

Interfacing LCD with Arduino using AVR-GCC

Arpita Mishra and G V V Sharma*

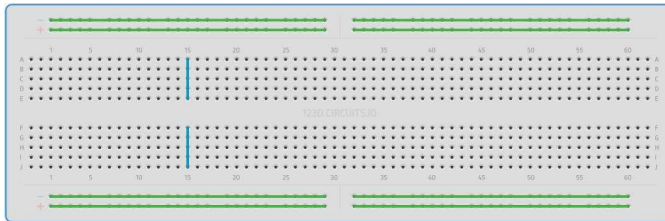


Fig. 1: Breadboard

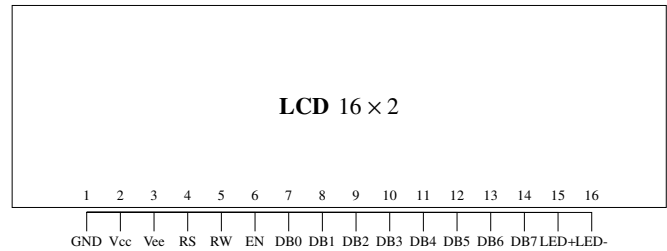


Fig. 2: LCD

Abstract—This manual shows how to interface an Arduino to a 16×2 LCD display using AVR-GCC. This framework provides a useful platform for displaying the output of AVR-Assembly programs.

1 COMPONENTS

Component	Value	Quantity
Breadboard		1
Arduino	Uno	1
LCD	16×2	1
Jumper Wires		20

TABLE I

2 DISPLAY NUMBER ON LCD

Problem 1. Plug the LCD in Fig. 2 to the breadboard.

Problem 2. Connect the Arduino pins to LCD pins as per Table II.

Problem 3. Display the number 5 on the LCD

Solution:

Arpita Mishra is an intern with the TLC, IIT Hyderabad. *The author is with the Department of Electrical Engineering, Indian Institute of Technology, Hyderabad 502285 India e-mail: gadepall@iith.ac.in. All content in the manual released under GNU GPL. Free to use for anything.

TABLE II: Arduino to LCD Pin Connection.

Arduino Pins	LCD Pins	LCD Pin Label	LCD Pin Description
GND	1	GND	
5V	2	Vcc	
GND	3	Vee	Contrast
D8	4	RS	Register Select
GND	5	R/W	Read/Write
D9	6	EN	Enable
D10	11	DB4	Serial Connection
D11	12	DB5	Serial Connection
D12	13	DB6	Serial Connection
D13	14	DB7	Serial Connection
5V	15	LED+	Backlight
GND	16	LED-	Backlight

```
#include <avr/io.h>
#include <util/delay.h>
#include <stdlib.h>
```

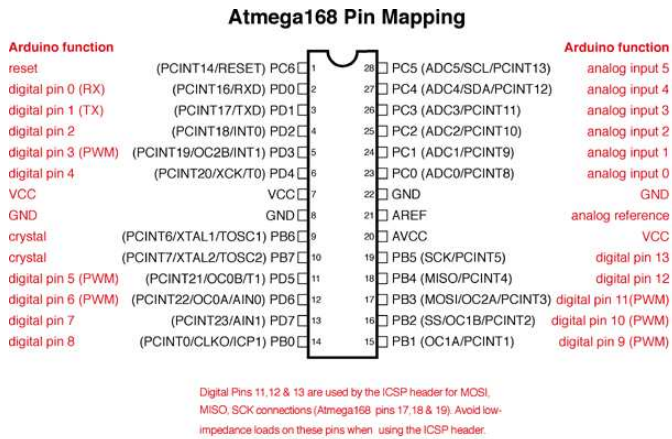


Fig. 3: Arduino-ATMEGA328P pin map.

```
// TYPEDEFS
typedef uint8_t byte; // changed
                        the name

// -----
//LCD DRIVER ROUTINES
//
// Routines:
// LCD_Init initializes the LCD
// controller
// LCD_Cmd sends LCD controller
// command
// LCD_Char sends single ascii
// character to display
// LCD_Clear clears the LCD
// display & homes cursor
// LCD_Integer displays an integer
// value
// LCD_Message displays a string
// PortB is used for data
// communications with the HD44780-
// controlled LCD.
// The following defines specify
// which port pins connect to the
// controller:
#define ClearBit(x,y) x &= ~_BV(y)
// equivalent to cbi(x,y)
#define SetBit(x,y) x |= _BV(y) //
// equivalent to sbi(x,y)
#define LCD_RS 0 // pin for LCD R/
// S (eg PB0)
#define LCD_E 1 // pin for LCD
// enable
```

```
#define DAT4 2 // pin for d4
#define DAT5 3 // pin for d5
#define DAT6 4 // pin for d6
#define DAT7 5 // pin for d7
// The following defines are
// controller commands
#define CLEARDISPLAY 0x01

void PulseEnableLine ()
{
    SetBit(PORTB,LCD_E); // take LCD
// enable line high
    _delay_us(40); // wait 40
// microseconds
    ClearBit(PORTB,LCD_E); // take
// LCD enable line low
}

void SendNibble(byte data)
{
    PORTB &= 0xC3; // 1100.0011 =
// clear 4 data lines
    if (data & _BV(4)) SetBit(PORTB,
// DAT4);
    if (data & _BV(5)) SetBit(PORTB,
// DAT5);
    if (data & _BV(6)) SetBit(PORTB,
// DAT6);
    if (data & _BV(7)) SetBit(PORTB,
// DAT7);
    PulseEnableLine(); // clock 4
// bits into controller
}

void SendByte (byte data)
{
    SendNibble(data); // send upper 4
// bits
    SendNibble(data<<4); // send
// lower 4 bits
    ClearBit(PORTB,5); // turn off
// boarduino LED
}

void LCD_Cmd (byte cmd)
{
    ClearBit(PORTB,LCD_RS); // R/S
// line 0 = command data
    SendByte(cmd); // send it
}

void LCD_Char (byte ch)
{
    SetBit(PORTB,LCD_RS); // R/S line
```

```

    l = character data
    SendByte(ch); // send it
}
void LCD_Init()
{
    LCD_Cmd(0x33); // initialize
                    controller
    LCD_Cmd(0x32); // set to 4-bit
                    input mode
    LCD_Cmd(0x28); // 2 line , 5x7
                    matrix
    LCD_Cmd(0x0C); // turn cursor off
                    (0x0E to enable)
    LCD_Cmd(0x06); // cursor
                    direction = right
    LCD_Cmd(0x01); // start with
                    clear display
    _delay_ms(3); // wait for LCD to
                    initialize
}
void LCD_Clear() // clear the LCD
                    display
{
    LCD_Cmd(CLEARDISPLAY);
    _delay_ms(3); // wait for LCD to
                    process command
}

void LCD_Message(const char *text)
    // display string on LCD
{
    //while (*text) // do until /0
                    character
    LCD_Char(*text++); // send char &
                    update char pointer
}

void LCD_Integer(int data)
    // displays the integer value of
    DATA at current LCD cursor
    position
{
    char st[8] = ""; // save enough
                    space for result
    itoa(data, st, 10); //
    LCD_Message(st); // display in on
                    LCD
}

// MAIN PROGRAM

```

```

int main(void)
{
    // use PortB for LCD interface
    DDRB = 0xFF; // 1111.1111; set
                    PB0-PB7 as outputs

    LCD_Init(); // initialize LCD
                    controller

    while(1)
    {
        LCD_Clear();
        LCD_Integer(5); // show counter
        _delay_ms(600); // set
                        animation speed
    }
}

```

Problem 4. Modify the above code so that the numbers from 0 to 9 are displayed repeatedly.

Solution: Replace LCD_Integer(5) by the following code.

```

for (byte count=0; count<10; count
    ++)
{
    LCD_Integer(count); // show
                        counter
    _delay_ms(600); // set
                        animation speed
}

```