EE2016: Experiment 5

Group:19

Sparsh Gupta
EE23B117

Charan Srikkanth
EE23B127

Kaushik Iyer
EE23B135

September 12, 2024

1 Introduction

This experiment involves an AVR Atmega8 microcontroller on a breadboard. The microcontroller (on the bread board) is connected to a USBASP board and in turn, this is connected to the personal computer (desktop). The USBASP board itself includes a microcontroller and this provides an elegant arrangement to program the Atmega8 on the breadboard.

2 Objectives

The primary objectives of this experiment are:

1. Make the connections between USBASP,atmega8 chip and LEDs on a breadboard to implement the tasks .
2. Use burn-o-mat software to write the atmega chip
3. Code the different tasks given to us in microchip studio using C language .
4. Get an output from the atmega chip using LEDs and show it to the TAs

2.1 Header code used for all tasks:

used define to set int8 to show unsigned character in all the tasks

```c
#define int8 unsigned char
#include <util/delay.h>
#include <avr/io.h>
```

3 Theory involved

1. DDRX sets which pins are input and output in binary form in the Xth port, if the pin at that position is:
   (a) 1: that pin is output pin
   (b) 0: that pin is meant to take input
2. PORTX is used to set output to be given in X's output ports
3. PINX is used to access the input received from outside
4. delay(1000) used to wait 1 second before next line is executed

4 Make LED glow

The task is to make the LED glow permanently

4.1 C-function to implement

```c
int8 led_glow(){
    DDRD = 0x03;
    while(1){
        PORTD = 0x03;
    }
}
```

5 Make the LED blink with about 1 second gap.

For this part we need to make the LED on every alternate second.So we add delay of 1 second before the next line of code is executed, the standard dimensions of delay are ms, Hence delay(1000) is used

5.1 C-function to implement

```c
    int8 led_blink(){
    DDRD = 0x03;
    while (1) {
        PORTD  = 0x01;
        _delay_ms(1000);
        PORTD = 0x00;
        _delay_ms(1000);  }
}
```

6 Make 2 LEDs blink alternately

We make the 2 LEDs be ON in alternate times of their duty cycle.

6.1 C-function to implement

```c
    int8 led_blink(){
    DDRD = 0x03;
    while (1) {
        PORTD  = 0x01;
        _delay_ms(1000);
        PORTD = 0x10;
        _delay_ms(1000);  }
}
```

7 Make 2 LEDs produce 2-bit gray code sequence

we make the LEDs glow in gray code sequence: 00, 01, 10, 11, 00, ...

7.1 C-function to implement

```c
int8 gray_code(){
    DDRD = 0x03;
    while (1) {
        PORTD  = 0x00;
        _delay_ms(1000);
        PORTD = 0x01;
        _delay_ms(1000);
        PORTD  = 0x03;
        _delay_ms(1000);
        PORTD = 0x02;
        _delay_ms(1000);
    }
}
```

8 Perform addition for two 2-bit numbers

We hard code a pair of 2-bit numbers in our program and perform addition of the numbers.To show the results of the addition, we use three LEDs.

8.1 C-function to implement

```c
int8 add_2bit(int8 num1,int8 num2){
    DDRD= 0x07;
    int8 sum=num1+num2;
    while(1){
        PORTD= sum;
    }
}
```

9 Set up 3-bit Johnson Counter

The Johnson counter is designed by connecting three instances of the D flip-flop with reset in a sequential manner. The output of the last flip-flop is fed back as input to the first flip-flop, with appropriate connections to form the Johnson counter. It will have states given by 000, 100, 110, 111, 011 and 001 and this sequence repeats.Here, we have implemented the Johnson counter using LEDs and used Atmega8 to control them.

9.1 C-function to implement

```c
int8 johnson(){
    DDRD = 0x07;
    while (1) {
        PORTD  = 0x00;
        _delay_ms(1000);
        PORTD = 0x01;
        _delay_ms(1000);
        PORTD  = 0x03;
        _delay_ms(1000);
        PORTD = 0x07;
        _delay_ms(1000);
        PORTD = 0x06;
        _delay_ms(1000);
        PORTD = 0x04;
        _delay_ms(1000);
    }
}
```

10 Use port C instead of port D for output in the previous tasks

We do the exact same experiment as above but by using port C instead of D,We changed the connections of LEDs and resistors from D to C's corresponding pins, and change the code by changing 'x' at the end of all port names from D to C.

10.1 C-function to implement

```
int8 add_2bit(int8 num1,int8 num2){
    DDRC= 0x07;
    int8 sum=num1+num2;
    while(1){
        PORTC= sum;
}
}
int8 johnson(){
    DDRC = 0x07;
    while (1) {
        PORTC  = 0x00;
        _delay_ms(1000);
        PORTC = 0x01;
        _delay_ms(1000);
        PORTC  = 0x03;
        _delay_ms(1000);
        PORTC = 0x07;
        _delay_ms(1000);
        PORTC = 0x06;
        _delay_ms(1000);
        PORTC = 0x04;
        _delay_ms(1000);
    }
}
```

11 Perform addition for two 4-bit numbers by hard-coding the numbers

We perform addition of the two 4-bit numbers and show the results on five LEDs connected to Port C.

11.1 C-function to implement

```
int8 add_4bit(int8 num1,int8 num2){
    DDRC= 0x1F;
    int8 sum=num1+num2;
    while(1){
        PORTC= sum;
    }
}
```

12 BONUS: Performing addition for two 4-bit numbers by taking input on breadboard

We perform addition of the two 4-bit numbers and show the results on five LEDs connected to Port C while taking input from port D.

12.1 C-function to implement

```c
void add_4bit_with_input(){
    DDRD=0x00;
    DDRC=0x1F;
    while(1){
        PORTC = (PIND & 0x0F) + ((PIND & (0xF0)) >> 4);
    }
}
```

13 Comments

1. We learnt how to code using C language in AVR. We also learnt how to wire the Atmega8 chip on breadboard.
2. We were not able to write the chip initially due to setting on USBASP being at 3.3V instead of 5V. After figuring that out, rest of the experiment went well.