ECE 375 LAB 2

Introduction to AVR Development Tools

Lab Time: Friday 2-4

Hao Truong

INTRODUCTION

The purpose of this lab is to introduce the configuration of the I/O ports of the ATmega128 microcontroller. Students will write a simple C program for the ATmega128 microcontroller and compile their program using the Atmel Studio GCC compiler. The program will make the TekBot perform the basic BumpBot routine as introduced in lab 1.

PROGRAM OVERVIEW

The BumpBot program provide some basic behaviors that allows the TekBot to respond to whisker inputs which are right and left whiskers (buttons). The BumpBot initially goes forward by default until left or right whiskers are triggered. If the right whisker is hit, the TekBot will back up for a second, then turn left for a second, and then continue forward. If the left whisker is hit, the TekBot will back up for a second, then turn right for a second, and then also continue forward. If both whiskers are triggered at the same time, the TekBot will behave as it does when right whisker is triggered.

INITIALIZATION ROUTINE

The initial routine provides a one-time initialization of the key registers that allow the BumpBot program to execute correctly. First, we set 7-4th bits on port B as outputs and turn on 6-5th LEDs as it indicates that the TekBot is moving forward by default. Then we set port D as inputs and enable pull-up registers.

MAIN ROUTINE

The main function is a simple while loop that keeps the grogram running forever. Inside the loop, it will check for any input, that comes from Port Input Register for port D, by reading 8-bits of data from PIND and then execute corresponding function based on the input. If either right whisker or both whiskers get hit at the same time, it will call HitRight function. If left whisker is hit, then it will call HitLeft function.

SUBROUTINES

1. HitRight Routine

The HitRight function first move the TekBot backwards for a second by clear 6-5th bits on PORTB followed by __delay_ms(1000) function to make it wait for one second. Then it turns left for a second by setting bit 5 on PORTD and then another call to __delay_ms(1000). Finally, it resume to its forward motion by setting 6-5th bits on PORTD.

2. HitLeft Routine

The Hitleft function behaves similarly with HitRight function except that instead of turning left after reversing, it turns right.

ADDITIONAL QUESTIONS

1) 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 benefit of writing code in a language like C than can be cross compiled is tha an executable code is created that can be run on different platforms or devices and there is no need to rewrite the code. However, the downside is that it is hard to maintain and expand as a result. Sometimes code does not compile.

2) 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 size of my Lab 1 and Lab 2 output .hex files are 530 and 856 bytes respectively. There is a size difference between these two files, even though they both perform the same BumpBot behavior maybe because of the way the instructions are written and how memory is fetched and used in assembly.

CONCLUSION

In this lab, we were required to write a simple C program which performed the same basic behavior for the TekBok as seen in lab 1. We learned how to configure the I/O ports of the ATmega128 microcontroller and compile our program using the Atmel Studio GCC compile. We also looked at the sizes of the Lab1 and Lab 2 output .hex files and it turned out that our C program took up more space than our assembly program even though they both performed the same behavior.

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.

```
/*
 * Hao_Truong_Lab2_sourcecode.c
 *
 * Created: 10/8/2021 2:39:51 PM
 * Author : Hao Truong
 */


/*
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
*/

#define F_CPU 16000000
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>


// Function prototypes
```

```
void HitRight();
void HitLeft();
void HitBoth();

int main(void)
{
        // Initialize relevant I/O Ports
        DDRB = 0b11110000;     // Set 7-4th bits as outputs
        PORTB = 0b01100000;    // Turn on 6-5th LEDs

        DDRD = 0b00000000;     // Set 7-0th bits as inputs
        PORTD = 0b11111111;    // Enable pull up resisters



        while (1) // loop forever
        {
                // Your code goes here
                if (PIND == 0b11111110) {     // if right whisker was hit
                        HitRight();                          // do HitRight function
                }
                if (PIND == 0b11111101) {     // if left whisker was hit
                        HitLeft();                                  // do HitLeft function
                }
                if (PIND == 0b11111100) {     // if both whiskers are triggered
                        HitRight();                          // do as HitRight
                }
        }
}

void HitRight() {
        // reverse for a second
        PORTB = 0b00000000;
        _delay_ms(1000);

        // turn left for a second
        PORTB = 0b00100000;
        _delay_ms(1000);

        // move forward again
        PORTB = 0b01100000;
}

void HitLeft() {
        // move backwards for a second
        PORTB = 0b00000000;
        _delay_ms(1000);

        // turn right for a second
        PORTB = 0b01000000;
        _delay_ms(1000);

        // move forward again
        PORTB = 0b01100000;
}
```