Nigerian Journal of Technology (NIJOTECH)
Vol. 33. No. 3, July 2014, pp. **359 – 366**

www.nijotech.com
http://dx.doi.org/10.4314/njt.v33i3.14

A BASIC APPROACH TO DESIGNING EMBEDDED SYSTEMS USING A SIMPLE CALCULATOR AND C PROGRAMMING LANGUAGE

F. Tanshi[1,*] and N. Bello[2]

[1]DEPARTMENT OF ELECTRICAL/ELECTRONIC ENG, FEDERAL UNIVERSITY OF PETROLEUM RESOURCES, EFFURUN, NIGERIA
[2]DEPARTMENT OF ELECTRICAL/ELECTRONIC ENGINEERING, UNIVERSITY OF BENIN, BENIN CITY, NIGERIA
E-mail addresses: *[1] foghor.ai@gmail.com, [2] nosabello@yahoo.com*

ABSTRACT
Embedded systems form the bedrock of most consumer appliances in today's world. Products such as automatic transmission cars, fuel pump metres, washing machines, digital cameras, DVDs, mobile phones, vending machines e.t.c depend on an embedded microcomputer to enable it perform its range of functions. For example a vending machine may have three main functions (namely - insert money, choose item to be purchased and send out item to buyer/user) that are controlled by an embedded chip through its peripherals which are linked to the appliance's user interface. This paper intends to be a complete introductory guide to embedded design considering a simple calculator that uses a microcontroller for input, processing and output operations.

Keywords: *Central Processing Unit (CPU), C Programming Language, Electronics, Embedded Systems, Integrated Circuit, Microprocessor and Microcontroller.*

1. INTRODUCTION
A calculator is a small hand held electronic device used to perform mathematical and logical operations. It often has a microprocessor or microcontroller chip embedded in it that enables it to perform these functions. The microprocessor came about as a programmable replacement for control circuits and calculator chips in the 1970s [1][2]. Up to this point, most control systems using digital logic were implemented using individual logic integrated circuits to create the design and as more functionality was required, the circuits became very bulky and unreliable. Many hundreds of these discreet logic chips were needed just to create a simple four function calculator. But with the invention of the microprocessor which has a complete processing unit inside a microchip, the reliability of digital logic circuits were greatly increased as a result of reduced size. Subsequently, there was need to include a permanent memory in the microprocessor which ultimately gave birth to the microcontroller which is the processor component used in this design. Unlike the microprocessor that doesn't have a program or data memory, [2] the microcontroller has a program and data memory in addition to a central processing unit (CPU).

An embedded microcontroller (also called a microcomputer and sometimes abbreviated µC, uC or MCU) contains a processor core, memory, and programmable input/output peripherals in one integrated circuit [3] [4].An integrated circuit is a complete electronic circuit made with a small piece of semiconductor material (also called a Chip) and is encapsulated in plastic and used as one component.
The simple calculator realized in this design is capable of performing four arithmetic operations: addition (+), subtraction (-), multiplication (*) and division (/) and can perform computation on a maximum of 16 digits at a time.

2. DESIGN CONSIDERATIONS
The first thing we considered in this design was the settings and additional peripherals that the microcontroller will require to perform as a functional calculator and how it would be made to interphase with them. Some of these peripherals which include; resistors, an oscillator, capacitors, a keypad and LCD could be connected to designated pins of the microcontroller as specified by the manufacturer in its datasheet and others to bidirectional general purpose data ports pins. Specifically, it is worthy to note here that the 4×4 keypad matrix used would require an 8-

* Corresponding author, Tel: +234-703-129-9216

bit data port because it has four rows and four columns. The LCD will require seven of the pins of an 8-bit data port when connected in 4-bit mode and can only display information that is first converted to a string of characters. Therefore the source code should include a float conversion function.

Furthermore, the software settings that would prepare the microcontroller to interact via these ports were also established. These software settings include the configuration bits, the direction of data flow across a port (whether input or output) and the type and frequency of the external oscillator which serves as a timer for the microcontroller to schedule its activities.

3. DESIGN METHODOLOGY

The major components in this design include a PIC16F876A, a 2×16 Monochrome Liquid Crystal Display (LCD) and a 4×4 keypad matrix. The calculator circuit was designed and simulated in Labcenter Electronics® Proteus®. The source code for the microcontroller was written in C and compiled with MikroC® Pro v6.00.

The Common Object File (COF) generated after compilation was downloaded to the microcontroller in Proteus in order to test and debug the source code. A COF file is a portable file format across many embedded programming platforms. The required corrections were made to the source code in MikroC® Pro and then burnt into the microcontroller ROM with a Top 2005+ programmer.

The circuit components including the microcontroller were then soldered on a Vero board and encased in Perspex, with the LCD and keypad placed at the top to make it accessible to the user. Finally, the entire circuit was tested after implementation.

4. DESIGN SPECIFICATION

4.1 Hardware Components

The list of hardware components used in the design together with their specifications and quantity are listed in Table 1

4.2 Programming

Usually before embarking on writing software for any purpose, an execution procedure of the program in view of the design considerations such as; described in section 2.0 should be spelt out and written down. This is called an Algorithm. And a pictorial representation of an algorithm is called a flowchart, shown in Figure 1.

4.3 Circuit Diagram and Analysis

For convenience the circuit diagram is divided into marked segments in other to explain its functional parts as shown in Figure 2.

Table 1: List of hardware components

Item	Value	Quantity
Resistors:	10kohm	5
Ceramic Capacitors	10pF	2
Hi-Watt Dry Cell	9v	1
Button Switch	No units	1
Toggle Switch	No units	1
Voltage Regulator	5v (L7805CV)	1
Jumper Wires	No Units	1 yard
Crystal Oscillator	4MHz	1
4*4 keypad Matrix	No Units	1
16×2 LCD: RT1602C, v3.0	No Units	1
PIC16F876A	No Units	1

4.3.1 Microcontroller Circuit

A button switch is connected through a 10Ω resistor in parallel to the master clear pin (resets the microcontroller) [6]. One instruction cycle consists of four oscillator periods; for an oscillator frequency of 4 MHz, this gives a normal instruction execution time of 1µs. [7].

The PIC16F876A has an 8k program memory holding the calculator source codes, 368byte of RAM which stores inputted and computed data temporarily.

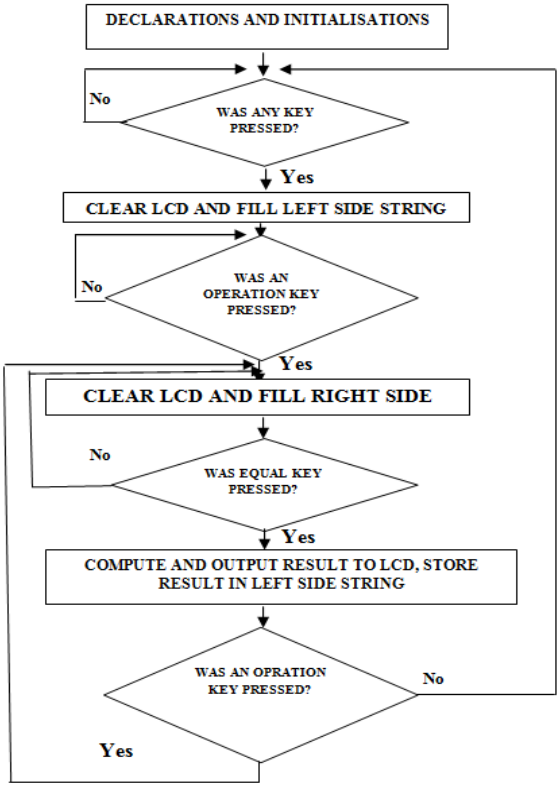

Figure.1: Flow Chart of Program

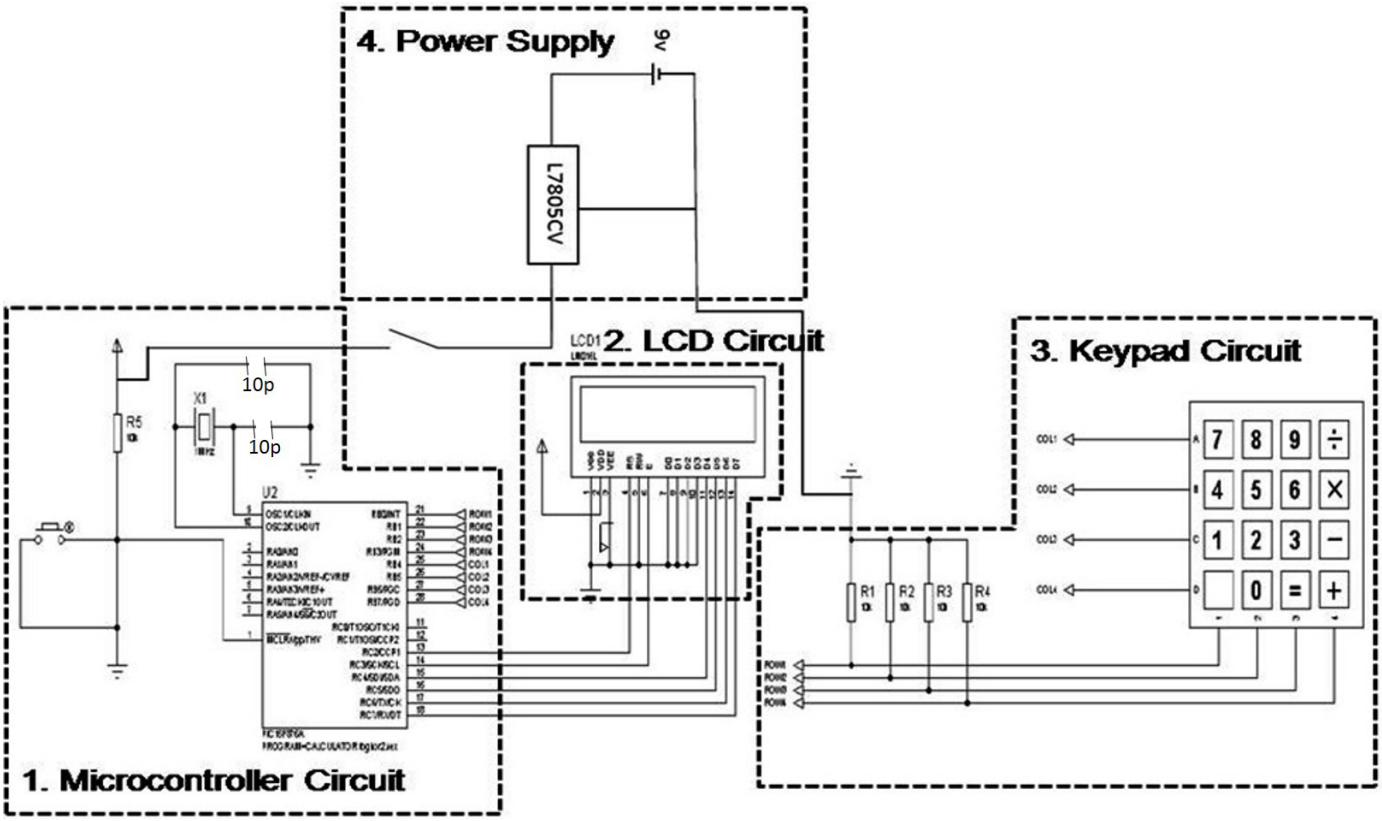

Figure 2: Simple Calculator Circuit Diagram

In order to burn code into a PIC microcontroller, its configuration word has to be determined. The configuration word is a group of 14-bits which make up the configuration Register located at address 2007h in program memory of the PIC16F876A. This memory location is beyond the user code program memory space and can be accessed only during programming of the chip [6]. These bits govern the response of the microcontroller to certain power states as well as configure its clock inputs to operate within a certain oscillator type and frequency.

There are 14-bits in the configuration register and bit-3 is the Power-up Timer Enable bit which allows a 72ms delay when the power button is pressed. Then the Oscillator Start-up Timer (OST) provides a delay of 1024 oscillator cycles (from OSC1 input) after the Power up Timer delay is over. This control bit ensures power supply to the circuit is steady and crystal oscillator or resonator has started and stabilized before microcontroller begins operation [6]. It therefore protects the microcontroller's internal circuitry from unsteady voltages. The configuration word used in programming the microcontroller is 0x2FC9.Immediately after initial declarations in the calculator program, is a do-while loop indicated by the link between the first and last diamonds on the flowchart and it causes the microcontroller to continuously scan the keypad in other to detect any key press. As keys are pressed, it decodes their value by the ASCII standard and stores them temporarily in its RAM pending when the operation to be performed is provided. Key inputs are stored as a string of characters which is sent to be displayed in the first row of the LCD. The first set of inputted characters are stored in a defined character string variable; "*leftsidestr" (see appendix)*, the operation key (i.e addition, subtraction, multiplication, division) is stored in a defined character variable; "*operation"(see appendix),*and the subsequent input keys are stored in a defined character string variable; "*rightsidestr"(see appendix)*.

4.3.2 Liquid Crystal Display (Lcd) Circuit

To set up the LCD for operation, VDD is positive supply to LCD, VCC is ground connection to LCD, VEE is used to adjust screen brightness and as a result it is connected to a variable resistor and then to ground. The Enable (E) line is connected to pin RC3, Read/Write line is grounded, Read Select line is connected to pin RC2, D0 to D3 are grounded and D4 to D7 are connected to pins RC4 to RC7 of the

microcontroller thereby configuring the LCD in 4-bit mode. I.e. it receives data 4-bit at a time.

Furthermore, the LCD has an LED/refractive material combination that forms its backlight. The value of resistance for powering the LED backlight is calculated as follows:

Power rating of calculator Circuit: 5V, Standard LED Bias Voltage: 4.0V

Standard LED Bias Current: 100mA [8].

$\frac{5-4}{0.1} = 10\Omega$

Therefore a 10Ω 1/2Watt resistor was used based on common resistors having a 5% tolerance.

4.3.3 4 × 4 Keypad Matrix

A keypad is a matrix of switches such that a key press results in the connection of a corresponding row and column thereby defining a unique character such as 1, 2, + e. t. c. All rows and columns are separately connected to the pins of PORTB as shown in Fig. 2. The value of each key is defined in the calculator code by the ASCII (American Standard Code for Information Interchange) standard e.g

55d = 37h = '1' and 56 d = 38 h = '2' e.t.c.

This way, the microcontroller is able to decipher the key that was pressed and displays it on the LCD. One of each end of four 10kΩ pull-down resistors are connected to the four columns and then to ground as shown in Figure. 2. The pull-down resistors ensure that the voltages of the columns are at zero potential. When a key is pressed, the corresponding column pulls the corresponding row input to zero. As a result the microcontroller detects an input value of '0' at the affected row and column with which it determines the key that was pressed according to the MikroC keypad library and the ASCII character standard. Pull-down resistor values usually range from 10kΩ – 50kΩ [9].

4.3.4 Power Supply Unit

The circuit requires 5V to operate and since a 5v battery could not be obtained, a 9V battery was regulated to 5V. The L7805CV is a 5V regulator as shown in the circuit Fig. 2.

4.3.5 Mode of Operation

For the calculator to provide results for computations, the inputs must be provided in the following order: a first string of figures, an operation key, a second string of figures and then an equal key is pressed. Similarly, in other to carry further computation on previous results, an operation key (such as addition) is pressed while the result of the previous computation is still on display, after which a second string of figures is inputted and then an equal key is pressed.

When the equals key is pressed, the microcontroller computes the result of the operation, converts it to a character string using the MikroC function "*FloaToStr"* then displays it on the second row of the LCD.

5. RESULTS AND ANALYSIS

It should be noted that when the circuit is drawn and simulated in Proteus, the power pins of the microcontroller and LCD are not visible and all other pins are re-organised for convenience. In addition, the source code is provided in the appendix.

First the .cof file was downloaded to the microcontroller in Proteus and the configuration bits and oscillator frequency were also imputed. The debug feature in Proteus allowed us to see the sequential flow of the program by running the simulation, pausing it and then using either of the step buttons (step into source line, or step over source line, step out of source line, execute) at the top right corner of the debug window.

We continued to edit the source code until the calculator responded as described in mode of operation (section 4.3.5). For example, displayed below are the results obtained when an addition computation is carried out between 18 and 23. Digit 1 is pressed first, then digit 8; shown in Figure 3.

After which, the addition sign (which is an operation) is pressed, then the screen is cleared awaiting the next string of digits to perform the operation with, as shown in Figure 4. Now entering the next string of digits; 2 and 3, which is then displayed on the screen as shown in the figures. When equals key is pressed, the answer is displayed on the second row of the LCD as shown in Figure 6

The result: 40.9999 is realized from the addition operation carried out between 18 and 23 instead of 41 because the "FloatToStr" function available in MikroC which converts a floating point number to a string of digits so it can be displayed on the LCD provides it this way. This leads to a loss of precision of 0.0001 which can be solved by writing another function to do the conversion in a case where better precision is required.

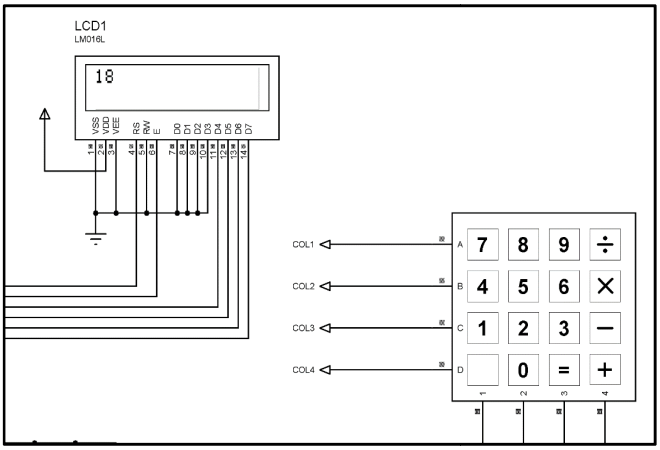

Figure. 3: Decimal Numeral 18 entered

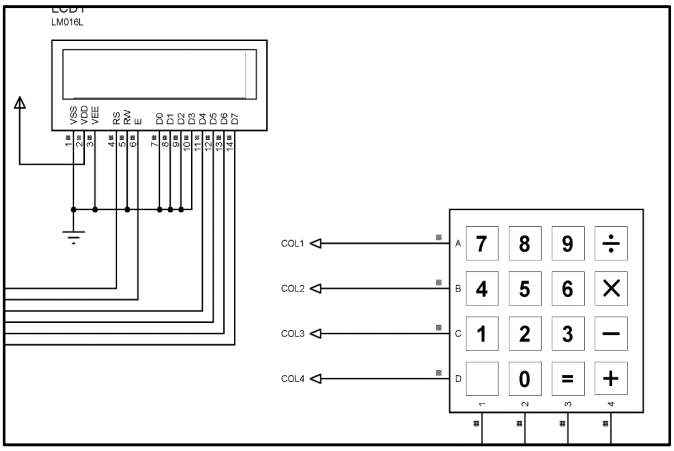

Figure. 4: Operation key is pressed

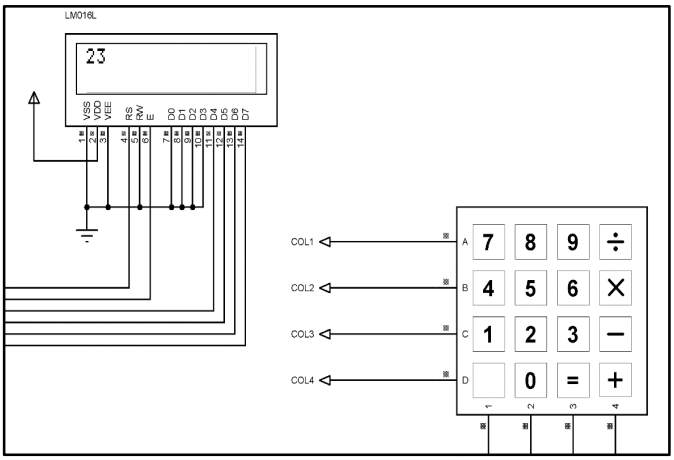

Figure. 5: Decimal Numeral 23 Entered

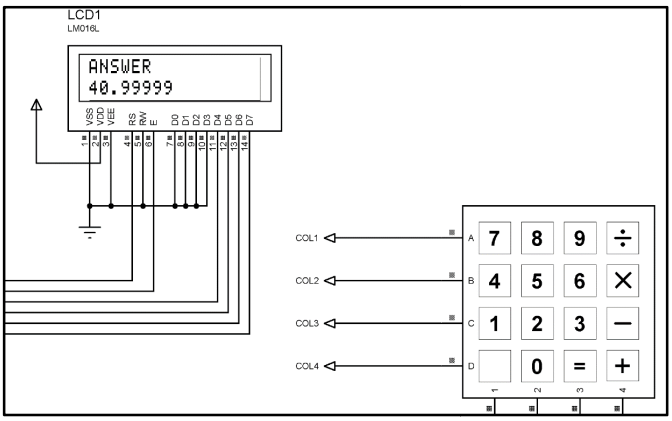

Figure. 6: Result is displayed after equal sign pressed

6. CONCLUSION

Embedded systems have a wide variety of applications in today's world and depending on the form of control one wishes to achieve, this paper will be a useful guide in all cases. It could be a standalone system like the calculator discussed or form a part of a larger system like a petrol pump control mechanism, a water level sensor [10], or an inverter [11].

7. REFERENCES

[1] Steve H.*Embedded System Design*, 2nd edition, Newnes, EBook, London, UK, 2003. Pg. 1 – 4

[2] Wilmshurst, T. Designing *Embedded Systems with PIC microcontrollers: Principles and Applications,* 1st Edition, Elsevier Ltd, EBook, Oxford, UK, 2007. Pg. 25-43, 386-407, 409-422.

[3] Barnett R. H, Larry O' Cull, and Cox S. *Embedded C Programming and the Microchip PIC,* 1stEdition,Delmar Publishers Inc., EBook, New York, USA, November 3, 2003.

[4] Wikipedia, 2014. "*PIC microcontroller*", available at http://en.wikipedia.org/wiki/PIC_microcontroller, accessed 1.20pm,20/03/2014.

[5] Cristian P. N, Vasilica C. *Calculator with keypad and LCD,* Silicon Service, EBook, USA, 2011.

[6] Microchip Technology Inc.HI-TECH C(R) for PIC10/12/16 user's guide,Microchip Technology Inc., EBook, USA, 2009. Pg. 5-13.

[7] Microchip TechnologyInc. PIC16F87XA Data Sheet: 28/40/44-Pin Enhanced Flash Microcontrollers, Microchip TechnologyInc.,EBook, USA, 2003.Pages 1-145, 148.

[8] Electronic Assembly, 2008. "LCD-MODULE 2x16 - 6.68mm INCL. CONTROLLER HD44780", Electronic Assembly, EBook, USA, Pg. 1-2.

[9] Stan Gibilisco. *The Illustrated Dictionary of Electronics*, 8th Edition, McGraw-Hill, EBook, USA, 2001.

[10] Anyanwu, CN, Mbajiorgu, C. C , Anoliefo, E .C "Design and implementation of a water level controller", *Nigerian Journal of Technology,* Vol. 31, No. 1, March, 2012, pp. 89 - 92.

[11] Agu, MU "A microprocessor-based control, scheme for a PWM voltage-fed inverter with an induction heating tank load", *Nigerian Journal of Technology* Vol. 23, No. 1, March 2004.

8. APPENDIX

Please note that the signs /* … */ indicates to the compiler that the enclosed word(s) are comments and not program codes.

Source Code

```
/*declaration of variables of type global outside the main function*/
charanswerstr[17],rightsidestr[17];
char leftsidestr[17], operation;
char leftsideflag, operationflag, rightsideflag;
char equalsignpressed;
char keypressed,counter;
float asctofloatvalueleft,asctofloatvalueright;
char asctofloatvaluestring[30];
float answer,leftside,rightside;
enum FLAGS {SET = 1,CLEAR = 0,MAXLCDCHARACTERS = 16,ADDITION = 43,
SUBTRACTION = 45,MULTIPLICATION =42,DIVISION =47,EQUALSIGN =61 };

/* Keypad module declaration*/
char keypadPort at PORTB;
/*End Keypad module connections*/

/* LCD module connections*/
sbit LCD_RS at RC2_bit;
sbit LCD_EN at RC3_bit;
sbit LCD_D4 at RC4_bit;
sbit LCD_D5 at RC5_bit;
sbit LCD_D6 at RC6_bit;
sbit LCD_D7 at RC7_bit;

sbit LCD_RS_Direction at TRISC2_bit;
sbit LCD_EN_Direction at TRISC3_bit;
sbit LCD_D4_Direction at TRISC4_bit;
sbit LCD_D5_Direction at TRISC5_bit;
sbit LCD_D6_Direction at TRISC6_bit;
sbit LCD_D7_Direction at TRISC7_bit;
/* End LCD module connections*/

/*function declaration and definition should be stated here*/
/* Start main function*/
void main() {
 ADCON1 = 0X07;                    /* configure portA as digital port*/
 counter =0;                    /* initialize counter used for locating character position on the lcd*/
 Keypad_Init();                    /* Initialize Keypad*/

 Lcd_Init();                    /* Initialize LCD*/
 Lcd_Cmd(_LCD_CLEAR);                /* Clear display*/
 Lcd_Cmd(_LCD_CURSOR_OFF);            /*turn Cursor off*/
 Lcd_Out(1,1,"A SIMPLE");            /* just some fancy initial display here or whatever you choose*/
 Lcd_Out(2,1,"CALCULATOR");
 counter =0; leftsideflag=SET;          /* this will enable our first SET of data entry to be stored on the leftside*/
 do {
  keypressed = CLEAR;                    /* Reset key code variable*/

  /* Wait for key to be pressed and released*/
  do
   // kp = Keypad_Key_Press();        /* Store key code in kp variable*/
    keypressed = Keypad_Key_Click();          /* Store key code in kp variable*/
  while (!keypressed);
 Lcd_Cmd(_LCD_CLEAR);                /* Clear display andprepare value for output, transform key to it's ASCII value*/
  switch (keypressed) {
   /* The following correspond to ascii value of the numbers shown based on the way connection was made from the keypad to the microcontroller using the 4 × 4 keypad library/arrangement present in labcenter proteus*/
    case  1: keypressed = 55 ; break;/*1*/
    case  2: keypressed = 56; break; /* 2*/
    case  3: keypressed = 57; break; /* 3*/
    case  4: keypressed = 47; break; /* divide*/
    case  5: keypressed = 52; break; /* 4*/
    case  6: keypressed = 53; break; /* 5*/
    case  7: keypressed = 54; break; /* 6*/
    case  8: keypressed = 42; break; /* multiply*/
    case  9: keypressed = 49 ; break; /* 7*/
    case 10: keypressed = 50; break; /* 8*/
    case 11: keypressed = 51; break; /* 9*/
    case 12: keypressed = 45; break; /* minus*/
    case 13: keypressed = 46; break; /* dot or full stop*/
    case 14: keypressed = 48; break; /* 0*/
    case 15: keypressed = 61; break; /* equal to*/
    case 16: keypressed = 43; break; /* addition*/

  }

/* Test if an operation key such as ADDITION,SUBTRACTION,DIVISION or MULTIPLICATION was pressed*/
   if(keypressed==DIVISION || keypressed==MULTIPLICATION || keypressed==SUBTRACTION || keypressed==ADDITION)
  {
   Lcd_Cmd(_LCD_CLEAR);   /*clear the lcd*/
```

```
   leftsideflag = CLEAR;  /* CLEAR leftsideflag so that
subsequent number is placed on the rightside of the
operation*/
   rightsideflag =SET;    /* This enable subsequent
numbers to be stored in the variable created to hold
rightside values*/
   answer = leftside;     /* This enable us to carry out
operation on previous answers obtained*/
   operation = keypressed;/* Help us to store the
operation that should be performed next*/
   counter=CLEAR;        /* CLEAR the counter so that we
start to fill lcd screen from leftmost position*/

  }
  if(keypressed==55 || keypressed==56 ||
keypressed==57 || keypressed==52 ||
keypressed==53  || keypressed==54 ||
keypressed==49 ||  keypressed==50 ||
  keypressed==51 ||  keypressed==46 ||
keypressed==48)
  {

       if(leftsideflag == SET)              /* Are we to fill the
leftside with data for the calculator since answer =
leftside + rightside*/
       {                            /* we fill the left side with
data first and when an operation key is pressed we start
to fill the right side*/
            if(counter==MAXLCDCHARACTERS)    /* Have
we got to the end of the lcd which can take a max of 16
characters*/
            {                       /* so that we rollover and
start again from the beginning, so this SET a limit*/
             counter = 0;                /* of our calculation to
16 character max and reset counter to start from*/
             Lcd_Cmd(_LCD_CLEAR);          /* the beginning
and CLEAR the lcd screen.*/
            }
leftsidestr[counter++] = keypressed;   /* Here we are
using our knowledge of pointers to accumulate the
numbers pressed and to display it as a string we
terminated with*/
       leftsidestr[counter] = '\0';        /* a null value of
\0 according to c standard*/
       Lcd_Out(1,1,leftsidestr);           /* show whatever
it is you are pressing on the lcd screen at row 1 column
1*/
 asctofloatvalueleft = atof(leftsidestr);   /* convert that
string shown on the lcd to a floating point number uisng
c function atof*/
       leftside = asctofloatvalueleft; /* store the floating
point result in variable leftside so it can be used for
computation subsequently     */
}
/* repeat above for right side*/
        if(rightsideflag == SET)
       {
            if(counter==MAXLCDCHARACTERS)
           {
            counter = 0;
            Lcd_Cmd(_LCD_CLEAR);
           }
        rightsidestr[counter++] = keypressed;
        rightsidestr[counter] = '\0';
        Lcd_Out(1,1,rightsidestr);
        asctofloatvalueright = atof(rightsidestr);
        rightside = asctofloatvalueright;


       }
  /*return a validnumber*/
  }
   if(keypressed==EQUALSIGN)             /* Was the key
pressed an equal sign, if it is then we have to process
leftside and rightside using the operation stored*/
  {                          /* stored previously and the
operation can be ADDITION, SUBTRACTION,DIVISION,or
MULTIPLICATION.*/

     Lcd_Cmd(_LCD_CLEAR);              /* CLEAR the lcd
before we start the operation*/
     if(operation == ADDITION)          /* was the
operation key pressed earlier ADDITION key? */
       {
        Lcd_Out(1,1,"ANSWER");         /* yes it was so
display answer at the top of the lcd screen*/
        answer = leftside + rightside;    /* use that
operation to carry out ADDITION since ADDITION was
pressed*/
        FloatToStr(answer,answerstr);    /* convert the
result to a string so we can show it on our lcd*/
        Lcd_Out(2,1,answerstr);         /* show it on the
second line of our lcd screen*/
       }
    if(operation == SUBTRACTION)
      {
       Lcd_Out(1,1,"ANSWER");
       answer = leftside - rightside;
       FloatToStr(answer,answerstr);
       Lcd_Out(2,1,answerstr);
      }

    if(operation == MULTIPLICATION)
      {
```

```
    Lcd_Out(1,1,"ANSWER");
    answer = leftside * rightside;
    FloatToStr(answer,answerstr);
    Lcd_Out(2,1,answerstr);
   }
if(operation == DIVISION)
   {
     if(rightside!=0)
       {
        Lcd_Out(1,1,"ANSWER");
        answer = leftside/ rightside;
        FloatToStr(answer,answerstr);
        Lcd_Out(2,1,answerstr);
       }
     else
                {
                 Lcd_Out(1,1,"ANSWER");
                 Lcd_Out(2,1,"ERR:DIV BY 0");
                }
        }
   leftsideflag = SET;
   rightsideflag = CLEAR;
   counter = CLEAR;
   rightside = 0;
   leftside = answer;
  }

 } while (1);
}
```