forward direction, and to be accepting input from any port on D. I additionally, instantiated variables to store the binary configuration for forward/backward movement and the turn left/right commands.

## Primary Objective

The primary objective for the lab was to have the TekBot move forward until it came into contact with some object, then reverse, turn away from the object, and then continue moving forward. This is done by executing a simple while() loop that loops forever and having some code check to see if one or both whiskers on the TekBot have been hit. If the right whisker was triggered, i.e. the TekBot came into contact with an object on its right side, or if both whiskers were triggered, then the TekBot would instantly move backward for a short period of time, turn to the left, and then continue forward. If the left whisker was triggered, then the bot would have turned right opposed to left.

## Challenge

The challenge objective for the lab was to have the TekBot move forward until it came into contact with some object, then continue pushing forward for some period of time, then reverse, then turn slightly toward the object, and then continue driving forward. This is done by, again, simply executing a while() loop that loops forever and having some code check to see if one or both whiskers on the TekBot have been hit. If either the right or left whisker was triggered, then the TekBot would continue moving forward for a short period of time, then the TekBot would move slightly backward, turn to towards the object, and then continue forward. This process should theoretically repeat, and the object should be continually pushed along the ground. When both bumpers are triggered at the same time, then the TekBot would similarly continue forward for a bit and then move backward, but, in this case, it would not turn prior to continuing forward.

# Additional Questions

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

A huge advantage that comes to mind is that you would be able to write code in a higher level language and then convert it to the .hex file that is required for the mega128 board to run. Higher level languages are generally easier to program in, easier to include complex functionality in, and are generally more commonly known by programmers. This might also come in handy when combining functions or classes which are written in different languages. There are also far more resources for learning how to program in C than in assembly language.

A major drawback, I would assume, is that the converted .hex file would have a considerable amount of excess code created during the conversion and, therefore, the final product would require additional space in memory to be stored. This is very common in any code conversions so I would assume it occurs here as well. You also tend to lose a lot of precision and efficiency working with memory, interrupts, micro-registers, etc. that is gained by using assembly language. Higher level languages also have much less portability benefits when compared to assembly language. Aside from that, it might rob people of the wonderful experience of programming in assembly language.

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?

Lab 1: 485 bytes Lab 2: 1012 bytes

C produces much larger programs that have a huge memory cost. They have much larger instruction sets due to their vast libraries and built-in functions. And, as stated before, I believe any language would have a considerable amount of excess code when “cross-compiled”.

# Difficulties

It was extraordinarily difficult for me to understand the process of configuring Port B and Port D for input/output. I could not wrap my head around why the variables were being created or what the values represented. Another difficult part was understanding why the bits for PORTD and PIND needed to be flipped and that a zero would represent a true condition opposed to a false condition.

# Conclusion

The requirement for this lab was to write a C program which would make the TekBot reverse and turn away from any object which it came into contact with before continuing forward. We wrote the program in AVRStudio loosely using a sample program as a reference, then compiled the code into a .hex file which we were then able to upload to the TekBot. The result of the program was a success; the TekBot would continue forward until one or both whisker was triggered then move backwards and turn away from the object. The lab was fun and it allowed me to better understand the difference between C and assembly language, and what commands in assembly must be executed in order to have similar functionality to C.

# Source Code

/\*

This code will cause a TekBot connected to a mega128 board to reverse and turn away from objects that it runs into. No pins are used as input, and four Port B pins are used for output.

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>

int main(void)

{

DDRB = 0b11111111; // configure Port B pins for input/output

PORTB = 0b01100000; // set initial value for Port B outputs

DDRD = 0b00000000; // configure Port B pins for input/output

PORTD = 0b11111111; // set initial value for Port B outputs

int MovFwd = 0b01100000; // move forward command

int MovBck = 0b00000000; // move backward command

int TurnR = 0b00100000; // turn left command

int TurnL = 0b01000000; // turn right command

while (1) { // loop forever

/\*

if(PIND == 0b11111110 || PIND == 0b11111100) { // if right or both are hit

PORTB = MovBck; // move backward

\_delay\_ms(1000); // wait for 1 s

PORTB = TurnL; // turn left

\_delay\_ms(1000); // wait for 1 s

PORTB = MovFwd; // make TekBot move forward

}

if(PIND == 0b11111101) { // if left bumper is hit

PORTB = MovBck; // move backward

\_delay\_ms(1000); // wait for 1 s

PORTB = TurnR; // turn right

\_delay\_ms(1000); // wait for 1 s

PORTB = MovFwd; // make TekBot move forward

}

\*/

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

// CHALLENGE

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

if(PIND == 0b11111100) { // if both bumpers are hit

\_delay\_ms(1500); // wait for 1.5 s

PORTB = MovBck; // move backward

\_delay\_ms(1000); // wait for 1 s

PORTB = MovFwd; // make TekBot move forward

}

if(PIND == 0b11111110) { // if right bumper is hit

\_delay\_ms(1500); // wait for 1.5 s

PORTB = MovBck; // move backward

\_delay\_ms(1000); // wait for 1 s

PORTB = TurnR; // turn right

\_delay\_ms(300); // wait for .3 s

PORTB = MovFwd; // make TekBot move forward

}

if(PIND == 0b11111101) { // if left bumper is hit

\_delay\_ms(1500); // wait for 1.5 s

PORTB = MovBck; // move backward

\_delay\_ms(1000); // wait for 1 s

PORTB = TurnL; // turn left

\_delay\_ms(300); // wait for .3 s

PORTB = MovFwd; // make TekBot move forward

}

}

}