ECE 375 LAB 2

C->Assembly->Machine Code->TekBot

Lab Time: Wednesday 10-12

Tu Lam

INTRODUCTION

The purpose of the second lab is to introduces the introduction of coding in the language C in Atmel Studio 7. The program following the same build from Lab 1 where BumpBot was introduce and learning to build your own command from that into C language. Also, you will learn how to use the binary system to learn about how 1s and 0s work with the register to produce the I/O of the BumpBot.

PROGRAM OVERVIEW

The BumpBot program provides the basic behavior that allows the BumpBot to react to whisker input. By default, the BumpBot continues to move forward until a whisker has been hit. If the right whisker is hit, the BumpBot will back up for a second and then turn left and continue forward. This is the same logic for the left whisker but in the opposite direction. Another feature that will be implemented into the program is if both whiskers were hit, then the BumpBot will perform the same logic as if the right whisker were hit.

Besides the standard MAIN routines within the program, there will be three additional routines were created and used. The RightWhisker and LeftWhisker routines provide the basic functionality for handling either a Right or Left whisker hit, respectively. Additionally, a Wait function will be implemented in the function to make sure the BumpBot have time to perform the correct respond when a whisker is trigger. Lastly, we will have a routine call BothWhiskers to replicate what will happen when both whiskers are hit.

INITIALIZATION ROUTINE

The initialization routine provides a one-time initialization of key registers that allow the BumpBot program to execute correctly. First, we will tell PORTB to be all as output from PIN 4-7 on the board while having PORTD PIN 0-1 to be our reading input of the command that would make the BumpBot respond.

MAIN ROUTINE

The Main routine executes a simple loop that checks to see if a whisker was hit. The loop keeps the BumpBot in a forward until it checks for condition of input. If the input is the right whisker, it jumps to that function and perform the routine that was written in the function. After it will jump out of the function and check to see other whiskers are hit like left whisker or even both whiskers. Then finally, the program loop back to the beginning of the 'while' loop forever and check the program again.

RIGHTWHISKER ROUTINE

The RightWhisker routine will first move the BumpBot backward by setting the PIN 5-6 to off. Then the BumpBot will turn left and continue forward. This will light the PIN 5 up and then PIN 6 demonstrate as it is moving forward. And this is the end of the routine.

LEFTWHISKER ROUTINE

The LeftWhisker routine is identical to the RightWhisker routine, except that it will turn right and then continue forward from the performance. There will be a wait function in all of the routines to help the BumpBot have time to perform correctly.

BothWhiskers Routine

The BothWhiskers routine will be a scenario case where both whiskers were hit at that moment. This routine will perform the exact way how the RightWhisker routine works. So basically, the BumpBot will move backward, the turn left and continue 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.

Writing in a language like C that can be "cross compiled" has it benefits and drawbacks to the system. The benefit is that since the system is common and when it comes to exchanging code and file to another device as there is no need to rewrite the program and such. The downside of the having cross compiled is that the system is written to compile that program, so it has hard time executing code the runtime that you want or having control over what processor that the compiler is running.

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 Lab 1 .hex file came to about 485 bytes and the Lab 2 came to around 1,200 bytes for the .hex file. The size difference between the two files maybe have to do with the compiler and the way how each set of instruction is using memory to fetch and execute the file even though they perform the same behavior.

Conclusion

For the Lab #2, the lab helps to demonstrate coding in C language and how to program the language to make the BumpBot move. Learning about compiling last week for Lab #1, the content transfer over this week and our object to build the same command that was demonstrated last week into this week on our own. The lab overall familiarize yourself with C language and using that language as a way to setup your PORTx and learn how the command works on the board.

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 reformate the code so it looks nice and is readable.

```
/*****************************************************************
 * BumpBot.c
 *
 * This code will cause a TekBot connected to the AVR board to
 * move forward and when it touches an obstacle, it will reverse
 * and turn away from the obstacle and resume forward motion.

 * 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
 ****************************************************************
 *
 * Author: Tu Lam
 * Date: October 11, 2020
 *
 ****************************************************************/

#define F_CPU 16000000                          //This include all the definition files
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>

int main(void)
{
        DDRB = 0b11110000;                      //This configure Port B[Pin 4 - 7] to be the outputs
        PORTB = 0b11110000;                     //Set up the Pin 4 - 7 to turn on it LED

        while (1) // loop forever
        {
                PORTB = 0b01100000;             //This set the BumpBot to move forward forever until a
trigger is hit
                _delay_ms(500);                 //Wait for 500 ms


                if (PIND == 0b11111110) {       //Look if the right whisker hit
                        RightWhisker();
                }

                else if (PIND == 0b11111101) {  //Look if the left whisker hit
                        LeftWhisker();
                }

                else if(PIND == 0b11111100) {   //Look if the left & right whisker hit
                        BothWhiskers();
                }
        }
}

/*************************************************************
* Function: RightWhisker
* Description: If the right whisker hit on PIN 0 in PORTD,
               The BumpBot will turn left and move forward.
*************************************************************/
void RightWhisker() {

        PORTB = 0b00000000;                     //Move backward
        _delay_ms(500);                         //Wait for 500ms
        PORTB = 0b00100000;                     //Turn left for a second
        _delay_ms(1000);                        //Wait for 1s
        PORTB = 0b01100000;                     //Continue forward
        _delay_ms(500);                         //Wait for 500ms
}

/*************************************************************
* Function: LeftWhisker
* Description: If the right whisker hit on PIN 1 in PORTD,
               The BumpBot will turn right and move forward.
*************************************************************/
void LeftWhisker() {

        PORTB = 0b00000000;                     //Move backward
        _delay_ms(500);                         //Wait for 500ms
        PORTB = 0b01000000;                     //Turn right for a second
```

```
        _delay_ms(1000);                        //Wait for 1s
        PORTB = 0b01100000;                     //Continue forward
        _delay_ms(500);                         //Wait for 500ms
}

/*********************************************************
* Function: BothWhiskers
* Description: If the both whiskers hit on PIN 1 & 0 in PORTD,
               The BumpBot will turn left and move forward.
*********************************************************/
void BothWhiskers() {

        PORTB = 0b00000000;                     //Move backward
        _delay_ms(500);                         //Wait for 500ms
        PORTB = 0b00100000;                     //Turn left for a second
        _delay_ms(1000);                        //Wait for 1s
        PORTB = 0b01100000;                     //Continue forward
        _delay_ms(500);                         //Wait for 500ms
}
```