BIRZEIT UNIVERSITY

Electrical and Computer Engineering

ENCS4110 | Computer Design LAB

Experiment No. 9: LCD Interfacing with TM4C123

Objectives

In this experiment, you will learn to interface a 16×2 LCD with TM4C123 Tiva C Launchpad. As you know that 16×2 LCD is widely used in embedded projects to display different types of sensor data. Firstly, we will discuss the pinout of LCD and its connection diagram with Tiva Launchpad. After that, we will discuss the 16×2 LCD driver for TM4C123 Tiva Launchpad using Keil uvision IDE.

16×2 LCD Introduction

This LCD can display 32 ASCII characters. It consists of two rows and one column. Each row can display one ASCII character. Hence, the position of each character is defined in terms of rows and columns order pairs (x,y). For example, (0,0) means the first row and first column, and (1,15) means the second row and 15th column.

Function Description:

Registers

The LCD controller has two 8-bit registers, an instruction register (IR) and a data register (DR). The IR stores instruction codes, such as display clear and cursor shift, and address information for display data RAM (DDRAM) and character generator RAM (CGRAM).

The IR can only be written from the MPU. The DR temporarily stores data to be written into DDRAM or CGRAM and temporarily stores data to be read from DDRAM or CGRAM.

Data written into the DR from the MPU is automatically written into DDRAM or CGRAM by an internal operation. The DR is also used for data storage when reading data from DDRAM or CGRAM. When address information is written into the IR, data is read and then stored into the DR from DDRAM or CGRAM by an internal operation.

Memory

In 16×2 LCD controller, there are three memory are available to store characters, numbers and special symbols. Which are DDRAM (data display RAM) which stores ASCII codes, CGROM (character generating ROM) which is responsible for stored standard character pattern, and CGRAM (character generating RAM) which holds custom character pattern space total 8 in 2×16 module.

Display Data RAM (DDRAM)

Display data RAM (DDRAM) stores display data represented in 8-bit character codes. Its extended capacity is 80×8 bits, or 80 characters. The area in display data RAM (DDRAM) that is not used for display can be used as general data RAM.

Character Generator ROM (CGROM)

The character generator ROM that is responsible for stored standard character pattern generates 5×8 dot or 5×10 dot character patterns from 8-bit character codes. It can generate 208 5×8 dot character patterns and 32 5×10 dot character patterns.

Character Generator RAM (CGRAM)

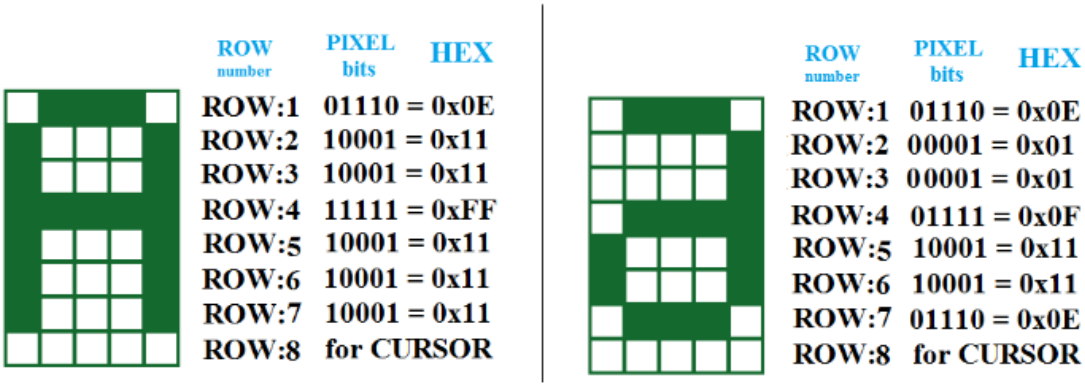
The character generating RAM which holds custom character pattern has only 8 memory location available to store user defined characters with address 0x00 - 0x07 , which is shown in the figure 3.1.

Displaying Standard Character on LCD

Out of these three memory locations, DDRAM and CGROM are used to generate regular standard characters (ASCII characters). By using these three memory locations, a user can generate different character fonts and symbols on LCD display. A character font describes the shape and style of the character. Each shape of a character is designed by taking the number of pixels in mind. For example, in 16x2 LCD there are 16 segments available per single line. Each segment contains pixels in 5x7 or 5x10 matrix forms.

For example, in 16×2 LCD there are 16 segments available per single line. Each segment contains pixels in 5×8 or 5×10 matrix forms.

For example, a character in both uppercase ‘A’ and lowercase ‘a’ is designed by energizing corresponding pixels as shown below.



All these eight hexadecimal codes (referred as character pattern) of each character are stored in character generator ROM (CGROM) area.

The Display Data RAM (DDRAM) stores the ASCII code of a character which is sent by the microcontroller. Now the LCD controller maps the corresponding ASCII

Code in DDRAM with CGROM address to bring the hexadecimal codes (character pattern) of that particular character. By using those hexadecimal codes the 5x7 matrix segment will light according to that character pattern to display corresponding character on it as shown in the figure below.

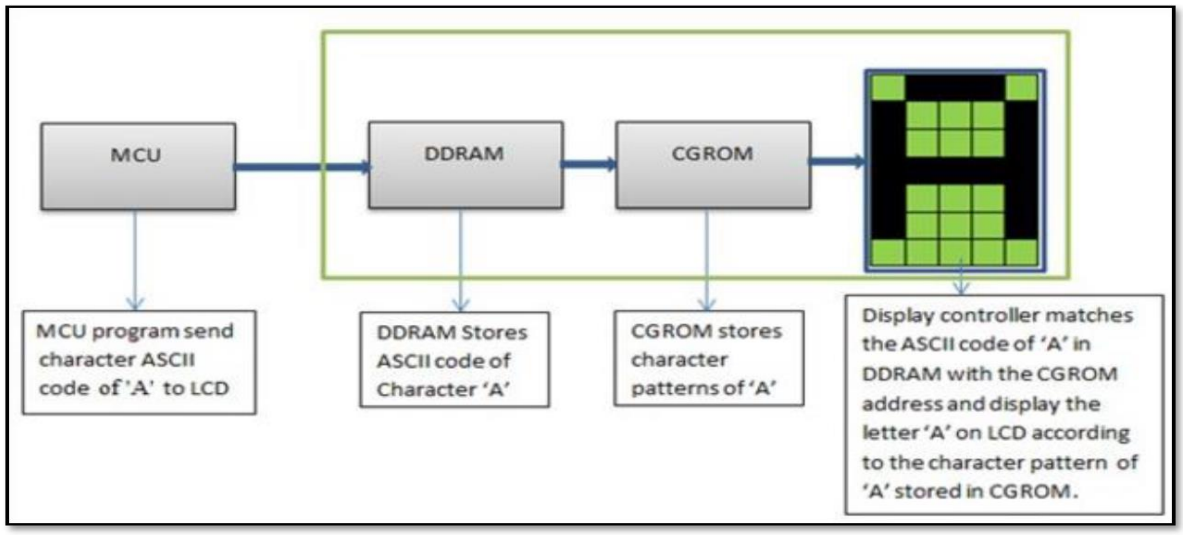


Figure 3.2 : block diagram shows character generation on LCD.

Pinout Diagram

Following diagram shows the pinout diagram of 16×2 LCD.

16×2 LCD Pin Details

Let's first discuss the pin details of LCD. It consists of 16 pins such as data, control, power supply, and backlight LED pins.

Data Pins

Pins D0 to D7 are data pins which are used to send data to be displayed or commands to LCD controller. Hence, these lines will be connected with TM4C123 GPIO pins to transfer data. But this LCD can be used either in 4-bit (only D0 to D3 pins are used) or 8-bit mode(all D0 to D7 pins are used).

Control Pins:

LCD Contrast (Vo) : It is used to adjust the contrast of LCD with respect to text display. This contrast can be set through by using a potential divider circuit with a variable resistor.

Register Select (RS) : With the help of this pin, TM4C123 microcontroller informs the LCD controller either we are sending commands or data to LCD. In other words, it helps to differentiate between data and commands.

For example, when we want to send commands to LCD from TM4C123 microcontroller, we set this pin active low by sending an active signal from the GPIO pin of Tiva Launchpad. These commands are setting cursor position, cursor on or off, scrolling text left or right, clearing text from LCD, etc. On the contrary, when we want to send data to LCD, we provide active high signal to this pin from TM4C123 microcontroller

Read/Write (R/W)

As its name suggests, R/W is used to select read or write mode of LCD. When this pin is set to active high, LCD will be in read mode which means we can read data from the LCD. Similarly, when this pin is set to active low, we will be able to write data to the LCD. We will connect this pin to the ground. Because we are using a 16×2 LCD as an output device only.

Enable (E)

This pin is used to enable and disable LCD. When this pin is active low, LCD controller will be disabled. That means control pins and data pins will not have any effect on the display. On the other hand, when the enable pin is set to active high, the LCD will work normally and process all data and control instructions.

Backlight LED Pins

These pins are cathode and anode pins of back light LED. They are used to provide +5 volts and ground to anode and cathode pins.

Power Supply Pins

+5 volt power supply pins.

Difference Between 4-bit and 8-bit mode

HD47780 LCDs can be interfaced with TM4C123 Tiva Launchpad either in 4-bit or 8-bit mode. For a 4-bit interface, we need to use 6 GPIO pins of Launchpad. In 4-bit mode, data transfers from microcontroller to the LCD in two consecutive half bytes. On the contrary for 8-bit mode, one byte data transfers to the LCD in one go. Consequently, we need to use 10 GPIO pins of TM4C123 microcontroller. Therefore, to save GPIO pins, we will use 4-bit mode of 16×2 LCD. Because, displaying data on LCD is not a time-critical action. Hence. 4-bit use will save us 4 GPIO pins of the Tiva launchpad.

16×2 LCD Commands

The following table lists some of the most commonly used commands which are used to configure and control LCD. You can consult the datasheet of the 16×2 controller to know about more commands. As we mentioned earlier, the RS should be made active low to send commands to LCD using D0-D7 data lines.

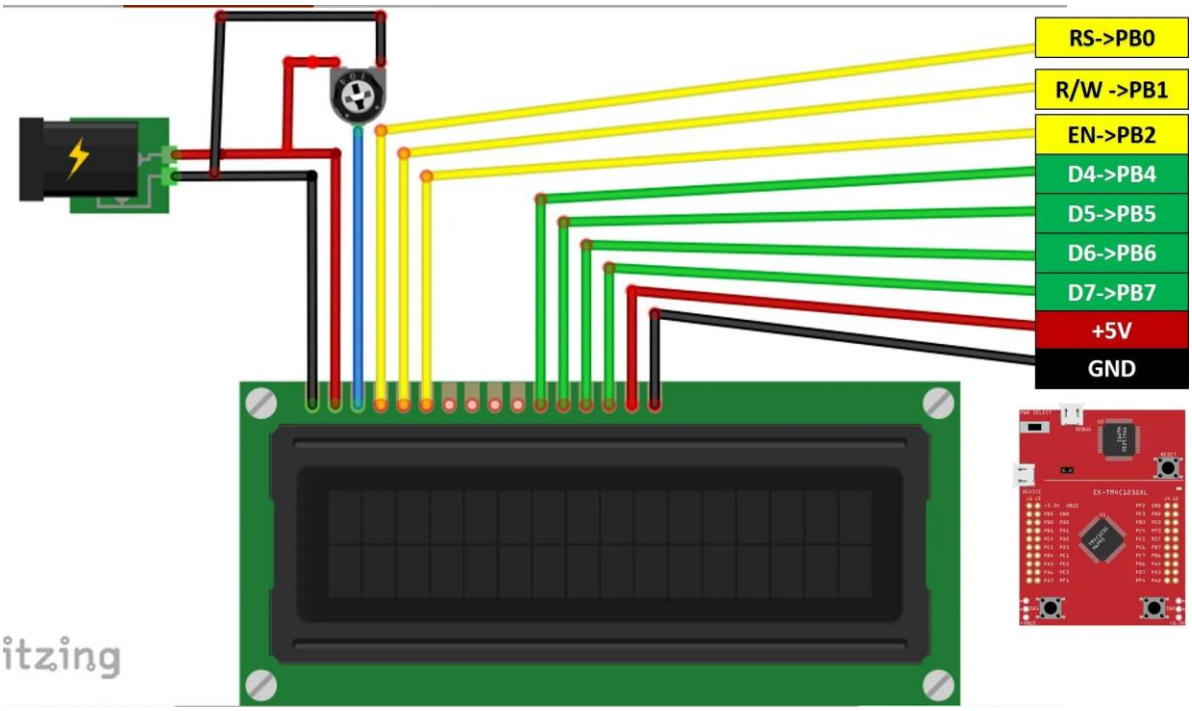
In the later section of this experiment, we will see how to send commands from TM4C123 Tiva Launchpad to LCD.

Command	Function
0x01	Clear display screen
0x02	Return cursor to home i.e starting position
0x06	Shift the cursor one position right
0x0F	Turn on LCD display
0x80	Takes cursor to beginning of first row
0xC0	Takes cursor to beginning of Second row
0x28	Select 4-Bit Mode of LCD
0x38	Select 8-Bit Mode of LCD

Note: HD47780 controller which is connected with a 16×2 display works at very low speed than our TM4C123 microcontroller. Therefore, in order to synchronize the HD47780 controller with our microcontroller, we should add some delay between sending consecutive commands to LCD. We will see in the programming part how to do it.

LCD Interfacing with TM4C123 Tiva Launchpad in 4-bit Mode

As we mentioned earlier, we will use a 4-bit mode to interface 16×2 LCD with TM4C123 microcontroller. Now make the connection with 16×2 LCD with TM4C123 Tiva C Launchpad according to this schematic diagram.



In this schematic, we use GPIO PORTB of TM4C123GH6PM microcontroller to send commands and data to the LCD. PB0 to PB2 pins are used to provide control signals such as RS, E, and R/W. Similarly, PB4-PB7 pins are used to send data. Also, connect a potentiometer with a contrast adjust pin. Connect +5 volts and ground to other terminals of the potentiometer.

16×2 LCD Programming with TM4C123 Microcontroller

Now let's see how to send commands and data to the 16×2 LCD from TM4C123 Tiva C Launchpad. Firstly, we will write functions to send commands, data and initializations of the display controller.

Write 4-bit Nibble to LCD

As we mentioned earlier, we will be using 16×2 LCD in 4-bit mode. Therefore, first we need to create a function to send data or commands to LCD in 4-bit mode. This LCD_Write_Nibble() routine is used to send a nibble. The same function works for both data and commands. The second argument (unsigned char control), which is a RS signal for LCD, differentiates between data or command. Because when we want to send data we set RS to active high and when we want to send commands we clear the RS signal.

Inside this function, for data to be transmitted to D0-D3 pins, first we extract the upper bits from the data input argument. Because LCD data pins are connected to the PB4-PB7 pins of TM4C123 microcontroller. After that, we extract the lower bits from the data input argument. Because LCD control pins are connected to the PB0-PB2 pins of TM4C123 microcontroller.

In order to perform a write operation, first, we make the RS and R/W pins active low. After that, we give an active high to low transition pulse to enable pin to perform a write operation on the LCD. Finally, send nibble to LCD.

```
void LCD4bits_Data(unsigned char data)
{
        LCD_Write4bits(data & 0xF0 , RS);   //upper nibble first
        LCD_Write4bits(data << 4 , RS);     //then lower nibble
        delayUs(40);                        //delay for LCD (MCU is faster than LCD)
}
```

LCD Commands Send Function

This function is used to send commands to LCD. All commands are of 8-bits. Hence, first this routine sends the upper nibble of command to LCD and after that it sends the lower nibble. As we mentioned, the HD77480 controller operates at much lower speed than the TM4C123 microcontroller. Therefore, we need to add delay between sending commands. Command 1 and 2 takes 1.64ms and all other commands take 40us. Therefore, we add a delay of 2ms if commands are less than 4 and 40us for all other commands.

```
void LCD4bits_Cmd(unsigned char command)
{
        LCD_Write4bits(command & 0xF0 , 0);    //upper nibble first
        LCD_Write4bits(command << 4 , 0);   //then lower nibble

        if(command < 4)
                delayMs(2);      //commands 1 and 2 need up to 1.64ms
        else
                delayUs(40);     //all others 40 us
}
```

LCD Initialization Function

To initialize LCD, first we need to configure PORTB GPIO pins as digital output pins.

LCD4bits_init() routine enables the clock to TM4C123 GPIOB which is connected to control and data pins of 16×2 LCD. After we send various initialization commands to LCD from control pins of TM4C123 microcontroller such as:

- Set the character font size to 5×7. Because fonts on 16×2 LCD can be used in different sizes such as 5×7, 5×10 ang 5xn. Here 5 represents the number of rows and n represents the number of columns.
- Select 4-bit Mode to transfer data or commands in two nibbles
- Setting to move cursor position right after displaying each character
- Clear the screen
- Enable light and Cursor blinking

```
/* LCD and GPIOB initialization Function */
void LCD4bits_Init(void)
{
        SYSCTL->RCGCGPIO |= 0x02;     //enable clock for PORTB
        delayMs(10);                   //delay 10 ms for enable the clock of PORTB
         LCD->DIR = 0xFF;              //let PORTB as output pins
        LCD->DEN = 0xFF;               //enable PORTB digital IO pins
        LCD4bits_Cmd(0x28);           //2 lines and 5x7 character (4-bit data, D4 to D7)
        LCD4bits_Cmd(0x06);           //Automatic Increment cursor (shift cursor to right)
        LCD4bits_Cmd(0x01);           //Clear display screen
        LCD4bits_Cmd(0x0F);           //Display on, cursor blinking
}
```

LCD Write Character Function

LCD_Write4bits () function prints a character on the current cursor position of the display. A character type of data consists of 8-bits. But we are using a 4-bit mode of LCD. Hence, data transfers in two nibbles or 2 pieces of 4-bits by using LCD4bits_Data () function. The one important point to note here is that we are sending control signal RS which will be defined as an active high micro in the code. Because we set RS signal to active high when we want to transmit data from TM4C123 Tiva Launchpad to LCD.

```
void LCD_Write4bits(unsigned char data, unsigned char control)
{
        data &= 0xF0;                          //clear lower nibble for control
        control &= 0x0F;                       //clear upper nibble for data
        LCD->DATA = data | control;            //Include RS value (command or data ) with data
        LCD->DATA = data | control | EN;       //pulse EN
        delayUs(0);                            //delay for pulsing EN
        LCD->DATA = data | control;            //Turn off the pulse EN
        LCD->DATA = 0;                         //Clear the Data }
```

LCD String Write Function

LCD_WriteString () routine writes string to LCD starting from current cursor position. It uses LCD4bits_Data () function to send all characters of a string one by one. LCD_WriteString () function keeps writing characters till the NULL character is found.

```
void LCD_WriteString(char * str)
{
        volatile int i = 0;        //volatile is important

        while(*(str+i) != '\0')      //until the end of the string
        {
                LCD4bits_Data(*(str+i));    //Write each character of string
                i++;                        //increment for next character
        }
```

```
}
```

Complete Code

This sample code of TM4C123 Tiva C Launchpad prints characters and strings on 16×2 LCD.

This code displays “ENCS4110” on the first row and “Lab” on the second row of the LCD.

```
#include "TM4C123.h"    // Device header

#define LCD GPIOB               //LCD port with Tiva C
#define RS 0x01                 //RS -> PB0 (0x01)
#define RW 0x02                 //RW -> PB1 (0x02)
#define EN 0x04                 //EN -> PB2 (0x04)

//Functions Declaration
void delayUs(int);   //Delay in Micro Seconds
void delayMs(int);   //Delay in Milli Seconds
void LCD4bits_Init(void); //Initialization of LCD Dispaly
void LCD_Write4bits(unsigned char, unsigned char); //Write data as (4 bits) on LCD
void LCD_WriteString(char*);   //Write a string on LCD
void LCD4bits_Cmd(unsigned char);    //Write command
void LCD4bits_Data(unsigned char);   //Write a character

int main(void)
{
        char* str1 = "ENCS4110…";    //Write any string you want to display on the first row of LCD
        char* str2 = "Lab …";        //Write any string you want to display on the second row of LCD

        LCD4bits_Init();             //Initialization of LCD
        delayMs(500);                //delay 500ms for LCD initialization

        LCD4bits_Cmd(0x01);          //Clear the display
        delayMs(500);
        LCD4bits_Cmd(0x80);          //Force the cursor to beginning of 1st line
        delayMs(500);                //delay 500ms for LCD (MCU is faster than LCD)
        LCD_WriteString(str1);       //Write the string on LCD
        delayMs(500);                //Delay 500 ms to let the LCD diplays the data

        LCD4bits_Cmd(0xC0);          //Force the cursor to beginning of 2nd line
        delayMs(500);                //delay 500ms for LCD (MCU is faster than LCD)
        LCD_WriteString(str2);       //Write the string on LCD
        delayMs(500);                //Delay 500 ms to let the LCD diplays the data

}
```

```
void LCD4bits_Init(void)
{
	SYSCTL->RCGCGPIO |= 0x02;    //enable clock for PORTB
	delayMs(10);               //delay 10 ms for enable the clock of PORTB
	LCD->DIR = 0xFF;           //let PORTB as output pins
	LCD->DEN = 0xFF;           //enable PORTB digital IO pins
	LCD4bits_Cmd(0x28);        //2 lines and 5x7 character (4-bit data, D4 to D7)
	delayMs(100);
	LCD4bits_Cmd(0x06);        //Automatic Increment cursor (shift cursor to right)
	delayMs(100);
	LCD4bits_Cmd(0x01);        //Clear display screen
	delayMs(100);
	LCD4bits_Cmd(0x0F);        //Display on, cursor blinking
}


void LCD_Write4bits(unsigned char data, unsigned char control)
{
	data &= 0xF0;                  //clear lower nibble for control
	control &= 0x0F;                //clear upper nibble for data
	LCD->DATA = data | control;       //Include RS value (command or data ) with data
	LCD->DATA = data | control | EN;    //pulse EN
	delayUs(10);  //delay for pulsing EN
	LCD->DATA = data | control;  //Turn off the pulse EN
	LCD->DATA = 0;       //Clear the Data
}

void LCD_WriteString(char * str)
{
	volatile int i = 0;        //volatile is important

	while(*(str+i) != '\0')      //until the end of the string
	{
		LCD4bits_Data(*(str+i));    //Write each character of string
		i++;                  //increment for next character
	}
}

void LCD4bits_Cmd(unsigned char command)
{
	LCD_Write4bits(command & 0xF0 , 0);    //upper nibble first
	LCD_Write4bits(command << 4 , 0);       //then lower nibble

	if(command < 4)
		delayMs(2);      //commands 1 and 2 need up to 1.64ms
	else
		delayUs(40);     //all others 40 us
```

```
}

void LCD4bits_Data(unsigned char data)
{
        LCD_Write4bits(data & 0xF0 , RS);   //upper nibble first
        LCD_Write4bits(data << 4 , RS);     //then lower nibble
        delayUs(40);    //delay for LCD (MCU is faster than LCD)
}


void delayMs(int n)
{
        volatile int i,j;           //volatile is important for variables incremented in code
        for(i=0;i<n;i++)
                for(j=0;j<3180;j++)        //delay for 1 msec
                {}
}

void delayUs(int n)
{
        volatile int i,j;  //volatile is important for variables incremented in code
        for(i=0;i<n;i++)
                for(j=0;j<3;j++)          //delay for 1 micro second
                {}
}
```

In-Lab Tasks:

1) Modify the code above to display your name on the first line and your ID on the second line.

2) Write a program that displays your name on LCD with movement. Your program should allow the user to control the direction of the movement (shift left, shift right) using the two on-board push buttons.

3) Write a program that displays your name at the 1st raw and your ID on the 2nd row on the LCD. The upper word should be firstly appears from the left of the LCD then it is shifted continually to the other side. The lower word must have the opposite movement at the time .All that happen after a button press from the user.

```
void LCD_set_cursor_pos(int row, int col)
{
   unsigned char cp;
   if (row == 0)
      cp = 0x80;
```

```
    else
        cp = 0xc0;
    cp |= col;
    LCD4bits_Cmd(cp);
}
```