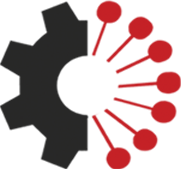
**FYP LAB MANUAL (MCT-498)**



**Submitted by: SYED FARAZ ABBAS.**

**Supervisor: DR. AHSAN NAEEM.**

DEPARTMENT OF MECHATRONICS & CONTROL ENGINEERING.

**University of Engineering and Technology, Lahore**

**.**DECEMBER 24, 2021

Contents

[INTRODUCTION: 2](#_Toc92916775)

[KEYPAD 4\*4: 2](#_Toc92916776)

[SPECIFICATIONS: 3](#_Toc92916777)

[APPLICATIONS: 3](#_Toc92916778)

[DATASHEET: 4](#_Toc92916779)

[POTENTIOMETRE: 4](#_Toc92916780)

[USAGE: 5](#_Toc92916782)

[APPLICATIONS: 5](#_Toc92916783)

[CIRCUIT DIAGRAM: 6](#_Toc92916784)

[FORMULA: 6](#_Toc92916785)

[CHARACTER L.C.D: 6](#_Toc92916786)

[CODE: 7](#_Toc92916787)

[STEPS: 16](#_Toc92916788)

[CONNECTIONS: 17](#_Toc92916789)

**LAB MANUAL:**

INTERFACING KEYPAD 4\*4 AND CHARACTER L.C.D WITH TIVA LAUNCHPAD.

# INTRODUCTION:

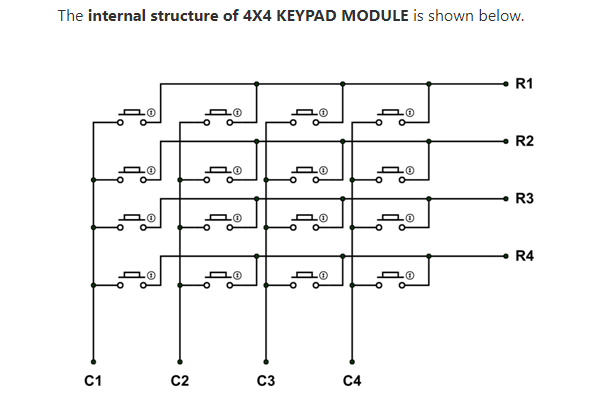
## KEYPAD 4\*4:

KEYPAD 4\*4 is used to enter input in projects. It can be used with different microcontrollers. In our project, we shall be using KEYPAD 4\*4 with Tiva Launchpad to enter input. And then that input will be displayed on the LCD.

4X4 KEYPAD MODULES are available in different sizes and shapes. But they all have same pin configuration. It is easy to make 4X4 KEYPAD by arranging 16 buttons in matrix formation by yourself.

|  |  |
| --- | --- |
| **ROWS** | **DESCRIPTION** |
| 1 | PIN1 is taken out from 1st  ROW |
| 2 | PIN2 is taken out from 2nd  ROW |
| 3 | PIN3 is taken out from 3rd  ROW |
| 4 | PIN4 is taken out from  4th ROW |
| **COLUMNS** | **DESCRIPTION** |
| 5 | PIN5 is taken out from 1st  COLUMN |
| 6 | PIN6 is taken out from 2nd  COLUMN |
| 7 | PIN7 is taken out from 3rd  COLUMN |
| 8 | PIN8 is taken out from 4th COLUMN |
|  |  |

As given in above table, a **4X4 KEYPAD** will have **EIGHT TERMINALS**. In them four are **ROWS of MATRIX** and four are **COLUMNS of MATRIX**. These 8 PINS are driven out from 16 buttons present in the MODULE. Those 16 alphanumeric digits on the MODULE surface are the 16 buttons arranged in MATRIX formation.



## SPECIFICATIONS:

* Maximum Voltage across EACH SEGMENT or BUTTON: 24V
* Maximum Current through  EACH SEGMENT or BUTTON: 30mA
* Maximum operating temperature: 0°C to + 50°C
* Ultra-thin design
* Adhesive backing
* Easy interface
* Long life

## APPLICATIONS:

* Security systems.
* Vending machines.
* Industrial machines.
* Engineering systems.
* Measuring instruments.



* Data entry for Embedded Systems
* Hobby projects.
* Basically anywhere INPUT device is needed

## DATASHEET:

<https://components101.com/sites/default/files/component_datasheet/4x4%20Keypad%20Module%20Datasheet.pdf>

## POTENTIOMETRE:

A potentiometer is **a manually adjustable variable resistor with 3 terminals**. ... A potentiometer is also commonly known as a potmeter or pot. The most common form of potmeter is the single turn rotary potmeter. This type of pot is often used in audio volume control (logarithmic taper) as well as many other applications.

There are multiple types of potentiometers that rae available in the market:

* Digital potentiometer
* Membrane potentiometer

A potentiometer is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider. If only two terminals are used, one end and the wiper, it acts as a variable resistor or rheostat.

## C:\Users\dell\Desktop\ppp.jpg

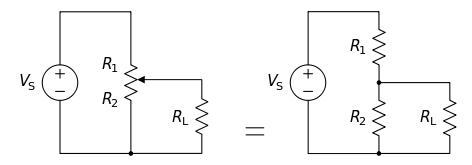
## USAGE:

It is used in our project to control the brightness of the LCD screen. We will be using Rotary potentiometer. Rotary potentiometer (the most common type) vary their resistive value as a result of an angular movement. Rotating a knob or dial attached to the shaft causes the internal wiper to sweep around a curved resistive element. The most common use of a rotary potentiometer is the volume-control pot.

## APPLICATIONS:

* Audio control
* Television
* Motion control
* Transducers
* Computation

## CIRCUIT DIAGRAM:

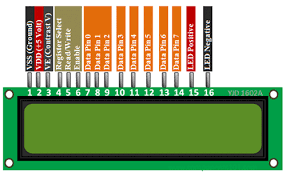


## FORMULA:

**C:\Users\dell\Desktop\FR.PNG**

## CHARACTER L.C.D:

We will be displaying numbers on the character lcd. These LCDs are ideal for displaying text/characters only, hence the name ‘Character LCD’. The display has an LED backlight and can display 32 ASCII characters in two rows with 16 characters on each row. Most of the LCD Displays available in the market are 16\*2 (That means, the LCD displays are capable of displaying 2 lines each having 16 Characters a), 20X4 LCD Displays (4 lines, 20 characters).

****

GND should be connected to the ground of Tiva launchpad

VCC is the power supply for the LCD which we connect the 5 volts pin on the Tiva launchpad.

Vo (LCD Contrast) controls the contrast and brightness of the LCD. Using a simple voltage divider with a potentiometer, we can make fine adjustments to the contrast.

RS (Register Select) pin lets the Tiva launchpad tell the LCD whether it is sending commands or the data. Basically this pin is used to differentiate commands from the data.

For example, when RS pin is set to LOW, then we are sending commands to the LCD (like set the cursor to a specific location, clear the display, scroll the display to the right and so on). And when RS pin is set on HIGH we are sending data/characters to the LCD.

R/W (Read/Write) pin on the LCD is to control whether or not you’re reading data from the LCD or writing data to the LCD. Since we’re just using this LCD as an OUTPUT device, we’re going to tie this pin LOW. This forces it into the WRITE mode.

E (Enable) pin is used to enable the display. Meaning, when this pin is set to LOW, the LCD does not care what is happening with R/W, RS, and the data bus lines; when this pin is set to HIGH, the LCD is processing the incoming data.

D0-D7 (Data Bus) are the pins that carries the 8 bit data we send to the display. For example, if we want to see the uppercase ‘A’ character on the display we will set these pins to 0100 0001(according to the ASCII table) to the LCD.

A-K (Anode & Cathode) pins are used to control the backlight of the LCD.



# CODE:

Here is your code.

#include "TM4C123.h" /\* include register defintion file of TM4C123GH6PM \*/

#define LCD GPIOB /\* Define "LCD" as a symbolic name for GPIOB \*/

#define RS 0x01 /\* PORTB BIT5 mask \*/

#define RW 0x02 /\* PORTB BIT6 mask \*/

#define EN 0x04 /\* PORTB BIT7 mask \*/

#define HIGH 1

#define LOW 0

/\*define useful symbolic names for LCD commands \*/

#define clear\_display 0x01

#define returnHome 0x02

#define moveCursorRight 0x06

#define moveCursorLeft 0x08

#define shiftDisplayRight 0x1C

#define shiftDisplayLeft 0x18

#define cursorBlink 0x0F

#define cursorOff 0x0C

#define cursorOn 0x0E

#define Function\_set\_4bit 0x28

#define Function\_set\_8bit 0x38

#define Entry\_mode 0x06

#define Function\_8\_bit 0x32

#define Set5x7FontSize 0x20

#define FirstRow 0x80

/\*

//---SYSTEM CONTROL REGISTERS---LED//

#define SYSCTL\_RCGCGPIO\_R (\*((volatile unsigned long \*) 0x400FE608))

#define GPIO\_PORTF\_DEN\_R (\*((volatile unsigned long \*) 0x4002551C))

#define GPIO\_PORTF\_DIR\_R (\*((volatile unsigned long \*) 0x40025400))

#define GPIO\_PORTF\_DATA\_R (\*((volatile unsigned long \*) 0x40025038))

#define GPIO\_PORTF\_CLK\_EN 0x20

#define GPIO\_PORTF\_PIN1\_EN 0x02

#define GPIO\_PORTF\_PIN2\_EN 0x04

#define GPIO\_PORTF\_PIN3\_EN 0x08

#define LED\_ON1 0x02

#define LED\_ON2 0x04

#define LED\_ON3 0x08

\*/

#define DELAY\_VALUE 9000

void Delay(unsigned long );

void delay(unsigned long);

void Delay(unsigned long n)

{

volatile unsigned long i;

for(i=0;i<n;i++);

}

/\* KEYPAD\*/

void Init\_GPIO\_C(void);

unsigned char scan\_keypad(void);

void Init\_GPIO\_C()

{

  SYSCTL->RCGCGPIO |= 0x24; //PORTC & F clock

  GPIOC->DIR |=0x00; //PORTC rows

  GPIOC->PDR |=0xFF;

  GPIOC->DEN |=0xFF;

  GPIOF->DIR |=0xFF; //PORTF colummns

  GPIOF->DEN |=0xFF;

}

static int c=0;

static unsigned int col;

static unsigned const char key[4][4] ={   //KEYPAD INTERFACE

                  {'1','2','3','A'},

                  {'4','5','6','B'},

                  {'7','8','9','C'},

                  {'\*','0','#','D'}

                };

                /\* prototypes of LCD functions \*/

void delay\_ms(int n); /\* mili second delay function \*/

void delay\_us(int n); /\* micro second delay function \*/

void LCD\_init(void); /\* LCD initialization function \*/

void LCD\_Cmd(unsigned char command); /\*Used to send commands to LCD \*/

void LCD\_Write\_Char(unsigned char data); /\* Writes ASCII character \*/

void LCD\_Write\_Nibble(unsigned char data, unsigned char control); /\* Writes 4-bits \*/

void LCD\_String (char \*str);  /\* Send string to LCD function \*/

unsigned char scan\_keypad(void)

{

  unsigned int i=0,j=0,x=0,y=0;

  //int i=0;

//  unsigned char input='x';

//  unsigned char read=(unsigned char)GPIOC->DATA;

//  unsigned char read1;

  while(1) {

//    read1=GPIOC->DATA&0xF0;

//  read1=read1>>3;

//  GPIOF->DATA = read1;

//  return read1;

/\*read=(unsigned char)GPIOC->DATA;

    Delay(50);

read=~read;

read=read>>3;

GPIOF->DATA=read;

Delay(50);

\*/

//read=read>>3;

  for( i=0;i<4;i++) //COLUMN TRAVERSAL

  {

    GPIOF->DATA = 0x01<<(i+1);

  //  read1=(unsigned char)GPIOF->DATA;

  //  read1= (unsigned char)(read1<<3) ;

  //  Delay(100);

  //  read=(unsigned char)GPIOC->DATA;

    Delay(50000);

  //  read=~read;

    //  GPIOF->DATA = 0x04;

    for( j=0;j<4;j++)

    {

      //Delay(4500);

      if(((GPIOC->DATA)&0xF0)&(1U<<(j+4)))

      {

        x=j;

        y=i;

        j=0;

        i=0;

        return key[x][y];

      //  LCD\_Write\_Char(key[x][y]);

      //Delay(18000);

      }

    //  else LCD\_Write\_Char('X');

    }

/\*    if(~(GPIOC->DATA)& 0x10)

    {

        while(~(GPIOC->DATA) & 0x10){}

    //  input= key[0][i];

      //  input='1';

    //  if(read1&0x10)

    //    return key[0][0];

    //  if(read1&0x20)

      //  return key[0][1];

    //  else return key[0][2];

          return key[0][i];

    }

     if((~(GPIOC->DATA)) & 0x20)

    {

      while((~(GPIOC->DATA)) & 0x20){}

    //  input= key[1][i];

    //    input='2';

    //    return '2';

        return key[1][i];

    }

     if((~(GPIOC->DATA)) & 0x40)

    {

      while((~(GPIOC->DATA)) & 0x40){}

      //input= key[2][i];

      //  input='3';

        return '3';

    }

     if((~(GPIOC->DATA)) & 0x80)

    {

      while((~(GPIOC->DATA)) & 0x80){}

    //  input= key[3][i];

      //  input='4';

        return '4';

    }

    //Delay(9000);

    //if(input!='x')\*/

  }

  //return input;

}

}

#define DELAY\_VALUE 9000

void Delay(unsigned long );

void delay(unsigned long);

int main()

{

  /\*LED BLINKING CODE

  SYSCTL\_RCGCGPIO\_R |= GPIO\_PORTF\_CLK\_EN; //enable clock for PORTF

GPIO\_PORTF\_DEN\_R |= GPIO\_PORTF\_PIN1\_EN; //enable pins 1 on PORTF

GPIO\_PORTF\_DIR\_R |= GPIO\_PORTF\_PIN1\_EN; //make pins 1 as output pins

GPIO\_PORTF\_DEN\_R |= GPIO\_PORTF\_PIN2\_EN; //enable pins 2 on PORTF

GPIO\_PORTF\_DIR\_R |= GPIO\_PORTF\_PIN2\_EN; //make pins 2 as output pins

GPIO\_PORTF\_DEN\_R |= GPIO\_PORTF\_PIN3\_EN; //enable pins 3 on PORTF

GPIO\_PORTF\_DIR\_R |= GPIO\_PORTF\_PIN3\_EN; //make pins 3 as output pins

\*/

  //LCD CODE

   LCD\_init();

  Init\_GPIO\_C();

unsigned char input1='\0';

  //GPIOF->DATA=0x01;

  while(1)

{

 LCD\_Cmd(clear\_display);

LCD\_Cmd(FirstRow); /\* Force cusor to begining of first row \*/

// delay\_ms(500);

  //   for(unsigned int i=1;i<=250;i++)

  //{

  //  Delay(9000);

  //}

  /\*

for (unsigned char i=0;i<4;i++)

{

  for (unsigned char j=0;j<4;j++)

    LCD\_Write\_Char(key[i][j]);

   for(unsigned int k=1;k<=250;k++)

  {

    Delay(9000);

  }

}

  \*/

// LCD\_Write\_Char('L');

//delay\_ms(500);

LCD\_Write\_Char('K');

LCD\_Write\_Char('E');

LCD\_Write\_Char('Y');

LCD\_Write\_Char(':');

LCD\_Write\_Char('-');

     for(unsigned int i=1;i<=250;i++)

  {

    Delay(9000);

  }

input1=scan\_keypad();

LCD\_Write\_Char(input1);

//  GPIOF->DATA=GPIOF->DATA+1;

   for(unsigned int i=1;i<=250;i++)

  {

    Delay(27000);

  }

  //   LCD\_Cmd(0x71);

  //   LCD\_Write\_Char(0x71);

    Delay(24);

  //delay\_us(40);

  //  LCD\_Cmd(0xFC);

  //delay\_us(40);

}

}

/\* LCD and GPIOB initialization Function \*/

void LCD\_init(void)

{

SYSCTL->RCGCGPIO |=(1<<1); /\* Enable Clock to GPIOB \*/

LCD->DIR |=0xFF; /\* Set GPIOB all pins a digital output pins \*/

LCD->DEN |=0xFF; /\* Declare GPIOB pins as digital pins \*/

LCD\_Cmd(Set5x7FontSize); /\* select 5x7 font size and 2 rows of LCD \*/

LCD\_Cmd(Function\_set\_4bit); /\* Select 4-bit Mode of LCD \*/

LCD\_Cmd(moveCursorRight); /\* shift cursor right \*/

LCD\_Cmd(clear\_display); /\* clear whatever is written on display \*/

LCD\_Cmd(cursorBlink); /\* Enable Display and cursor blinking \*/

}

void LCD\_Cmd(unsigned char command)

{

LCD\_Write\_Nibble(command & 0xF0, 0); /\* Write upper nibble to LCD \*/

LCD\_Write\_Nibble((command << 4)&0xF0, 0); /\* Write lower nibble to LCD \*/

if (command < 4)

// delay\_ms(2); /\* 2ms delay for commands 1 and 2 \*/

        Delay(9000);

    //Delay();

else

// delay\_us(40); /\* 40us delay for other commands \*/

    Delay(24);

}

void LCD\_Write\_Nibble(unsigned char data, unsigned char control)

{

data &= 0xF0; /\* Extract upper nibble for data \*/

control &= 0x0F; /\* Extract lower nibble for control \*/

// LCD->DATA = data | control; /\* Set RS and R/W to zero for write operation \*/

LCD->DATA = data | control | EN; /\* Provide Pulse to Enable pin to perform wite operation \*/

  // LCD->DATA = data | control;

//delay\_ms(20000);

  Delay(9000);

  //  delay\_us(2000);

// LCD->DATA = data; /\*Send data \*/

    LCD->DATA = data | control;

// LCD->DATA = 0; /\* stop writing data to LCD \*/

}

void LCD\_Write\_Char(unsigned char data)

{

LCD\_Write\_Nibble(data & 0xF0, RS); /\* Write upper nibble to LCD and RS = 1 to write data \*/

LCD\_Write\_Nibble((data << 4)&0xF0, RS); /\* Write lower nibble to LCD and RS = 1 to write data \*/

// delay\_us(40);

    Delay(24);

  /\* GPIO\_PORTF\_DATA\_R = 0x02; /

  /Turn on RED LED

  Delay(); //Delay almost 1 sec

  GPIO\_PORTF\_DATA\_R = 0x00; //Turn off LED

  Delay(); //Delay almost 1 sec

  \*/

}

void LCD\_String (char \*str) /\* Send string to LCD function \*/

{

  int i;

  for(i=0;str[i]!=0;i++) /\* Send each char of string till the NULL \*/

  {

    LCD\_Write\_Char(str[i]); /\* Call LCD data write \*/

  }

}

/\* Mili seconds delay function /\*

/\*void delay\_ms(int n)

{

int i,j;

for(i=0;i<n;i++)

  {

for(j=0;j<3180;j++)

{}

}

}

\*/

void delay\_ms(int n)

{

int i,j;

for(i=0;i<n;i++)

for(j=0;j<31800;j++);

}

/\* Micro seconds delay function \*/

void delay\_us(int n)

{

int i,j;

for(i=0;i<n;i++)

for(j=0;j<3;j++);

}

# STEPS:

* Write that code on keil.
* Build it
* Connect TIVA Launchpad with your laptop.
* Load code in your microcontroller.
* Then make connections of L.C.D and keypad as shown in figure below.

# CONNECTIONS:

|  |  |
| --- | --- |
| **LCD PINS** | **TIVA PINS** |
| **16** | **grd** |
| **16** | **potentiometre** |
| **16** | **1 lcd -----** |
| **15** | **vbus** |
| **15** | **Potentiometer same** |
| **14** | **Pb7** |
| **13** | **Pb6** |
| **12** | **Pb5** |
| **11** | **Pb4** |
| **1** | **16 lcd------** |
| **2** | **Potentiometer sameee** |
| **3** | **resistor** |
| **4** | **Pb0** |
| **5** | **Pb1** |
| **6** | **Pb2** |
| **Potentiometer middle** | **resistor** |
|  |  |
| **KEYPAD PINS** | **TIVA PINS** |
| **1** | **Pf0** |
| **2** | **Pc5** |
| **4** | **Pc7** |
| **3** | **Pc6** |
| **8** | **Pf4** |
| **7** | **Pf5** |
| **5** | **Pf1** |
| **6** | **Pf2** |