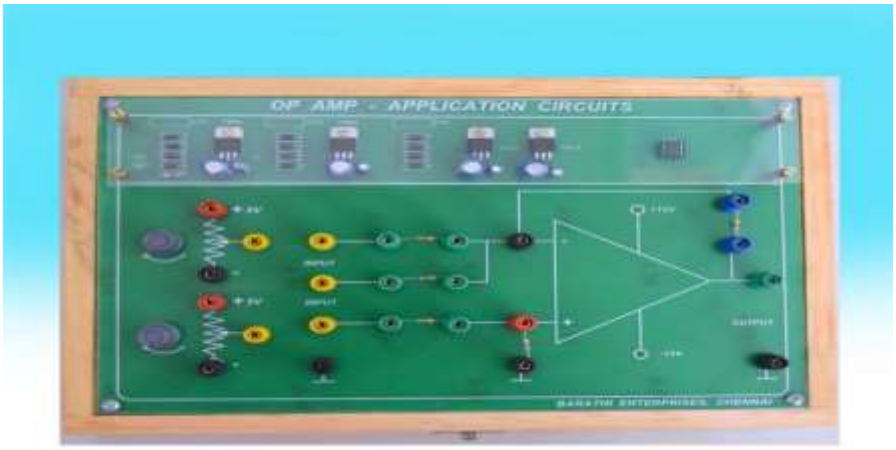
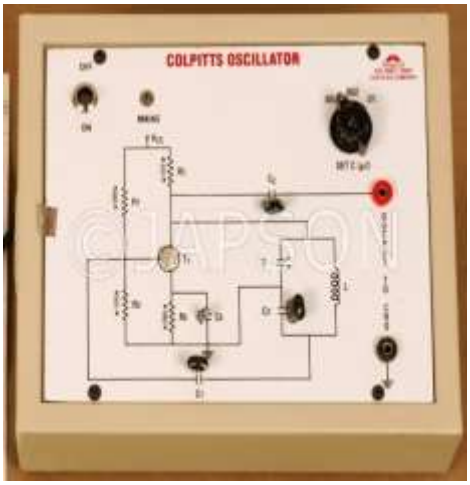
SeethalakshmiRamaswami College (Autonomous)
Affiliated to Bharathidasan University
Accredited with "A" Grade (3rd cycle) by NAAC
Tiruchirappalli – 620 002

M.Sc. PROGRAMME

Department of Electronics





16-pin LCD, Pin 15 Led+ and Pin 16 is LED-

Pin No.	Name	Function
1	V_{ss}	Ground
2	V_{dd}	+ve supply
3	V_{ee}	Contrast
4	RS	Register Select
5	R/W	Read/Write
6	E	Enable
7	D0	Data bit 0
8	D1	Data bit 1
9	D2	Data bit 2
10	D3	Data bit 3
11	D4	Data bit 4
12	D5	Data bit 5
13	D6	Data bit 6
14	D7	Data bit 7

Source: Everyday Practical Electronics, 1997

Applications of Operational Amplifier

Aim: To study the Operational amplifier as Adder, Subtractor, Comparator, Integrator&Differentiator.

Components and Equipments required:IC741, Regulated DC power supply (2), Resistors (10k Ω (4), 1k Ω (2)), Capacitors (0.01 μF (2)), Multimeter, Signal generator, CRO, CRO probes, Bread board and connecting wires.

Part-1: Adder, Subtractor& Comparator

Circuit Diagrams:

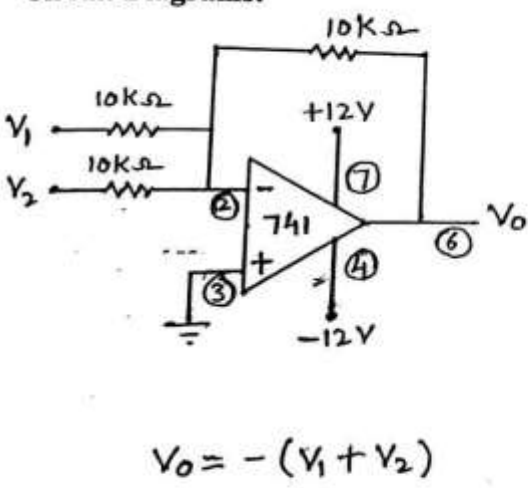


Figure 1: Inverting Adder

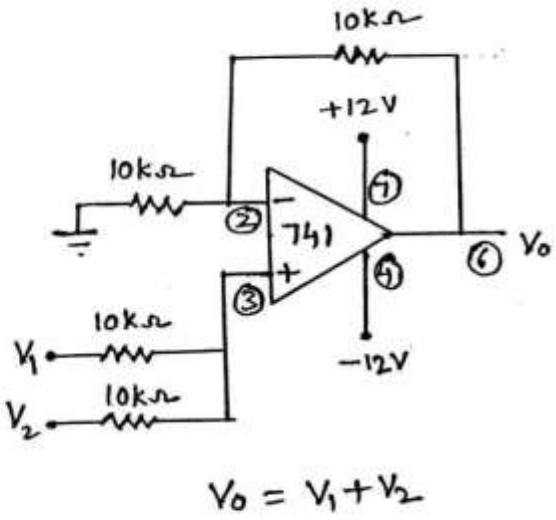


Figure 2: Non-inverting Adder

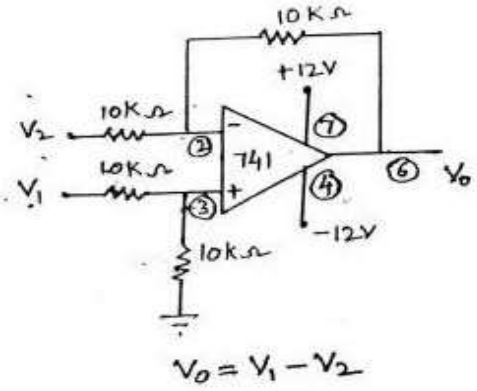


Figure 3: Subtractor

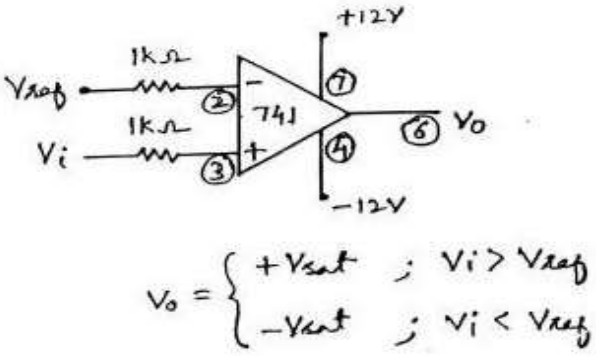


Figure 4: Comparator

Procedure:

1. Connect the circuits as shown in Fig. 1, Fig. 2, Fig. 3 & Fig. 4.
2. Apply the inputs from a regulated power supply.
3. Measure the output voltage using a multimeter.
4. Repeat the above steps for different values of inputs.

Observations:

Inverting Adder

S. No.	V_1	V_2	V_o

Non-inverting Adder

S. No.	V_1	V_2	V_o

Subtractor

S. No.	V_1	V_2	V_o

Comparator

S. No.	V_1	V_2	V_o

Part-2: Integrator & Differentiator

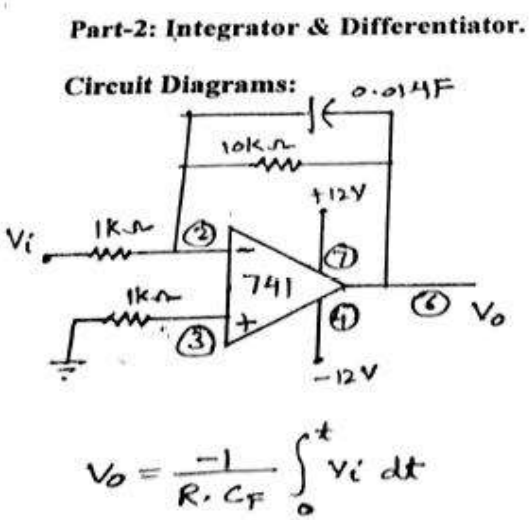


Figure 5: Integrator

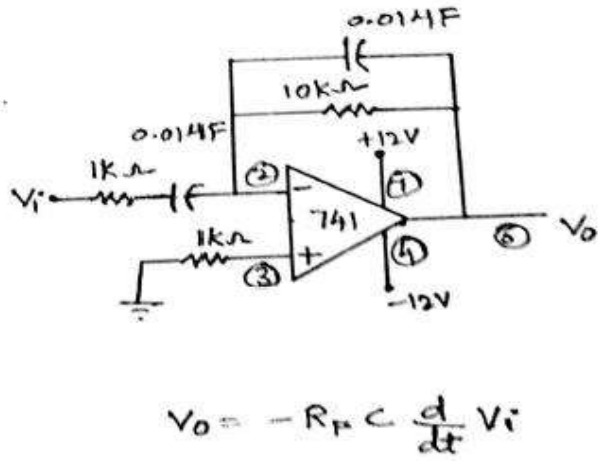


Figure 6: Differentiator

Procedure:

1. Connect the circuits as shown in Fig. 5 & Fig. 6.
2. Apply a square wave of 3 kHz frequency from Function generator, as input to the Integrator. Observe the output waveform.
3. Apply a triangular wave of 3 kHz frequency from Function generator, as input to the Differentiator. Observe the output waveform.
4. Draw the input & output waveforms for above circuits on a graph sheet.

Expected Waveforms:

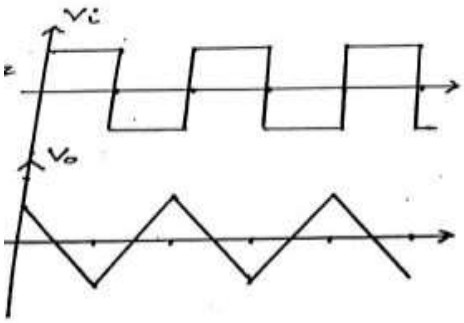


Figure 7: Integrator

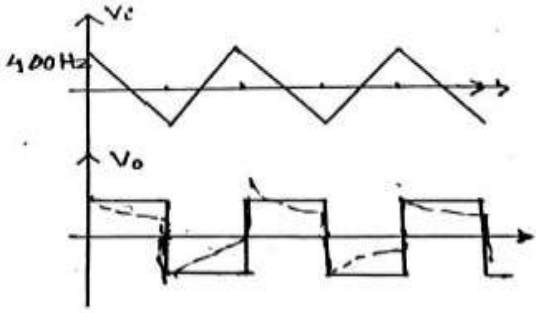


Figure 8: Differentiator

Results: The Operational amplifier is studied as Adder, Subtractor, Comparator, Integrator &Differentiator, and their outputs are verified.

Hartley &Colpitts Oscillators

Aim: To study the operation of Hartley &Colpitts oscillator circuits and to determine theirfrequency of oscillations.

Components and Equipments required:HI-Q electronics Hartley oscillator trainer, HI-Qelectronics Colpitts oscillator trainer, CRO, CRO probe & connecting patch cords.

Circuit Diagram:

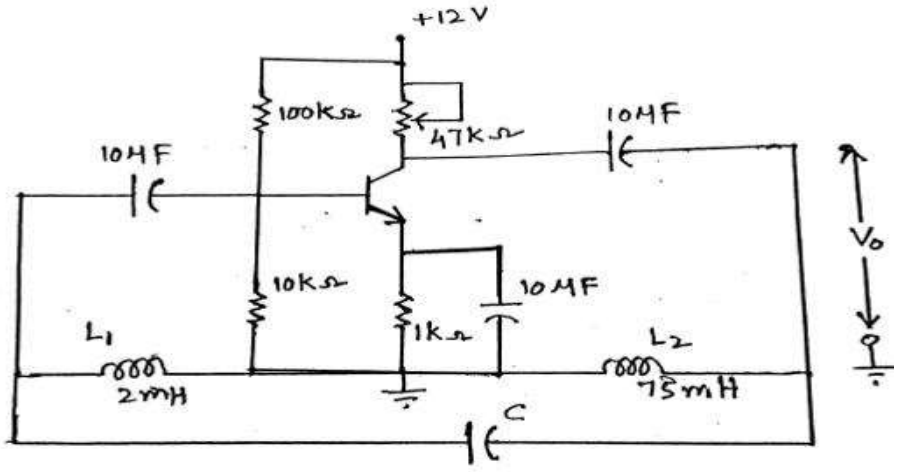


Fig. 1: Hartley oscillator

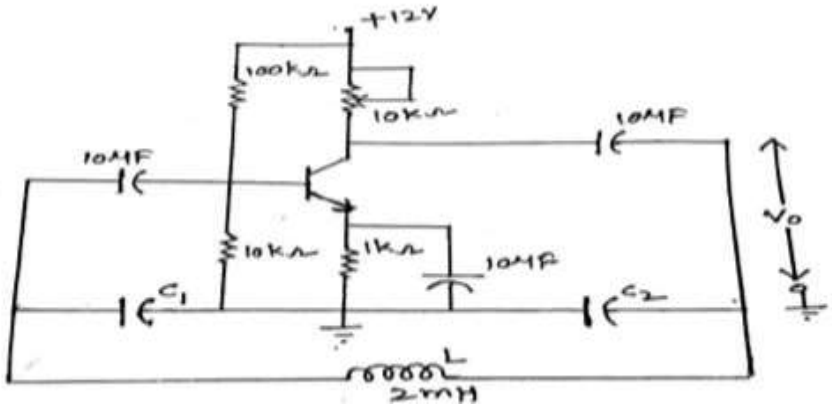


Fig : 2:Colpitts oscillator

Procedure:

Part-1: Hartley Oscillator

1) Complete the circuit as shown in Fig. 1, by connecting the capacitor C1 provided on panel of the trainer.
2) Switch ON the power supply for trainer.
3) Connect the CRO to the output terminals & vary the Rc till the stable oscillations are obtained on the CRO.
4) Now measure the frequency of oscillations practically.
5) Repeat the above steps for different values of capacitors C2, C3 & C4.
6) Draw the oscillations obtained on a graph sheet.

Part-2: Colpitts Oscillator

1) Complete the circuit as shown in Fig. 2, by connecting the capacitors C1 & C2 provided on panel of the trainer.
2) Switch ON the power supply for trainer.
3) Connect the CRO to the output terminals & measure the frequency of oscillations practically.
4) Repeat the above steps by selecting another pair of capacitors C1 & C2.
5) Draw the oscillations obtained on a graph sheet.

Hartley Oscillator

S. No.	C	L1	L2	$f_0 = \frac{1}{2\pi\sqrt{(L_1+L_2)C}}$	T	f = 1/T

Colpitts Oscillator

S. No.	C1	C2	L	$f_0 = \frac{1}{2\pi}\sqrt{\frac{C_1+C_2}{LC_1C_2}}$	T	f = 1/T

Results:The operation of Hartley &Colpitts oscillator circuits is studied, & their frequency ofoscillations is verified with the theoretical values.

1. LCD INTERFACING

```
#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>
#include "LCD_16x2_H_file.h"
#define LCD_Dir  DDRB
#define LCD_Port PORTB

int main()
{
	LCD_Init();
	LCD_String("WELCOME TO THE");
	LCD_Command(0xC0);
	LCD_String("DEPARTMENT OF ELECTRONICS");
	while(1);
}
```

LCD HEADERFILE

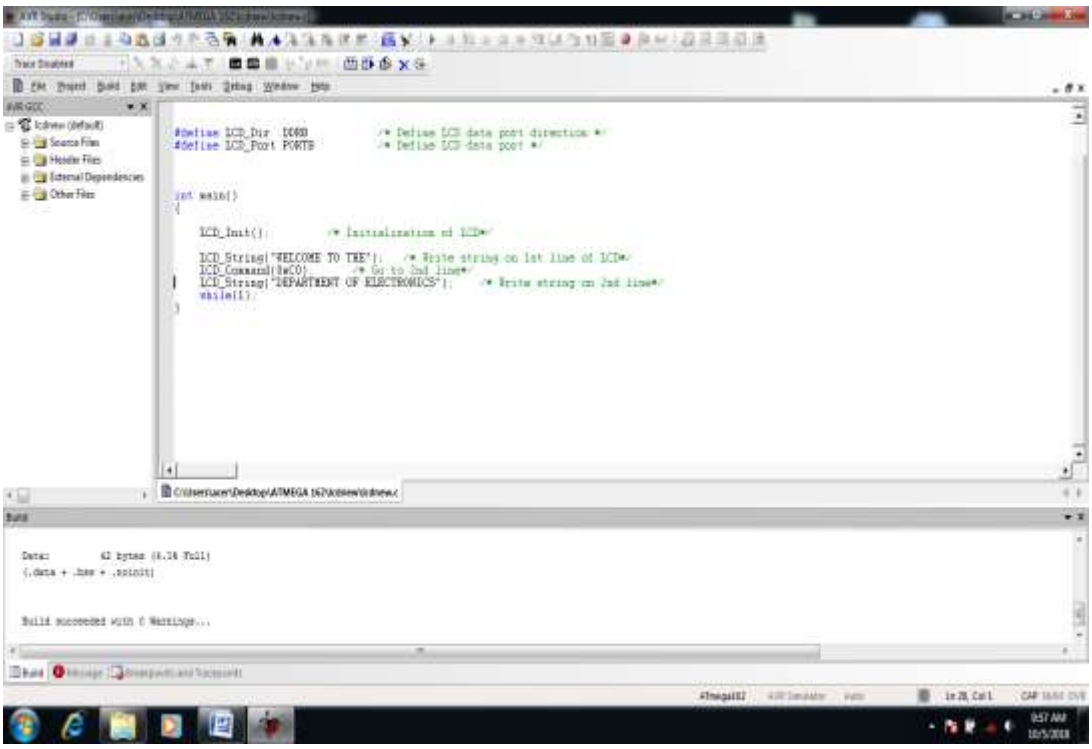
```
#ifndef LCD_16x2_H_H_
#define LCD_16x2_H_H_
#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>
```

```
#define LCD_Data_Dir DDRB

#define LCD_Command_Dir DDRC

#define LCD_Data_Port PORTB

#define LCD_Command_Port PORTC

#define RS PC2

#define RW PC3

#define EN PC4


voidLCD_Command (char);

voidLCD_Char (char);

voidLCD_Init (void);

voidLCD_String (char*);

voidLCD_String_xy (char,char,char*);

voidLCD_Clear (void);

voidLCD_Command (char cmd)

{

        LCD_Data_Port = cmd;
        LCD_Command_Port&= ~((1<<RS)|(1<<RW));
        LCD_Command_Port |= (1<<EN);

        _delay_us(1);

        LCD_Command_Port&= ~(1<<EN);

        _delay_ms(3);

}

voidLCD_Char (char char_data)

{
```

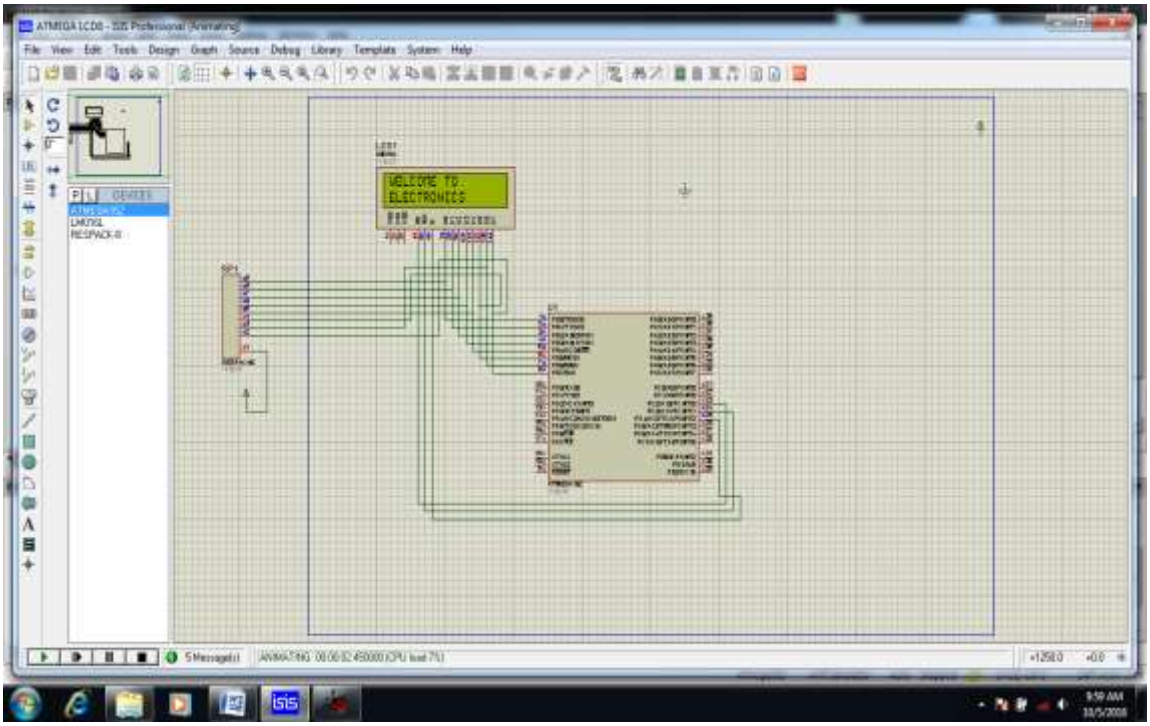
```
        LCD_Data_Port = char_data;
        LCD_Command_Port&= ~(1<<RW);
        LCD_Command_Port |= (1<<EN)|(1<<RS);

        _delay_us(1);

        LCD_Command_Port&= ~(1<<EN);

        _delay_ms(1);

}

voidLCD_Init (void)

{

        LCD_Command_Dir |= (1<<RS)|(1<<RW)|(1<<EN);
        LCD_Data_Dir = 0xFF;

        _delay_ms(20);
        LCD_Command (0x38);
        LCD_Command (0x0C);
        LCD_Command (0x06);
        LCD_Command (0x01);
        LCD_Command (0x80);

}


voidLCD_String (char *str)

{

        int i;

        for(i=0;str[i]!=0;i++)

        {

                LCD_Char (str[i]);

        }

}
```

```
voidLCD_String_xy (char row, char pos, char *str)
{
        if (row == 1)
                LCD_Command((pos& 0x0F)|0x80);
        else if (row == 2)
                LCD_Command((pos& 0x0F)|0xC0);
        LCD_String(str);
}

voidLCD_Clear (void)
{
        LCD_Command (0x01);
        /*LCD_Command (0x80);
        }
```

BUILD PROJECT



PROTEUSSIMULATION



2. TOGGLE THE PORT USING LED

```
int main(void)
{
DDRD=0b11111111;
DDRC=0b00000000;

int input=0;

        LCD_Init();
        LCD_String("LED         ");
        LCD_Command(0xC0);
        LCD_String("BLINKING     ");
while (1)
   {

        input=PINC&0b00000001;
if(input==0b00000001)
        {
        PORTD=0X55;

        }
else
{

PORTD=0XAA;
}
   }
}
```

LCD HEADERFILE

```
#ifndef LCD_16x2_H_H_

#define LCD_16x2_H_H_

#define F_CPU 8000000UL
```

```
#include <avr/io.h>
#include <util/delay.h>
#define LCD_Data_Dir DDRB
#define LCD_Command_Dir DDRC
#define LCD_Data_Port PORTB
#define LCD_Command_Port PORTC
#define RS PC2
#define RW PC3
#define EN PC4

voidLCD_Command (char);
voidLCD_Char (char);
voidLCD_Init (void);
voidLCD_String (char*);
voidLCD_String_xy (char,char,char*);
voidLCD_Clear (void);
voidLCD_Command (char cmd)
{
        LCD_Data_Port = cmd;
        LCD_Command_Port&= ~((1<<RS)|(1<<RW));
        LCD_Command_Port |= (1<<EN);
        _delay_us(1);
        LCD_Command_Port&= ~(1<<EN);
        _delay_ms(3);
}
voidLCD_Char (char char_data)
```

```
{
        LCD_Data_Port = char_data;
        LCD_Command_Port&= ~(1<<RW);
        LCD_Command_Port |= (1<<EN)|(1<<RS);

        _delay_us(1);

        LCD_Command_Port&= ~(1<<EN);

        _delay_ms(1);

}

voidLCD_Init (void)

{

        LCD_Command_Dir |= (1<<RS)|(1<<RW)|(1<<EN);
        LCD_Data_Dir = 0xFF;

        _delay_ms(20);
        LCD_Command (0x38);
        LCD_Command (0x0C);
        LCD_Command (0x06);
        LCD_Command (0x01);
        LCD_Command (0x80);

}


voidLCD_String (char *str)

{

        int i;

        for(i=0;str[i]!=0;i++)

        {

                LCD_Char (str[i]);

        }

}
```

```
voidLCD_String_xy (char row, char pos, char *str)
{
        if (row == 1)
                LCD_Command((pos& 0x0F)|0x80);
        else if (row == 2)
                LCD_Command((pos& 0x0F)|0xC0);
        LCD_String(str);
}

voidLCD_Clear (void)
{
        LCD_Command (0x01);
        /*LCD_Command (0x80);
        }
```

3. SENSOR INTERFACING

```
int main(void)
{
DDRD=0b11111111;
DDRC=0b00000000;
int SENSOR=0;
        LCD_Init();

while (1)
   {

        SENSOR=PINC&0b00000001;
if(SENSOR==0b00000001)
        {
        PORTD=0b00000001;

        LCD_String("SENSOR       ");
        LCD_Command(0xC0);
        LCD_String("SENSING       ");

        }
else
{
        LCD_String("SENSOR       ");
        LCD_Command(0xC0);
        LCD_String("NOT SENSING     ");
PORTD=0b00000000;
}

   }
}
```

LCD HEADERFILE

```
#ifndef LCD_16x2_H_H_
```

```
#define LCD_16x2_H_H_

#define F_CPU 8000000UL

#include <avr/io.h>

#include <util/delay.h>

#define LCD_Data_Dir DDRB

#define LCD_Command_Dir DDRC

#define LCD_Data_Port PORTB

#define LCD_Command_Port PORTC

#define RS PC2

#define RW PC3

#define EN PC4


voidLCD_Command (char);

voidLCD_Char (char);

voidLCD_Init (void);

voidLCD_String (char*);

voidLCD_String_xy (char,char,char*);

voidLCD_Clear (void);

voidLCD_Command (char cmd)

{

        LCD_Data_Port = cmd;
        LCD_Command_Port&= ~((1<<RS)|(1<<RW));
        LCD_Command_Port |= (1<<EN);

        _delay_us(1);

        LCD_Command_Port&= ~(1<<EN);

        _delay_ms(3);
```

```
}
voidLCD_Char (char char_data)
{
        LCD_Data_Port = char_data;
        LCD_Command_Port&= ~(1<<RW);
        LCD_Command_Port |= (1<<EN)|(1<<RS);
        _delay_us(1);
        LCD_Command_Port&= ~(1<<EN);
        _delay_ms(1);
}
voidLCD_Init (void)
{
        LCD_Command_Dir |= (1<<RS)|(1<<RW)|(1<<EN);
        LCD_Data_Dir = 0xFF;
        _delay_ms(20);
        LCD_Command (0x38);
        LCD_Command (0x0C);
        LCD_Command (0x06);
        LCD_Command (0x01);
        LCD_Command (0x80);
}

voidLCD_String (char *str)
{
        int i;
        for(i=0;str[i]!=0;i++)
        {
                LCD_Char (str[i]);
```

```
        }
}

voidLCD_String_xy (char row, char pos, char *str)
{
        if (row == 1)
                LCD_Command((pos& 0x0F)|0x80);
        else if (row == 2)
                LCD_Command((pos& 0x0F)|0xC0);
        LCD_String(str);
}

voidLCD_Clear (void)
{
        LCD_Command (0x01);
        /*LCD_Command (0x80);
        }
```

4. RELAY INTERFACING

```
int main(void)
{
DDRD=0b11111111;
DDRC=0b00000000;
int input=0;
        LCD_Init();
        LCD_String("RELAY        ");
        LCD_Command(0xC0);
        LCD_String("INTERFACING   ");
while (1)
   {
        input=PINC&0b00000001;
if(input==0b00000001)
        {
        PORTD=0b00000001;
        }
else
{
PORTD=0b00000000;
}
   }
}
```

LCD HEADERFILE

```
#ifndef LCD_16x2_H_H_
#define LCD_16x2_H_H_
#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>
#define LCD_Data_Dir DDRB
#define LCD_Command_Dir DDRC
#define LCD_Data_Port PORTB
#define LCD_Command_Port PORTC
#define RS PC2
#define RW PC3
#define EN PC4

voidLCD_Command (char);
voidLCD_Char (char);
voidLCD_Init (void);
voidLCD_String (char*);
voidLCD_String_xy (char,char,char*);
voidLCD_Clear (void);
voidLCD_Command (char cmd)
{
```

```
        LCD_Data_Port = cmd;
        LCD_Command_Port&= ~((1<<RS)|(1<<RW));
        LCD_Command_Port |= (1<<EN);

        _delay_us(1);

        LCD_Command_Port&= ~(1<<EN);

        _delay_ms(3);

}

voidLCD_Char (char char_data)

{

        LCD_Data_Port = char_data;
        LCD_Command_Port&= ~(1<<RW);
        LCD_Command_Port |= (1<<EN)|(1<<RS);

        _delay_us(1);

        LCD_Command_Port&= ~(1<<EN);

        _delay_ms(1);

}

voidLCD_Init (void)

{

        LCD_Command_Dir |= (1<<RS)|(1<<RW)|(1<<EN);
        LCD_Data_Dir = 0xFF;

        _delay_ms(20);
        LCD_Command (0x38);
        LCD_Command (0x0C);
        LCD_Command (0x06);
        LCD_Command (0x01);
        LCD_Command (0x80);

}


voidLCD_String (char *str)
```

```
{
        int i;
        for(i=0;str[i]!=0;i++)
        {
                LCD_Char (str[i]);
        }
}

voidLCD_String_xy (char row, char pos, char *str)
{
        if (row == 1)
                LCD_Command((pos& 0x0F)|0x80);
        else if (row == 2)
                LCD_Command((pos& 0x0F)|0xC0);
        LCD_String(str);
}

voidLCD_Clear (void)
{
        LCD_Command (0x01);
        /*LCD_Command (0x80);
        }
```

5. KEYPAD INTERFACING

```
unsigned char keypad[4][4] = {      {'7','8','9','/'},
                                    {'4','5','6','*'},
                                    {'1','2','3','-'},
                                    {' ','0','=','+'}};
unsigned char colloc, rowloc;
charkeyfind()
{
        while(1)
        {
                KEY_DDR = 0xF0;
                KEY_PRT = 0xFF;
                do
                {
                        KEY_PRT &= 0x0F;
                        asm("NOP");
                        colloc = (KEY_PIN & 0x0F);
                }while(colloc != 0x0F);

                do
                {
                        do
                        {
                                _delay_ms(20);
                                colloc = (KEY_PIN & 0x0F);
```

```
        }while(colloc == 0x0F);
        _delay_ms (40);
        colloc = (KEY_PIN & 0x0F);
}
while(colloc == 0x0F);
KEY_PRT = 0xEF;
asm("NOP");
colloc = (KEY_PIN & 0x0F);
if(colloc != 0x0F)
{
        rowloc = 0;
        break;
}
KEY_PRT = 0xDF;
asm("NOP");
co  lloc = (KEY_PIN & 0x0F);
if(colloc != 0x0F)
{
        rowloc = 1;
        break;
}

KEY_PRT = 0xBF;
asm("NOP");
colloc = (KEY_PIN & 0x0F);
```

```
            if(colloc != 0x0F)
            {
                rowloc = 2;
                break;
            }

            KEY_PRT = 0x7F;
            asm("NOP");
            colloc = (KEY_PIN & 0x0F);
            if(colloc != 0x0F)
            {
                rowloc = 3;
                break;
            }
        }

        if(colloc == 0x0E)
            return(keypad[rowloc][0]);
        else if(colloc == 0x0D)
            return(keypad[rowloc][1]);
        else if(colloc == 0x0B)
            return(keypad[rowloc][2]);
        else
            return(keypad[rowloc][3]);
}
```

```
int main(void)
{
	LCD_Init();
	LCD_String_xy(1,0,"Press a key");
while(1)
	{
		LCD_Command(0xc0);
		LCD_Char(keyfind());
	}
}
```

LCD HEADERFILE

```
#ifndef LCD_16x2_H_H_
#define LCD_16x2_H_H_
#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>
#define LCD_Data_Dir DDRB
#define LCD_Command_Dir DDRC
#define LCD_Data_Port PORTB
#define LCD_Command_Port PORTC
#define RS PC2
#define RW PC3
```

```
#define EN PC4


voidLCD_Command (char);

voidLCD_Char (char);

voidLCD_Init (void);

voidLCD_String (char*);

voidLCD_String_xy (char,char,char*);

voidLCD_Clear (void);

voidLCD_Command (char cmd)

{

        LCD_Data_Port = cmd;
        LCD_Command_Port&= ~((1<<RS)|(1<<RW));
        LCD_Command_Port |= (1<<EN);

        _delay_us(1);

        LCD_Command_Port&= ~(1<<EN);

        _delay_ms(3);

}

voidLCD_Char (char char_data)

{

        LCD_Data_Port = char_data;
        LCD_Command_Port&= ~(1<<RW);
        LCD_Command_Port |= (1<<EN)|(1<<RS);

        _delay_us(1);

        LCD_Command_Port&= ~(1<<EN);

        _delay_ms(1);

}

voidLCD_Init (void)
```

```
{
        LCD_Command_Dir |= (1<<RS)|(1<<RW)|(1<<EN);
        LCD_Data_Dir = 0xFF;

        _delay_ms(20);
        LCD_Command (0x38);
        LCD_Command (0x0C);
        LCD_Command (0x06);
        LCD_Command (0x01);
        LCD_Command (0x80);

}


voidLCD_String (char *str)

{

        int i;

        for(i=0;str[i]!=0;i++)

        {

                LCD_Char (str[i]);

        }

}


voidLCD_String_xy (char row, char pos, char *str)

{

        if (row == 1)

                LCD_Command((pos& 0x0F)|0x80);

        else if (row == 2)

                LCD_Command((pos& 0x0F)|0xC0);
        LCD_String(str);

}
```

```
voidLCD_Clear (void)

{

        LCD_Command (0x01);
        /*LCD_Command (0x80);

        }
```

6. STEPPER MOTOR INTERFACING

```
{
LCD_Init();
        LCD_String("STEPPER       ");
        LCD_Command(0xC0);
        LCD_String("MOTOR         ");
        int period;
        DDRD = 0x0F;
        period = 100;
        while (1)
        {
                        for(int i=0;i<12;i++)
                {
                        PORTD = 0x09;
                        _delay_ms(period);
                        PORTD = 0x08;
                        _delay_ms(period);
                        PORTD = 0x0C;
                        _delay_ms(period);
                        PORTD = 0x04;
                        _delay_ms(period);
                        PORTD = 0x06;
                        _delay_ms(period);
                        PORTD = 0x02;
```

```
                _delay_ms(period);
                PORTD = 0x03;
                _delay_ms(period);
                PORTD = 0x01;
                _delay_ms(period);
        }
        PORTD = 0x09;
        _delay_ms(period);
        _delay_ms(1000);

for(int i=0;i<12;i++)
        {
                PORTD = 0x09;
                _delay_ms(period);
                PORTD = 0x03;
                _delay_ms(period);
                PORTD = 0x06;
                _delay_ms(period);
                PORTD = 0x0C;
                _delay_ms(period);
        }
        PORTD = 0x09;
        _delay_ms(period);
        _delay_ms(1000);
}
```

```
}
```

LCD HEADERFILE

```
#ifndef LCD_16x2_H_H_
#define LCD_16x2_H_H_
#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>
#define LCD_Data_Dir DDRB
#define LCD_Command_Dir DDRC
#define LCD_Data_Port PORTB
#define LCD_Command_Port PORTC
#define RS PC2
#define RW PC3
#define EN PC4

voidLCD_Command (char);
voidLCD_Char (char);
voidLCD_Init (void);
voidLCD_String (char*);
voidLCD_String_xy (char,char,char*);
voidLCD_Clear (void);
voidLCD_Command (char cmd)
```

```
{
        LCD_Data_Port = cmd;
        LCD_Command_Port&= ~((1<<RS)|(1<<RW));
        LCD_Command_Port |= (1<<EN);

        _delay_us(1);

        LCD_Command_Port&= ~(1<<EN);

        _delay_ms(3);

}

voidLCD_Char (char char_data)

{

        LCD_Data_Port = char_data;
        LCD_Command_Port&= ~(1<<RW);
        LCD_Command_Port |= (1<<EN)|(1<<RS);

        _delay_us(1);

        LCD_Command_Port&= ~(1<<EN);

        _delay_ms(1);

}

voidLCD_Init (void)

{

        LCD_Command_Dir |= (1<<RS)|(1<<RW)|(1<<EN);
        LCD_Data_Dir = 0xFF;

        _delay_ms(20);
        LCD_Command (0x38);
        LCD_Command (0x0C);
        LCD_Command (0x06);
        LCD_Command (0x01);
        LCD_Command (0x80);

}
```

```
voidLCD_String (char *str)
{
        int i;
        for(i=0;str[i]!=0;i++)
        {
                LCD_Char (str[i]);
        }
}

voidLCD_String_xy (char row, char pos, char *str)
{
        if (row == 1)
                LCD_Command((pos& 0x0F)|0x80);
        else if (row == 2)
                LCD_Command((pos& 0x0F)|0xC0);
        LCD_String(str);
}

voidLCD_Clear (void)
{
        LCD_Command (0x01);
        /*LCD_Command (0x80);
        }
```

7.PROGRAM FOR TIMER

```
int main(void)
{
        DDRB = 0xFF;
        PORTB = 0;
while(1)
  {
                PORTB= ~ PORTB;
                T0delay();
        }
}
void T0delay()
{
        TCCR0 = (1<<CS02) | (1<<CS00);
        TCNT0 = 0xB2;
        while((TIFR&0x01)==0);
        TCCR0 = 0;
        TIFR = 0x1;
}
```

8. ADC INTERFACING

```
#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <stdlib.h>
#include "LCD_16x2_H_file.h"

voidADC_Init()
{
DDRA = 0x00;
ADCSRA = 0x87;
ADMUX = 0x40;
}
intADC_Read(char channel)
{
        intAin,AinLow;
        ADMUX=ADMUX|(channel & 0x0f);
        ADCSRA |= (1<<ADSC);
        while((ADCSRA&(1<<ADIF))==0);
        _delay_us(10);
        AinLow = (int)ADCL;
        Ain = (int)ADCH*256;
        Ain = Ain + AinLow;
        return(Ain);
```

```
}
int main()
{
        char String[5];
        int value;
        ADC_Init();
        LCD_Init();
        LCD_String("ADC value");
        while(1)
        {
                LCD_Command(0xc4);
                value=ADC_Read(0);
                itoa(value,String,10);
                LCD_String(String);
                LCD_String("  ");
        }
        return 0;
}
```

LCD HEADERFILE

```
#ifndef LCD_16x2_H_H_
#define LCD_16x2_H_H_
#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>
#define LCD_Data_Dir DDRB
```

```
#define LCD_Command_Dir DDRC

#define LCD_Data_Port PORTB

#define LCD_Command_Port PORTC

#define RS PC2

#define RW PC3

#define EN PC4


voidLCD_Command (char);

voidLCD_Char (char);

voidLCD_Init (void);

voidLCD_String (char*);

voidLCD_String_xy (char,char,char*);

voidLCD_Clear (void);

voidLCD_Command (char cmd)

{

        LCD_Data_Port = cmd;
        LCD_Command_Port&= ~((1<<RS)|(1<<RW));
        LCD_Command_Port |= (1<<EN);

        _delay_us(1);

        LCD_Command_Port&= ~(1<<EN);

        _delay_ms(3);

}

voidLCD_Char (char char_data)

{

        LCD_Data_Port = char_data;
        LCD_Command_Port&= ~(1<<RW);
        LCD_Command_Port |= (1<<EN)|(1<<RS);
```

```
        _delay_us(1);
        LCD_Command_Port&= ~(1<<EN);
        _delay_ms(1);
}
voidLCD_Init (void)
{
        LCD_Command_Dir |= (1<<RS)|(1<<RW)|(1<<EN);
        LCD_Data_Dir = 0xFF;
        _delay_ms(20);
        LCD_Command (0x38);
        LCD_Command (0x0C);
        LCD_Command (0x06);
        LCD_Command (0x01);
        LCD_Command (0x80);
}

voidLCD_String (char *str)
{
        int i;
        for(i=0;str[i]!=0;i++)
        {
                LCD_Char (str[i]);
        }
}

voidLCD_String_xy (char row, char pos, char *str)
{
```

```
        if (row == 1)
                LCD_Command((pos& 0x0F)|0x80);
        else if (row == 2)
                LCD_Command((pos& 0x0F)|0xC0);
        LCD_String(str);
}

voidLCD_Clear (void)
{
        LCD_Command (0x01);
        /*LCD_Command (0x80);
        }
```