College of Engineering, Pune

Dept. of Electronics and Telecommunications Engineering

SY MICRO PROJECT 2021-22

PROJECT NAME:- DIGITAL CALCULATOR USING 8051 MICROCONTROLLER

DONE BY:- 112007065 SINGH ARCHITA

112007062 SHIVANI GIRI

112007049 RUPALI DEBNATH

112007048 SANSKRUTI RATHOD

GUIDED BY:- Prof. Yogita Kapse

Content

1. Abstract ……………………………………………………………………………2
2. Introduction …………………………………………………………………………..2
3. Objectives ……………………………………………………………………………7
4. Block Diagram ………………………………………………………………………..8
5. Implementation ………………………………………………………………………9
6. Results ………………………………………………………………………………18
7. Advantages …………………………………………………………………………..18
8. Disadvantages ………………………………………………………………………..19
9. Conclusion …………………………………………………………………………...19
10. References …………………………………………………………………………...19

Abstract:

This paper focuses on designing and implementation of an 8051 microcontroller based digital calculator which will perform simple arithmetic like addition, subtraction, multiplication and division using a calculator keypad, which includes digits from 0 to 9 and the operators for the given arithmetic operations along with an equal to sign, and a 16x2 Liquid Crystal Display (LCD).

This program is limited to single digit input and double digit results. This allows the program for the arithmetic operations to be simple, while the same principles can be extended to multi-digit calculations.

Introduction:

Advancement in technology has led to building electronic devices with simple circuitry. Introduction of microcontrollers has majorly helped in making designing of electronic circuits an easy and simple process.

A microcontroller is known as a computer on a chip. It is essential for the operation of devices such as mobile phones, video cameras, electrical appliances and most self-contained electronic systems.

Microcontroller has the following elements:

- CPU
- RAM
- ROM or Program memory
- Input Output Ports (I/Os)
- Clock
- Peripherals

<u>CPU:</u>

CPU is similar to a processor in a computer, which basically consists of Arithmetical and Logical Unit (ALU), Control Unit and Register Array.

ALU performs all arithmetic and logical operations on the data received from input devices or memory.

Register array consists of a series of registers like accumulator (A), B, C, C etc. which act as temporary fast access memory locations for processing data.

Control Unit controls the flow of instructions and data throughout the system.

RAM:

RAM stands for Random Access Memory. Similar to a computer, RAM is used to store data dynamically while the microcontroller is executing instructions. It is a volatile memory, means when the power goes off, all data is gone.

ROM or Programmable Memory:

ROM stands for Read Only Memory. In old microcontrollers flash memory was one time programmable that is why it is called ROM. But in latest microcontrollers it is re-programmable, ie. EEPROM (Electrically Erasable Programmable Read Only Memory). ROM is used to store the program or instructions which needs to be executed.

Input Output Ports (I/Os):

Microcontrollers provide multiple general purpose input output (GPIO) pins which can be configured as an input or output pin by writing to particular configuration registers. This pins can read or write HIGH or LOW state from/to its pins, making it possible to interface with external world.

Clock:

A microcontroller requires clock as it executes and is driven by sequential logic. Clock source can be external like crystal oscillator or internal like RC oscillator. Different microcontrollers will have different options for clocking.

Peripherals:

Microcontrollers also have other peripherals like:

- UART, SPI, I2C for serial communication
- Timers/Counters
- Capture/Compare/PWM modules
- Analog to Digital converter

The IC used for the project is: AT89C51

Features:

- 4K bytes of Reprogrammable Flash Memory
- 128 x 8 bit internal RAM
- 32 programmable I/O lines
- Two 16 bit Timers/Counters
- Six interrupt sources
- Programmable Serial channel

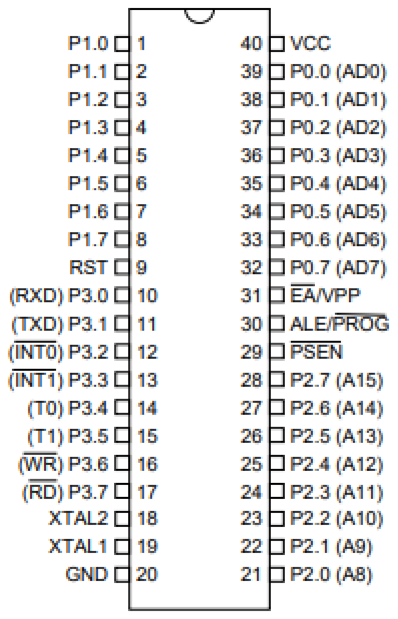


Fig.1 _ Pin configuration of AT89C51 IC

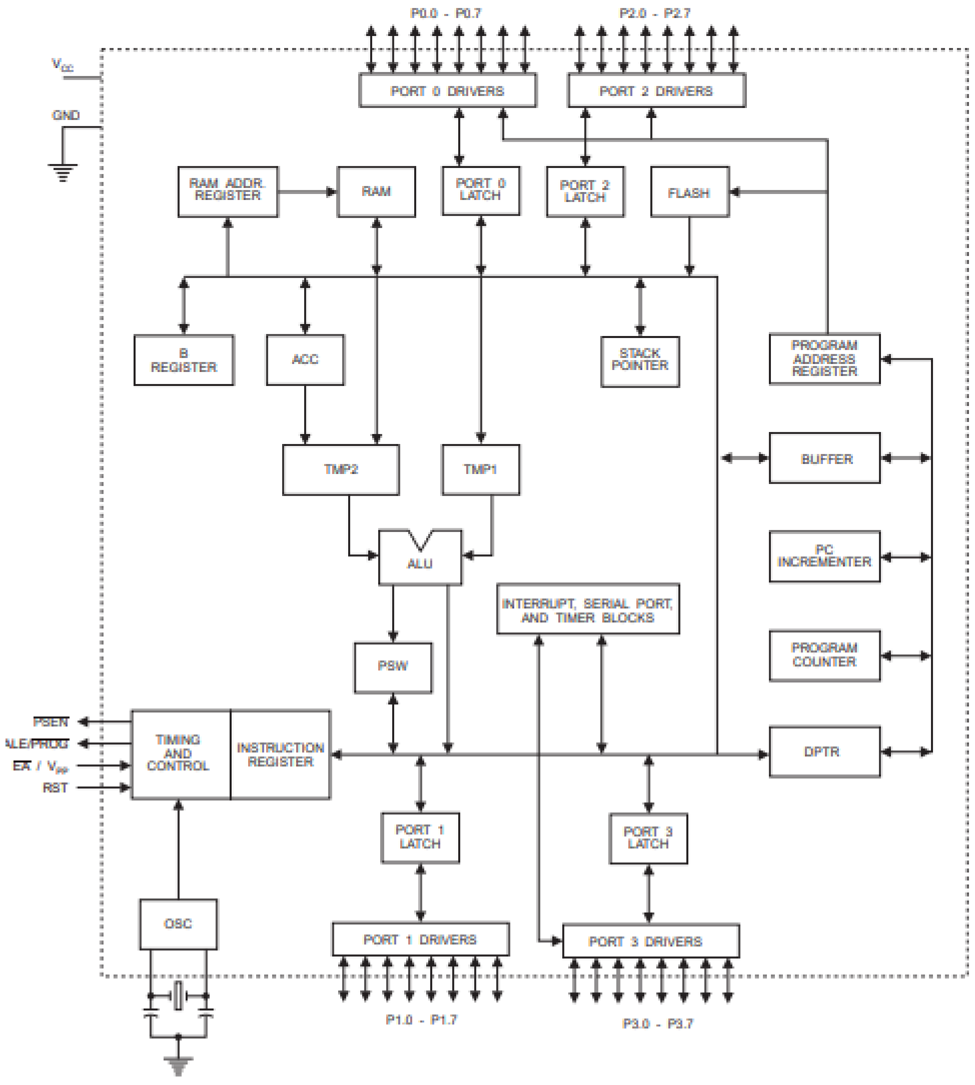


Fig.2 _ Block diagram of AT89C51

Pin Description:

Vcc - Supply voltage

GND - Ground

Port 0, 1, 2, 3 – 8 bit bidirectional I/O port

RST – reset input

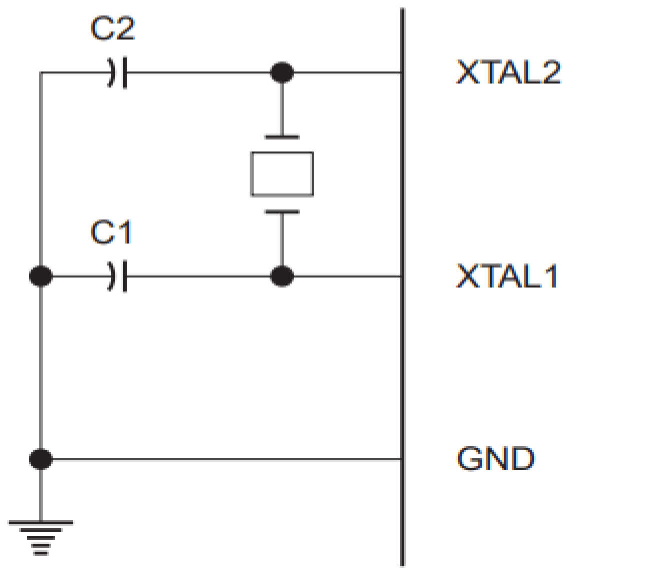
ALE/PROG – Address Latch Enable output pulse and also Program Pulse Input during Flash programming

PSEN – Program Store Enable, read strobe to external program memory

EA/VPP – External Access Enable and also 12 V programming enable voltage during Flash programming.

XTAL1 - Input to the inverting oscillator amplifier and input to the internal clock operating circuit

XTAL2 – output from the inverting oscillator amplifier



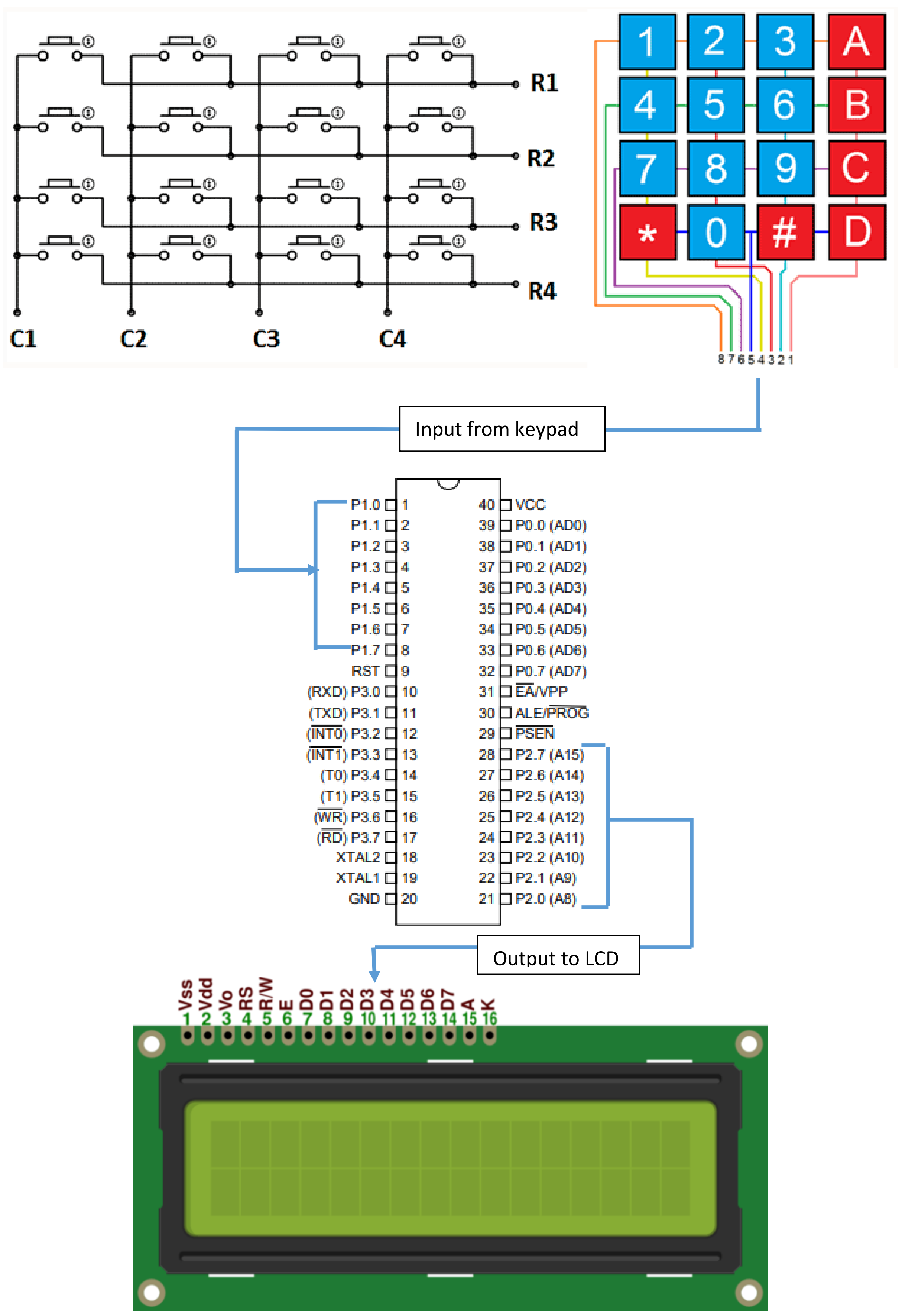
Note: C1, C2 = 30 pF ± 10 pF for Crystals
= 40 pF ± 10 pF for Ceramic Resonators

Fig.3 _ Oscillator Connections

Objectives:

1. Interfacing a 16x2 LCD to the microcontroller
2. Interfacing a 4x4 keypad to the microcontroller
3. Coding to operate the circuit as a digital calculator to perform basic operations with one digit input and 4 preset operations such as addition, subtraction, multiplication and division.

<u>Block Diagram:</u>



Implementation:

1. Connect the 11.59 MHz crystal oscillator in parallel with two 33pF capacitors, in series with each other and ground, to XTAL1 and XTAL2 pins of the IC.
2. Connect a 10uF electrolytic capacitor, in series with +5V source, and a 8.2k resistance, in series with ground, in parallel to the RST pin of the IC.
3. Connect the EA pin to a +5V power source.
4. Connect the input pins P1.0 to P1.7 of the IC to the keypad with P1.0 to P1.3 being the column pins (A, B, C, D) and P1.4 to P1.7 being the row pins (1, 2, 3, 4).
5. Connect the VSS pin of the LCD to ground.
6. Connect the VDD pin of the LCD to a +5V power source.
7. Connect the RS, RW and E pins of the LCD to P3.7, P3.5 and P3.6 of the IC respectively.
8. Connect the data pins D0 to D7 of the LCD to output pins P2.0 to P2.7 of the IC respectively.
9. Code a program to take input from keypad, perform the operations and display it on the LCD.
10. Burn the code in the IC and observe the output.

Circuit Diagram:

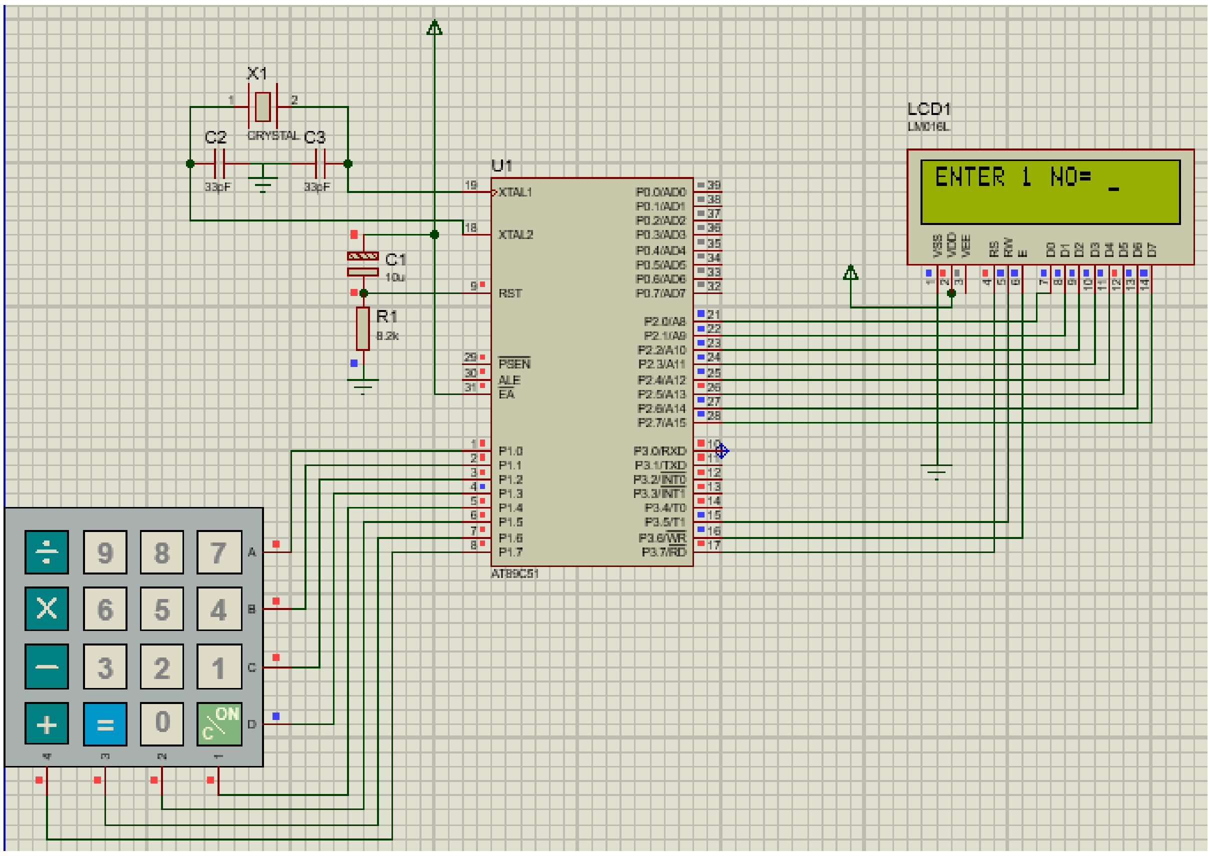


Fig.4 _ Circuit Diagram

The program outline used to implement the the calculator is shown below:

1. Single digit calculator produces two digit results.
2. Hardware: x12 keypad, 2x16 LCD, P16F887 MCU
3. MAIN
4. Initialise
5. PortC = keypad
6. RC0 – RC3 = output rows
7. RC4 – RC7 = input columns
8. PortD = LCD
9. RD1, RD2 = control bits
10. RD4– RD7 = data bits
11. CALL Initialise display
12. Scan Keypad
13. REPEAT

```
14. CALL Keypad input, Delay 50ms for debounce
15. CALL Keypad input, Check key released
16. IF first key, load Num1, Display character and restart loop
17. IF second key, load sign, Display character and restart loop
18. IF third key, load Num2 Display character and restart loop
19. IF fourth key, CALL Calculate result
20. IF fifth key, Clear display
21. ALWAYS
22. SUBROUTINES
23. Included LCD driver routines
24. Initialise display
25. Display character
26. Keypad Input
27. Check row A, IF key pressed, load ASCII code
28. Check row B, IF key pressed, load ASCII code
29. Check row C, IF key pressed, load ASCII code
30. Check row D, IF key pressed, load ASCII code
31. ELSE load zero code
32. Calculate result
33. IF key = ‘+’, Add
34. IF key = ‘-‘, Subtract
35. IF key = ‘x’, Multiply
36. IF key = ‘/’, Divide
37. Add Add Num1 + Num2
38. Load result, CALL Two digits
39. Subtract Subtract Num1 – Num2
40. IF result negative, load minus sign, CALLDisplay character
41. Load result, CALL Display character
42. Multiply
43. REPEAT
44. Add Num1 to Result
45. Decrement Num2
46. UNTIL Num2= 0
47. Load result, CALL Two digits
```

```
48. Divide
49. REPEAT
50. Subtract Num2 from Num1
51. Increment Result
52. UNTIL Num1 negative
53. Add Num2 back onto Num1 for Remainder
54. Load Result, CALL Display character
55. Load Remainder, CALL Display character
56. Two digits
57. Divide result by 10, load MSD, CALL Display character
58. Load LCD, CALL Display character
```

Code:

```
#include<reg51.h>

void lcdcmd(unsigned char);
void lcddata(unsigned char);
void MSDelay(unsigned int);
void disp_num(float num);
int get_num(char ch);
void lcdinit();
char scan_key(void);
unsigned char s[30]={"ENTER 1 NO= "};
unsigned char s1[30]={"ENTER 2 NO= "};
unsigned char s2[30]={"OPERATOR = "};
sfr ldata = 0xA0;
sbit rs = P3^7;
sbit rw = P3^5;
sbit en = P3^6;
sbit r0=P1^0;
sbit r1=P1^1;
sbit r2=P1^2;
sbit r3=P1^3;
sbit c0=P1^4;
sbit c1=P1^5;
sbit c2=P1^6;
sbit c3=P1^7;

void lcdinit(){
 MSDelay(15000);
            lcdcmd(0x30);
            MSDelay(4500);
            lcdcmd(0x30);
            MSDelay(300);
            lcdcmd(0x30);
            MSDelay(600);
    lcdcmd(0x38);
    lcdcmd(0x0F);
    lcdcmd(0x01);
    lcdcmd(0x06);
    lcdcmd(0x80);
}


int main (void)
  {
   while(1){
   unsigned int k=0,m=0,n=0;int k2,k1; char key,key1;unsigned char ch2;
   lcdinit();


                        while(s[k]!='\0')
                        {
                                    lcddata(s[k]);
                                    k++;
                        }
                        key=scan_key();
			k2=get_num(key);
			lcddata(key);
			lcdcmd(0x01);

			while(s2[n]!='\0')
			{
				lcddata(s2[n]);
				n++;
			}
			ch2=scan_key();
		lcddata(ch2);
			lcdcmd(0x01);

			while(s1[m]!='\0')
			{
				lcddata(s1[m]);
				m++;
			}
			key1=scan_key();
			k1=get_num(key1);
			lcddata(key1);
		lcdcmd(0x01);
	switch(ch2)
	{
	case '+':
	disp_num(k1+k2);
	break;
		case '-':
	disp_num(k2-k1);
	break;
		case '*':
	disp_num(k2*k1);
	break;
		case '/':
```

```
disp_num(k2/k1);
break;
}
return 0;
}
}
void lcdcmd(unsigned char value)
  {
    ldata = value;
    rs = 0;
    rw = 0;
    en = 1;
    MSDelay(50);
    en = 0;
            MSDelay(50);

  }
void lcddata(unsigned char value)
  {
    ldata = value;
    rs = 1;
    rw = 0;
    en = 1;
    MSDelay(50);
    en = 0;
    MSDelay(50);
  }
void MSDelay(unsigned int itime)
  {
    unsigned int i, j;
    for(i=0;i<itime;i++)
      for(j=0;j<5;j++);
  }
char scan_key()
```

```
{
        unsigned char c;
        c='s';
        while(!(c=='0' && c=='1' &&  c=='2' && c=='3' && c=='4' && c=='5' && c=='6'
&& c=='7' && c=='8' && c=='9' && c=='+' && c=='-' && c=='*' && c=='/'  && c=='C' &&
c=='='))
        {
r0=0;r1=1;r2=1;r3=1;
        if(c0==0 && r0==0 ){lcddata('7');MSDelay(100000);return c='7';}
   if(c1==0 && r0==0){ lcddata('8');MSDelay(100000);return c= '8';}
        if(c2==0 && r0==0){ lcddata('9');MSDelay(100000);return c= '9';}
        if(c3==0 && r0==0){ lcddata('/');MSDelay(100000);return c= '/';}

r0=1;r1=0;r2=1;r3=1;

        if(c0==0 && r1==0){ lcddata('4');MSDelay(100000);return c= '4';}
   if(c1==0 && r1==0){ lcddata('5');MSDelay(100000);return c= '5';}
        if(c2==0 && r1==0){ lcddata('6');MSDelay(100000);return c= '6';}
        if(c3==0 && r1==0){ lcddata('*');MSDelay(100000);return c= '*';}

r0=1;r1=1;r2=0;r3=1;

        if(c0==0 && r2==0){ lcddata('1');MSDelay(100000);return c= '1';}
   if(c1==0 && r2==0){ lcddata('2');MSDelay(100000);return c= '2';}
        if(c2==0 && r2==0){ lcddata('3');MSDelay(100000);return c= '3';}
        if(c3==0 && r2==0){ lcddata('-');MSDelay(100000);return c= '-';}

r0=1;r1=1;r2=1;r3=0;

        if(c0==0 && r3==0){ lcddata('C');MSDelay(100000);return c= 'C';}
   if(c1==0 && r3==0){ lcddata('0');MSDelay(100000);return c= '0';}
        if(c2==0 && r3==0){ lcddata('=');MSDelay(100000);return c= '=';}
        if(c3==0 && r3==0){ lcddata('+');MSDelay(100000);return c= '+';}
```

```
}
return 0;
}

int get_num(char ch)         //convert char into int
{
        switch(ch)
        {
                case '0': return 0; break;
                case '1': return 1; break;
                case '2': return 2; break;
                case '3': return 3; break;
                case '4': return 4; break;
                case '5': return 5; break;
                case '6': return 6; break;
                case '7': return 7; break;
                case '8': return 8; break;
                case '9': return 9; break;
        }
        return 0;
}

void disp_num(float num)           //displays number on LCD
{
        unsigned char UnitDigit  = 0;  //It will contain unit digit of numb
        unsigned char TenthDigit = 0;  //It will contain 10th position digit of numb
        unsigned char decimal = 0;
        int j;
        int numb;
        j=(int)(num*10);

        numb=(int)num;
        if(numb<0)
        {
            numb = -1*numb;  // Make number positive
            lcddata('-');       // Display a negative sign on LCD
        }

        TenthDigit = (numb/10);                 // Findout Tenth Digit

        if( TenthDigit != 0)            // If it is zero, then don't display
        lcddata(TenthDigit+0x30);       // Make Char of TenthDigit and then display it on LCD

        UnitDigit = numb - TenthDigit*10;

        lcddata(UnitDigit+0x30);        // Make Char of UnitDigit and then display it on LCD
        lcddata('.');

        decimal=(j%10)+0x30;
        lcddata(decimal);
        MSDelay(2000000);
    }
```

Result:

There is need for a portable, reliable, low cost and faster means of calculation with simple design. This study designed and implemented a Microcontroller based calculator for easy and speedy calculation. The Microcontroller AT89C51 was programmed with C-language and compiled using an 8051 code compiler and proteus ISIS Professional 7.8 portable simulation software. Results of the calculator were found to agree with the other calculators.

Advantages:

- Portable
- Reliable
- Low cost
- Easy to make at home
- Fast

- Simple design

Disadvantages:

- Can only perform simple arithmetic operations
- Only takes a one digit input
- Easily disassembled

Conclusion:

It is a comparatively simple and easy to make calculator. It helps in implementation of a lot of concepts that students have learnt in their Second Year of engineering. It can be further developed into a higher working calculator.

References:

https://electrosome.com/microcontroller/

http://ww1.microchip.com/downloads/en/devicedoc/doc0265.pdf

Thank You