EE2016 - Experiment 5

Nishanth Senthil Kumar EE23B049
Gopalkrishna Ramanujan EE23B029
Arjun Krishnaswamy EE23B009

Group Number : 27

September 24, 2024

1 Introduction

This experiment involves an AVR Atmega8 microcontroller on a breadboard. The microcontroller (on the breadboard) is connected to a USBASP board as shown in Figure 1 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 Port Configuration

The Atmega8 microcontroller has 3 ports: PortB, PortC, and PortD. Each of these ports is associated with 3 registers - DDRx, PORTx and PINx which set, respectively, the direction, output and input functionality of the ports. Each bit in these registers configures a pin of a port. Bit0 of a register is mapped to Pin0 of a particular port, Bit1 to Pin1, Bit2 to Pin2 and so on.
DDR stands for Data Direction Register and 'x' indicates a port alphabet. As the name suggests, this register is used to set the direction of port pins to either input or output. For input, we set to 0 while for output, we set to 1

3 Task 3

3.1 Objective

A program to turn on an LED permanently.

3.2 Code

```c
#include <avr/io.h>
#include <util/delay.h>

int main(void){
    DDRD=0xFF;
    while (1){

        PORTD=0x01;
    }
}
```

3.3 Explanation

- Two libraries have been imported, ”io.h” contains AVR register names like ”PORTx” and ”PINx” which can be directly used instead of manually typing in the addresses of the input/output registers.
- We set the Data Direction Register of D to 0xFF, which means that all the pins are set to output.
- Now,after setting the D register to output, we write 0x01 to the register, which means that the pin 0 of the D register is set to 1, and remaining are set to 0. The command need not be written in hex, it could be written in binary or decimal.
- Since we want the LED to glow indefinitely, the while loop is not necessary in this case.
- The LED must be connected to pin 0 of the D register, with cathode connected to the ground, and the anode connected to the pin on the microcontroller. A resistor of around 330 ohms should also be connected to it, because the LED operates at around 2-3V, and the output pin when set to high gives 5V.
- Hence to ensure that excess current doesn’t flow through the LED and fry it, it is required to connect a resistor. If a resistor of high value is connected, the LED will glow with less brightness.

4 Task 4

4.1 Objective

A program to make an LED connected to PORTD, blink with a time period of approximately 1 second.

4.2 Code

```c
#include <avr/io.h>
#include <util/delay.h>

int main (void){
    DDRD = 0x01;
    while (1) {
        PORTD = 0x01;
        _delay_ms(500);
        PORTD = 0x00;
        _delay_ms(500);
    }
 }
```

4.3 Explanation

- The ”delay.h” allows us to use delay functions directly. We could make our own delay function using empty for loops, however, the GCC compiler may optimise the code, and when it sees an empty for loop, it may ignore it. In order to use an empty for loop, the optimisation of the compiler should be set to the minimum value.
- We set the Data Direction Register of D to 0x01, which means only one pin is set to output.
- Now,after setting the D register to output, we write 0x01 to the register, which means that the pin 0 of the D register is set to 1, and remaining are set to 0.
- We use the delay ms function from delay.h to create a delay of 500 ms after which we write 0x00 to the register which sets pin 0 to 0
- Thus, the LED is ON for 0.5 seconds and OFF for 0.5 seconds and which goes on forever leading to the blinking effect. .

5 Task 5

5.1 Objective

To make 2 LED's connected to PORTD blink alternatively every second.

5.2 Code

```c
#include <avr/io.h>
#include <util/delay.h>

int main (void){
    DDRD = 0x03;
    while (1) {
        PORTD = 0x01;
        _delay_ms(1000);
        PORTD = 0x02;
        _delay_ms(1000);
    }
}
```

5.3 Explanation

- We set the Data Direction Register of D to 0x03, which means we use 2 pins as output pins.
- Now,after setting the D register to output, we write 0x01 to the register, which means that the pin 0 of the D register is set to 1, and pin 1 is set to 0.
- We use the delay ms function from delay.h to create a delay of 500 ms after which we write 0x10 to the register which sets pin 1 to 1 and pin 0 to 0
- We connect 2 LEDs at pin 0 and pin 1 respectively and the output sequence of 01-10-01-10 ... from PORTD leads to alternate blinking of the LEDs

6 Task 6

6.1 Objective

To make the two LEDs produce the following sequence: 00, 01, 10, 11, 00,...

6.2 Code

```c
#include <avr/io.h>
#include <util/delay.h>

int main (void){
    DDRD=3;
    while (1) {
        PORTD = 0;
        _delay_ms(1000);
        PORTD = 1;
        _delay_ms(1000);
        PORTD = 2;
        _delay_ms(1000);
        PORTD = 3;
        _delay_ms(1000);
    }
 }
```

6.3 Explanation

- We set the Data Direction Register of D to 0x03, which means we use 2 pins as output pins.
- Now we use the exact same approach as the previous task but instead, vary the pins 0 and 1 in such a manner that we get the required sequence, that is, we set the ports to 0 1 2 3 and back again which leads to the PORTD outputs 00-01-10-11-00....

7 Task 7

7.1 Objective

To hard code a pair of 2-bit numbers in your program and perform addition of the numbers. Show the results of the addition on three LEDs

7.2 Code

```
#include <avr/io.h>
#include <util/delay.h>

int main (void){
    unsigned char a=5;
    unsigned char b=1;
    DDRD = 0x07;
    while (1) {
        PORTD = a+b;
    }
}
```

7.3 Explanation

- We set the Data Direction Register of D to 0x07(or just 7), which means we use 3 pins as output pins.
- This is needed since the sum of 2 two-bit numbers may require 3 bits to be shown
- The numbers to be added (a and b) are initialised as unsigned char since we know that our numbers don't require more than 8 bits to be store.
- The addition process is very simple and the code segment PORTD = a + b serves the purpose
- As usual, LEDS are connected to output pins and in this case, we use pin 0, pin 1 and pin 2 to show the sum

8 Task 8

8.1 Objective

To setup the three LEDs to implement a 3-bit Johnson counter.

8.2 Code

```c
#include <avr/io.h>
#include <util/delay.h>

int main (void){
    DDRD=7;
    while (1) {
       PORTD=0;
       _delay_ms(1000);
       PORTD=4;
       _delay_ms(1000);
       PORTD=6;
       _delay_ms(1000);
       PORTD=7;
       _delay_ms(1000);
       PORTD=3;
       _delay_ms(1000);
       PORTD=1;
       _delay_ms(1000);
    }
 }
```

8.3 Explanation

- We set the Data Direction Register of D to 0x07(or just 7), which means we use 3 pins as output pins.
- The 3-bit johnson counter follows the pattern 000-100-110-111-011-001 and back again to 000 which when translated to decimal is the sequence 0-4-6-7-3-1 and 0
- This idea is used to sequentially set the output pins of PORTD and give adequte delay in between in order to display the sequence
- As usual, LEDS are connected to output pins and in this case, we use pin 0, pin 1 and pin 2 to show the sum

9 Task 9

9.1 Objective

Doing task 7 and 8 with Register C

9.2 Code

```c
#include <avr/io.h>
#include <util/delay.h>

int main (void){
    unsigned char a=5;
    unsigned char b=1;
    DDRC = 0x07;
    while (1) {
        PORTC = a+b;
    }
}
```

```c
#include <avr/io.h>
#include <util/delay.h>

int main (void){
    DDRC=7;
    while (1) {
       PORTC=0;
       _delay_ms(1000);
       PORTC=4;
       _delay_ms(1000);
       PORTC=6;
       _delay_ms(1000);
       PORTC=7;
       _delay_ms(1000);
       PORTC=3;
       _delay_ms(1000);
       PORTC=1;
       _delay_ms(1000);
    }
 }
```

9.3 Explanation

- This is task 7 and 8 all over again but with 3 bits of PORTC set as output using Data Direction Register of C, which is set to 0x07

10 Task 10

10.1 Objective

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

10.2 Code

```c
#include <avr/io.h>
#include <util/delay.h>
int main(void)
{
    DDRC = 0xFF;
    unsigned char a = 8;
    unsigned char b = 7;
    while (1)
    {
        PORTC = a + b;
    }
}
```

10.3 Image of Adder working

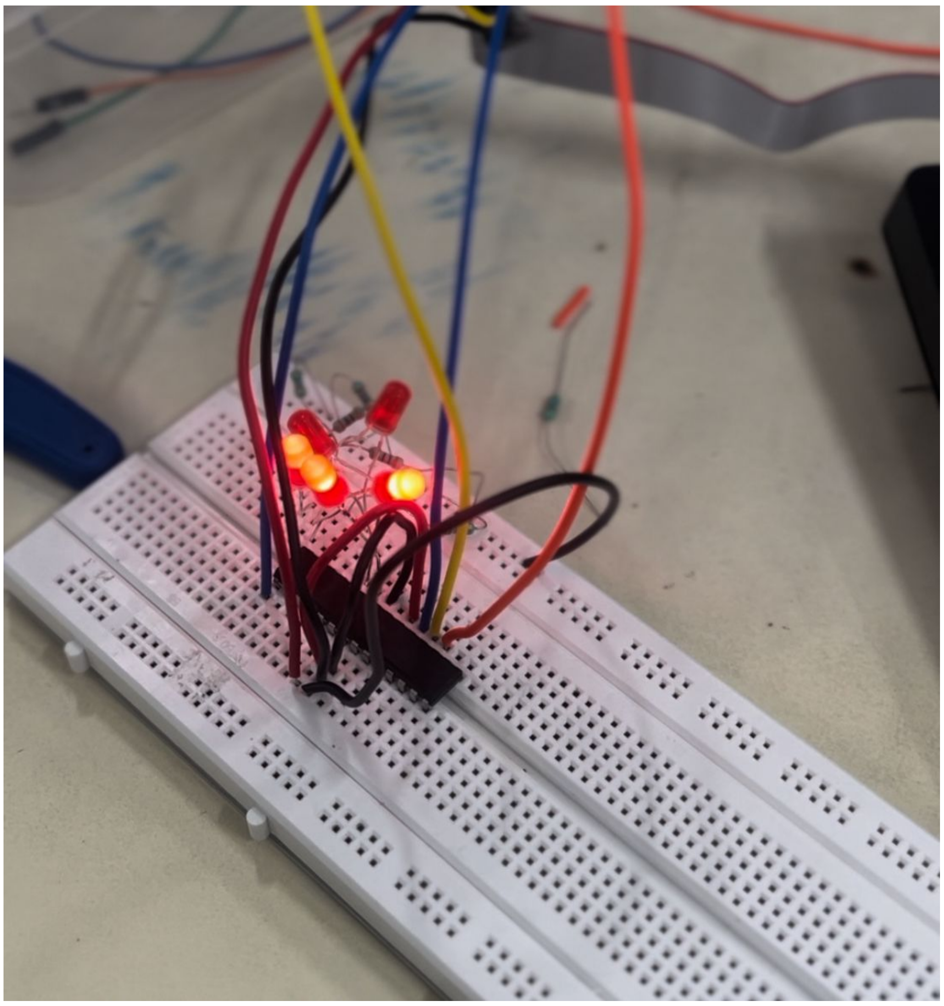

Figure 1: Adder in practice

10.4 Explanation

- We set the Data Direction Register of Cx to 0xFF which means all the pins of PORT are made output pins

- The sum of 2 four-bit numbers may require 5 bits to display
- The exact same method used for adding 2 two-bit numbers is used. PORTC = a + b does the job for us again.
- 5 LEDs are connected to the 5 output pins of PORTC in order to display the sum

11 Programming and Debugging experience

- The C programming part was pretty smooth and direct since all the tasks pretty much involved usage of Ports and configuring input and output pins.
- Since the 4 bit sum involved usage of 5 LEDs to display the sum, it again lead to loose wiring which took us some significant time to correct.
- We faced issues while performing circuit wiring with USBASP which caused issues initially. The USBASP board connection diagram was a little confusing because the board looked a little different in real life.
- Whenever we tried burning the code onto the ATmega8, it threw an error saying "Board not found", we initially believed that the Atmega8 was malfunctioning, but the problem was that it was not pushed into the bread board well.