**Microprocessor Systems and Interfacing**

**Lab Report**

**Lab04**



|  |  |
| --- | --- |
| Group Members Name & Reg #: | **Muhammad Haris Irfan**  **(FA18-BCE-090)** |
|  |  |
| Class | Microprocessor Systems and Interfacing CPE342 (**BCE-6B**) |
| Instructor’s Name | Dr. Omer Ahmad |

**Pre-Lab Tasks**

## Task-1

Read the theory section of this lab thoroughly.

**Task-2**

|  |
| --- |
| Write an assembly language program that is able to display the numbers ‘0’ to ‘9’ on an LCD as connected in figure 5.2. *[hint: look up ASCII table]* |

Code

.equ RS = 0

.equ En = 1

.org 0x0000

rjmp start

.org 0x0034

start:

ldi R16, 0xFF ; Set PortB to output

out DDRB, R16

out DDRD, R16 ; Set PortD to output

rcall LCD\_Init ; Initiallize the LCD

rcall DisplayNums ; Display Numbers 0-9 on the LCD

forever:

nop

rjmp forever ; Do nothing in a never ending loop

delay\_1ms: ; This function will generate a delay of approximately 1 ms on an 8-MHz Atmega328p

push R16

push R17

ldi R17, 8

L1:

ldi R16, 250

L11:

nop

dec R16

brne L11

dec R17

brne L1

pop R17

pop R16

ret

delay\_50ms:

push R16

ldi R16, 50

L2:

rcall delay\_1ms

dec R16

brne L2

pop R16

ret

LCD\_Send\_Command: ; This function assumes that command byte is in R18.

cbi PORTB, RS ; Clear RS for command

out PORTD, R18 ; Output the command byte on PORTD

rcall LCD\_Pulse\_En ; Send a 1 ms pulse on En (PB1)

ret

LCD\_Send\_Data: ; This function assumes that data byte is in R18.

sbi PORTB, RS ; Set RS for data

out PORTD, R18 ; Output the data byte on PORTD

rcall LCD\_Pulse\_En ; Send a 1 ms pulse on En (PB1)

ret

LCD\_Pulse\_En:

sbi PORTB, En ; Set En high

rcall delay\_1ms ; wait for 1ms

cbi PORTB, En

ret

LCD\_Init:

rcall delay\_50ms ; wait for more than 40ms

ldi R18, 0x30 ; send command 0x30

rcall LCD\_Send\_Command

rcall delay\_1ms ; delay of more than 4.1 ms

rcall delay\_1ms

rcall delay\_1ms

rcall delay\_1ms

ldi R18, 0x30 ; send command 0x30

rcall LCD\_Send\_Command

rcall delay\_1ms ; wait more than 100 us

ldi R18, 0x38 ; send command 0x38 (2 lines, 5x7 size)

rcall LCD\_Send\_Command

ldi R18, 0x0E ; send command 0x08 (Display off)

rcall LCD\_Send\_Command

ldi R18, 0x01 ; send command 0x01 (Clear display)

rcall LCD\_Send\_Command

ldi R18, 0x0F ; send command 0x0F (Entry mode set)

rcall LCD\_Send\_Command

ret

DisplayNums:

push R18 ; save R18 to the stack

push R17 ; save R17 to the stack

ldi R18, '0' ; R18 holds ASCII value for 0

ldi R17, ('9'+1) ; Loop has to run till digit 9 has been displayed

L3:

rcall LCD\_Send\_data

inc R18

cp R18, R17

BRNE L3

pop R17 ; restore R17

pop R18 ; restore R18

ret

**Simulation**

**Diagram

Description automatically generated**

## Task-3

|  |
| --- |
| Consider the basic wiring shown between an ATmega328P chip and an LCD in figure 5.2. Write and execute a C-program on Proteus that is able print your name on the first row of the LCD and your roll-number on the second row of the LCD |

**Code:**

#include <avr/io.h>

#define *F\_CPU* 1000000

#include <util/delay.h>

#define RS PC0

#define EN PC1

void lcd\_comm (char);

void lcd\_data(char);

void lcd\_init (void);

int main(void)

{

char name[16]={'H','A','R','I','S', 'I','R', 'F', 'A', 'N',' ',' ',' ',' ',' ',' '};

char rnum[16]={'F','A','1','8','-','B','C','E','-','0','9','0',' ',' ',' ',' '} ;

while(1)

{

DDRD = 0xFF;

DDRC = 0x03;

lcd\_comm(0x38); //2 lines

lcd\_comm(0x0C); //cursor off

//lcd\_comm(0x01); clear screen

lcd\_comm(0xD80); //force to first line

for(int i=0; i<16;i++)

{

*\_delay\_ms*(500);

lcd\_data(name[i]);

lcd\_comm(0x06);

if(i==4)

lcd\_comm(20);

//increametn cur

}

lcd\_comm(192);

for(int j=0; j<16;j++)

{

*\_delay\_ms*(50);

lcd\_data(rnum[j]);

lcd\_comm(0x06);

//increametn cur

}

return 0;

}

}

void lcd\_comm(char x){

PORTD = x;

PORTC &= ~(1<<RS);

PORTC |= (1<<EN);

*\_delay\_ms*(5);

PORTC &= ~(1<<EN);

}

void lcd\_data(char x){

PORTD = x;

PORTC |= (1<<RS);

PORTC |= (1<<EN);

*\_delay\_ms*(50);

PORTC &= ~(1<<EN);

}

**Simulation:**

**Diagram, schematic

Description automatically generated**

**In Lab Tasks**

**Task 1:**

Wire your Arduino Uno / Nano / ATmega328P to an LCD on your breadboard and execute the program performed in Task-2 from ‘Pre-Lab’ Tasks

A picture containing text

Description automatically generated

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Task 2:**

Consider your controller connected to an 8×dipswitch, a push-button and an LCD. Program your controller in a way that whenever the push-button is pressed, the input from the dipswitch should be pushed on to the LCD as an ASCII character. Execute your code on Proteus.

**Code:**

#define *F\_CPU* 8000000UL /\* Define CPU Frequency e.g. here its 8MHz \*/

#include <avr/io.h> /\* Include AVR std. library file \*/

#include <util/delay.h> /\* Include inbuilt defined Delay header file \*/

#define LCD\_Dir DDRD /\* Define LCD data port direction \*/

#define LCD\_Port PORTD /\* Define LCD data port \*/

#define RS PD0 /\* Define Register Select (data reg./command reg.) signal pin \*/

#define EN PD1 /\* Define Enable signal pin \*/

void LCD\_Init (void);

void LCD\_String (char \*str);

void LCD\_Command( unsigned char cmnd );

void LCD\_data( unsigned char data );

int main()

{

LCD\_Init(); /\* Initialization of LCD\*/

DDRC = 0x03;

DDRB = 0x00;

PORTB = 0xff;

unsigned char a = 0b00000000;

if ((PINC & (1<<PC4)) == 0)

{

a |= (PINB & 0xff);

}

while(1)

{

if((PINC & (1<<PC4)) != 0)

{

LCD\_data(a);

*\_delay\_ms*(10000);

}

}

}

void LCD\_Command( unsigned char cmnd )

{

LCD\_Port &= 0x0F ;

LCD\_Port |= (cmnd & 0xF0); /\* sending upper nibble \*/

LCD\_Port &= ~ (1<<RS); /\* RS=0, command reg. \*/

LCD\_Port |= (1<<EN); /\* Enable pulse \*/

*\_delay\_us*(1);

LCD\_Port &= ~ (1<<EN);

*\_delay\_us*(200);

LCD\_Port &= 0x0F;

LCD\_Port |= (cmnd << 4); /\* sending lower nibble \*/

LCD\_Port |= (1<<EN);

*\_delay\_us*(1);

LCD\_Port &= ~ (1<<EN);

*\_delay\_ms*(2);

}

void LCD\_data( unsigned char data )

{

LCD\_Port = (LCD\_Port & 0x0F) | (data & 0xF0); /\* sending upper nibble \*/

LCD\_Port |= (1<<RS); /\* RS=1, data reg. \*/

LCD\_Port|= (1<<EN);

*\_delay\_us*(1);

LCD\_Port &= ~ (1<<EN);

*\_delay\_us*(200);

LCD\_Port = (LCD\_Port & 0x0F) | (data << 4); /\* sending lower nibble \*/

LCD\_Port |= (1<<EN);

*\_delay\_us*(1);

LCD\_Port &= ~ (1<<EN);

*\_delay\_ms*(2);

}

void LCD\_Init (void) /\* LCD Initialize function \*/

{

LCD\_Dir = 0xFF; /\* Make LCD command port direction as o/p \*/

*\_delay\_ms*(20); /\* LCD Power ON delay always >15ms \*/

LCD\_Command(0x33);

LCD\_Command(0x32); /\* send for 4 bit initialization of LCD \*/

LCD\_Command(0x28); /\* Use 2 line and initialize 5\*7 matrix in (4-bit mode)\*/

LCD\_Command(0x0c); /\* Display on cursor off\*/

LCD\_Command(0x06); /\* Increment cursor (shift cursor to right)\*/

LCD\_Command(0x01); /\* Clear display screen\*/

*\_delay\_ms*(2);

LCD\_Command (0x80); /\* Cursor 1st row 0th position \*/

}

void LCD\_String (char \*str) /\* Send string to LCD function \*/

{

int i;

for(i=0;str[i]!=0;i++) /\* Send each char of string till the NULL \*/

{

LCD\_data (str[i]);

}

}

**Simulation**

**Diagram

Description automatically generated**

**Post Lab Tasks**

**Task 1:**

Implement the in-lab task for the 4-pin communication mode on Proteus. Additionally, add a 5×dip-switch array at an input. Use the dip switch to pass 8-bit ASCII code, one nibble at a time, to be displayed on the LCD. Toggle the 5th switch to indicate that one nibble is ready to load.

**Simulation**

**Diagram

Description automatically generated with medium confidence**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_