Experiment No: 10 **Date :**

Roll No: _____

Aim : **To perform AVR I/O programming in C**

Theory :

- **In AVR microcontroller family, there are many ports available for I/O operations, depending on which family microcontroller you choose.**
- **For the ATmega32 40-pin chip 32 Pins are available for I/O operation. The four ports PORTA, PORTB, PORTC, and PORTD are programmed for performing desired operation.**

- **The 40-pin AVR has four ports for using any of the ports as an input or output port, it must be accordingly programmed.**
- **The Registers Addresses for ATmega32 Ports is given below:**

  PORTx,

  DDRx

  PINx
- **Each port in AVR microcontroller has three I/O registers associated with it. They are designated as.**
- **Each of I/O registers is 8 bits wide, and each port has a maximum of 8 pins, therefore each bit of I/O registers affects one of the pins.**
- **For accessing I/O registers associated with the ports the common relationship between the registers and the pins of AVR microcontroller.**

DDRx: Data Direction Register

- **Before reading or writing the data from the ports, their direction needs to be set. Unless the PORT is configured as output, the data from the registers will not go to controller pins.**
- **This register is used to configure the PORT pins as Input or Output.**

- **Writing 1's to DDRx will make the corresponding PORTx pins as output.**
- **Similarly writing 0's to DDRx will make the corresponding PORTx pins as Input.**

DDRx	Port Type
1's	O/P
0's	I/P

PORTx:

- **when port is configured as output then PORTx register is used.**
- **When we set bits in DDRx to 1, corresponding pins becomes output pins.**
- **Now we can write the data into respective bits in PORTx register.**
- **This will immediately change the output state of pins according to data we have written on the ports.**

```
DDRA = 0xFF;            //make port A as outputs

PORTA = x;
```

PINx register:

- **PINx register used to read the data from port pins. In order to read the data from port pin, first we have to change the portxs data direction to input.**
- **This is done by setting bits in DDRx to zero. If port is made output, then reading PINx register will give a data that has been output on port pins.**
- **There are two input modes.**
- **Either we can use port pins as internal pull up or as tri stated inputs. It will be explained as shown below:**
- **For reading the data from port A.**

```
DDRA = 0x00;    //Set port A as input

x = PINA;       //Read contents of port a
```

Write C program to toggle LED connected to PORT B.

```
#include <avr/io.h>
#define F_CPU 16000000UL  // 16 MHz
#include <util/delay.h>

int main(void)
{
DDRB = 0xFF;
    while (1)
    {
        PORTB = 0b00010000; //Port D bit4 On
        _delay_ms(1000);
        PORTB = 0b00000000; //Port D bit4 Off
        _delay_ms(1000);
    }
}
```

Conclusion :