Project Report

Embedded Systems Design Lab ICT 3111

Submitted by:

Name	Roll Number	Registration No.
Ishita Gangal	63	140911460
Tushya Gautam	64	140911464
Sunny Bhalotia	65	140911462

Submitted to:

Mr. Santhosha Rao

Ms. Smitha A.

SEMESTER : 5th

DEPARTMENT : INFORMATION AND COMMUNICATION TECHNOLOGY

BRANCH : INFORMATION TECHNOLOGY

SECTION : A

BATCH : A2

PROJECT

Problem given:

Write an Embedded C program to simulate full fledged calculator by interfacing 4 X 4 matrix keyboard and LCD to 32 bit ARM microcontroller.

Contents:

ALGORITHM/PSEUDOCODE

Project once loaded starts with a message displayed using timer interrupts: “ENTER AN EXPRESSION“, after which user can give the input expression to the calculator.

This calculator allows an entire expression, complete with opening and closing parenthesis, as input. Operations include addition (‘+’), subtraction (‘-‘), multiplication (‘*’), division (‘/’), factorial (‘f’), sine (‘s’), cosine (‘c’), tangent (‘t’), square (‘q’), root (‘r’), natural log (‘l’), delete/backspace and clear screen. Precedence of operators is considered.

There is no limitation on the number of digits that can be input for a number. Multiple digits of a number are getting encoded to characters and then later decoded for evaluation.

The characters are used as keys to identify the multiple digits or whether sin/cos/log/tan and so on are in the expression. This is needed because conversion to postfix can be done only if single characters separated by operators are present.

Error handling for cases such as division by 0 has been done. The input expression is evaluated and the result (float or integer) printed on the LCD.

The algorithm/Pseudocode is as following:

1. Define LCD port lines (Enable, Reset and Data lines)
2. Initialize variables used for stack, evaluation and calculations
3. Initialize variables used for lcd display, expression, final result and timer
4. Initialize an array for interfacing matrix keyboard (mode 1 and mode 2)
5. Initialize an array to store the values according to matrix keyboard input for evaluation
6. Initialize an array to display on lcd and an array for the message

//Start of main

7. Initialize LCD functions:
    a. Set 4 bit mode
    b. Set display on
    c. Set increment mode on
    d. Clear display
8. Initially point cursor to first position in first line
9. Initialize timer functions
    a. Set PR and MR0 to required timer values
    b. Set MCR to 0X03 to generate interrupt
    c. Accordingly configure other timer configurations
10. Enable IRQ Handler. Timer0_IRQHandler is invoked every 1ms.
    a. For each ms, the ISS is invoked and a letter is displayed on the LCD
    b. Hence a flowing message appears
11. Configure rows to output (P2.10 to P2.13)I
12. Configure decoder to output (P0.15 to P0.18)

While(1):

13. Continuously scan rows one by one and send to output by using FIOPIN for Port 0.
14. Scan to get column value when a key is pressed
15. If(key is pressed)
    a. If(row=3 and col=3) ->Mode switch button
        i. Increment counter
        ii. Counter keeps track of which mode the calculator is on
    b. Else if(row=3 and col=2) -> Equal to button
        i. Evaluate the expression by converting from infix to postfix
        ii. Invoke contopfix() and Evaluatepostfix()

c. If(mode=1)
    i. Take input character and store in an array to be displayed
    ii. Store the input character in an array to be evaluated later
d. Else mode->2
    i. If(input char='D')
        1. Delete last character by removing it from array and reset cursor
    ii. If(input char='C')
        1. Clear all characters and reset cursor to beginning of display
    iii. Else
        1. Store character in array to display
    iv. Store character in array to evaluate
    v. Reset cursor to beginning of display

//End of main

//LCD functions

Lcd_com():

1. Get temp value for command
2. Shift to set to P0.23 to P0.26 accordingly
3. Send to wr_cn

Wr_cn():

1. Clear RS_CTRL to set to command mode
2. Enable data lines

Lcd_data():

1. Get temp value for data
2. Shift to set to P0.23 to P0.26 accordingly
3. Send to wr_dn

Wr_dn():

1. Set RS_CTRL to set to data mode
2. Enable data lines

Lcd_puts():

1. Get character string
2. Send character by character to lcd_data() to get displayed

//Columns Scan; Polling

Scan():

1. Scan columns and assign co according to which key is pressed

//Infix to postfix expression and evaluation

Push(): Push an element into the stack

Pop():Pop an element from the stack

//Multiple digits of a number are getting encoded to characters and then later decoded. The characters are used as keys to identify the multiple digits or whether sin/cos/log/tan and so on are in the expression. This is needed because conversion to postfix can be done only if single characters separated by operands are present.

Modify_infix():

1. Receives stack which is to be evaluated
2. Checks each operator in the stack
3. If multiple digits then
    a. While(element isdigit)
        i. Repeatedly multiply by 10 and add to get multiple digits
4. Check if stack element=s(sin) or c(cos) or t(tan) or q(square) or l(ln) or r(root) or f(factorial)
5. Here each character is encoded (by taking letter input) and decoded to perform the operation
6. According to the input the corresponding character is decoded and the function is evaluated and stored
7. Operators are added

Contopfix():

Converts infix expression to postfix form by taking precedence into consideration

Prec();

Precedence of each operator is sent to Contopfix() during evaluation

Evaluatepfix():

1. While(array is not empty)
    a. If character between ‘a’ and ‘z’ then send character
    b. If(digit) then pop digit
    c. If(operator) then evaluate and push back into the stack
2. Send expression to lcd_puts() to be displayed on the lcd
3. Convert top of stack to final_result(character array) by using sprint
4. Send final_result to lcd_puts() to be displayed on the lcd

//Timer intitialized to display message one character at a time with appropriate delay between the display of each character of the message : ENTER AN EXPRESSION

Init_timer():

Initialize timer configurations as mentioned in main

TIMER0_IRQHandler():

To display message as mentioned in main

EMBEDDED C SOURCE CODE

```
/*****************************************************************************
 LCD:
    * Port lines used: Data1 to Data4 - P0.23 to P0.26
    * En - P0.28. RS - P0.27, RW - Ground
    * Connection : CND to CNAD. Short jumper JP16

 Matrix Keyboard:
    * Connection: 4x4 Key Matrix to CNB
 *****************************************************************************/

#include <lpc17xx.h>
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <math.h>

#define SIZE 50
#define RS_CTRL  0x08000000  //P0.27 for register select
#define EN_CTRL  0x10000000  //P0.28 for enable
#define DT_CTRL  0x07800000  //P0.23 to P0.26 for the data lines


/*Functions to intialize and display on the LCD*/
void lcd_init(void);
void wr_cn(void);
void clr_disp(void);
void delay_lcd(unsigned int);
void lcd_com(void);
void wr_dn(void);
void lcd_data(void);
void clear_ports(void);
void lcd_puts(unsigned char *);
char getkeycode(void);
void scan(void);

/*Functions to convert infix expression to postfix*/
void push(char);
char pop(void);
signed int prec(char);
void contopfix(void);
void modifyinfix(void);

/*Functions to evalute postfix */
void evaluate_pfix(void);
void push1(float);
float pop1(void);
float s[SIZE];


/*Timer interrupt used to display message at the beginning*/
void TIMER0_IRQHandler(void);
void init_timer0(void);

/* Global varible declarations */
signed int top=-1,x;
unsigned char st_arr[50],infix[50],pfix[50],infix_copy[50];
float op1,op2,temp5,output;

signed int top1=-1;
float arr[50];                              //to store all log,sin etc.
```

```
unsigned int index1=0,y=97, i=0, i1=0, i2, l, precedence, p, z, i5=0;
unsigned char c, ch, c3, e1, e2;

unsigned char col, row, flag;
unsigned long int temp1=0, temp2=0, temp, temp3,i3=0, a, xz=0;
unsigned char msg1;
unsigned char msg2;
unsigned char keychar, original_char;


/*Arrays used as lookup for display on the LCD,
seven_code1 has hex values for 0-9,+,*,-,/,= corresponding to original_code1
seven_code2 has values for other operations, corresponding to original_code2, being
used when mode is switched*/

unsigned char seven_code[4][4], temp4[50], original_code[4][4], final_result[10];
unsigned char
seven_code1[4][4]={0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x2A,0x2B,0x2D,0x2
F,0x3D,0x7A};
unsigned char
seven_code2[4][4]={0x28,0x29,0x73,0x63,0x74,0x6C,0x71,0x72,0x66,0x2F,0x2A,0x2B,0x2D,0x2
F,0x3D,0x7A};      //mode 2
unsigned char
seven_code3[16]={0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x2A,0x2B,0x2D,0x2F,
0x3D,0x7A};

unsigned char equalarray[2]={0x3D,'\0'};
unsigned char original_code1[4][4]={'0','1','2','3','4','5','6','7','8','9','*','+','-
','/','=','z'};       //z for mode switch
unsigned char
original_code2[4][4]={'(',')','s','c','t','l','q','r','f','/','*','+','D','C','=','z'};
//z for mode switch
unsigned int counter=0;

unsigned char myCharPointer[50];
unsigned int infix_index=0, lcdindex=0, length=0, dig;
float output_copy;
unsigned int func_name;
int multi_dig=0;              //variable used to enable multiple digits for input

/*Variables used to display initial message*/
unsigned char message1[20]={'E','N','T','E','R',' ','A','N',''
','E','X','P','R','E','S','S','I','O','N',':'};  //initial message to be displayed
int msg_count=0, msg_disp=0;


/*Main function, execution starts here*/
int main(void)
{

    SystemInit();
    SystemCoreClockUpdate();

    lcd_init();                          //Initialize LCD
    delay_lcd(3200);

    temp1 = 0x80;                       //First message on LCD 1st line
    lcd_com();
    delay_lcd(800);

    init_timer0();                      //Timer interrupt; to display a message
    NVIC_EnableIRQ(TIMER0_IRQn);        //Timer interrupt handler is enabled



    LPC_GPIO2->FIODIR = 0x00003C00;  //set 7 segment display to output (P2.10 to P2.17)
```

```
while(1)
{
    for(row = 0; row<4; row++)          //Scan row
    {
        if(row == 0)
            temp = 0x00000400;
        else if(row == 1)
            temp = 0x00000800;
        else if(row == 2)
             temp = 0x00001000;
         else if(row == 3)
             temp = 0x00002000;
         LPC_GPIO2->FIOPIN = temp;  //Set row selected to 1(high)
         flag= 0;
         scan();
         if(flag == 1)
         {

            if((row==3)&&(col==3)) //Mode switch button is selected, to change mode
            {
                counter++;               //Counter to keep track of mode
                for(a=0; a<=70000; a++);
                break;
            }
            if((row==3)&&(col==2))       //Equal to(=) button is pressed
            {
                modifyinfix();
                contopfix();             //Infix is converted to postfix to evaluate
                evaluate_pfix();         //Postfix expression is evaluated
                for(a=0; a<=70000; a++);
                break;
            }
            if(counter%2==0)          //Change mode; selects from first matrix
            {

                temp4[lcdindex]=seven_code1[row][col];
                //Character to be displayed is stored here
                original_char=original_code1[row][col];
                //Stores the character to evaluate
                lcdindex=lcdindex+1;
                //Getting the actual character
                infix_copy[infix_index++]=original_char;
                //Stores the entire expression to be evaluated
            }
            else
            {

                original_char=original_code2[row][col];
                //Button pressed to delete a single character(backspace)
                if(original_char=='D')
                {
                    temp4[--lcdindex]=0x20;
                    infix_copy[--infix_index]='\0';
                    //Element is removed from the array that has the expression
                    for(a=0; a<=10000; a++);
                    clear_ports();
                    temp1 = 0x80;              //1st message on LCD 1st line
                    lcd_com();
                    lcd_puts(temp4);
                    delay_lcd(800);
                }



                //Key to be pressed to clear the entire display
                else if(original_char=='C')
                {
```

```
                        for(a=0; a<=lcdindex; a++)
                        {
                            temp4[a]=0x20;   //blank space

                        }
                        for(a=0; a<=infix_index; a++)
                        {
                            infix_copy[a]='\0';
                            //Entire expression array is set to null
                        }
                        lcdindex=0;         //Reset display pointer
                        infix_index=0;      //Reset pointer that is used for evaluation
                    }
                    else
                    {
                        temp4[lcdindex]=seven_code2[row][col];
                        func_name = temp4[lcdindex];
                        lcdindex=lcdindex+1;    //Get the actual character
                        infix_copy[infix_index++]=original_char;
                        //Character is stored in array "infix_copy" to be evaluated
                    }
                }


                for(a=0; a<=70000; a++);
                clear_ports();
                temp1 = 0x80;       //1st message on LCD 1st line, 0X80
                lcd_com();
                lcd_puts(temp4);
                delay_lcd(800);

            }
        }
     }
    while(1);
}

/*Function to intialize the LCD Display
Sends: 0x30 thrice, 0x20 to set 4 bit mode, 0x28 to display in two lines
       0x0C to set display on, 0x06 to set increment mode
       0x01 to clear display and 0x80 to set cursor to beginning of first line */
void lcd_init()
{

    LPC_PINCON->PINSEL1 &= 0xFC003FFF;   //Ports initialized as GPIO ; P0.23 to P0.28

    LPC_GPIO0->FIODIR |= DT_CTRL;        // Setting the directions as output
    LPC_GPIO0->FIODIR |= RS_CTRL;
    LPC_GPIO0->FIODIR |= EN_CTRL;

    clear_ports();
    delay_lcd(3200);

    temp2 = (0x30<<19);                 //To wake the LCD; 0x30 sent three times
    wr_cn();
    delay_lcd(30000);

    temp2 = (0x30<<19);
    wr_cn();
    delay_lcd(30000);

    temp2 = (0x30<<19);
    wr_cn();
    delay_lcd(30000);

    temp2 = (0x20<<19);              //Set two 4 bit mode
```

```
    wr_cn();
    delay_lcd(30000);

    temp1 = 0x28;                    //Enable two lines
    lcd_com();
    delay_lcd(30000);

    temp1 = 0x0c;                    //Set display on
    lcd_com();
    delay_lcd(800);

    temp1 = 0x06;                    //Set to increment mode
    lcd_com();
    delay_lcd(800);

    temp1 = 0x01;                    //Clear initially
    lcd_com();
    delay_lcd(10000);

    temp1 = 0x80;                    //Set cursor to beginning of first line i.e, 0x80
    lcd_com();
    delay_lcd(800);
    return;
}

/*Function to send command values to be written to lcd display */
void lcd_com(void)
{
    temp2 = temp1 & 0xf0;           //Move data (26-8+1) times : 26 - HN place, 4 -
Bits
    temp2 = temp2 << 19;            //Data lines from 23 to 26
    wr_cn();
    temp2 = temp1 & 0x0f;           //26-4+1
    temp2 = temp2 << 23;
    wr_cn();
    delay_lcd(1000);
    return;
}

/* Command nibble o/p routine */
void wr_cn(void)                    //Write to command reg
{
    clear_ports();
    LPC_GPIO0->FIOPIN = temp2;      //Assign the value to the data lines
    LPC_GPIO0->FIOCLR = RS_CTRL;    //Clear bit RS
    LPC_GPIO0->FIOSET = EN_CTRL;    //EN=1
    delay_lcd(25);
    LPC_GPIO0->FIOCLR = EN_CTRL;    //EN =0
    return;

}

/*Command o/p routine which also outputs high nibble first and lower nibble next */
void lcd_data(void)
{
    temp2 = temp1 & 0xf0;
    temp2 = temp2 << 19;
    wr_dn();
    temp2= temp1 & 0x0f;
    temp2= temp2 << 23;
    wr_dn();
    delay_lcd(1000);
    return;
}
/*Data o/p routine which also outputs high nibble first and lower nibble next */
void wr_dn(void)
{
```

```
    clear_ports();

    LPC_GPIO0->FIOPIN = temp2;      //Assign the value to the data lines
    LPC_GPIO0->FIOSET = RS_CTRL;    //set bit  RS
    LPC_GPIO0->FIOSET = EN_CTRL;    //EN=1
    delay_lcd(25);
    LPC_GPIO0->FIOCLR = EN_CTRL;    //EN =0
    return;
}

 /*Function provide delay */
void delay_lcd(unsigned int r1)
{
    unsigned int r;
    for(r=0;r<r1;r++);
    return;
}

/*Function to clear the LCD display*/
void clr_disp(void)
{
    temp1 = 0x01;
    lcd_com();
    delay_lcd(10000);
    return;
}

/* Clearing the lines at power on */
void clear_ports(void)
{

    LPC_GPIO0->FIOCLR = DT_CTRL;    //Clearing data lines
    LPC_GPIO0->FIOCLR = RS_CTRL;    //Clearing RS line
    LPC_GPIO0->FIOCLR = EN_CTRL;    //Clearing Enable line
  return;
}

/*Function to display string onto the LCD
  Retrieves the character array and sends it to lcd_data() to be displayed*/
void lcd_puts(unsigned char *buf1)
{
    i5=0;
    while(buf1[i5]!='\0')
    {
        temp1 = buf1[i5];
        lcd_data();
        i5++;
        if(i5==16)
        {
            temp1 = 0xc0;
            lcd_com();
        }

    }
    return;
}

/*Function to read column input; Polling*/
void scan(void)
{
     temp3=LPC_GPIO1->FIOPIN;        //Checks which column is pressed
     temp3=temp3&0x07800000;


     if(temp3!=0x0)
     {
         flag=1;
```

```
          if(temp3==0x00800000)
              col=0;
          else if(temp3==0x01000000)
              col=1;
          else if(temp3==0x02000000)
              col=2;
          else if(temp3==0x04000000)
              col=3;
      }
}

/*Function to push element entered onto stack*/
void push(char symbol)
{
    if(top==49)
    {
        return;
    }
    else
        st_arr[++top]=symbol;
}

/*Function to pop an element from the stack*/
char pop(void)
{
    if(top==-1)
        return '#';
    else
        return(st_arr[top--]);
}

/*Function to check if the stack is empty and return accordingly*/
int isempty(void)
{
    if(top==-1)
        return 1;
    else
        return 0;
}

/*Function to take care of multiple digits as input*/
void modifyinfix(void)
{
    for(i=0;infix_copy[i]!='\0';i++)
    {
        c=infix_copy[i];
        if(isdigit(c))              //If the entered character is a digit
        {
            xz=xz*10+(c-'0');
            //If multiple digits are entered they are stored in xz one by one
            if(!isdigit(infix_copy[i+1]))   //If the entered character is not a digit
            {
                infix[index1++]=(char)(y);
                arr[y-97]=xz;           //The final result is stored
                y++;
                xz=0;
            }
            continue;

        }
        if((c>='a' && c<='z'))
        {


            if(c=='l')          //ln function
            {
                infix[index1++]=(char)(y);
```

```
        while(isdigit(infix_copy[++i]))
        {
            multi_dig=multi_dig*10+(infix_copy[i]-'0');
       //If multiple digits are entered they are stored in multi_dig one by one
        }
        arr[y-97]=log(multi_dig);  //Final log value
        i--;
        multi_dig=0;
        y++;
        continue;
    }

    if(c=='s')           //sin function
    {
        infix[index1++]=(char)(y);
        while(isdigit(infix_copy[++i]))
        {
            multi_dig=multi_dig*10+(infix_copy[i]-'0');
       //If multiple digits are entered they are stored in multi_dig one by one
        }
        arr[y-97]=sin((multi_dig*3.14)/180);    //Final sin value
        i--;
        multi_dig=0;
        y++;
        continue;
    }

    if(c=='c')       //cos function
    {
        infix[index1++]=(char)(y);
        while(isdigit(infix_copy[++i]))
        {
            multi_dig=multi_dig*10+(infix_copy[i]-'0');
        }
        arr[y-97]=cos((multi_dig*3.14)/180);    //Final cos value
        i--;
        multi_dig=0;
        y++;
        continue;
    }
    if(c=='t')      //Tan function
    {
        infix[index1++]=(char)(y);
        while(isdigit(infix_copy[++i]))
        {
            multi_dig=multi_dig*10+(infix_copy[i]-'0');
        }
        arr[y-97]=tan((multi_dig*3.14)/180);    //Final tan value
        i--;
        multi_dig=0;

        y++;
        continue;
    }
    if(c=='q')          //Square funtion
    {
        infix[index1++]=(char)(y);
        while(isdigit(infix_copy[++i]))
        {
            multi_dig=multi_dig*10+(infix_copy[i]-'0');
        }
        arr[y-97]=(multi_dig)*multi_dig;         //Final square value
        i--;
        multi_dig=0;
        y++;
        continue;
```

```
            }
            if(c=='r')          //root function
            {
                infix[index1++]=(char)(y);
                while(isdigit(infix_copy[++i]))
                {
                    multi_dig=multi_dig*10+(infix_copy[i]-'0');
                }
                arr[y-97]=pow(multi_dig,0.5);            //final root value
                i--;
                multi_dig=0;
                y++;
                continue;
            }
            if(c=='f')          //Factorial
            {
                infix[index1++]=(char)(y);
                while(isdigit(infix_copy[++i]))
                {
                    multi_dig=multi_dig*10+(infix_copy[i]-'0');
                }
                while(multi_dig>0)      //Loop to evaluate factorial of number entered
                {
                    fact*=multi_dig;
                    multi_dig--;
                }
                arr[y-97]=fact;
                i--;
                multi_dig=0;
                y++;
                continue;
            }
        }
        else
        {
            infix[index1++]=c;      //for operators
        }
    }
}

/*Function to convert infix to pfix*/
void contopfix(void)
{
    p=0;
    for(i2=0;infix[i2]!='\0';i2++)
    {
        e1=infix[i2];
        switch(e1)
        {
            case '(':push(e1);
                     break;
            case ')':e2=pop();
                    while(e2!='(')
                    {
                        pfix[p++]=e2;
                        e2=pop();
                    }
                    break;
            case '+':
            case '-':
            case '*':
            case '/':if(!isempty())
                     {
                        precedence=prec(e1);
                        e2=pop();
                        while(precedence<=prec(e2))
                        {
```

```
                            pfix[p++]=e2;
                            if(!isempty())
                                e2=pop();
                            else
                            break;
                        }
                        if(precedence>prec(e2))
                            push(e2);
                    }
                    push(e1);
                    break;
            default:pfix[p++]=e1;
                    break;
        }
    }
    while(!isempty())
        pfix[p++]=pop();
}

/*Function to check precedence of input expression*/
int prec(char symbol)
{
    switch(symbol)
    {

        case '/':
        case '*':return 3;
        case '+':
        case '-':return 2;
        case '(':return 0;
        default:return -1;
    }
}

/* Function for PUSH operation */
void push1(float elem)
{
    s[++top1]=elem;
}

/* Function for POP operation */
float pop1(void)
{
    return(s[top1--]);
}

/*Function invoked when '=' is pressed*/
void evaluate_pfix(void)
{
    while( (ch=pfix[i1++]) != '\0')
    {
        if((ch>='a' && ch<='z'))
        {
            x=ch-'a';
            temp5=arr[x];
            push1(temp5);
            continue;
        }
        if(isdigit(ch))
            push1(ch-'0');          //Push the operand
        else
        {                           //Operator,pop two  operands
        op2=pop1();
        op1=pop1();
        switch(ch)
        {
```

```
            case '+':push1(op1+op2);break;
            case '-':push1(op1-op2);break;
            case '*':push1(op1*op2);break;
            case '/':push1(op1/op2);break;
        }
        }
    }
    lcd_puts(equalarray);                //Expression is sent to lcd to be displayed
    sprintf(final_result, "%f", s[top1]);//Integer array is converted to a character
array
    lcd_puts(final_result);              //Final result is sent to lcd to be displayed
    delay_lcd(800);
}

/*Timer0 used to display a message to "Enter an expression";
  Generates an interrupt when TC matched MR0 */
void init_timer0(void)
{
    LPC_TIM0->TCR=0x02;           //Reset
    LPC_TIM0->MR0=3000;
    LPC_TIM0->PR=0;
    LPC_TIM0->MCR=0X03;           //Interrupt is generated when TC matches with MR0
    LPC_TIM0->TCR=0X01;           //Enable
}
/*ISS for Timer0
  Displays one character at a time of the intiial message*/
void TIMER0_IRQHandler(void)
{
    LPC_TIM0->IR=0X01;
    msg_count++;
    if(msg_count==150)
    {
        temp4[0]= message1[msg_disp];   //Character is sent to lcd to be displayed
every time interrupt is generated
        lcd_puts(temp4);
        if(msg_disp==8)
        {
            temp1 = 0xC0;
            lcd_com();
            delay_lcd(10000);
        }
        if(msg_disp==19)                //End of message string is reached
        {
            LPC_TIM0->MCR=0X04;         //Stop timer
            clear_ports();              //Clear ports
            temp1 = 0x01;               //Clear
            lcd_com();
            delay_lcd(10000);
        }
        msg_disp++;
        msg_count=0;
    }
 }
```