Department of School of Technology

**Lab Manual**

**SUBJECT: Embedded Systems (18IC311T)**

**6th Semester (B. Tech)**

**(Branch: ICT)**

**Prepared By: Team 5**

Jay Bhgaiya (18BIT040)

Jehil Thakkar (18BIT043)

Jai Lohana (18BIT035)

Deep Mistry (18BIT015)

Kathan Pathak (18BIT051)

****

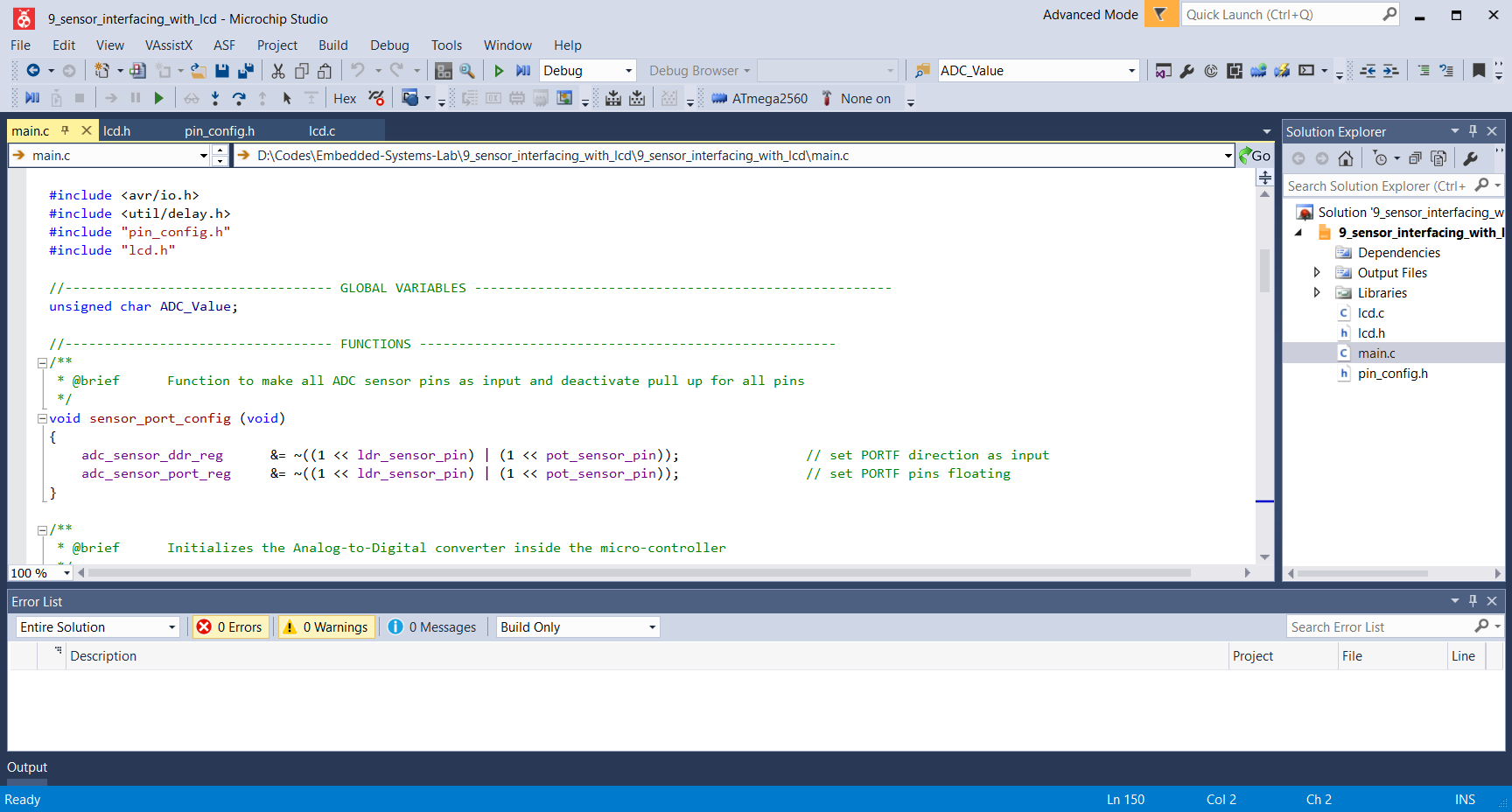
**Pandit Deendayal Energy University**

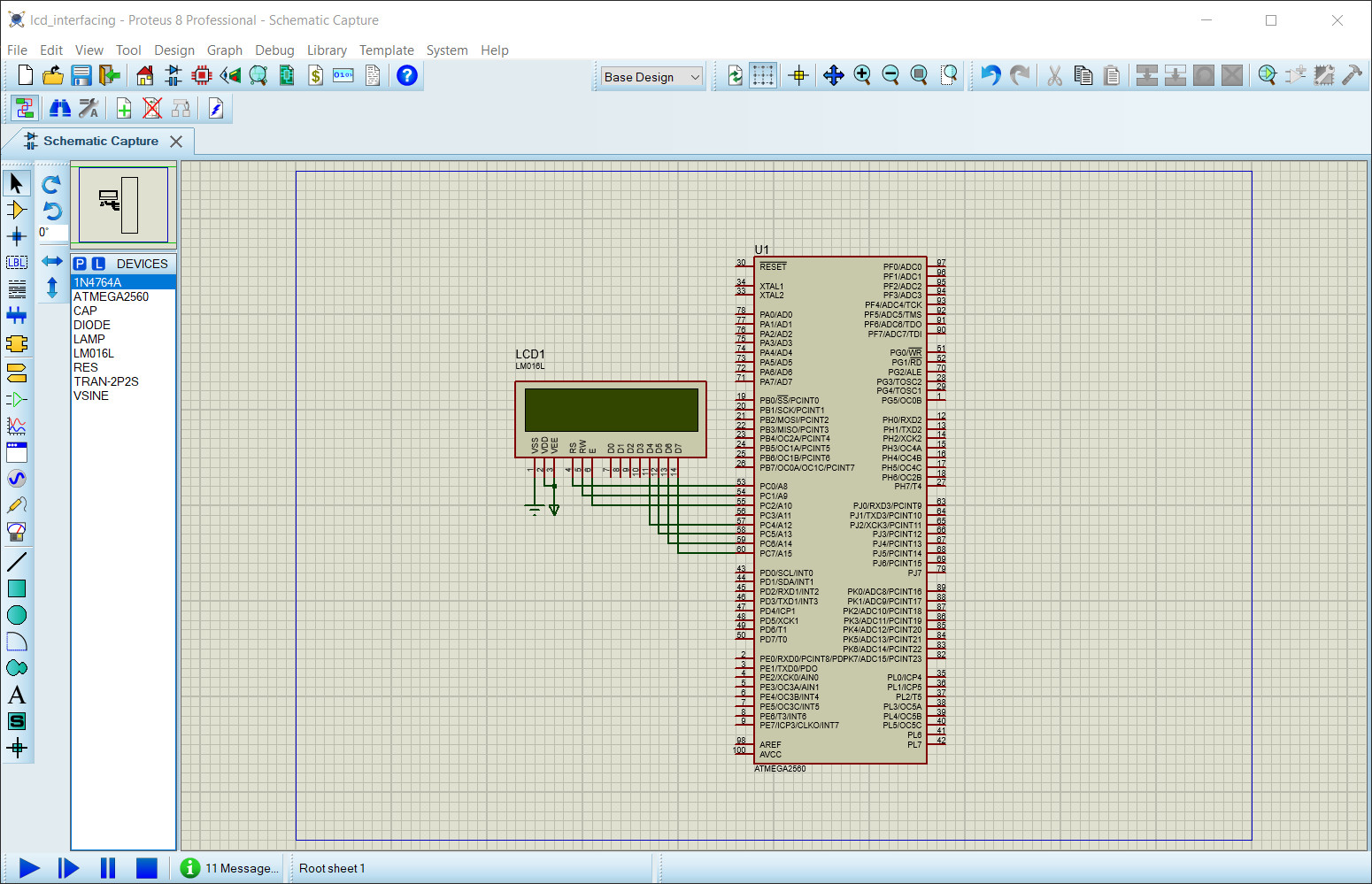
**School of Technology, Gandhinagar - 382426, Gujarat**

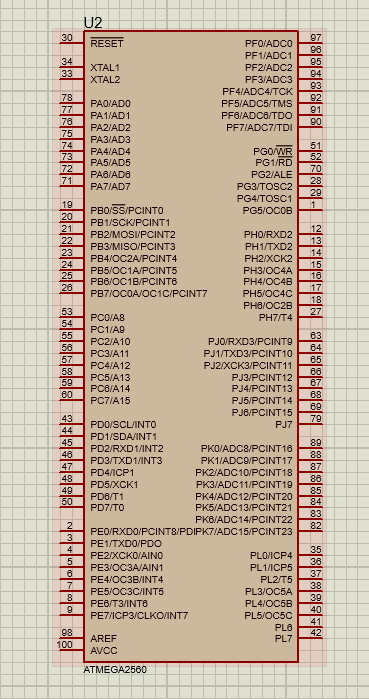
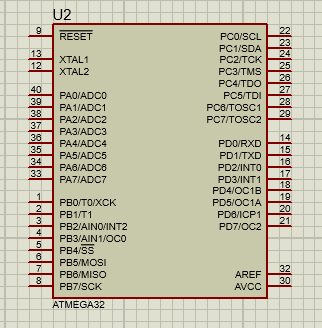
**List of Experiments**

|  |  |  |
| --- | --- | --- |
| Ex. No. | Title | Page No. |
| 1 | Familiarization with IDE and trainer kits/boards. | 3 |
| 2 | Program for blinking LED, pattern generation, timing, sequence generation etc. | 5 |
| 3 | Program for interfacing multi-digit 7 segment display and implementing counter etc. | 8 |
| 4 | Program for interfacing toggle and push button switches. | 10 |
| 5 | Program for interfacing simple keypad and matrix keypad, controlling LEDs using switches. | 13 |
| 6 | Program for interfacing DC motor with AVR microcontroller. | 16 |
| 7 | Program for interfacing stepper motor with AVR microcontroller. | 14 |
| 8 | Program for interfacing LCD and displaying text on it. | 18 |
| 9 | Program for interfacing various sensors and displaying quantity on LCD. | 20 |
| 10 | Program for interfacing RS 232 serial modules and file transfer using it. Use of soft wares like terminal and hyper-terminal. | 22 |

1. **Familiarization with IDE and trainer kits/boards.**

**Atmel Studio: For Programming**

**Proteus 8: For Simulation**

**AVR Micro-Controller: atmega2560 and atmega32**

**2. Program for blinking LED, pattern generation, timing, sequence generation.**

**a) LED Blinking:**

#ifndef *F\_CPU*

#define *F\_CPU* 16000000UL //clock speed is 16MHz

#endif

#include <avr/io.h>

#include <util/delay.h>

int main(void)

{

DDRA = 0xFF;

while (1)

{

PORTA = 0x01;

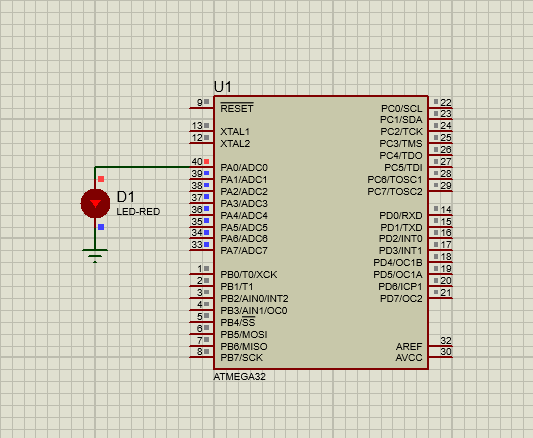
*\_delay\_ms*(1000);

PORTA = 0x00;

*\_delay\_ms*(1000);

}

}

**Output:**

**b) LED Blinking 1 to 8 (Pattern 1)**

#ifndef *F\_CPU*

#define *F\_CPU* 16000000UL //clock speed is 16MHz

#endif

#include <avr/io.h>

#include <util/delay.h>

int main(void)

{

DDRA = 0xFF;

while (1)

{

for (int i=0; i<8; i++)

{

PORTA = (0x01 << i);

*\_delay\_ms*(500);

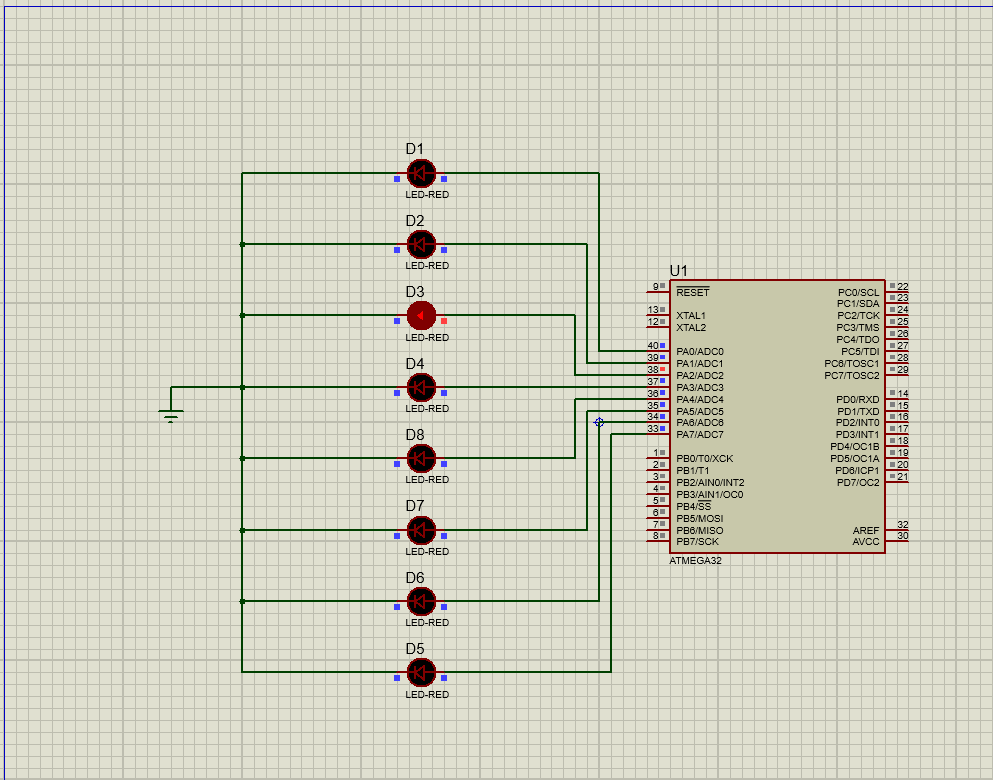
PORTA = 0x00;

}

}

}

**Output:**

****

**c) Led Blinking From Mid (Pattern 2)**

#ifndef *F\_CPU*

#define *F\_CPU* 16000000UL //clock speed is 16MHz

#endif

#include <avr/io.h>

#include <util/delay.h>

int main(void)

{

DDRA = 0xFF;

while (1)

{

for (int i=3; i>=0; i--)

{

PORTA |= (1 << i);

PORTA |= (1 << (7-i));

*\_delay\_ms*(500);

}

for (int i=0; i<4; i++)

{

PORTA &= ~(1 << i);

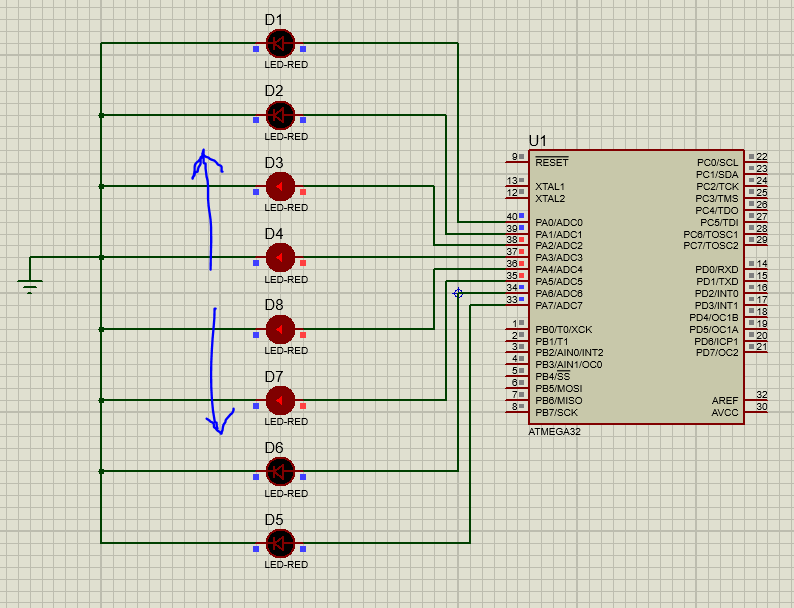
PORTA &= ~(1 << (7-i));

*\_delay\_ms*(500);

}

}

}

**Output:**

**3. Program for interfacing multi-digit 7 segment display and implementing counter.**

**a) Seven Segment LED**

#ifndef *F\_CPU*

#define *F\_CPU* 16000000UL //clock speed is 16MHz

#endif

#include <avr/io.h>

#include <util/delay.h>

int main(void)

{

DDRB = 0xFF;

while (1)

{

char array[] = {0xFC, 0x60, 0xDA, 0xF2, 0x66, 0xB6, 0xBE, 0xE0, 0xFE, 0xF6};

for (int i=0; i<=9; i++)

{

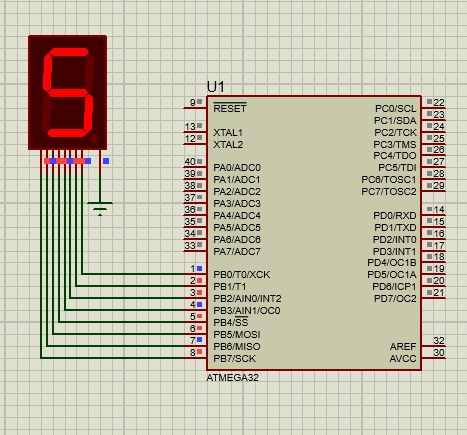
PORTB = array[i];

*\_delay\_ms*(1000);

}

}

}

**Output:**

**b) Seven Segment LED (Counter)**

#ifndef *F\_CPU*

#define *F\_CPU* 16000000UL //clock speed is 16MHz

#endif

#include <avr/io.h>

#include <util/delay.h>

int main(void)

{

DDRB = 0xFF;

DDRA = 0xFF;

char array[] = {0xFC, 0x60, 0xDA, 0xF2, 0x66, 0xB6, 0xBE, 0xE0, 0xFE, 0xF6};

int tenth\_num = 0;

while (1)

{

PORTB = array[tenth\_num++];

for(int i=0; i<10; i++)

{

PORTA = array[i];

*\_delay\_ms*(500);

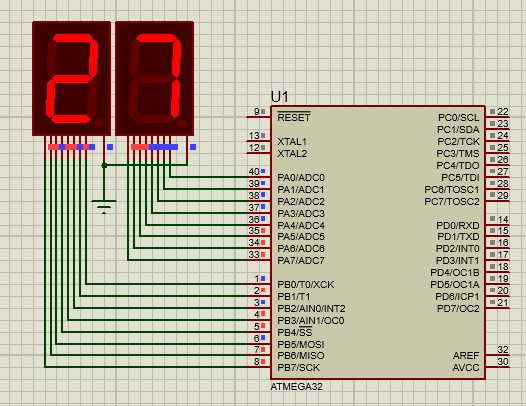
}

if(tenth\_num == 10)

tenth\_num = 0;

}

}

**Output:**

**4. Program for interfacing toggle and push button switches.**

#include <avr/io.h>

#include <util/delay.h>

// Definitions for ATmega2560 and Interfacing of Toggle and push button

#if defined(\_\_AVR\_ATmega2560\_\_)

#define toggle\_button\_ddr\_reg DDRF

#define toggle\_button\_port\_reg PORTF

#define toggle\_button\_pin\_reg PINF

#define toggle\_button\_pin PF1

#define push\_button\_ddr\_reg DDRF

#define push\_button\_port\_reg PORTF

#define push\_button\_pin\_reg PINF

#define push\_button\_pin PF0

#define led\_ddr\_reg DDRA

#define led\_port\_reg PORTA

#define led\_1\_pin PA0

#define led\_2\_pin PA1

#endif

//---------------------------------- FUNCTIONS ---------------------------------------------------------

//-----------------------------CONFIGURATION FUNCTIONS --------------------------------------------------

/\*\*

\* @brief Function to make \*\*ONLY\*\* Toggle and push button Switch pin as input and pull it up internally

\*/

void toggle\_and\_push\_button\_switch\_config (void) {

// Make \*\*ONLY\*\* Toogle Switch pin as input

toggle\_button\_ddr\_reg &= ~( 1 << toggle\_button\_pin );

// Make \*\*ONLY\*\* Toggle Switch pin internally pull-up

toggle\_button\_port\_reg |= ( 1 << toggle\_button\_pin );

// Make \*\*ONLY\*\* Push button Switch pin as input

push\_button\_ddr\_reg &= ~( 1 << push\_button\_pin );

// Make \*\*ONLY\*\* Push button Switch pin internally pull-up

push\_button\_port\_reg |= ( 1 << push\_button\_pin );

}

/\*\*

\* @brief Function to make \*\*ONLY\*\* 'led\_1\_pin' and 'led\_2\_pin' as output and initially set it to low

\*/

void led\_pin\_config (void) {

// Make 'led\_1\_pin' as output

led\_ddr\_reg |= ( 1 << led\_1\_pin );

// Set 'led\_1\_pin' to low initially

led\_port\_reg &= ~( 1 << led\_1\_pin );

// Make 'led\_2\_pin' as output

led\_ddr\_reg |= ( 1 << led\_2\_pin );

// Set 'led\_2\_pin' to low initially

led\_port\_reg &= ~( 1 << led\_2\_pin );

}

//------------------------------------- LED RELATED FUNCTIONS -----------------------------------------

/\*\*

\* @brief Function to set LED 1 pin to high, hence turn on LED 1

\*/

void led\_1\_on(void){

// Turn on all LEDs

led\_port\_reg |= (1 << led\_1\_pin);

}

/\*\*

\* @brief Function to set LED 1 pin to low, hence turn off LED 1

\*/

void led\_1\_off(void){

// Turn off all LEDs

led\_port\_reg &= ~(1 << led\_1\_pin);

}

/\*\*

\* @brief Function to set LED 2 pin to high, hence turn on LED 2

\*/

void led\_2\_on(void){

// Turn on all LEDs

led\_port\_reg |= (1 << led\_2\_pin);

}

/\*\*

\* @brief Function to set LED 2 pin to low, hence turn off LED 2

\*/

void led\_2\_off(void){

// Turn off all LEDs

led\_port\_reg &= ~(1 << led\_2\_pin);

}

//---------------------------------- MAIN ----------------------------------------------------------------

int main(void)

{

// Initialize the necessary devices (Led, Toggle and push button switch) required for the experiment.

toggle\_and\_push\_button\_switch\_config();

led\_pin\_config();

while (1)

{

// If the push button Switch is NOT pressed

if ((push\_button\_pin\_reg & (1 << push\_button\_pin)) == (1 << push\_button\_pin)) {

*\_delay\_ms*(100);

led\_1\_off(); //Turn off LED 1

}

else {

*\_delay\_ms*(100);

led\_1\_on(); //Turn on LED 1

}

if ((toggle\_button\_pin\_reg & (1 << toggle\_button\_pin)) == (1 << toggle\_button\_pin)) {

*\_delay\_ms*(100);

led\_2\_off(); //Turn off LED 2

}

else {

*\_delay\_ms*(100);

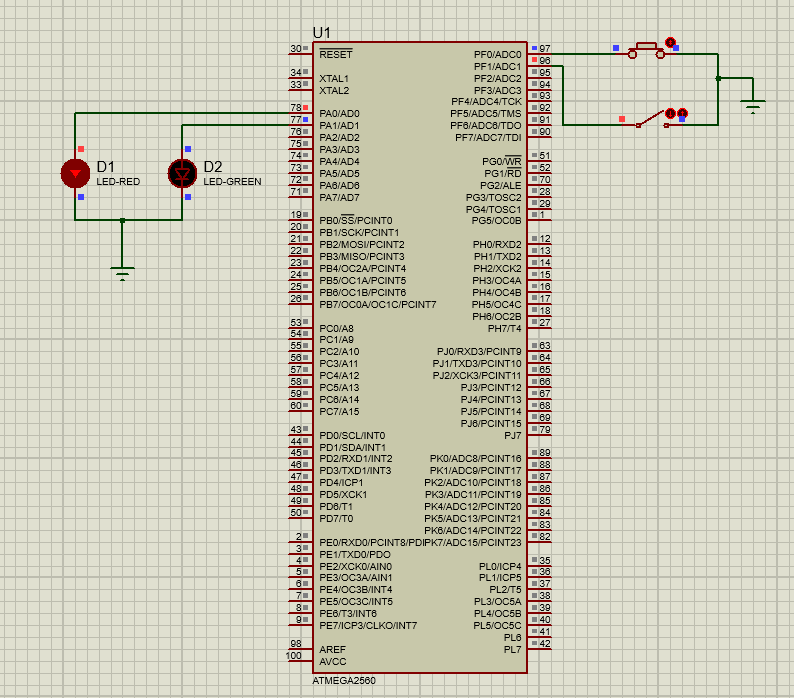
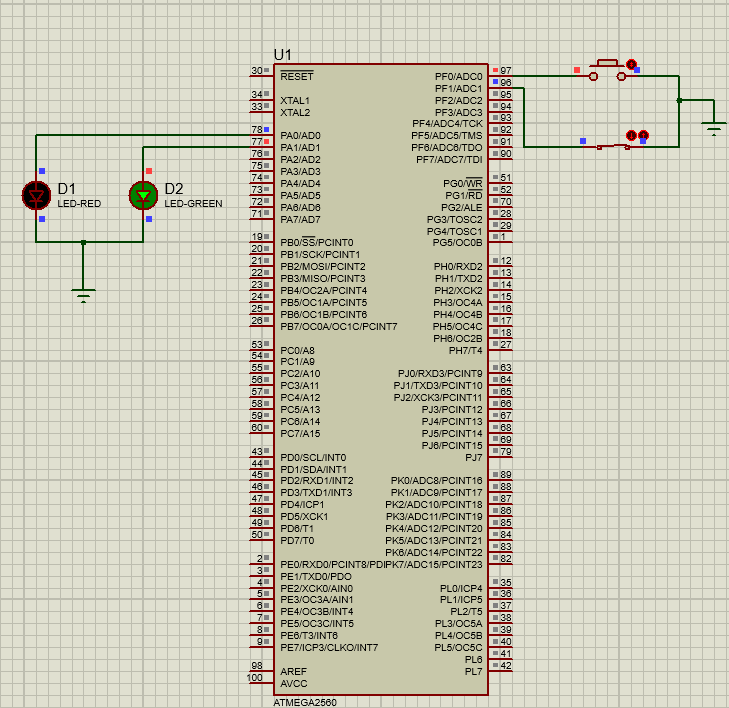
led\_2\_on(); //Turn on LED 2

}

}

}

**Output:**

****

**5. Program for interfacing simple keypad and matrix keypad, controlling LEDs using switches.**

#include <avr/io.h>

#include <util/delay.h>

#if defined(\_\_AVR\_ATmega2560\_\_)

#ifndef *F\_CPU*

#define *F\_CPU* 14745600

#endif

//---------------------------------- INPUT / OUTPUT PERIPHERALS -----------------------------------------------------

// Seven Segment definitions

#define seven\_seg\_ddr\_reg DDRA

#define seven\_seg\_port\_reg PORTA

// Matrix Column definitions

#define matrix\_col\_ddr\_reg DDRF

#define matrix\_col\_port\_reg PORTF

#define matrix\_col\_1\_pin PF0

#define matrix\_col\_2\_pin PF1

#define matrix\_col\_3\_pin PF2

#define matrix\_col\_4\_pin PF3

// Matrix Row Definitions

#define matrix\_row\_ddr\_reg DDRB

#define matrix\_row\_port\_reg PORTB

#define matrix\_row\_pin\_reg PINB

#define matrix\_row\_1\_pin PB0

#define matrix\_row\_2\_pin PB1

#define matrix\_row\_3\_pin PB2

#define matrix\_row\_4\_pin PB3

#endif

// ----------------------------------------- Functions -----------------------------------------

//-----------------------------CONFIGURATION FUNCTIONS --------------------------------------------------

/\*\*

\* @brief Function to make \*\*ONLY\*\* Configuration of Seven Segment Display.

\*/

void seven\_seg\_config(void)

{

// Make \*\*ONLY\*\* all pins as output

seven\_seg\_ddr\_reg = 0xFF;

// Make \*\*ONLY\*\* set all pins initially low

seven\_seg\_port\_reg = 0x00;

}

/\*\*

\* @brief Function to make \*\*ONLY\*\* Configuration of Columns of matrix.

\*/

void matrix\_col\_pin\_config(void)

{

// Make \*\*ONLY\*\* Four column pins as output

matrix\_col\_ddr\_reg |= ((1 << matrix\_col\_1\_pin) | (1 << matrix\_col\_2\_pin) | (1 << matrix\_col\_3\_pin) | (1 << matrix\_col\_4\_pin));

// Make \*\*ONLY\*\* Disabling all columns witout disturbing reamianing pins

matrix\_col\_port\_reg &= ~((1 << matrix\_col\_1\_pin) | (1 << matrix\_col\_2\_pin) | (1 << matrix\_col\_3\_pin) | (1 << matrix\_col\_4\_pin));

}

/\*\*

\* @brief Function to make \*\*ONLY\*\* Configuration of Rows of matrix.

\*/

void matrix\_row\_pin\_config(void)

{

// Make \*\*ONLY\*\* Four row pins defined as input

matrix\_row\_ddr\_reg &= ~((1 << matrix\_row\_1\_pin) | (1 << matrix\_row\_2\_pin) | (1 << matrix\_row\_3\_pin) | (1 << matrix\_row\_4\_pin));

// Make \*\*ONLY\*\* Disabling all pull-up resister on four rows witout disturbing reamianing pins

matrix\_row\_port\_reg &= ~((1 << matrix\_row\_1\_pin) | (1 << matrix\_row\_2\_pin) | (1 << matrix\_row\_3\_pin) | (1 << matrix\_row\_4\_pin));

}

// ----------------------------------------- Main -----------------------------------------

int main(void)

{

seven\_seg\_config();

matrix\_col\_pin\_config();

matrix\_row\_pin\_config();

while (1)

{

char array[] = {0xFC, 0x60, 0xDA, 0xF2, 0x66, 0xB6, 0xBE, 0xE0, 0xFE, 0xF6, 0xEE, 0x3E, 0x9C, 0x7A, 0x9E, 0x8E};

// { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}

int key=0, column = 0, temp = 0;

for (column=1,temp=1; column<=4; temp\*=2,column++)

{

// Make \*\*ONLY\*\* Disabling all pull-up resister on four rows witout disturbing reamianing pins

matrix\_col\_port\_reg &= ~((1 << matrix\_col\_1\_pin) | (1 << matrix\_col\_2\_pin) | (1 << matrix\_col\_3\_pin) | (1 << matrix\_col\_4\_pin));

matrix\_col\_port\_reg |= temp;

// Reading rows data and identify the key

switch(matrix\_row\_pin\_reg & ((1 << matrix\_row\_1\_pin) | (1 << matrix\_row\_2\_pin) | (1 << matrix\_row\_3\_pin) | (1 << matrix\_row\_4\_pin)))

{

case (1 << matrix\_row\_1\_pin):{ // row1

key = column;

}break;

case (1 << matrix\_row\_2\_pin):{ // row2

key = 4 + column;

}break;

case (1 << matrix\_row\_3\_pin):{ // row3

key = 8 + column;

}break;

case (1 << matrix\_row\_4\_pin):{ // row4

key = 12 + column;

}break;

}

*\_delay\_ms*(10); // Key debounce

}

if((key < 0) || (key > 15) ){

seven\_seg\_port\_reg = array[0];

}

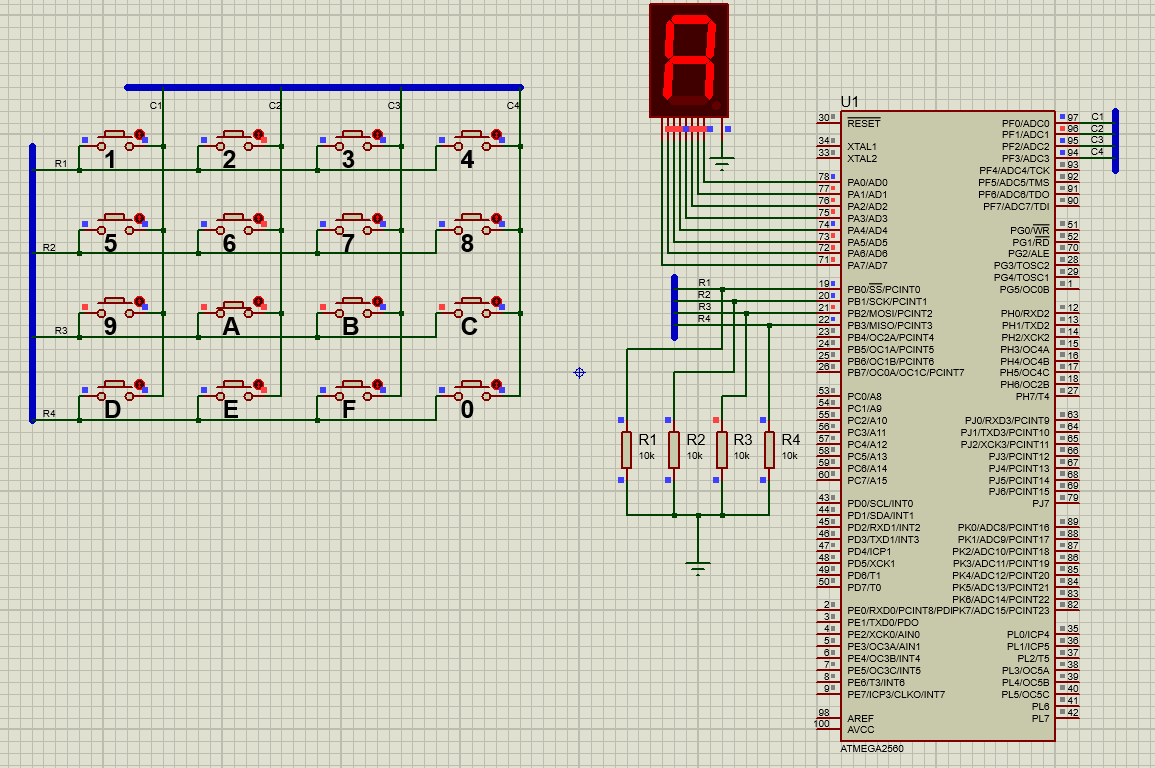
else{

seven\_seg\_port\_reg = array[key];

}

}

}

**Output:**

**6. Program for interfacing DC motor with AVR microcontroller.**

#include<stdio.h>

#define *F\_CPU* 16000000UL

#include<avr/io.h>

#include<util/delay.h>

int main(void){

DDRA = 0xFF;

while(1)

{

PORTA =0x00;

*\_delay\_ms*(100);

PORTA =0x06;

*\_delay\_ms*(100);

PORTA =0x28;

*\_delay\_ms*(100);

PORTA =0x1E;

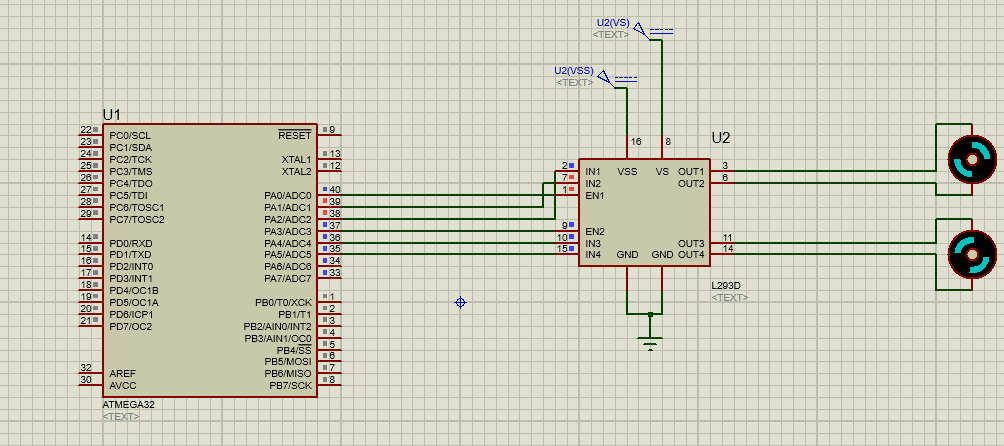
*\_delay\_ms*(100);

PORTA =0x2D;

*\_delay\_ms*(100);

}

}

**Output:**

**7. Program for interfacing stepper motor with AVR microcontroller.**

#define *F\_CPU* 16000000UL

#include <avr/io.h>

#include <util/delay.h>

int main(void)

{

int period;

DDRC = 0x0F; /\* Make PORTD lower pins as output \*/

period = 100; /\* Set period in between two steps \*/

while (1)

{

/\* Rotate Stepper Motor clockwise with Half step sequence \*/

for(int i=0;i<12;i++)

{

PORTC = 0x09;

*\_delay\_ms*(period);

PORTC = 0x08;

*\_delay\_ms*(period);

PORTC = 0x0C;

*\_delay\_ms*(period);

PORTC = 0x04;

*\_delay\_ms*(period);

PORTC = 0x06;

*\_delay\_ms*(period);

PORTC = 0x02;

*\_delay\_ms*(period);

PORTC = 0x03;

*\_delay\_ms*(period);

PORTC = 0x01;

*\_delay\_ms*(period);

}

PORTC = 0x09; /\* Last step to initial position \*/

*\_delay\_ms*(period);

*\_delay\_ms*(1000);

/\* Rotate Stepper Motor Anticlockwise with Full step sequence \*/

for(int i=0;i<12;i++)

{

PORTC = 0x09;

*\_delay\_ms*(period);

PORTC = 0x03;

*\_delay\_ms*(period);

PORTC = 0x06;

*\_delay\_ms*(period);

PORTC = 0x0C;

*\_delay\_ms*(period);

}

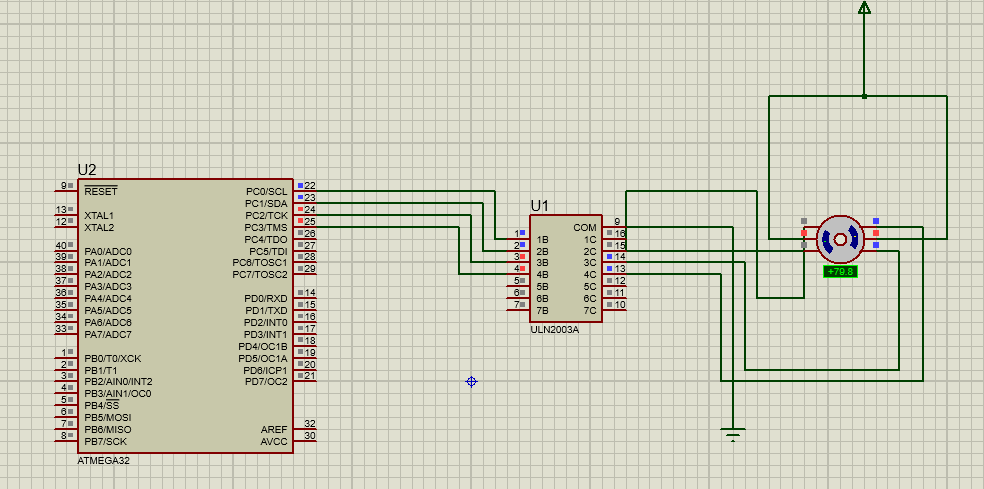
PORTC = 0x09;

*\_delay\_ms*(period);

*\_delay\_ms*(1000);

}

}

**Output:**

**8. Program for interfacing LCD and displaying text on it.**

#include <avr/io.h>

#define *F\_CPU* 16000000UL

#include <stdio.h>

#include <util/delay.h>

void command (unsigned char cmd)

{

PORTC = 0X02;

PORTD = cmd;

PORTC = 0X00;

*\_delay\_ms*(15);

}

void lcd\_data(unsigned char data)

{

PORTC = 0X03;

PORTD = data;

PORTC = 0X01;

*\_delay\_ms*(15);

}

void lcd\_print(char \*p)

{

while(\*p)

{

lcd\_data(\*p++);

}

}

int main(void)

{

DDRC=0XFF;//This register is used for selecting the R/S and R/W pin.

DDRD=0XFF;//This register is used to give the data or commands.

command(0x38);//Activated 2 lines in 8-bit mode.

command(0X0F);//Display is ON, cursor is blinking.

command(0x01);//Clearing the display.

while(1)

{

command(0X80);//Forced the cursor to first position of first line.

lcd\_print("Embedded Systems");

*\_delay\_ms*(1000);

command(0XC0);//Forced the cursor to the first position of second line.

lcd\_print("Exp No. : 8");

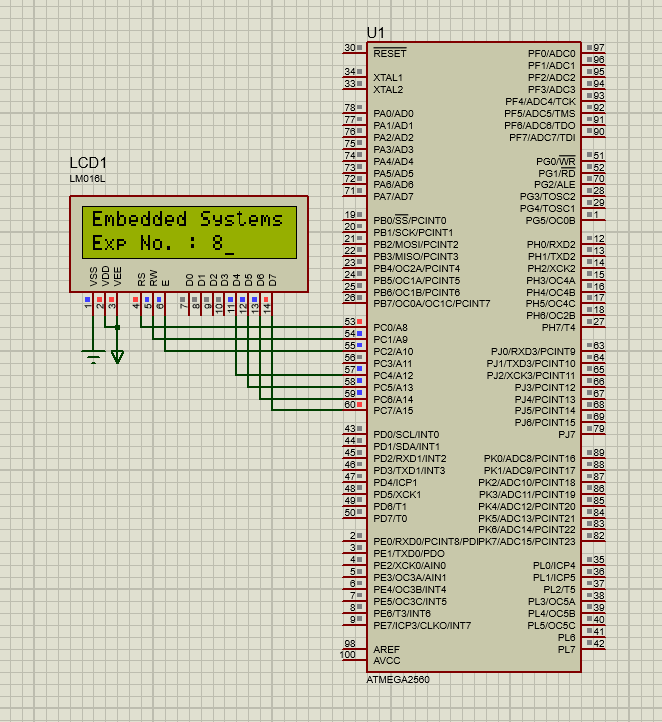
*\_delay\_ms*(1000);

command(0X01);//Clearing the display.

*\_delay\_ms*(1000);

}

}

**Output:**

**9. Program for interfacing various sensors and displaying quantity on LCD.**

#ifndef *F\_CPU*

#define *F\_CPU* 1600000UL

#endif

#include <avr/io.h>

#include <util/delay.h>

#include "LCD/lcd.h"

void adc\_init()

{

// AREF = AVcc

ADMUX = (1<<REFS0);

// ADC Enable and prescaler of 128

ADCSRA = (1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);

}

// read adc value

*uint16\_t* adc\_read(*uint8\_t* ch)

{

// select the corresponding channel 0~7

ch &= 0b00000111; // AND operation with 7

ADMUX = (ADMUX & 0xF8)|ch;

// start single conversion

// write '1' to ADSC

ADCSRA |= (1<<ADSC);

// wait for conversion to complete

// ADSC becomes '0' again

while(ADCSRA & (1<<ADSC));

return (ADC);

}

int main()

{

DDRB=0xff;

*uint16\_t* adc\_result0;

int temp;

int far;

char buffer[10];

// initialize adc and lcd

adc\_init();

lcd\_init(LCD\_DISP\_ON\_CURSOR); //CURSOR

lcd\_clrscr();

lcd\_gotoxy(0,0);

*\_delay\_ms*(50);

while(1)

{

adc\_result0 = adc\_read(0); // read adc value at PA0

temp=adc\_result0/2.01; // finding the temperature

//lcd\_gotoxy(0,0);

//lcd\_puts("Adc=");

//itoa(adc\_result0,buffer,10); //display ADC value

//lcd\_puts(buffer);

lcd\_gotoxy(0,0);

itoa(temp,buffer,10);

lcd\_puts("Temp="); //display temperature

lcd\_puts(buffer);

lcd\_gotoxy(7,0);

lcd\_puts("C");

far=(1.8\*temp)+32;

lcd\_gotoxy(9,0);

itoa(far,buffer,10);

lcd\_puts(buffer);

lcd\_gotoxy(12,0);

lcd\_puts("F");

*\_delay\_ms*(1000);

if(temp>=30)

{lcd\_clrscr();

lcd\_home();

lcd\_gotoxy(0,1);

lcd\_puts("FAN ON");

PORTB=(1<<PINB0);

}

if (temp<=30)

{

lcd\_clrscr();

lcd\_home();

lcd\_gotoxy(7,1);

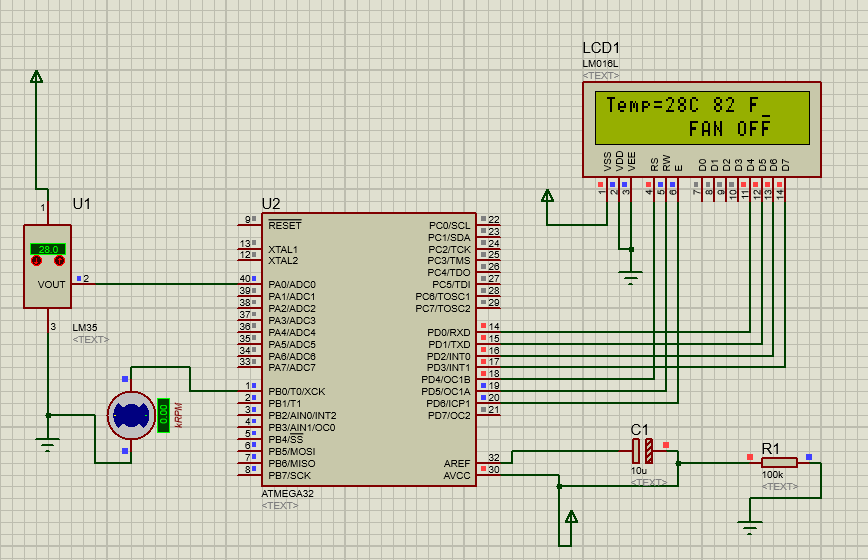
lcd\_puts("FAN OFF");

PORTB=(0<<PINB0);

}

}

}

**Output:**

**10. Program for interfacing RS 232 serial modules and file transfer using it. Using of software like terminal and hyper-terminal.**

// Program to receive data from USART and displaying it on LCD

/\*

Receive data from serial port and display it on LCD

LCD DATA port----PORT A

ctrl port------PORT B

rs-------PB0

rw-------PB1

en-------PB2

using external clock frequency 12MHz

\*/

#define *F\_CPU* 8000000UL

#define USART\_BAUDRATE 9600 // Baud Rate value

#define BAUD\_PRESCALE (((*F\_CPU* / (USART\_BAUDRATE \* 16UL))) - 1)

#include<avr/io.h>

#include<util/delay.h>

#define LCD\_DATA PORTA //LCD data port

#define ctrl PORTB

#define en PB2 // enable signal

#define rw PB1 // read/write signal

#define rs PB0 // register select signal

void LCD\_cmd(unsigned char cmd);

void init\_LCD(void);

void LCD\_write(unsigned char data);

void LCD\_clear();

void usart\_init();

void usart\_putch(unsigned char send);

unsigned int usart\_getch();

int main()

{

unsigned char value;

DDRA=0xff; // LCD\_DATA port as output port

DDRB=0x07; // signal as out put

init\_LCD(); //initialization of LCD

*\_delay\_ms*(50); // delay of 50 milli seconds

usart\_init(); // initialization of USART

while(1)

{

value=usart\_getch(); // Call a function to get data from serial port

LCD\_cmd(0xC0); // to go in second line and zeroth position on LCD

LCD\_write(value); // write data to LCD

}

return 0;

}

void init\_LCD(void)

{

LCD\_cmd(0x38); // initialization of 16X2 LCD in 8bit mode

*\_delay\_ms*(1);

LCD\_cmd(0x01); // clear LCD

*\_delay\_ms*(1);

LCD\_cmd(0x0E); // cursor ON

*\_delay\_ms*(1);

LCD\_cmd(0x80); // ---8 go to first line and --0 is for 0th position

*\_delay\_ms*(1);

return;

}

void LCD\_cmd(unsigned char cmd)

{

LCD\_DATA=cmd;

ctrl =(0<<rs)|(0<<rw)|(1<<en);

*\_delay\_ms*(1);

ctrl =(0<<rs)|(0<<rw)|(0<<en);

*\_delay\_ms*(50);

return;

}

void LCD\_write(unsigned char data)

{

LCD\_DATA= data;

ctrl = (1<<rs)|(0<<rw)|(1<<en);

*\_delay\_ms*(1);

ctrl = (1<<rs)|(0<<rw)|(0<<en);

*\_delay\_ms*(50);

return ;

}

void usart\_init()

{

UCSRB |= (1 << RXEN) | (1 << TXEN); // Turn on the transmission and reception circuitry

UCSRC |= (1 << URSEL) | (1<<USBS) | (1 << UCSZ0) | (1 << UCSZ1);

// Use 8-bit character sizes

UBRRL = BAUD\_PRESCALE;

// Load lower 8-bits of the baud rate value into the low byte of the UBRR register

UBRRH = (BAUD\_PRESCALE >> 8); // Load upper 8-bits of the baud rate value..

// into the high byte of the UBRR register

}

unsigned int usart\_getch()

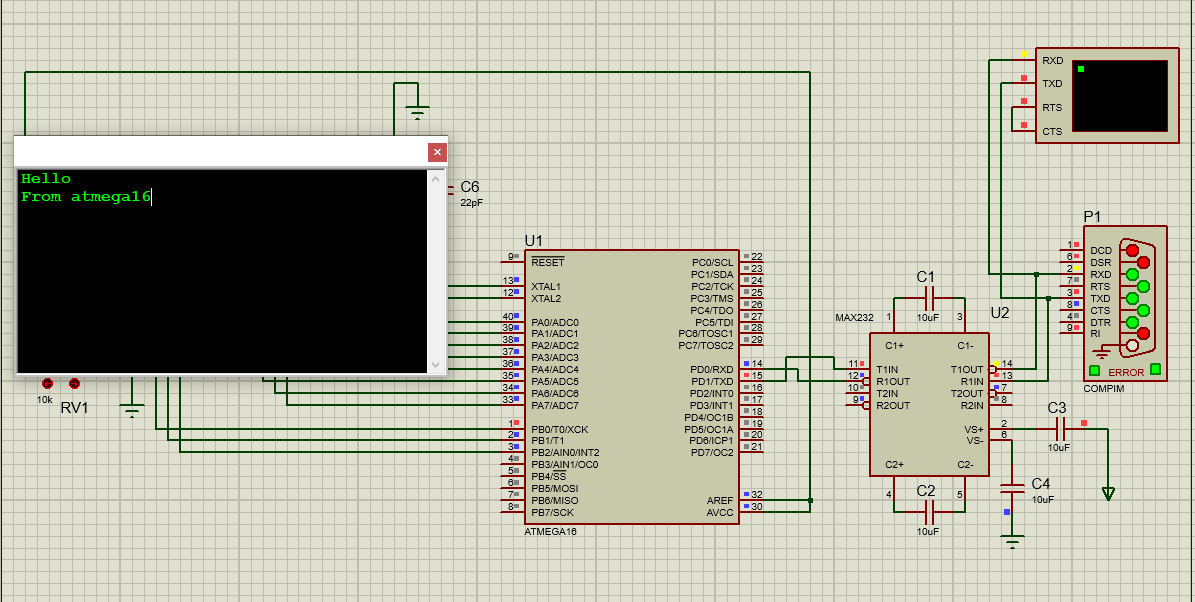
{

while ((UCSRA & (1 << RXC)) == 0);

// Do nothing until data has been received and is ready to be read from UDR

return(UDR); // return the byte

}

**Output:**