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]*

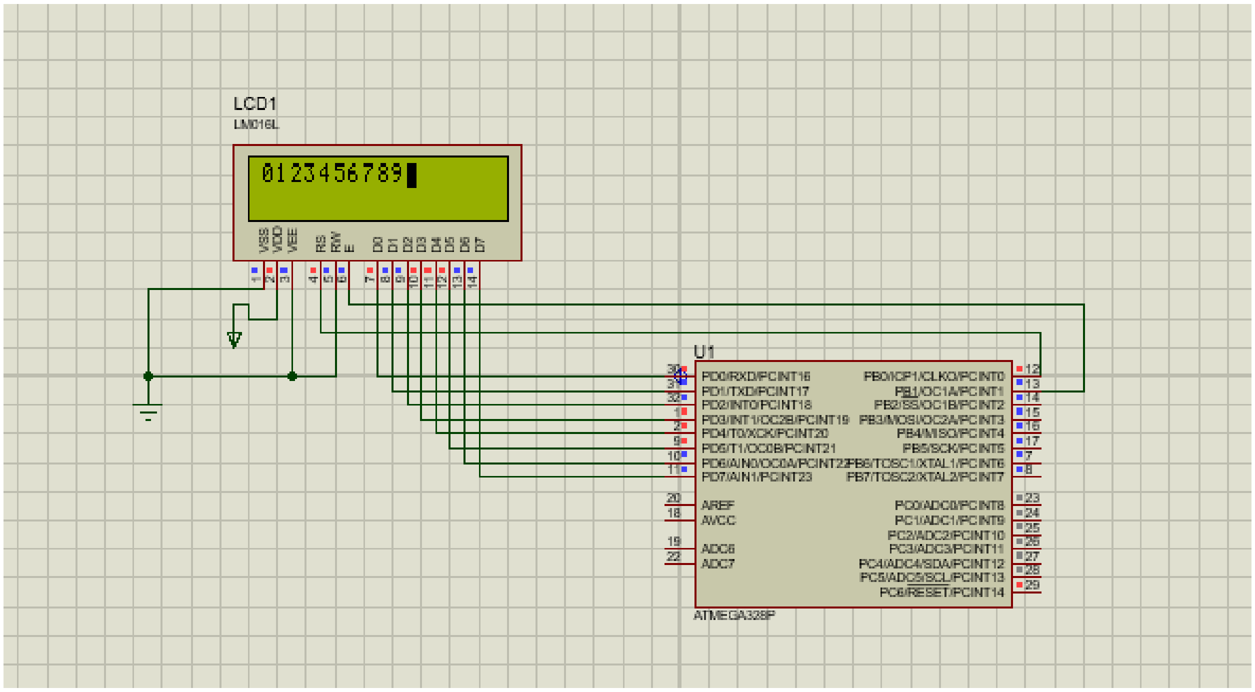
Code

```
.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



Task-3

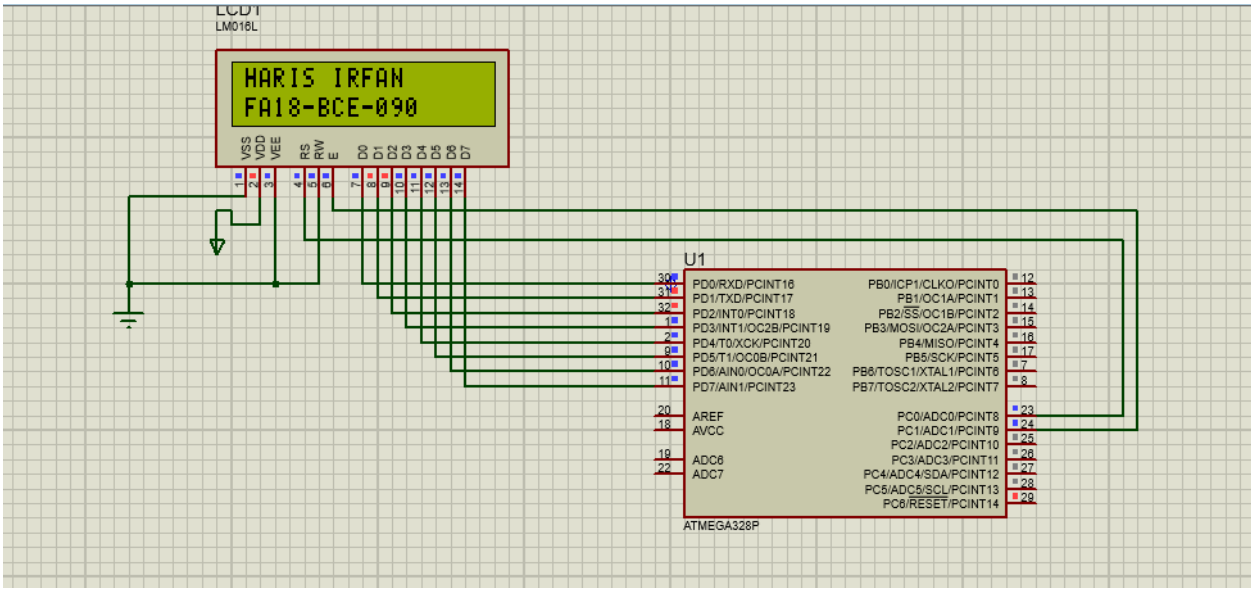
Consider the basic wiring shown between an ATmega328P chip and an LCD in figure 5.2. Write and execute a C-program on Proteus that is able print your name on the first row of the LCD and your roll-number on the second row of the LCD

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);
}
```

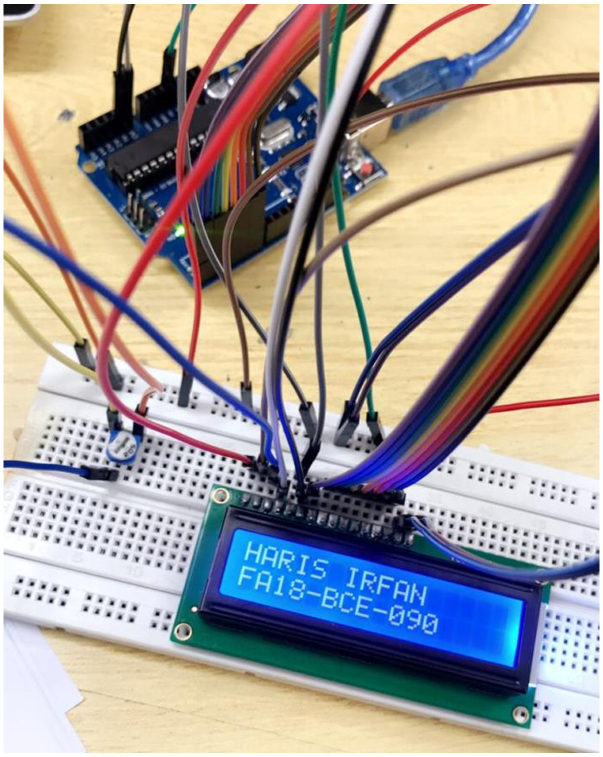
Simulation:



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



Task 2:

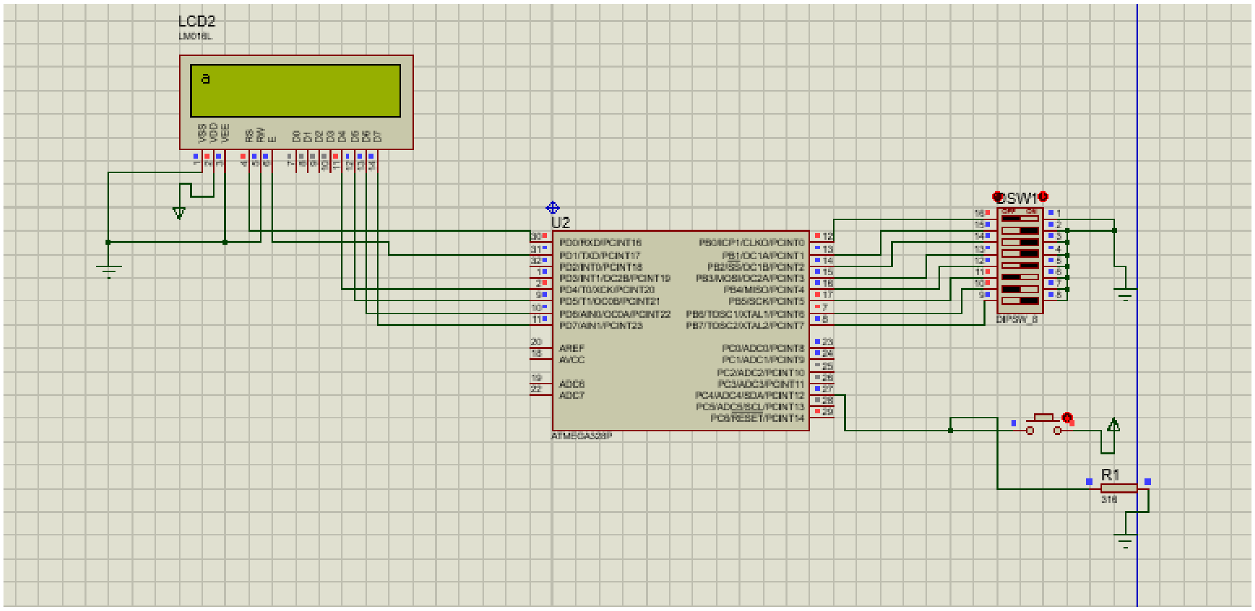
Consider your controller connected to an 8×dipswitch, a push-button and an LCD. Program your controller in a way that whenever the push-button is pressed, the input from the dipswitch should be pushed on to the LCD as an ASCII character. Execute your code on Proteus.

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

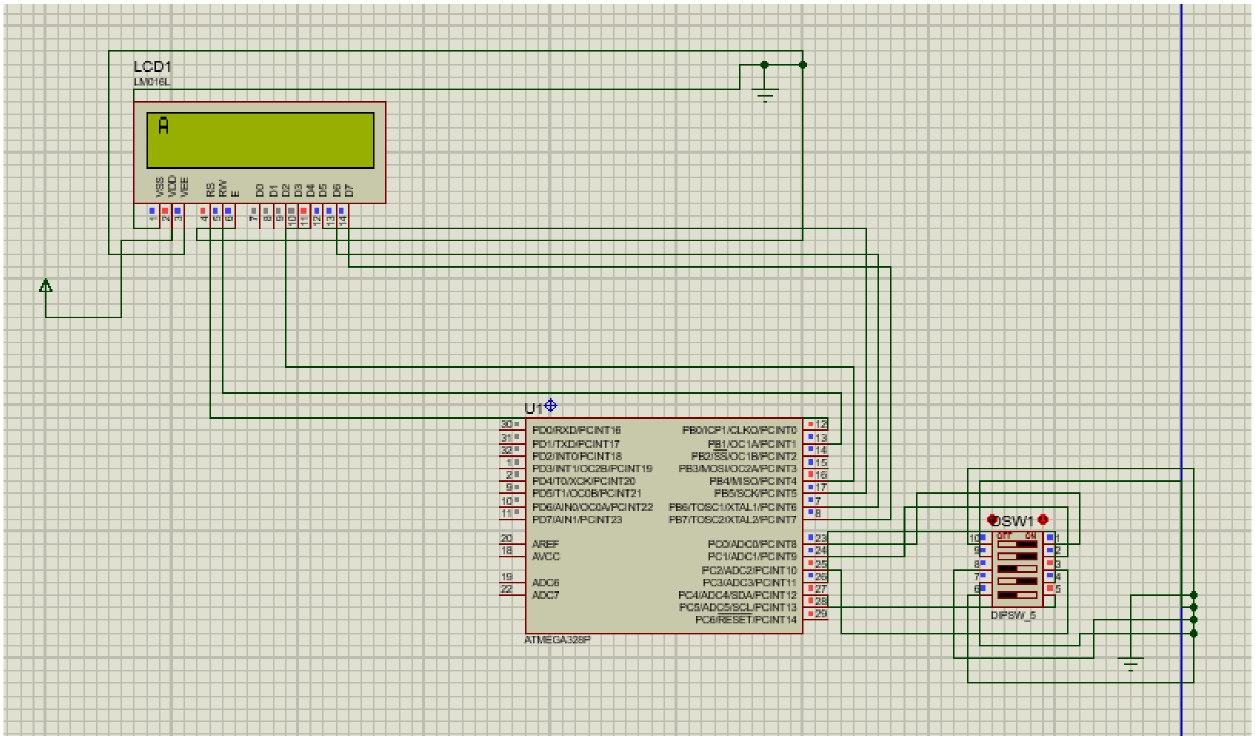


Post Lab Tasks

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


LCD1
LM016L
A
VSS
VDD
VEE
RS
RW
E
D0
D1
D2
D3
D4
D5
D6
D7
1
2
3
4
5
6
7
8
9
10
11
12
13
14
U1
PD0/RXD/PCINT16
PD1/TXD/PCINT17
PD2/INT0/PCINT18
PD3/INT1/OC2B/PCINT19
PD4/T0/XCK/PCINT20
PD5/T1/OC0B/PCINT21
PD6/AIN0/OC0A/PCINT22
PD7/AIN1/PCINT23
AREF
AVCC
ADC6
ADC7
PB0/ICP1/CLKO/PCINT0
PB1/OC1A/PCINT1
PB2/SS/OC1B/PCINT2
PB3/MOSI/OC2A/PCINT3
PB4/MISO/PCINT4
PB5/SCK/PCINT5
PB6/TOSC1/XTAL1/PCINT6
PB7/TOSC2/XTAL2/PCINT7
PC0/ADC0/PCINT8
PC1/ADC1/PCINT9
PC2/ADC2/PCINT10
PC3/ADC3/PCINT11
PC4/ADC4/SDA/PCINT12
PC5/ADC5/SCL/PCINT13
PC6/RESET/PCINT14
ATMEGA328P
DSW1
OFF
ON
DIPSW_5