USB display Linux driver

A Linux kernel driver for the
HITACHI HD44780U LCD display
connected via USB bridge controller CH341A

Version 1.0 dated 03/14/2023

written by Ingo A. Kubbilun

DISCLAIMER

THE DOCUMENT IN-HAND AND ANY RELATED SOFTWARE IS PROVIDED BY **THE AUTHOR** "*AS IS*" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.

IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION).

Document revision

Version:	Date:	Author:	Comments / changes:
0.1	03/11/2023	Ingo A. Kubbilun	document creation
1.0	03/14/2023	Ingo A. Kubbilun	1st published version

Table of Contents

Document revision i
Table of Contents ii
List of Tables iv
List of Figures v
Web resources / bibliography vi
Acronyms & Abbreviations vii
1 Introduction 8
1.1 Starting point (kernel driver) 8
2 The display 9
2.1 Hardware description 9
2.1.1 Organization of the DDRAM 11
2.1.2 HD44780U commands 11
3 USB bridge controller CH341A 14
3.1 Linux kernel driver 14
3.1.1 IOCTLs 14
4 The userland library 15
4.1 The scroll and auto-scroll feature 15
4.2 The character generator RAM 15
5 The command line tool 17
A Appendix 18
A.1 Building the software 18
A.1.1 Required (Debian) packages 18
A.1.2 Building the driver, library, and tool 18
A.1.3 Installing the software 18

List of Tables

Table 1: prefix bytes (command codes) CH341A..... 11

Table 2: Command bytes (prefix bytes)..... 12

List of Figures

Figure 1: Default character dot matrices (from https://de.wikipedia.org/wiki/HD44780#/media/Datei:Charset.gif)..... 10

Figure 2: Organization of the DDRAM (HD44780U)..... 11

Figure 3: dot matrix of the German umlaut ‘a’: ‘ä’..... 16

Web resources / bibliography

Tag	Description	URL
[CH341A]	Nanjing Qinheng Microelectronics Co., Ltd.	http://www.wch-ic.com/products/CH341.html
[ABACOM]	ABACOM Electronic-Software-Shop, USB display 4x20 characters green/white	https://www.electronic-software-shop.com/hardware/displays-usb/
[HITACHI]	Hitachi HD44780U (LCD-II), Dot Matrix Liquid Crystal Display Controller/Driver	https://de.wikipedia.org/wiki/HD44780 (German)
[HD44780U]	PDF specification HD44780U	https://cdn-shop.adafruit.com/datasheets/HD44780.pdf

Acronyms & Abbreviations

Acronym	Meaning
CH341A	USB bridge controller CH341x (here: A)
HD44780U	HITACHI LCD display module HD44780U
USB	Universal Serial Bus

1 Introduction

The German company ABACOM (see [ABACOM]) provides a 4 x 20 characters LCD display unit, which is connected via the CH341A – an USB bridge controller – (see [CH341A]) to the USB of a host PC.

The display is a Hitachi HD 44780U (see [HITACHI] – Wikipedia).

This unit has very good support for MS Windows operating systems but lacks a good support for Linux. For this reason, the document in-hand provides a Linux kernel driver supporting this USB/display unit.

1.1 Starting point (kernel driver)

A quite outdated Linux kernel driver is provided by the CH341A manufacturer and is available on GitHub: https://github.com/zoobab/ch341-parport (from 2013).

This driver does not work on current Linux kernels (kernel 5.10.x from Debian Bullseye (11) is used throughout this document). The nowadays used Intel/AMD CPU feature SMAP (Supervisory Mode Access Prevention) generates a General Protection (GP) Fault because the user-provided data (e.g. during a file system write operation) is not properly copied to kernel space.

Nevertheless, this kernel code is a good starting point because it ‘documents’ how the CH341A is programmed to drive e.g. the HD 44780U LCD display.

2 The display

This chapter describes the HD 44780U LCD display. It is organized in a 4 x 20 character matrix. Each character consists of 5x8 pixels (although 5x10 is supported, too).

2.1 Hardware description

According to [HD44780U], the HD44780U has two 8-bit registers:

1. instruction register (IR);
2. data register (DR).

Two types of memory (RAM) are embedded in the display unit:

1. DDRAM (display data RAM);
2. CGRAM (character generator RAM).

The DDRAM just consists of 80 memory cells 0x00 through 0x4F storing the displayed character codes.

The CGRAM contains the dot matrices of the characters. It is accompanied by the CGROM (character generator ROM) containing the fixed dot matrices of the built-in characters. 208 5x8 dot characters can be stored. The following figure (from https://de.wikipedia.org/wiki/HD44780) illustrates the default mapping:

Figure 1: Default character dot matrices (from https://de.wikipedia.org/wiki/HD44780#/media/Datei:Charset.gif).

Please note that my sample device has a different character dot matrix programmed in the CGROM (character generator ROM, as opposite to the character generator RAM, CGRAM).

No German umlauts as shown in Figure 1 are available in my device. For this reason a special library function will be introduced later, which provides the German umlauts ‘Ä’, ‘Ö’, ‘Ü’, ‘ä’, ‘ö’, ‘ü’, and ‘ß’ as user-defined characters. Up to eight characters as 5x8 dot matrices can be programmed in the LCD display.

2.1.1 Organization of the DDRAM

The display data ram stores the character codes of the four rows with twenty characters each. The memory accesses are **not** monotonically increasing. Figure 2 illustrates the memory layout of the DDRAM:

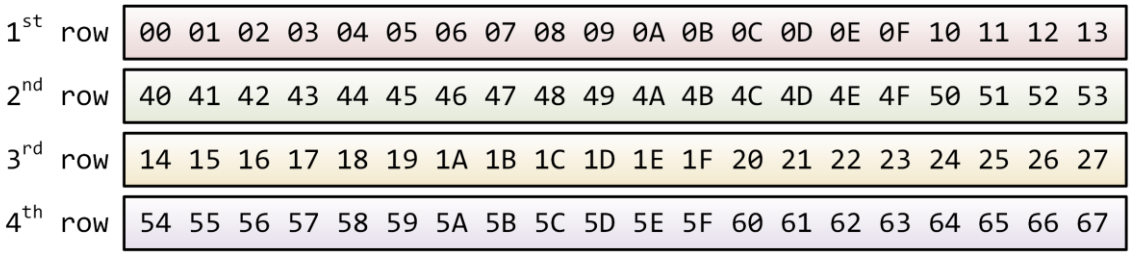

Figure 2: Organization of the DDRAM (HD44780U).

2.1.2 HD44780U commands

The display is connected to the host via the CH341A USB bridge controller, which has to be switched to 'MEM' mode (see chapter X).

USB bulk (out) transfer are used to communicate with the display. A prefix byte controls what is sent to the display. It is followed by a data byte. The following prefixes have been extracted from the original Linux kernel driver provided by [CH341A]:

Prefix byte:	Symbolic definition:	Meaning:
0xA0	CH34x_PARA_CMD_STS	get status of parallel
0xA1	CH34x_CMD_SET_OUTPUT	set parallel output
0xA2	CH34x_CMD_IO_ADDR	MEM IO address
0xA3	CH34x_CMD_PRINT_OUT	print output
0xA6	**CH34x_PARA_CMD_W0**	**write data$_0$ to parallel**
0xA7	**CH34x_PARA_CMD_W1**	**write data$_1$ to parallel**
0xA8	CH34x_CMD_SPI_STREAM	SPI command
0xA9	CH34x_CMD_SIO_STREAM	SIO command
0xAA	**CH34x_CMD_I2C_STREAM**	**I²C command**
0xAB	CH34x_CMD_UIO_STREAM	UIO command
0xAC	CH34x_PARA_CMD_R0	read data$_0$ from parallel
0xAD	CH34x_PARA_CMD_R1	read data$_1$ from parallel

Table 1: prefix bytes (command codes) CH341A.

Only the three highlighted command codes are used in the Linux kernel driver presented here. One I²C command is issued during device initialization. This is the sequence **0xAA,0x61,0x00**, which was captured using

Wireshark in USBCap mode using the MS Windows driver. I did not find out what this sequence is good for…

The vast majority of the commands sent to the CH341A (and thus to the HD44780U) either use **`0xA6`** (write data$_0$) or **`0xA7`** (write data$_1$), respectively.

The instruction register (IR) is just data$_0$, the data register (DR) is data$_1$. To make a long story short, data$_0$ is used to send instructions, data$_1$ is used to store a character code into the DDRAM (or to program user-defined dot matrices in the CGRAM):

Command byte:	Meaning:
`0x01`	clear display and set DDRAM address 0 in the address counter (AC)
`0x02`	return home, i.e. set AC to zero (0)
`0x04`	set cursor move direction and specify display shift (options in bits 1 and 0 ORed to the command byte)
`0x08`	display on/off, cursor on/off, cursor blinking on/off (options in bits 2, 1, and 0 ORed in the command byte)
`0x10`	‘cursor or display shift’, options in bits 1 and 0 ORed in the command byte
`0x20`	‘function set’, set e.g. 4 or 8 bit, etc.; options in bits 4, 3, and 2
`0x40`	set CGRAM address, address 0x00..0x3F in bits 0 through 5
`0x80`	set DDRAM address, address 0x00..0x7F in bits 0 through 6

Table 2: Command bytes (prefix bytes).

The cursor and the next write address in DDRAM is programmed via:

`0xA6,0x80+<`*`offset`*`>`

A capital ‘A’ (code 0x41) is then written by:

`0xA7,0x41`

Each write automatically increases[1] the address counter AC by one (1).

All of these byte pairs are sent via an USB bulk out request to the CH341A USB bridge controller. Normally, one should read back the so called BF = ‘busy flag’ to determine if the CH341A is able to receive the next command.

[1] The HD44780U can also be programmed to decrease the counter, I think this is for RTL = right-to-left writing and reading. The current driver implementation **does not** support RTL, though.

The Linux kernel driver presented here just waits 2ms after each write to limit the number of USB bulk transfers on the USB.

A full LCD display update takes 20 characters x 4 rows x (approx.) 2ms = 160ms.

3 USB bridge controller CH341A

The CH341A (vendor ID **0x1A86**, product ID **0x5512**) is connected to the HITACHI HD44780U LCD display.

Please note:

There are several Linux kernel drivers around for exactly the same VID:PID, which **do not** work for the HD44780U. Please ensure that none of these drivers is loaded while using the display!

Some examples for these drivers include (but are not limited to):

https://github.com/gschorcht/spi-ch341-usb

https://github.com/WCHSoftGroup/ch341ser_linux

https://github.com/zoobab/ch341-parport (matches the original driver)

3.1 Linux kernel driver

The Linux kernel driver `ch341ahd44780u.ko` (implemented by a single C source code `ch341ahd44780u.c`) was derived from the original kernel driver provided by [CH341A].

During initialization of the driver, a USB control transfer is performed to switch the CH341A into mem(ory) mode.

After that, only USB bulk (out) transfers are performed transmitting the Instruction Register (IR) or Data Register (DR) byte pairs – one-by-one. After the I/O completion, the driver waits 2ms before the next pair can be transferred.

3.1.1 IOCTLs

The kernel driver just supports the opening and closing of the device `/dev/hd44780u_dp`***N***, which N is 0, 1, 2, ... By this, you can attach more than one LCD display at once.

Furthermore, two IOCTLs are supported to program the HD44780U:

1. code `0x80000000` with an unsigned long (64bit) as the parameter. This sends a single command to the display (please refer to the source code for details);
2. code `0x8`*nnnnnnn* with an unsigned long (64bit) interpreted as a memory buffer address. '*nnnnnnn*' has to be the buffer size (> 0) in bytes. This IOCTL sends a sequence of commands to the display.

4 The userland library

To ease the programming of the LCD display HD44780U, a Linux shared object (userland library) is provided: `libusbhd44780u.so`

The library backs up the display buffer of the HD44780U with an in-memory buffer on the host. A scrolling mechanism is provided by the library, too.

To minimize the number of USB bulk transfers to the display, the library always computes the '*delta*' of the in-memory representation of the display buffer and the real hardware buffer (DDRAM) in the HD44780U.

The C header file `usbhd44780u.h` contains Doxygen documentation for each publicly available (exported) function.

4.1 The scroll and auto-scroll feature

The library maintains an '*auto-scroll*' flag, which can be toggled. If it is off (false), then new characters are written in 'wrap mode', i.e. once cursor position x=19, y=3 was written, then x=0, y=0 will be the next position (no scrolling).

If the auto-scroll flag is true, then the library automatically scrolls the LCD display line-by-line once the 3rd row was completely filled (or a newline '`\n`' character is written if the cursor is in the 3rd row).

Please recall that this is a software feature of the library, the LCD display HD44780U does not support hardware scrolling.

4.2 The character generator RAM

The CGRAM of the HD44780U is able to store eight user-defined characters with a 5x8 dot matrix each.

The character codes of these user-defined characters are just 0..7.

Figure 3 shows the German umlaut 'a', which is 'ä':

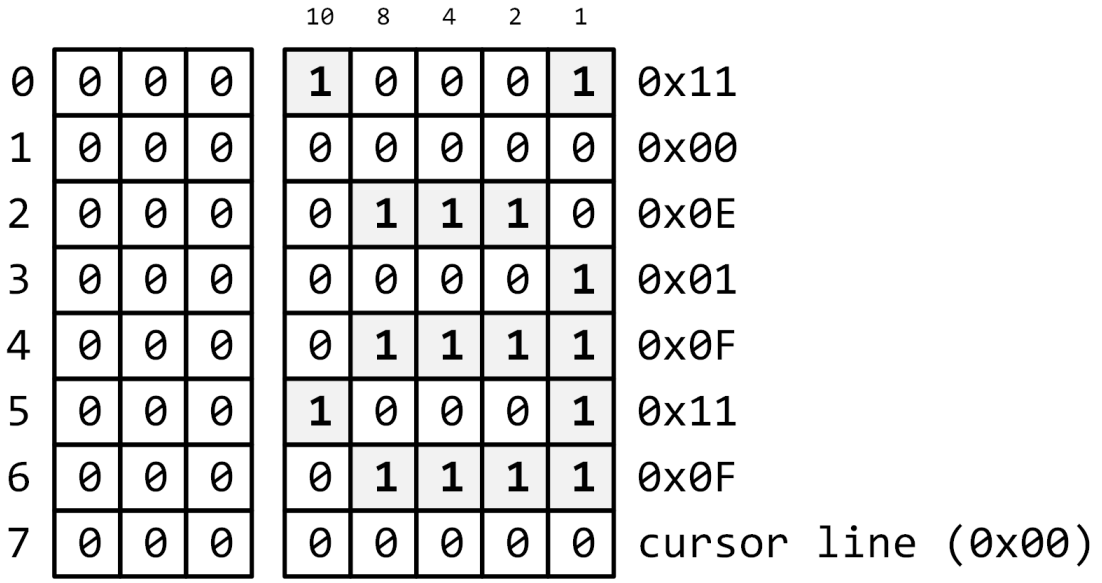

Figure 3: dot matrix of the German umlaut 'a': 'ä'.

You have to program the sequence `0x11, 0x00, 0x0E, 0x01, 0x0F, 0x11, 0x0F, 0x00` into the CGRAM addresses 0..7 to provide the German umlaut 'ä' as the character code `0x00`.

5 The command line tool

Normally, you should program the display in your own code using the shared object (please refer to chapter 4).

The command line tool `usbhd44780u` can be used to program the HD44780U on the command line.

If you execute the tool without arguments or with '`--help`', it displays:

```
HD44780U LCD display cmdline tool

usage: usbhd44780u [<options>...]
------

  --help            (or no options at all): print this help.
The following options may be repeated:
  --dev=<num>       specify device (default: 0)
  --clear           clear display
  --conf=DCBA       leave empty or specifc one or more of 'D', 'C', 'B':
                    'D' display on (if omitted, then off)
                    'C' cursor on (if omitted, then off)
                    'B' cursor blinking on (if omitted, then off)
                    'A' autoscroll on (if omitted, then off)
  --cursor=x,y      set cursor position; x=0..19, y=0..3
  --scroll=rows     rows=1..4, scroll this many rows
  --print="<msg>" print this message, '\n' is newline
                    '\"' is double quote, '\xNN' is code (hex).
                    '\'' is prime; you can alternatively use:
                    --print='<msg>'
  --wait=millis     delay this amount of milliseconds.
  --test            run INTERACTIVE test suite (all other options
                    except for --dev are IGNORED.
```

The kernel driver has to be loaded first if it is not already loaded when the device is attached to the USB.

Because the display is switched off (default), you always have to configure the display using the '--conf=' argument, e.g.

```
usbhd44780u --dev=0 --conf=D --print="Hello world."
```

This selects device 0 (this is the default, you can omit the device argument). Then, the display is switched on. Finally, the string "Hello world." is printed.

If you want to see the blinking cursor, then:

```
usbhd44780u --dev=0 --conf=DCB --print="Hello world.\n"
```

By appending '\n', the cursor moves to the next row, column 0 (a line feed).

To execute the full (interactive!) test suite, just execute:

```
usbhd44780u --dev=0 --test
```

A Appendix

A.1 Building the software

This software was tested on Debian Buster (10) with kernel 4.19.x and on Debian Bullseye (11) with kernel 5.10.x.

A.1.1 Required (Debian) packages

You have to install the full build chain, the DKMS (Dynamic Kernel Module System), and the Linux kernel headers:

```
sudo apt install build-essential dkms linux-headers-`uname -r`
```

A.1.2 Building the driver, library, and tool

Just enter:

```
make
```

A.1.3 Installing the software

Just enter:

```
sudo make install
```

The install target of the `Makefile` installs the tool into `/usr/bin`, the library into `/usr/lib`. The kernel module is installed by the DKMS. Finally, the `Makefile` removes any known conflicting kernel drivers and executes `modprobe` to load the newly compiled kernel driver.