Title: Developing a Calculator using ARM LPC1768 Microcontroller for interfacing peripherals.

Objective: To design and implement a scientific calculator using an ARM LPC1768 Microcontroller that interfaces with peripherals such as a keypad for user input and an LCD display for output.

Group Members:

Pranav Mohan- 210905204

Samarth Parashar- 210905206

Aditya Singhvi-210905216

Lakshay Saxena - 210905384

MV Balaji - 210905400

Description:

A scientific calculator is a device which can perform complex mathematical operations and functions. It typically features a range of specialized functions such as trigonometric, logarithmic, and exponential functions, as well as statistical, probability, and engineering functions.

In addition to basic arithmetic operations, such as addition, subtraction, multiplication, and division,we have implemented the ability to calculate more advanced mathematical operations, such as square roots, exponents, and logarithms.

This scientific calculator is capable of performing the following operations:

1. Addition(+)
2. Subtraction(-)
3. Multiplication(*)
4. Division(/)
5. Sin
6. Cos
7. Tan
8. Log
9. Log to the base
10. Exponential
11. Square
12. Cube
13. Square Root
14. nPr
15. nCr
16. Factorial

This calculator will be interfaced with the help of the 16x2 LCD Panel and the 4X4 Key Matrix found on the LPC1768 .The input for this calculator will be given using the 4x4 Key Matrix board and the output will be shown on the 16 x 2 LCD Panel.

Thorough testing and debugging has been conducted to ensure correct functionality and reliability, with attention given to edge cases and user input errors. The final outcome is a functional scientific calculator that showcases the capabilities of the LPC1768 microcontroller for interfacing with peripherals and performing arithmetic calculations accurately and efficiently.

This project will demonstrate the integration of hardware and software components and the utilization of LPC1768 for developing an embedded system application. The resulting calculator can be used in various real-world applications where efficient and accurate arithmetic calculations are required, showcasing the practical implementation of a calculator using an ARM Cortex M microcontroller for interfacing with peripherals.

Configurations:

1. LCD Display:

The LPC1768 board features a small LCD panel that allows for basic visual feedback and interaction with the device. This LCD panel is usually a monochrome, backlit display that can show up to 16 characters per line, with two lines available each of which can display a single character, and two rows of these character positions, one on top of the other. Each character position is usually made up of a 5x8 dot matrix, which can be used to display a variety of characters and symbols.

This LCD can be used to display the message from the controller.16 pin small LCD has to be mounted to the connector CN11. 10 pin connector CNAD is used to interface this LCD from the controller. Only higher 4 data lines are used among the 8 LCD data lines.

2. 4x4 Key Matrix:

The switches SW3 to SW18 are organized as 4 rows X 4 columns matrix. One end of all the switches are configured as columns. The other end of the matrix is configured as rows. A row line will always be an output from the controller. Column lines are pulled to ground.Input signals to be read by a microcontroller or other device using only a few input pins.

Each switch in the matrix is connected to a unique row and column in the matrix. When a switch is pressed, it connects the corresponding row and column together, creating a connection that can be detected by the microcontroller. By scanning through each row and column of the matrix, the microcontroller can determine which switches are pressed and read their corresponding signals.

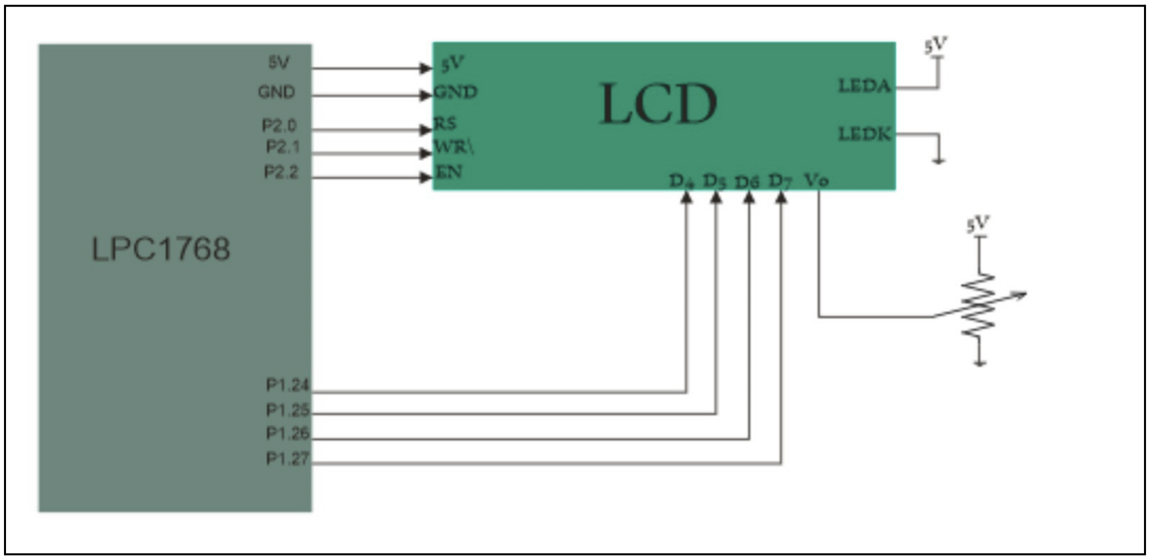
A high level sent from the row will appear at column end if the switch is pressed.

Hardware setup:

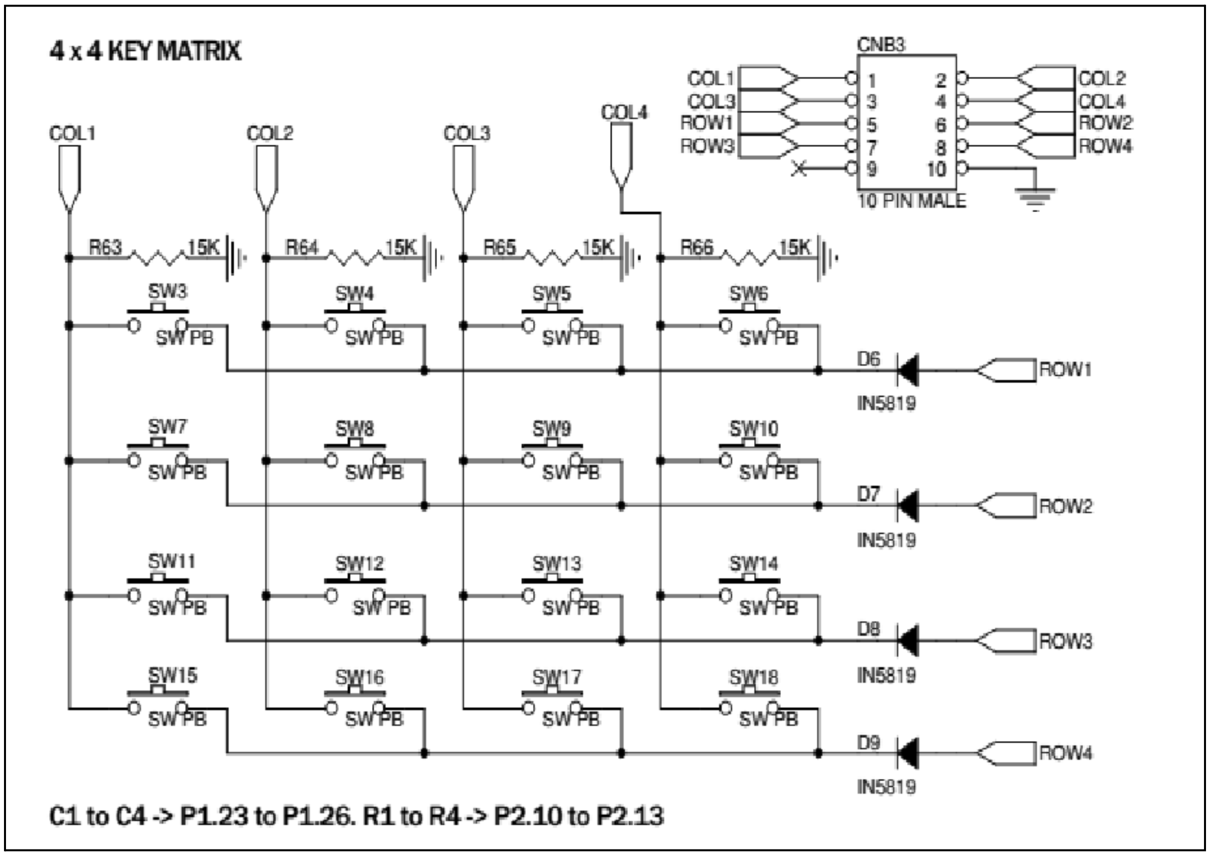
For the LCD Panel connect 10 pin FRC cable from CND to CNAD.

For the 4x4 Key Matrix connect 10 pin FRC cable from CNB to CNB3

Block Diagram:

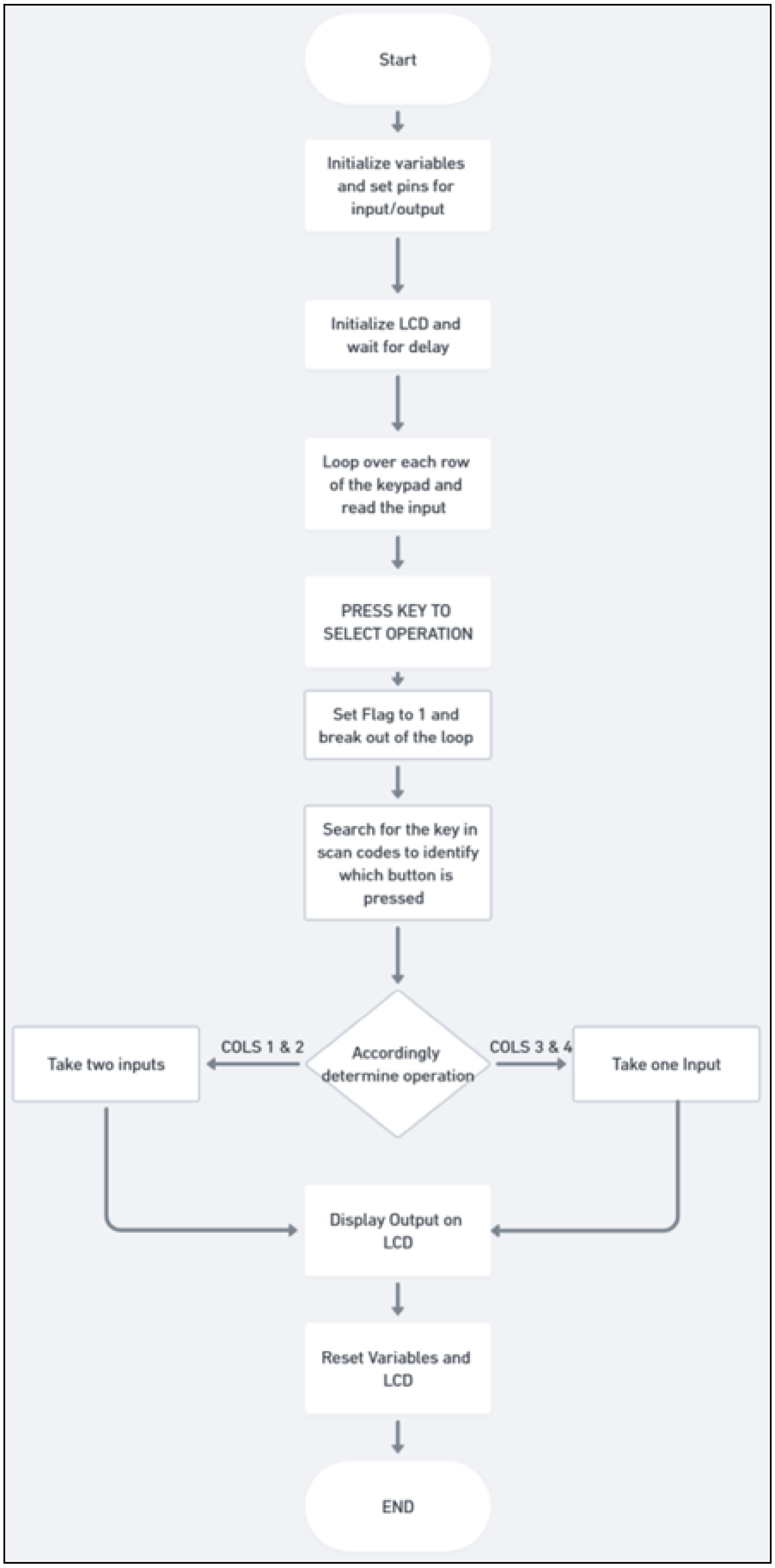


Block Diagram of 16x2 LCD



Block Diagram of 4x4 Switch Key Matrix

Block Diagram of Scientific Calculator:


Start
Initialize variables and set pins for input/output
Initialize LCD and wait for delay
Loop over each row of the keypad and read the input
PRESS KEY TO SELECT OPERATION
Set Flag to 1 and break out of the loop
Search for the key in scan codes to identify which button is pressed
Accordingly determine operation
COLS 1 & 2
Take two inputs
COLS 3 & 4
Take one Input
Display Output on LCD
Reset Variables and LCD
END

Project Code:

Main Code:

```
#include <LPC17xx.h>

#include "lcdmsh.h"

#include <math.h>

void scan(void);

unsigned char row,flag,key;

unsigned long int i,j,var1,temp,temp2,temp3;

unsigned char scan_code[16]={0x11,0x21,0x41,0x81,0x12,0x22,0x42,0x82,

   0x14,0x24,0x44,0x84,0x18,0x28,0x48,0x88};

unsigned char ascii_code[16]={'0','1','2','3','4','5', '6','7','8','9','A','B','+','-','*','/'};

int idx = 3;

int ans = 0;

unsigned char finans[5] = {'0','0','0','0','\0'};

int a = 0;

int b = 0;



char* op;

int count = 0;

unsigned char* operations[4][4] = {

   {"+","-","*","/"},

   {"log","pow","nCr","nPr"},

   {"10log","sin","cos","tan"},

   {"sqrt","square","cube","factorial"}
```

```
};



int num_operands = 0;

int operands[2] = {0};

int result = 0;



int factorial(int n) {

  int result = 1;

        for(int i = 1; i <= n; i++) {

        result *= i;

        }

        return result;

}



int main(void)

{

  LPC_GPIO2->FIODIR = 0x3c00;

  LPC_GPIO1->FIODIR = 0xf87fffff;

        lcd_init();

        temp1 = 0x80;

        lcd_com();

        delay_lcd(800);
```

```
while(1)

{

for(row = 1; row < 5; row++)

{

      if(row == 1)

    var1 = 0x400;

      else if(row == 2)

    var1 = 0x800;

      else if(row == 3)

    var1 = 0x1000;

      else if(row == 4)

    var1 = 0x2000;



      temp = var1;

  LPC_GPIO2->FIOCLR = 0x3c00;

  LPC_GPIO2->FIOSET = var1;



      flag = 0;

  scan();



  if(flag == 1)

      {
        count++;
        break;
            }
}
if(flag == 1)
            break;
}
for (i = 0; i < 4; i++)
{
for (j = 0; j < 4; j++)
{
            if (key == scan_code[i*4+j])
            {
            op = operations[i][j];
            if (i < 2)
            num_operands = 2;
        else
            num_operands = 1;
        break;
            }
}
}

switch (num_operands) {
case 1:
```

```
while(count<1)
            {
        while(1)
            {
            for(row=1; row<5; row++)
            {
                if(row==1)
                        var1 = 0x400;
                else if(row==2)
                        var1 = 0x800;
                else if(row==3)
                        var1 = 0x1000;
                else if(row==4)
                         var1 = 0x2000;
                temp = var1;
                LPC_GPIO2->FIOCLR = 0x3c00;
                LPC_GPIO2->FIOSET = var1;
                flag = 0;
                scan();
             if(flag==1)
                {
                        count++;
                        break;
                }
            }
            if(flag==1)
                break;
            }
        for(i=0; i<16; i++)
            {
            if(key==scan_code[i])
            {
                key = ascii_code[i];
                lcd_puts(&key);
                        delay_lcd(50000);
                if(count==1)
                        a = key-48;
                break;
            }
            }
            }
switch (op) {
            case "10log":
        ans=log(a)/log(10);
        break;
            case "sin":
        ans=sin(a);
        break;
            case "cos":
              ans=cos(a);
        break;
           case "tan":
        ans=tan(a);
        break;
           case "sqrt":
        ans=sqrt(a);
        break;
           case "square":
           ans=pow(a, 2);
        break;
           case "cube":
        ans=pow(a, 3);
        break;
           case "factorial":
        ans=factorial(a);
        break;
    default:
        break;
}
break;
case 2:
while(count<2)
           {
        while(1)
           {
```

```
 for(row=1; row<5; row++)
 {
    if(row==1)
          var1 = 0x400;
    else if(row==2)
          var1 = 0x800;
    else if(row==3)
          var1 = 0x1000;
    else if(row==4)
          var1 = 0x2000;
    temp = var1;
    LPC_GPIO2->FIOCLR = 0x3c00;
  LPC_GPIO2->FIOSET = var1;
    flag = 0;
    scan();
    if(flag==1)
    {
          count++;
          break;
    }
 }
 if(flag==1)
    break;
 }
for(i=0; i<16; i++)
```

```
            {
                if(key==scan_code[i])
            {
               key = ascii_code[i];
               lcd_puts(&key);
               delay_lcd(50000);
               if(count==1)
                    a = key-48;
               else if (count==2)
                      b = key-48;
               break;
            }
            }
            }
switch (op) {
            case "+":
        ans=a+b;
        break;
            case "-":
        ans=a-b;
        break;
            case "*":
        ans=a*b;
        break;
            case "/":
```

```
        ans=a/b;
        break;
           case "log":
        ans=log(a)/log(b);
        break;
           case "pow":
        ans=pow(a, b);
        break;
           case "nCr":
        ans=factorial(a)/(factorial(a-b)*factorial(b));
        break;
           case "nPr":
        ans=factorial(a)/factorial(a-b);
      break;
     default:
        break;
}
break;
default:
break;
}
temp1 = 0xc0;
lcd_com();
delay_lcd(800);
while(ans!=0)
```

```
            {
            finans[idx--] = (ans%10)+48;
            ans = ans/10;
            }
    lcd_puts(&finans[0]);
            return 0;

}
void scan(void)
{
            temp3 = LPC_GPIO1->FIOPIN;
            temp3 &= 0x07800000;
            if(temp3!=0)
            {
            flag = 1;
        temp3>>=19;
        temp>>=10;
            key = temp3|temp;
            }
}
```

Header File:

```
 #include<LPC17xx.h>
```

```
#define RS_CTRL 0x08000000;

#define EN_CTRL 0x10000000;

#define DT_CTRL 0x07800000;


unsigned long int temp1 = 0, temp2 = 0;

void lcd_init(void);

void lcd_com(void);

void clear_ports(void);

void delay_lcd(unsigned int);

void lcd_puts(unsigned char*);

void lcd_data(void);

void wr_cn(void);

void wr_dn(void);



void lcd_init(void)

{

	LPC_GPIO0->FIODIR |= DT_CTRL;

	LPC_GPIO0->FIODIR |= RS_CTRL;

	LPC_GPIO0->FIODIR |= EN_CTRL;

	clear_ports();

	delay_lcd(3200);

	temp1 = 0x33;

	lcd_com();

	delay_lcd(800);
	temp1 = 0x32;
	lcd_com();
	delay_lcd(800);
	temp1 = 0x28;
	lcd_com();
	delay_lcd(800);
	temp1 = 0x0c;
	lcd_com();
	delay_lcd(800);
	temp1 = 0x06;
	lcd_com();
	delay_lcd(800);
	temp1 = 0x01;
	lcd_com();
	delay_lcd(10000);
	return;
}

void clear_ports(void)
{
	LPC_GPIO0->FIOCLR = DT_CTRL;
	LPC_GPIO0->FIOCLR = RS_CTRL;
	LPC_GPIO0->FIOCLR = EN_CTRL;
}
```

```
void delay_lcd(unsigned int r1)
{
        unsigned int r;
        for(r=0; r<r1; r++);
        return;
}

void lcd_com(void)
{
        temp2 = temp1 & 0xf0;
        temp2 = temp2<<19;
        wr_cn();
        delay_lcd(30000);
        temp2 = temp1 & 0x0f;
        temp2 = temp2<<23;
        wr_cn();
        delay_lcd(30000);
        return;
}

void wr_cn()
{
        LPC_GPIO0->FIOPIN = temp2;
        LPC_GPIO0->FIOCLR = RS_CTRL;
        LPC_GPIO0->FIOSET = EN_CTRL;
        delay_lcd(25);

        LPC_GPIO0->FIOCLR = EN_CTRL;

}


void lcd_puts(unsigned char* buf)

{

        unsigned int i=0;

        while(buf[i]!='\0')

        {

        temp1=buf[i];

        lcd_data();

        delay_lcd(800);

        i++;

        if(i==16)

        {

        temp1=0xc0;

        lcd_com();

        delay_lcd(800);

        }

        }

        return;

}


void lcd_data(void)

{
        temp2 = temp1 & 0xf0;

        temp2 = temp2<<19;

        wr_dn();

        delay_lcd(30000);

        temp2 = temp1 & 0x0f;

        temp2 = temp2<<23;

        wr_dn();

        delay_lcd(30000);

        return;

}


void wr_dn()

{

        LPC_GPIO0->FIOPIN = temp2;

        LPC_GPIO0->FIOSET = RS_CTRL;

        LPC_GPIO0->FIOSET = EN_CTRL;

        delay_lcd(25);

        LPC_GPIO0->FIOCLR = EN_CTRL;

        return;

}
```