

Application Note

Interfacing a Microcontroller with a Character LCD using a Hitachi HD44780 Display Driver

Daniel Svoboda

April 5, 2013

Keywords: character LCD, display driver, embedded system, microcontroller

Abstract: Hitachi display drivers have become a standard choice for interfacing a microcontroller with a character Liquid Crystal Display (LCD). Their combination of a simple command protocol and low power demands have made them an obvious choice for hobbyists and professionals alike. This document will describe the necessary hardware and software setup in order to reliably send commands and strings to an LCD using one of these drivers.

Note: This document will refer to the datasheet for the Hitachi HD44780 driver, available at: <https://www.sparkfun.com/datasheets/LCD/HD44780.pdf>

Introduction

The Hitachi HD44780 is a versatile integrated circuit (IC) capable of driving a wide variety of Liquid Crystal Displays (LCDs). A Microcontroller (MCU) or other processing unit can control the HD44780 Driver using a standard set of commands sent over parallel data lines, as well as control signals on several dedicated-purpose lines. While these commands and signals are highly standardized and fairly simple, they can still be quite difficult to control correctly. This document aspires to address the majority of the configuration – both hardware and software – that must take place in order to reliably interface with a screen through the HD44780.

Hardware Setup

Generally the LCD and HD44780 come already attached to the same printed circuit board (PCB). The embedded system developer's first task is to correctly connect the MCU to this PCB. Additional circuitry is also necessary to provide power and contrast adjustment to the LCD.

Terminal Placement

Most HD44780 driven character LCDs feature 14 pins, in one of two configurations. These pins appear as through-holes on the PCB. They are usually a row of 14 (fig. 1.b) at the bottom of the LCD or as 7 rows of 2 on the side (fig. 1.a). For the purposes of this Application Note, the 7 x 2 arrangement will be used. The pin functionality on 14 x 1 pin PCBs is identical, only the arrangement differs.

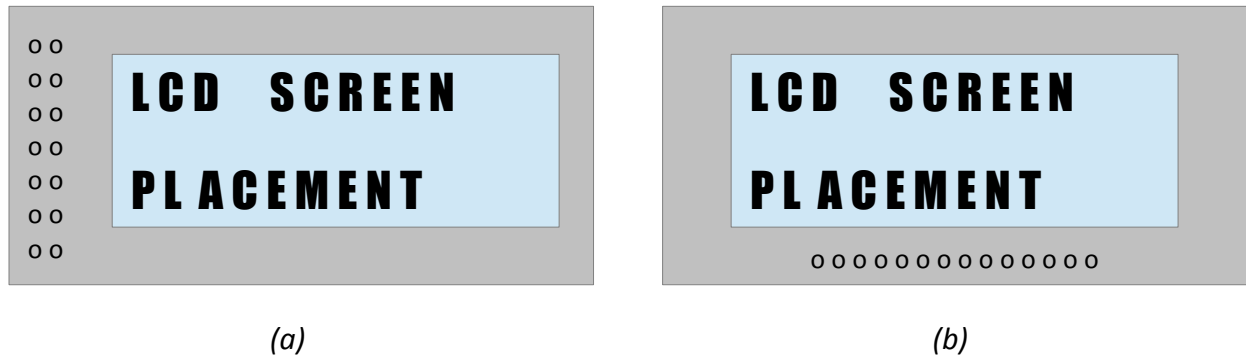


Figure 1: Arrangement of pins on PCB

Pin Functions

The following is a description of the function of each pin.

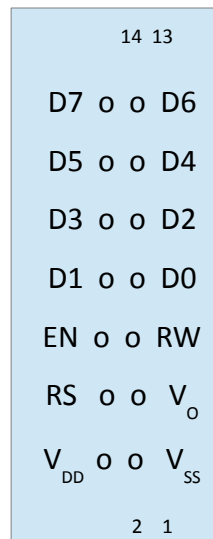


Figure 2: Placement and names of pins

- V_{DD} and V_{SS} : These provide power and ground to the display driver.
- V_O : This voltage controls the contrast of the screen. A potentiometer is often used to vary this value, as shown in figure 3.
- Register Select (RS): The logic value on this pin tells the driver whether the data being sent is an instruction (Logical LOW) or a piece of data (Logical HIGH). Instructions control things like cursor position and scrolling, whereas data contains actual characters to be

displayed on the screen.

- Read/Write (RW): This pin enables the MCU to either write data to the LCD (Logical LOW) or read data from it (Logical HIGH). Since, in this application, there is never a need to read data back from the display, this pin can be tied to ground (ensuring permanent write mode). This has the beneficial effect of freeing up an additional I/O pin on the MCU for other purposes.
- Enable (EN): This pin is toggled on and off to clock data from the HD44780's data pins onto the chip.
- Data Pins 0 – 7 (D0 – D7): Eight pins are available for data transfer to and from the HD44780. These are digital I/O pins that accept a logical HIGH or LOW value. For this application, only 4 of the 8 pins will be used. This reduces the number of pins needed on the MCU to control the LCD.

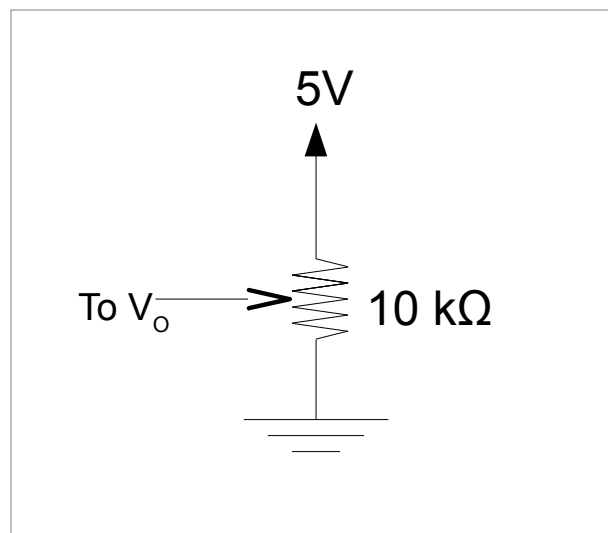


Figure 3: User-Settable Contrast Control

Example Setup with MSP430

Figure 4 (on next page) shows an example of the LCD and Driver assembly connected to a Texas Instruments MSP430 Microcontroller. Selection of digital pins on the MCU is arbitrary. Similar connections could clearly be made with any other MCU, provided it has at least 6 digital I/O pins.

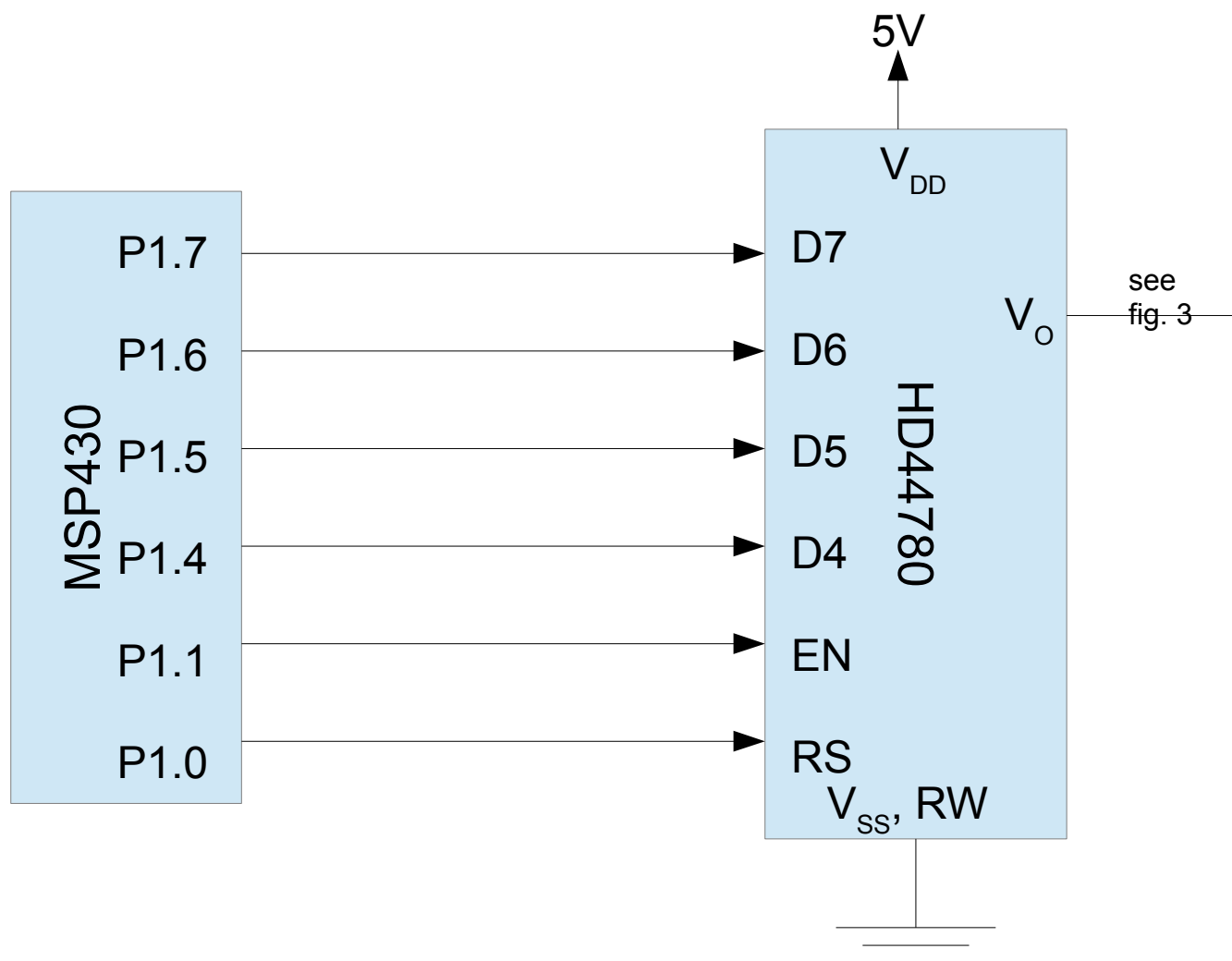


Figure 4: Example connections with MSP430

Software Setup

The remainder of the configuration is performed by code running on the MCU. This code is responsible for setting the HD44780 to 4 bit mode, initializing several important display parameters, sending data when the screen needs to be updated, and waiting for appropriate intervals between digital switching events, to ensure that transient behaviors are not inadvertently clocked in as (false) data.

Entering 4 Bit Mode

Please refer to page 24 of the [Hitachi HD44780 Datasheet](#), which details the instructions available to control the driver. In order to receive data over 4 input lines instead of 8, the HD44780 must be set to 4 bit mode. This is achieved using the “Function set” instruction. Here are the relevant bits:

RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	DL	N	F	—	—

Figure 5: Function Set Instruction

DL stands for data length, and needs to be set to 0 in order for the data mode to be set to 4 bits. N indicates the number of lines on the display, and F sets the font size. This instruction must be sent to the HD44780 before all other commands in order for the display to work.

Other Important Initialization

Several other instructions from the datasheet are highly useful when controlling the HD44780. “Return home” and “Set DDRAM address” can be used to set the address in DDRAM where data is stored (this is the data displayed on the screen).

“Entry mode” is used to set the direction that the cursor moves each time a character is written. It can also be used to tell the entire display to shift when a character is written. This allows the application to scroll text across the screen without having to rewrite the entire display for each new character.

“Set CGRAM address” is used to create and address custom characters. This allows the programmer to display characters beyond the 256 that are part of the built in font. While this Application Note will not discuss this process in detail, ample explanation is available on pages 19-21 of the HD44780 datasheet.

Sending Strings to be Displayed

The first step in sending a string to the LCD for display is to use the “Set DDRAM address” instruction to set the DDRAM address at which the data will be written. Since these addresses

are directly mapped onto the screen, this operation is analogous with setting the cursor's position.

The next step is to switch the Register Select pin from LOW to HIGH, in order to notify the driver that the next set of bits are data, instead of an instruction. Since an 8 bit character needs to be sent over 4 bit lines, it will take two toggles of the enable pin to send one character. The upper nibble of the character is first placed on the MCU's data lines and EN is toggled. The lower nibble is set second.

After a single character of the string is sent to the HD44780, either the cursor will shift left or right one DDRAM address, or it will remain in place and the existing data will be shifted left or right. One of these four behaviors was selected in the previous step by using the "Entry mode" instruction. Character writes can continue until the entire string is displayed on the LCD.

Building in Appropriate Delays

After sending an instruction or a piece of data, the MCU can monitor data bit 7, which acts as a busy flag. This flag switches from 1 to 0 once internal calculation has finished and the HD44780 is ready to receive more data/instructions. While this feature is useful when the screen needs to be updated as quickly as possible, it also requires an additional interrupt on the MCU to detect the falling edge.

When ultra-fast screen refresh is not required, it can be much easier to include an appropriate delay between instructions/data, instead of monitoring the busy flag for a falling edge. Pages 24 – 25 of the datasheet include worst-case execution times for each of the instructions. By programming the MCU to wait longer than these worst-case times when a command is sent, the developer can ensure that command overlap will not take place.

Conclusions

Utilizing the above techniques, an engineer of an embedded system should be able to successfully interface a microcontroller with a character LCD using the Hitachi HD44780 display driver.

References

HD44780 datasheet – Hitachi

<https://www.sparkfun.com/datasheets/LCD/HD44780.pdf>

Programming an LCD - James Nevius

<http://inst.eecs.berkeley.edu/~ee40/calbot/webpage/ProgrammingLCD.htm>

Arduino Liquid Crystal Library

<http://arduino.cc/en/Tutorial/LiquidCrystalDisplay>

Microtips NMTC-S0802XRGHS datasheet

<http://www.microtipsusa.com/pdfs/products/Character/NMTC-S0802XRGHS.pdf>