EXPERIMENT NO. 01

CALCULATOR USING PIC16F877

DOP: **DOS:**

Project Members:

1)	Prasad Pawaskar	58
2)	Vishal Thakur	72

AIM: To implement basic calculator functionality using PIC microcontroller (PIC16F877)

Tools: Keil uVision 4, ISIS Proteus 7

Theory: A calculator is a simple device used to perform basic as well as complex operations of arithmetic. The most basic calculator includes algorithms and/or hardware to implement at the minimum, addition, subtraction, multiplication and division. This PIC microcontroller tutorial provides a simple calculator implementation for PIC16F877 microcontroller. This is a simple one digit calculator which implements only 4 functions addition(+), subtraction(-), multiplication(x) and division(/).

PIC 16F877 is one of the most advanced microcontroller from Microchip. This controller is widely used for experimental and modern applications because of its low price, wide range of applications, high quality, and ease of availability. It is ideal for applications such as machine control applications, measurement devices, study purpose, and so on. The PIC 16F877 features all the components which modern microcontrollers normally have.

Algorithm:1) Initialize keypad and LCD display.

2) Introduce delay for first number from keypad and display the result of LCD.

3) Introduce delay for getting function key and wait for second number and then equal sign.

4) According to the desired Function result is calculated and displayed on the screen.

Code: **// Code of the Calculator**

```
#include "Includes.h"
/* Pin configuration
  * RA0 = Enable pin for LCD
```

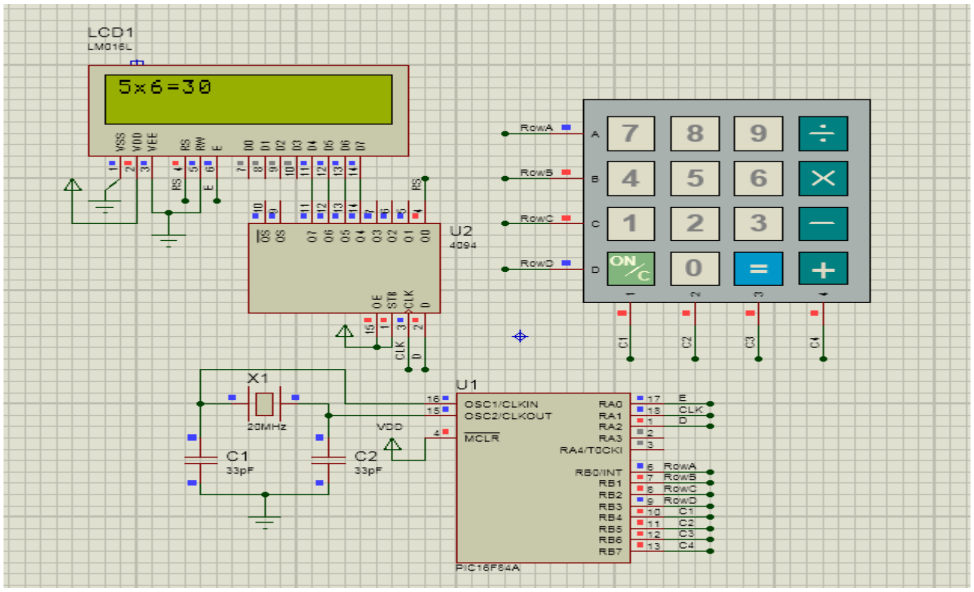
```
 * RA1 = CLK pin for LCD
 * RA2 = Data pin for LCD
 * PORTB = Keypad pins
 */
// Config word
__CONFIG(FOSC_HS & WDTE_OFF & PWRTE_ON &
    CP_OFF);
// Main function
void main(void)
{
    char key;                // Key char for keeping record of pressed key
    int num1 = 0;            // First number
    char func = '+';      // Function to be performed among two
    numbers
    int num2 = 0;            // Second number
    InitKeypad();            // Initialize Keypad
    InitLCD();                       // Initialize LCD


    while(1)
    {
      //get numb1
      key = GetKey();
      ClearLCDScreen();                // Clear LCD screen
     WriteDataToLCD(key);           // Echo the key pressed to LCD
     num1 = get_num(key);          // Get int number from char value,
    it checks for wrong input as well
       if(num1!=Error)                  // If correct input then proceed,
    num1==Error means wrong input
     {
            //get function
            key = GetKey();
            WriteDataToLCD(key);                    //Echo the key
    pressed to LCD
            func = get_func(key);                  //it checks for wrong
    func

            if(func!='e')                          //if correct input then
```

```
proceed, func=='e' means wrong input
    {
        //get numb2
        key = GetKey();
        WriteDataToLCD(key);          //Echo the key
pressed to LCD
        num2 = get_num(key);          //Get int number
from char value, it checks for wrong input as well
        if(num2!=Error)               //if correct input then
proceed, num2==Error means wrong input
        {
            //get equal sign
            key = GetKey();
            WriteDataToLCD(key);      //Echo the key
pressed to LCD
           if(key == '=')             //if = is pressed then
proceed
            {
                switch(func)          //switch on
function
                {
                case '+': disp_num(num1+num2);
break;
                case '-': disp_num(num1-num2);
break;
                case 'x': disp_num(num1*num2);
break;
                case '/': disp_num(num1/num2); break;
                }
            }
            else                                  //key other
then = here means error wrong input
            {
                if(key == 'C')        //if clear screen is
pressed then clear screen and reset
                    ClearLCDScreen(); // Clear LCD
screen
                else
                    DispError(0);         //Display
    wrong input error
                               }
                      }
              }
       }
    }
}
 * Functions used inside main for
 * making calculator are shown below
int get_num(char ch)             //convert char into int
{
    int num = 0;
    switch(ch)
    {
            case '0': num = 0; break;
            case '1': num = 1; break;
            case '2': num = 2; break;
            case '3': num = 3; break;
            case '4': num = 4; break;
            case '5': num = 5; break;
            case '6': num = 6; break;
            case '7': num = 7; break;
            case '8': num = 8; break;
            case '9': num = 9; break;
            case 'C': ClearLCDScreen(); num = Error; break;   //this is
    used as a clear screen and then reset by setting error
            default: DispError(0); num = Error; break;          //it means
    wrong input
    }
    return num;
}
char get_func(char chf)                 //detects the errors in inputted
    function
{
    if(chf=='C')                         //if clear screen then clear the LCD
    and reset
    {
```

```
                ClearLCDScreen();               //clear display
                return 'e';
        }

        if( chf!='+' && chf!='-' && chf!='x' && chf!='/' )   //if input is
        not from allowed funtions then show error
        {
                DispError(1);
                return 'e';
        }


        return chf;                             //function is correct so return
        the correct function
    }
void DispError(int numb)                //displays differet error messages
    {
        ClearLCDScreen();                       //clear display
        switch(numb)
        {
        case 0:         WriteStringToLCD("Wrong Input");         break;
        case 1:         WriteStringToLCD("Wrong Function");      break;
        default:        WriteStringToLCD("Wrong Input");         break;
        }
    }


    void disp_num(int numb)                     //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
        if(numb<0)
        {
                numb = -1*numb;                 // Make number positive
                WriteDataToLCD('-');        // Display a negative sign on
        LCD
        }
    TenthDigit = (numb/10);                     // Findout Tenth Digit
    if( TenthDigit != 0)                        // If it is zero, then
    don't display
        WriteDataToLCD(TenthDigit+0x30);    // Make Char of
    TenthDigit and then display it on LCD


    UnitDigit = numb - TenthDigit*10;
    WriteDataToLCD(UnitDigit+0x30);       // Make Char of
    UnitDigit and then display it on LCD
}
//LCD Display:
#include "Includes.h"
void ToggleEpinOfLCD(void)
{
    LCD_E = 1;                  // Give a pulse on E pin
    __delay_us(E_Delay);        // so that LCD can latch the
    LCD_E = 0;                  // data from data bus
    __delay_us(E_Delay);
}
void WriteByteToLCD(unsigned Byte)
{
    unsigned char BitCount = 0;


    for(BitCount=0;BitCount<8;BitCount++)
    {
        LCD_D = (((Byte>>BitCount)&0x1)!=0);    // Write bit
    value


        LCD_CLK = 1;                            // Toggle Clock
    pin to transfer it
        __delay_us(10);
        LCD_CLK = 0;
        __delay_us(10);
    }
}
void WriteCommandToLCD(unsigned char Command)
```

```
    {
        WriteByteToLCD((Command&0xF0)>>4);     // Write Upper
        nibble
        ToggleEpinOfLCD();
        WriteByteToLCD(Command&0x0F);            // Write Lower
        nibble
        ToggleEpinOfLCD();
    }
void WriteDataToLCD(char LCDChar)
    {
        WriteByteToLCD(((LCDChar&0xF0)>>4)|0x80);     // Write
        Upper nibble
        ToggleEpinOfLCD();
        WriteByteToLCD((LCDChar&0x0F)|0x80);            // Write
        Lower nibble
        ToggleEpinOfLCD();
    }
    oid InitLCD(void)
    {
        // Firstly make all pins output
        LCD_E      = 0;            // E = 0
        LCD_D      = 0;            // D = 0
        LCD_CLK   = 0;               // CLK = 0
        LCD_E_Dir     = 0;        // Make Output
        LCD_D_Dir     = 0;        // Make Output
        LCD_CLK_Dir = 0;        // Make Output
      ///////////////// Reset process from datasheet //////////////
        __delay_ms(40);


        WriteByteToLCD(0x03);
        ToggleEpinOfLCD();


        __delay_ms(6);
             WriteByteToLCD(0x03);
        ToggleEpinOfLCD();
    __delay_us(300);
    WriteByteToLCD(0x03);
    ToggleEpinOfLCD();


    __delay_ms(2);
    WriteByteToLCD(0x02);
    ToggleEpinOfLCD();


    __delay_ms(2);
  /////////////// Reset Process End ////////////////
    WriteCommandToLCD(0x28);      //function set
    WriteCommandToLCD(0x0c);      //display on,cursor off,blink off
    WriteCommandToLCD(0x01);      //clear display
    WriteCommandToLCD(0x06);      //entry mode, set increment
}
void ClearLCDScreen(void)          // Clear the Screen and return cursor to zero position
{
    WriteCommandToLCD(0x01);      // Clear the screen
    __delay_ms(2);                    // Delay for cursor to return at zero position
}
void WriteStringToLCD(const char *s)
{
    while(*s)
            WriteDataToLCD(*s++);
}
//Keypad :
#include "Includes.h"
// Function name: InitKeypad
void InitKeypad(void)
{
    PORTB = 0x00;              // Set PORTB pin values zero
    TRISB = 0xF0;              // PORTB4 to PORTB7 pins input, PORTB0 to PORTB3 pins output
    // Enable weak internal pull up on input pins
    OPTION_REG &= 0x7F;
}
// Function name: READ_SWITCHES
// Scan all the keypad keys to detect any pressed key.
char READ_SWITCHES(void)
{
    RowA = 0; RowB = 1; RowC = 1; RowD = 1;        //Test Row A
    if (C1 == 0) { __delay_ms(250); while (C1==0); return '7'; }
    if (C2 == 0) { __delay_ms(250); while (C2==0); return '8'; }
    if (C3 == 0) { __delay_ms(250); while (C3==0); return '9'; }
    if (C4 == 0) { __delay_ms(250); while (C4==0); return '/'; }


    RowA = 1; RowB = 0; RowC = 1; RowD = 1;        //Test Row B
    if (C1 == 0) { __delay_ms(250); while (C1==0); return '4'; }
    if (C2 == 0) { __delay_ms(250); while (C2==0); return '5'; }
    if (C3 == 0) { __delay_ms(250); while (C3==0); return '6'; }
    if (C4 == 0) { __delay_ms(250); while (C4==0); return 'x'; }

    RowA = 1; RowB = 1; RowC = 0; RowD = 1;        //Test Row C
    if (C1 == 0) { __delay_ms(250); while (C1==0); return '1'; }
    if (C2 == 0) { __delay_ms(250); while (C2==0); return '2'; }
    if (C3 == 0) { __delay_ms(250); while (C3==0); return '3'; }
    if (C4 == 0) { __delay_ms(250); while (C4==0); return '-'; }

    RowA = 1; RowB = 1; RowC = 1; RowD = 0;        //Test Row D
    if (C1 == 0) { __delay_ms(250); while (C1==0); return 'C'; }
    if (C2 == 0) { __delay_ms(250); while (C2==0); return '0'; }
    if (C3 == 0) { __delay_ms(250); while (C3==0); return '='; }
    if (C4 == 0) { __delay_ms(250); while (C4==0); return '+'; }


    return 'n';                    // Means no key has been pressed
}
// Function name: GetKey
// Read pressed key value from keypad and return its value
char GetKey(void)                        // Get key from user
{
        char key = 'n';                    // Assume no key pressed


        while(key=='n')                    // Wait untill a key is pressed
                key = READ_SWITCHES();     // Scan the keys again and
        again


        return key;                        //when key pressed then return its
        value
}
```

Output :



Conclusion: Thus, a basic Calculator was simulated using Proteus with the help of Keil. The results were visually verified using the Run feature in Proteus.