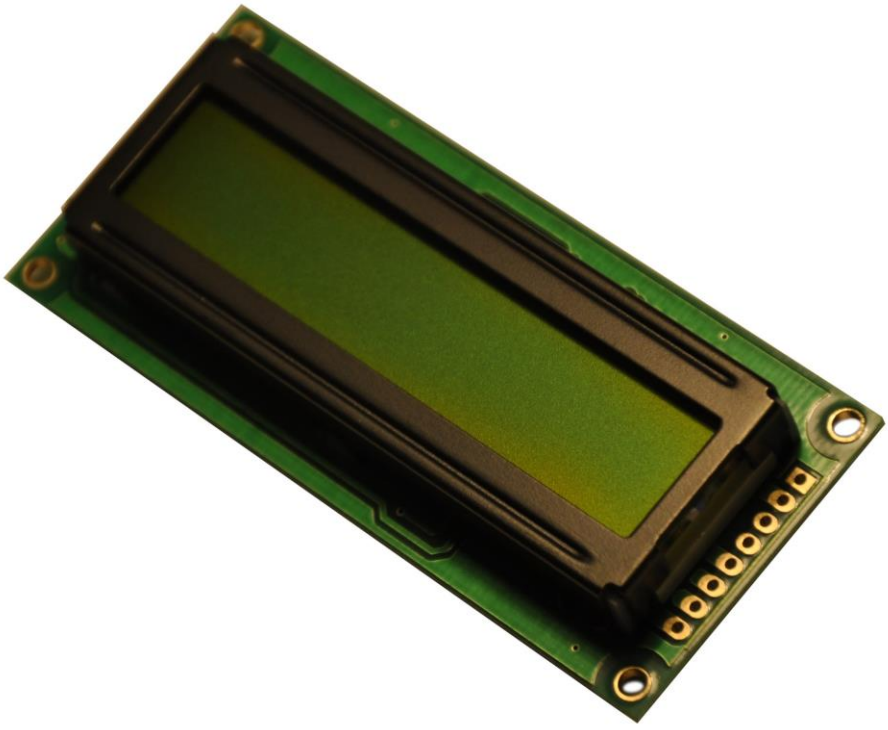
Device: LCD-1602

This document Version: 1

Matches module hardware Version: 1

Date: 2 June 2013

Description: 16 character x 2 line LCD display with Serial interface



Contents

Introduction

The LCD-1602 is a low power 16 character by 2 line liquid crystal display, with serial interface.

Features

With a very straightforward synchronous (ie, clocked) serial input, this display is easy to get up and running. With optional backlight, the display can be viewed in darkness and has a fully adjustable contrast.

It's perfect for making displaying more information about what is happening with your project. You need only dedicate 3 pins to getting a tonne of information displayed!

Construction

It's all pre-built! Just add female or male header pins, or solder directly to the board, and away you go.

Connections

The LCD-1602 has one connection port.

BLK	Backlight LED cathode (ground)
BLA	Backlight LED anode (positive supply)
E	Latch
CLK	Clock
DI	Data In
VO	Contrast adjustment (0V – VDD)
VDD	Positive supply (3.3V or 5V depending on model)
VSS	Ground connection

Technical details

Power

The LCD-1602 can be powered with 3.3V (LCD-1602-3.3V) or 5V (LCD-1602-5V). The display itself consumes around 2mA. The backlight can consume up to 80mA and can be controlled using PWM to adjust the brightness.

Contrast

The contrast setting requires a voltage between 0V and VDD, which can be achieved by using a 10K potentiometer. Simply connect one end of the potentiometer to VSS and the wiper to VO. Then as you turn the potentiometer, you will supply different voltages to the V0 pin, which will adjust the contrast. Usually you will only need to set this once.

Alternatively, you can supply a voltage using your microcontroller's digital to analogue converter, or use PWM and a filter to achieve the same result (both of which will give you programmable contrast adjustment).

Communicating with the display

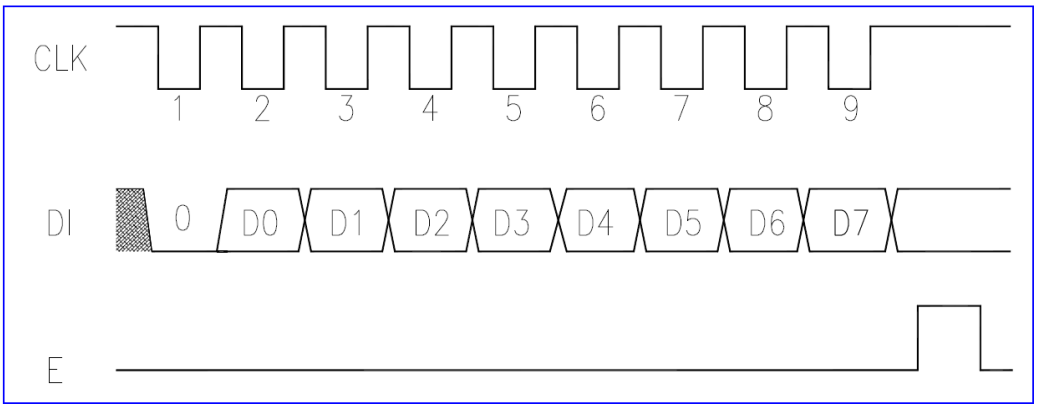
This display uses a synchronous serial interface to push data into it.

The clock is active low, meaning that you need to keep the CLK line high, and pulse it low and back to high to clock the data into the display.

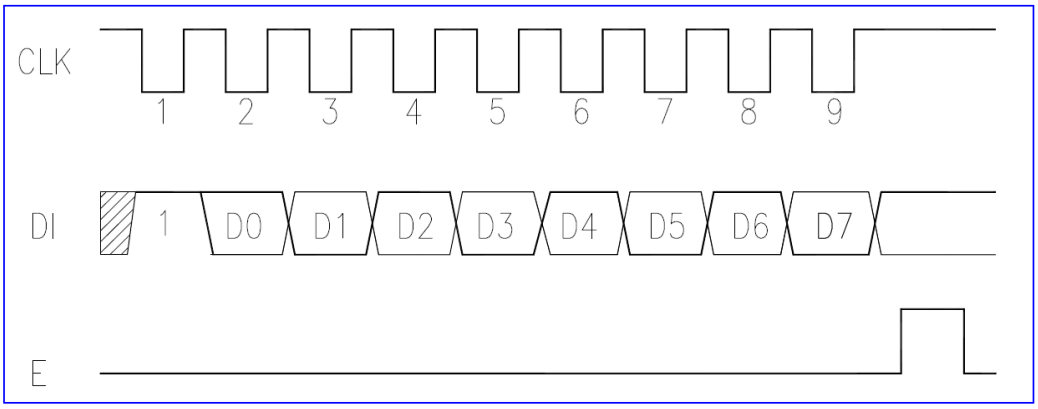
The latch is active high, meaning that you need to keep the E line low, and pulse it high and back to low to signal that you have completed sending a command instruction or data byte.

Each time you send data, 9 bits need to be sent. Firstly, clock in a 0 if you are sending a command instruction (eg, clear the display), and 1 if you are sending data (eg, characters to be displayed), on the DI line. Then clock in data bits 0 through 7 of your byte (that is, LSB first).

Once you have sent the 9 bits, pulse the latch (E) pin high and back to low. Here is an example of sending an instruction byte:



And here is an example of sending a data byte:



Initialising the display

Send the following commands to initialise the display:

```
// Set 16X2 display, 5X7 dots, 8 bit interface – do this four times
ea_lcd1602_write(0, 0b00111000);
delay_ms(3);
ea_lcd1602_write(0, 0b00111000);
delay_ms(3);
ea_lcd1602_write(0, 0b00111000);
delay_ms(3);
ea_lcd1602_write(0, 0b00111000);
delay_ms(3);

ea_lcd1602_write(0, 0b00001000);    // display off
ea_lcd1602_write(0, 0b00000001);    // reset
delay_ms(3);

ea_lcd1602_write(0, 0b00000110);    // increment cursor setting

ea_lcd1602_write(0, 0b00001100);    // display on, cursor off
```

Of course, this sample code assumes you have a routine called `ea_lcd1602_write` to send the data or instruction bit, and the following byte to the display. Naturally the `delay_ms` function will need to be changed for your microcontroller.

Display commands

Hidden behind the serial interface to this display is in fact the fabulous HD44780 LCD controller, which is in practically every LCD known to humankind. There are bucket loads of website covering the commands of this controller IC, but a few are described here for completeness' sake.

Command	7	6	5	4	3	2	1	0	Function
Clear	0	0	0	0	0	0	0	1	Clear Display
Return Home	0	0	0	0	0	0	1	0	Move cursor to top left corner
Entry mode	0	0	0	0	0	1	I	S	I=1 Increment cursor on write I=0 Decrement cursor on write S=1 Shift display on write S=0 No shift of display on write
Display control	0	0	0	0	1	D	C	B	D=1 Display on D=0 Display off C=1 Cursor On C=0 Cursor Off B=1 Cursor blinks B=0 Cursor static
Cursor/display shift	0	0	0	1	S	R	-	-	S=1 Shift display S=0 Shift cursor R=1 Shift right R=0 Shift left
Function set	0	0	1	D	N	F	-	-	D=1 8 bit interface D=0 4 bit interface N=1 Two display lines N=0 One display line F=1 Font set 1 (5x10) F=0 Font set 0 (5x8)
Set CGRAM addr	0	1	A	A	A	A	A	A	Set address to write to CGRAM
Set DRAM addr	1	A	A	A	A	A	A	A	Set address to write to DRAM (cursor position)

Displaying text

The key to displaying text is to send a "Set DRAM addr" command instruction (to set the DRAM location to where you want the text to appear) and then simply clock out ASCII data of the text.

Note that the first line occupies DRAM positions 0x00 to 0x0f and the second line occupies DRAM positions 0x40 to 0x4f.

So to set the cursor to the first character on the second line, send the command instruction 0b11000000 (remember that bit 7 is set to indicate a "Set DRAM addr" command).

Porting your existing LCD library

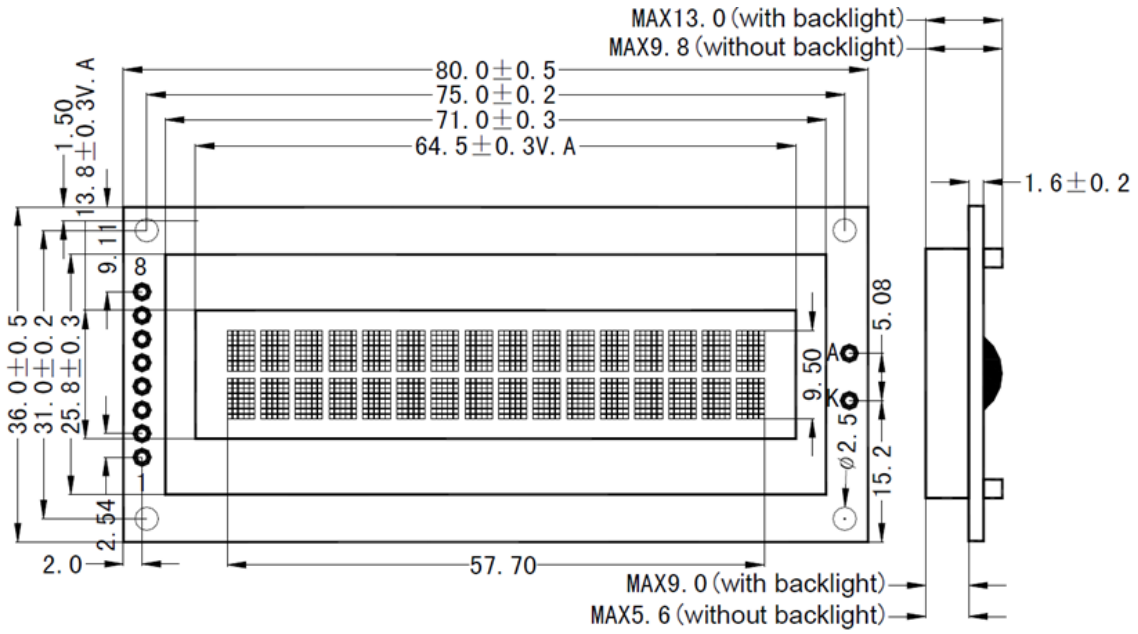
Your microcontroller probably has a library for this chip already and can be easily convinced to talk to this display.

You will probably find a routine in your LCD library that sets the RW, RS and data pins before clocking the data out – if so, you can simply replace that function with this one:

```
void ea_lcd1602_write(bit mode_data, uns8 data) {

    // data = 1, instruction = 0

    change_pin(lcd_di_port, lcd_di_pin, mode_data);

    // pulse clock to send data/instruction pin

    clear_pin(lcd_clk_port, lcd_clk_pin);
    delay_us(12);
    set_pin(lcd_clk_port, lcd_clk_pin);

      // send data or instruction byte

    for( uns8 count = 0 ; count < 8 ; count++ ) {
        change_pin(lcd_di_port, lcd_di_pin, data.0);
        data = data >> 1;
        clear_pin(lcd_clk_port, lcd_clk_pin);
        delay_us(12);
        set_pin(lcd_clk_port, lcd_clk_pin);
    }

      // pulse e (latch)

    set_pin(lcd_e_port, lcd_e_pin);
    delay_us(2);
    clear_pin(lcd_e_port, lcd_e_pin);

}
```

Note the timing of the latch and clock pins – allow 12 microseconds for the clock and 2 microseconds for the latch.

Dimensions



All dimensions in mm.

Versions

Doc Version	HW Version	Date	Comments
1	1	2 June 2013	Initial Version for board v1