2.5 INTERFACING 16x2 LCD WITH LPC1768 MICROCONTROLLER Learning

Outcomes:

After studying this interfacing project, students will be able to:

Understand the basics of 16x2 LCD

Interface a 16x2 LCD with LPC1768.

Write code for different applications.

2.5.1 THEORY

An LCD screen is an electronic display module that uses liquid crystal to produce a visible image. The 16×2 translates a display of 16 characters per line in 2 such lines. In this LCD, each character is displayed in a 5x7 pixel matrix.

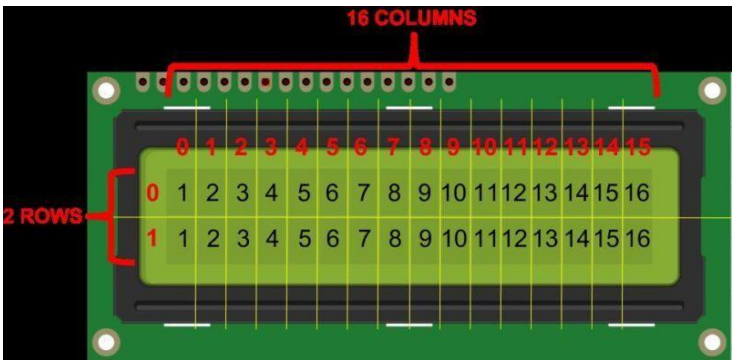

Fig.: 16x2 LCD

16X2 LCD Display Pinout Diagram

Category	Pin No.	Symbol	Function	Connection to MCU
Power Pins	1	VSS	Ground (0V)	GND
	2	VCC	Power Supply (+5V)	+5V
	3	VEE	Contrast Adjustment	Potentiometer (10KΩ) Output
Control Pins	4	RS	Register Select RS = 0, Command RS = 1, Data	GPIO Pin
	5	R/W	Read/Write (0 = Write, 1 = Read)	GPIO Pin (Usually GND for Write)
	6	E	Enable (Latching Data to LCD)	GPIO Pin
Data Pins	7	D0	Data Bit 0 (Used in 8-bit mode)	GPIO Pin (Optional in 4-bit mode)
	8	D1	Data Bit 1 (Used in 8-bit mode)	GPIO Pin (Optional in 4-bit mode)
	9	D2	Data Bit 2 (Used in 8-bit mode)	GPIO Pin (Optional in 4-bit mode)
	10	D3	Data Bit 3 (Used in 8-bit mode)	GPIO Pin (Optional in 4-bit mode)
	11	D4	Data Bit 4	GPIO Pin
	12	D5	Data Bit 5	GPIO Pin
	13	D6	Data Bit 6	GPIO Pin
	14	D7	Data Bit 7 (Busy Flag)	GPIO Pin
	15	A (LED+)	LED Backlight Anode (+)	+5V

Backlight Pins	16	K (LED-)	LED Backlight Cathode (-)	GND

2.5.2 INTERFACING DIAGRAM

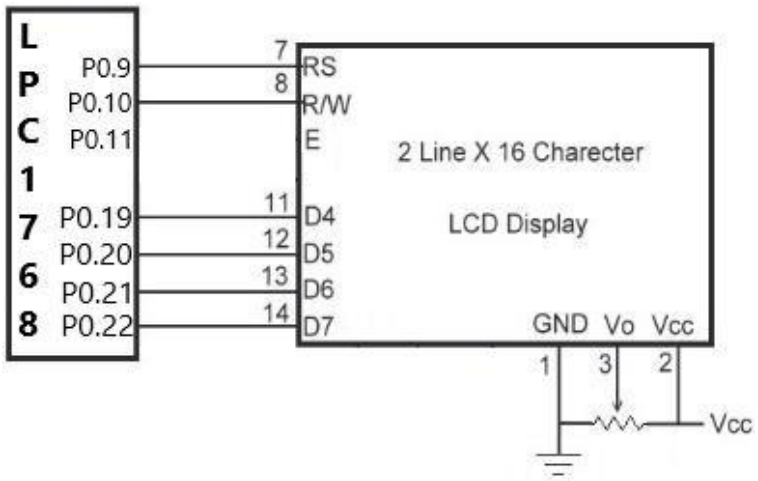

Fig.: Interfacing 16x2 LCD with LPC1768 Microcontroller

The 16x2 LCD (16 characters per row, 2 rows) is interfaced with the LPC1768 microcontroller in **4-bit mode**, which means only **four data lines (D4–D7)** are used instead of all eight (D0–D7). This reduces the required GPIO pins while still allowing full functionality.

Data to the LCD should be sent in ASCII Form

When interfacing a 16x2 LCD with a microcontroller, data must be sent in ASCII form for proper character display. The LCD controller has a built-in Character Generator ROM (CGROM) that maps ASCII values to corresponding characters. When we send an ASCII code, the controller looks up the corresponding character and displays it. **Busy Flag (BF) - D7**

- D7 acts as the Busy Flag (BF) when RS = 0 (Command mode) and R/W = 1 (Read mode). ○ The LCD sets D7 = 1 when it is busy processing a command or data.
  - The LCD sets D7 = 0 when it is ready to receive the next instruction.
- Why Check the Busy Flag?
  - Ensures that the next command is sent only when the LCD is ready.
  - Prevents data corruption and improves reliability.
  - Instead of using fixed delays, checking D7 (Busy Flag) makes code more efficient.

LCD Commands

a) Function Set Commands

Command (Hex)	Function
0x20	Select 4-bit Data Mode
0x28	4-bit Mode, 2-Line, 5x7 Font
0x30	Select 8-bit Data Mode
0x32	Initialize 4-bit Mode
0x33	Initialize 8-bit Mode
0x38	8-bit Mode, 2-Line, 5x7 Font

b) Display Control LCD Commands

Command (Hex)	Function
0x08	Display OFF, Cursor OFF
0x0C	Display ON, Cursor OFF
0x0E	Display ON, Cursor ON
0x0F	Display ON, Cursor Blinking

c) Cursor and Display Shift Commands

Command (Hex)	Function
0x10	Move Cursor Left
0x14	Move Cursor Right
0x18	Shift Entire Display Left
0x1C	Shift Entire Display Right

d) Clearing & Reset Commands

Command (Hex)	Function
0x01	Clear Display Screen
0x02	Return Home (Reset Cursor to 0,0)

e) Entry Mode Commands

Command (Hex)	Function
0x04	Decrement Cursor (Shift Left)
0x06	Increment Cursor (Shift Right)

f) Row Selection Commands

Command (Hex)	Function
0x80	Force Cursor to Beginning (1st Line)
0xC0	Force Cursor to Beginning (2nd Line)

2.5.3 EXAMPLE CODES

Write a C program to display a string on 16x2 LCD

```
#include <LPC17xx.h>

#define RS      9     //Register Select Pin
#define RW      10    //Read Write Pin
#define EN      11    //Enable Pin

void lcdchar(unsigned char data, unsigned char type);
void lcdinit(void); void lcdstring(char *str); void
delay(unsigned int count);

int main(void)
{
      lcdinit();
      lcdstring("VVCE Mysuru");
      while(1);
}

void lcdinit(void)
{
      //LCD Data Pins as Output Pins
      LPC_GPIO0->FIODIR |=(1<<22)|(1<<21)|(1<<20)|(1<<19);

      //Enable, Read/Write, Register Select Pins as Output Pins
      LPC_GPIO0->FIODIR |=(1<<11)|(1<<10)|(1<<9);

      //Function set to 8-bit mode.
      lcdchar(0x33,'C');
      delay(100);

      //Function set to 8-bit mode again.
      lcdchar(0x32,'C');
      delay(100);

      //Select 2-lines and 5x7 matrix D4-D7, 4 bits
lcdchar(0x28,'C');
      delay(100);

      //Display ON, Cursor OFF
lcdchar(0x0C,'C');
delay(100);
```

```
      //Increment cursor (Shift cursor to right)
lcdchar(0x06,'C');
      delay(100);

      //Clear display screen
lcdchar(0x01,'C');    delay(100);
}

void lcdstring(char *str)
{     unsigned char i = 0;

      while(str[i] != '\0')       //Send until NULL character found
      {
            lcdchar(str[i],'D');
            i++;
      }
}  void lcdchar(unsigned char data,unsigned char
type)
{
      //extract the lower nibble
      unsigned char lowerNibble = data & 0x0F;

      //extract the higher nibble      unsigned
char upperNibble =(data & 0xF0)>>4;

      //C - Command Select, D - Data Select
      if(type=='C')
            LPC_GPIO0->FIOCLR = (1<<RS);
else if(type=='D')                LPC_GPIO0-
>FIOSET = (1<<RS);

      // Write higher nibble value
      LPC_GPIO0->FIOSET|=(1<<EN);           // Make enable high
      LPC_GPIO0->FIOCLR|=(0x0F << 19);      // Clear all the bits
      LPC_GPIO0->FIOSET|=(upperNibble<<19);
      LPC_GPIO0->FIOCLR|=(1<<EN);           //Make enable low
delay(100);

      // Write lower nibble value
      LPC_GPIO0->FIOSET|=(1<<EN);           // Make enable high
      LPC_GPIO0->FIOCLR|=(0x0F << 19);      // Clear all the bits
      LPC_GPIO0->FIOSET|=(lowerNibble<<19);
      LPC_GPIO0->FIOCLR|=(1<<EN);           // Make enable low
delay(100);
}
```

```
void delay(unsigned int count)
{
      int i,j;
for(i=0;i<count;i++)
for(j=0; j<1275; j++);
}
```

Further exploration:

Modify the code to remove fixed delay function and implement using busy flag.

Try to display at different cursor positions and rows. Scroll the display left or right.

Can you display more than 16 characters in a row?