LCDproc User's Guide

Table of Contents

LCDproc User's Guide.....1
The Ultimate Guide to LCDproc 0.5.....1
Peter Marschall.....1
Rene Wagner.....1
Guillaume Filion.....1
William W. Ferrell.....1

ChapterÂ 1.Â Introduction.....8
About this Document.....8
Note.....8
What is LCDproc?.....8
The LCDproc Server - LCDd.....9
The LCDproc "Main" Client - lcdproc.....9
Other LCDproc Clients.....9

ChapterÂ 2.Â How to Obtain LCDproc.....10
Versioning.....10
Download Git Version of LCDproc as a Tarball.....10
Download The Latest Version of LCDproc from GitHub.....10

ChapterÂ 3.Â Installation.....11
Build LCDproc.....11
Note.....11
Install LCDproc Directly From The Sources.....11
Note.....12
Generate And Install Packages of LCDproc.....12
Note.....12
Tip.....12
Note.....12
Important.....12

ChapterÂ 4.Â LCDproc Configuration.....14
Configure LCDd.....14
Note.....14
LCDd.conf: The [Server] Section.....15
Warning.....16
Note.....17
Note.....17
LCDd.conf: The [Menu] Section.....18
Note.....18
LCDd.conf: The Driver Section.....19
The LCDproc Init Scripts.....19
Note.....19
init-LCDd.....19
init-lcdproc.....19
Note.....19
init-lcdexec.....20
init-lcdvc.....20

Table of Contents

ChapterÂ 5.Â LCDproc Drivers.....21
The bayrad Driver.....24
Features.....24
Configuration in LCDd.conf.....25
The CFontz Driver.....25
Configuration in LCDd.conf.....25
The CFontzPacket Driver.....25
Configuration in LCDd.conf.....26
Note.....26
Note.....26
Note.....27
The curses Driver.....27
Configuration in LCDd.conf.....27
The CwLnx Driver.....28
Features.....28
Configuration in LCDd.conf.....28
The ea65 driver.....29
EA65 front panel VFD.....29
Configuration in LCDd.conf.....29
The Eyebox One Driver (EyeboxOne).....30
Eyebox One LCD Module.....30
Eyebox One Driver and lcdproc client.....30
Copyright.....30
Configuration in LCDd.conf.....30
Note.....31
Note.....31
The Futaba Driver.....31
Displays.....31
Note.....32
Requirements.....32
Note.....32
Configuration in LCDd.conf.....33
Note.....33
futaba driver status.....33
Copyright.....33
The G15 Driver.....33
Features.....33
Configuration in LCDd.conf.....33
The glcd Driver.....33
Markus Dolze.....34
Note.....34
Connections.....34
Important.....34
Tip.....34
Configuration in LCDd.conf.....35
Note.....37
The glcdlib Driver.....37
Lucian Muresan.....37

Table of Contents

ChapterÂ 5.Â LCDproc Drivers
Connections.....37
Configuration in LCDd.conf.....38
Note.....38
The glk Driver.....39
Supported devices.....39
Note.....39
Configuration in LCDd.conf.....39
The HD44780 Driver.....39
Tip.....41
Connections.....41
Warning.....41
Tip.....45
Caution.....46
Tip.....46
Note.....46
Note.....53
Note.....54
Note.....56
Note.....57
Tip.....59
Note.....59
Note.....60
Note.....63
Warning.....65
Warning.....66
Important.....66
Warning.....67
Warning.....67
Note.....68
Warning.....68
Note.....69
Compiling.....70
Configuration in LCDd.conf.....70
Tip.....71
Note.....72
Note.....73
Note.....73
Miscellanea.....74
The i2500vfd Driver.....74
Features.....75
Options.....75
The icp_a106 Driver.....75
Configuration in LCDd.conf.....75
The imon Driver.....75
General.....75
Configuration in LCDd.conf.....76
Tip.....76

Table of Contents

ChapterÂ 5.Â LCDproc Drivers
- The imonlcd Driver 76
  - General 76
  - Configuration in LCDd.conf 77
- The IOWarrior Driver 77
  - General 77
  - Requirements 78
  - Note 78
  - Note 78
  - Configuration in LCDd.conf 78
- The irman Driver 79
  - Tip 79
  - Configuration in LCDd.conf 79
- The irtrans Driver 79
  - General 79
  - Configuration in LCDd.conf 80
- The Joystick Input Driver 80
  - Configuration in LCDd.conf 80
- The lb216 Driver 80
  - Chris Debenham 80
  - Configuration in LCDd.conf 81
- The lcdm001 Driver 81
  - Configuration in LCDd.conf 81
- The lcterm Driver 81
  - Configuration in LCDd.conf 82
- The Linux Event Device Input Driver 82
  - Configuration in LCDd.conf 82
- The lirc Driver 82
  - Checking Your LIRC Setup 82
  - Build LCDd with the lirc Driver 82
  - Configure LCDd to Use the lirc Driver 83
  - Configuration in LCDd.conf 83
  - Modify Your ~/.lircrc 83
- The lis Driver 84
  - Features 84
  - Requirements 84
  - Note 85
  - Configuration in LCDd.conf 85
- The MD8800 Driver 85
  - Features 85
  - Configuration in LCDd.conf 85
- The mdm166a driver 86
  - Markus Dolze 86
  - Configuration in LCDd.conf 86
  - Using the icons 86
  - Special device hints 87
- The ms6931 Driver 88
  - Configuration in LCDd.conf 88

Table of Contents

ChapterÂ 5.Â LCDproc Drivers
The mtc_s16209x Driver....88
Configuration in LCDd.conf....88
The MtxOrb Driver....88
Matrix Orbital LCD Modules....88
Matrix Orbital Hardware Installation....89
Warning....89
Before you start....89
Warning....90
Caution....90
Note....91
Configuration in LCDd.conf....91
Note....91
The MX5000 Driver....92
Features....92
Configuration in LCDd.conf....92
The NoritakeVFD Driver....92
Configuration in LCDd.conf....93
The Olimex_MOD_LCD1x9 Driver....93
Configuration in LCDd.conf....93
The *Really* Simple Serial interface (rawserial)....93
Connor Wolf....93
Features....93
Requirements....94
Configuration in LCDd.conf....94
The Mini-Box.com USB LCD picoLCD Driver....94
Displays....94
Requirements....95
Note....95
Configuration in LCDd.conf....95
Note....95
Note....97
picolcd driver status....97
Note....98
Copyright....98
The pyramid Driver....98
Stefan Reinauer....98
Features....98
Connector....98
Requirements....98
LED output....99
Configuration in LCDd.conf....100
The SDEC LCD Driver....100
Features....100
The sed1330 Driver....100
Connections....100
Warning....100
Keypad....102

Table of Contents

ChapterÂ 5.Â LCDproc Drivers
- Configuration in LCDd.conf ... 102
- Note ... 103
- The sed1520 Driver ... 103
  - Note ... 103
  - Connections ... 103
  - Warning ... 103
  - Configuration in LCDd.conf ... 105
- The serialPOS Driver ... 105
  - Connecting The Display ... 106
  - Note ... 106
  - Configuration in LCDd.conf ... 106
- The serialVFD Driver ... 107
  - Connecting The Display ... 108
  - Warning ... 108
  - Tip ... 110
  - Configuration in LCDd.conf ... 112
  - Note ... 112
- The shuttleVFD Driver ... 112
  - Features ... 113
  - Configuration in LCDd.conf ... 113
- The sli Driver ... 113
  - Note ... 113
  - Configuration in LCDd.conf ... 113
- The stv5730 Driver ... 114
  - Configuration in LCDd.conf ... 114
- The SureElec Driver ... 114
  - Configuration in LCDd.conf ... 114
- The svga Driver ... 114
  - Configuration in LCDd.conf ... 114
  - Note ... 115
- The Toshiba T6963 Driver ... 115
  - Connections ... 115
  - Warning ... 115
  - Compiling ... 116
  - Configuration in LCDd.conf ... 116
- The text Driver ... 116
  - Configuration in LCDd.conf ... 116
- The tyan Driver ... 117
  - Features ... 117
  - Configuration in LCDd.conf ... 117
- The ula200 Driver ... 117
  - General ... 117
  - Requirements ... 118
  - Note ... 118
  - Note ... 118
  - Known problems ... 118
  - Implementation note ... 118

Table of Contents

ChapterÂ 5.Â LCDproc Drivers
Configuration in LCDd.conf....118
The vlsys_m428 driver....119
General....119
Configuration....119
The xosd Driver....119
Configuration in LCDd.conf....119
Note....120
The yard2 driver....120
General....120
Configuration....120
Parallel Port Troubleshooting....120
Check The Wiring....121
Power Source Unregulated / Noisy....121
Ground Lift....121
Latchup....121
Contrast....121
Beware....121
Parallel Port Voltage....121
Enable Signal Rise Time....121
Note....122
EMI....122
One or Two Black Lines....122
Software Too Fast....122
LED Backlight....122
HD44780 Compatible....122
Miscellanea....122

ChapterÂ 6.Â Running LCDproc....123
Running LCDd....123
Running LCDd from the command line....123
Note....123
The Command Line Options of LCDd....123
Running lcdproc....124
The Command Line Options of lcdproc....124
Note....125

ChapterÂ 7.Â Contact Us....126
Errata....126
The LCDproc Mailing List....126
Reporting Bugs....126

AppendixÂ A.Â GNU Free Documentation License....128
PREAMBLE....128
APPLICABILITY AND DEFINITIONS....128
VERBATIM COPYING....129
COPYING IN QUANTITY....129
MODIFICATIONS....130

Table of Contents

AppendixÂ A.Â GNU Free Documentation License
COMBINING DOCUMENTS....131
COLLECTIONS OF DOCUMENTS....131
AGGREGATION WITH INDEPENDENT WORKS....132
TRANSLATION....132
TERMINATION....132
FUTURE REVISIONS OF THIS LICENSE....132
How to use this License for your documents....133

LCDproc User's Guide

The Ultimate Guide to LCDproc 0.5

Peter Marschall

<peter@adpm.de>

Rene Wagner

<reenoo@gmx.de>

Guillaume Filion

<gfk@logidac.com>

William W. Ferrell

<wwf@splatwerks.org>

0.5.x

Abstract

This document is a guide to LCDproc written for users. It covers LCDproc 0.5.x

---

Table of Contents

1. Introduction
    - About this Document
    - What is LCDproc?
    - The LCDproc Server - LCDd
    - The LCDproc "Main" Client - lcdproc
    - Other LCDproc Clients

2. How to Obtain LCDproc

- Versioning
- Download Git Version of LCDproc as a Tarball
- Download The Latest Version of LCDproc from GitHub

3. Installation
- Build LCDproc
- Install LCDproc Directly From The Sources
- Generate And Install Packages of LCDproc

4. LCDproc Configuration
- Configure LCDd
  - `LCDd.conf`: The `[Server]` Section
  - `LCDd.conf`: The `[Menu]` Section
  - `LCDd.conf`: The Driver Section
- The LCDproc Init Scripts
  - init-LCDd
  - init-lcdproc
  - init-lcdexec
  - init-lcdvc

5. LCDproc Drivers
- The bayrad Driver
  - Features
  - Configuration in LCDd.conf
- The CFontz Driver
  - Configuration in LCDd.conf
- The CFontzPacket Driver
  - Configuration in LCDd.conf
- The curses Driver
  - Configuration in LCDd.conf
- The CwLnx Driver
  - Features
  - Configuration in LCDd.conf
- The ea65 driver
  - EA65 front panel VFD
  - Configuration in LCDd.conf
- The Eyebox One Driver (EyeboxOne)
  - Eyebox One LCD Module
  - Eyebox One Driver and lcdproc client
  - Copyright
  - Configuration in LCDd.conf
- The Futaba Driver
  - Displays
  - Requirements
  - Configuration in LCDd.conf
  - futaba driver status
  - Copyright
- The G15 Driver
  - Features
  - Configuration in LCDd.conf
- The glcd Driver
  - Connections
  - Configuration in LCDd.conf
- The glcdlib Driver

    - Connections
    - Configuration in LCDd.conf
- The glk Driver
    - Supported devices
    - Configuration in LCDd.conf
- The HD44780 Driver
    - Connections
    - Compiling
    - Configuration in LCDd.conf
    - Miscellanea
- The i2500vfd Driver
    - Features
    - Options
- The icp_a106 Driver
    - Configuration in LCDd.conf
- The imon Driver
    - General
    - Configuration in LCDd.conf
- The imonlcd Driver
    - General
    - Configuration in LCDd.conf
- The IOWarrior Driver
    - General
    - Requirements
    - Configuration in LCDd.conf
- The irman Driver
    - Configuration in LCDd.conf
- The irtrans Driver
    - General
    - Configuration in LCDd.conf
- The Joystick Input Driver
    - Configuration in LCDd.conf
- The lb216 Driver
    - Configuration in LCDd.conf
- The lcdm001 Driver
    - Configuration in LCDd.conf
- The lcterm Driver
    - Configuration in LCDd.conf
- The Linux Event Device Input Driver
    - Configuration in LCDd.conf
- The lirc Driver
    - Checking Your LIRC Setup
    - Build LCDd with the lirc Driver
    - Configure LCDd to Use the lirc Driver
    - Configuration in LCDd.conf
    - Modify Your `~/.lircrc`
- The lis Driver
    - Features
    - Requirements
    - Configuration in LCDd.conf
- The MD8800 Driver

  - Features
  - Configuration in LCDd.conf
- The mdm166a driver
  - Configuration in LCDd.conf
  - Using the icons
  - Special device hints
- The ms6931 Driver
  - Configuration in LCDd.conf
- The mtc_s16209x Driver
  - Configuration in LCDd.conf
- The MtxOrb Driver
  - Matrix Orbital LCD Modules
  - Matrix Orbital Hardware Installation
  - Configuration in LCDd.conf
- The MX5000 Driver
  - Features
  - Configuration in LCDd.conf
- The NoritakeVFD Driver
  - Configuration in LCDd.conf
- The Olimex_MOD_LCD1x9 Driver
  - Configuration in LCDd.conf
- The *Really* Simple Serial interface (rawserial)
  - Features
  - Requirements
  - Configuration in LCDd.conf
- The Mini-Box.com USB LCD picoLCD Driver
  - Displays
  - Requirements
  - Configuration in LCDd.conf
  - picolcd driver status
  - Copyright
- The pyramid Driver
  - Features
  - Connector
  - Requirements
  - LED output
  - Configuration in LCDd.conf
- The SDEC LCD Driver
  - Features
- The sed1330 Driver
  - Connections
  - Keypad
  - Configuration in LCDd.conf
- The sed1520 Driver
  - Connections
  - Configuration in LCDd.conf
- The serialPOS Driver
  - Connecting The Display
  - Configuration in LCDd.conf
- The serialVFD Driver
  - Connecting The Display

        - Configuration in LCDd.conf
    - The shuttleVFD Driver
        - Features
        - Configuration in LCDd.conf
    - The sli Driver
        - Configuration in LCDd.conf
    - The stv5730 Driver
        - Configuration in LCDd.conf
    - The SureElec Driver
        - Configuration in LCDd.conf
    - The svga Driver
        - Configuration in LCDd.conf
    - The Toshiba T6963 Driver
        - Connections
        - Compiling
        - Configuration in LCDd.conf
    - The text Driver
        - Configuration in LCDd.conf
    - The tyan Driver
        - Features
        - Configuration in LCDd.conf
    - The ula200 Driver
        - General
        - Requirements
        - Known problems
        - Implementation note
        - Configuration in LCDd.conf
    - The vlsys_m428 driver
        - General
        - Configuration
    - The xosd Driver
        - Configuration in LCDd.conf
    - The yard2 driver
        - General
        - Configuration
    - Parallel Port Troubleshooting
        - Check The Wiring
        - Power Source Unregulated / Noisy
        - Ground Lift
        - Latchup
        - Contrast
        - Parallel Port Voltage
        - Enable Signal Rise Time
        - EMI
        - One or Two Black Lines
        - Software Too Fast
        - LED Backlight
        - HD44780 Compatible
        - Miscellanea
- 6. Running LCDproc
    - Running LCDd

Running LCDd from the command line
The Command Line Options of LCDd
Running lcdproc
The Command Line Options of lcdproc
7. Contact Us
Errata
The LCDproc Mailing List
Reporting Bugs
A. GNU Free Documentation License
PREAMBLE
APPLICABILITY AND DEFINITIONS
VERBATIM COPYING
COPYING IN QUANTITY
MODIFICATIONS
COMBINING DOCUMENTS
COLLECTIONS OF DOCUMENTS
AGGREGATION WITH INDEPENDENT WORKS
TRANSLATION
TERMINATION
FUTURE REVISIONS OF THIS LICENSE
How to use this License for your documents

List of Figures

5.1. HD44780: Connecting the contrast adjusting pin (V_{LC})
5.2. HD44780: Direct Keys
5.3. HD44780: Single Matrix Key
5.4. HD44780: Complete Key Matrix
5.5. HD44780: One Return Line
5.6. HD44780: Backlight Wiring
5.7. HD44780: Serial LPT wiring ('simple')
5.8. HD44780: Serial LPT wiring ('complex')
5.9. HD44780: Serial LPT - Keypad return lines buffered
5.10. HD44780: Serial LPT - Backlight extra circuit
5.11. hd44780/ftdi: Backlight Wiring
5.12. HD44780: PCF8574P port expander on I^2C bus
5.13. HD44780: ethlcd - block diagram
5.14. HD44780: USB-4-all brightness control
5.15. HD44780: USB-4-all contrast control
5.16. serialVFD: Serial Inverter

List of Tables

5.1. HD44780: Connection Types
5.2. HD44780: Power Connections
5.3. HD44780: Direct Key Mapping
5.4. HD44780: Matrix Keypad Layout
5.5. HD44780: 4bit Pinouts (1)
5.6. HD44780: 4bit Pinouts (2)
5.7. HD44780: 4bit Keypad Pinouts
5.8. HD44780: "Winamp" wiring

5.9. HD44780: "Winamp" wiring - Keypad
5.10. HD44780: "lcdtime" wiring
5.11. HD44780: "lcdtime" wiring - keypad
5.12. HD44780: Serial LPT - Keypad return lines
5.13. HD44780: 8bit FTDI
5.14. HD44780: 4bit FTDI
5.15. HD44780: Examples of I^{2}C port expander addresses
5.16. HD44780: USB-4-all - Display connection
5.17. HD44780: USB-4-all - Keypad connection
5.18. HD44780: USB-4-all - Brightness and contrast connection
5.19. HD44780: Default pin mapping for Raspberry Pi
5.20. HD44780: Example pin mapping for Beaglebone Black
5.21. Mapping between LCDproc keys and IrMan commands
5.22. mdm166a_output bitmask
5.23. USB Pinout
5.24. Mapping of `output` bits to LEDs
5.25. ConnectionType `classic` ordered by LPT port pins
5.26. ConnectionType `bitshaker` ordered by LPT port pins
5.27. SED1520 80-style wiring schematic
5.28. SED1520 68-style wiring schematic
5.29. serialPOS: Emulation Protocol Status
5.30. serialVFD: Display Status
5.31. T6963 wiring schematic

List of Examples

4.1. `LCDd.conf`: Specify which driver to use
5.1. HD44780: Configuration for FTDI 4bit
5.2. HD44780: Configuration for LCD2USB
5.3. HD44780: Configuration for I^{2}C with port expander
5.4. HD44780: Configuration for I^{2}C with port expander Alternative Wiring
5.5. HD44780: Configuration for I^{2}C with port expander alternative config
5.6. HD44780: Alternate configuration for Raspberry Pi GPIO pins
5.7. HD44780: Configuration example for Pi Plate connection type
5.8. HD44780: Configuration example for PiFace Control and Display connection type
5.9. HD44780: Example configuration for Beaglebone Black GPIO pins with libugpio
5.10. HD44780: Example configuration for Raspberry Pi 4 GPIO pins with libgpiod
5.11. `LCDd.conf`: Activate the lirc driver
5.12. `~/.lircrc`: Specify the associations from buttons to keys for the lirc driver
5.13. Matrix Orbital: keymap config
5.14. Pyramid: How to use the LED output from the client
6.1. LCDd -h
6.2. lcdproc -h

ChapterÂ 1.Â Introduction

Table of Contents

About this Document
What is LCDproc?
The LCDproc Server - LCDd
The LCDproc "Main" Client - lcdproc
Other LCDproc Clients

About this Document

This document was meant as a tutorial for LCDproc users. It tries to introduce you into the world of LCDproc giving you an overview of the project. After reading this document you will be able to set up your own LCDproc'ed system.

Note

Please note that this document is still under construction". We hope to finish it until the final release of LCDproc â £. If you run into any trouble feel free to write to the LCDproc mailing list. See https://lists.lcdproc.org/wws/info/lcdproc for details on how to subscribe to the list.

Therefore you might want to have a look at http://lcdproc.sourceforge.net/docs/, to get the latest version of this document (unless you want to generate it yourself from the docbook files in the Git).

This document was originally written for LCDproc 0.4.3, but has been dramatically updated since. At the time of writing there had already been the "LCDproc User's Guide" written by William W. Ferrel in 1999. His version covered an early version of LCDproc and therefore concentrated on Matrix Orbital displays.

William's document was "recycled" for the description of the Matrix Orbital display driver and for other parts of this document.

In several other places e-mails and other documents have been included in this document. The authors of those are listed below every such document.

What is LCDproc?

LCDproc is a client/server suite including drivers for all kinds of nifty LCD displays.

The server LCDd makes it possible to display text and other data on an LCD display. As well LCDd can handle certain input devices.

Support for devices is added by drivers. We distinguish between output and input drivers. LCDd currently supports only one single output driver, which may at the same time handle input. Nevertheless several input (only) drivers are supported.

Currently there are drivers for several serial devices: Matrix Orbital, Crystal Fontz, Bayrad, LB216, LCDM001 (kernelconcepts.de), Wirz-SLI and PIC-an-LCD; and some devices connected to the LPT port: HD44780, STV5730, T6963, SED1520 and SED1330. There are input (only) drivers for LIRC and joysticks.

Clients can connect to LCDd through common TCP sockets.

Various clients are available. The *main* client lcdproc, which is shipped with the LCDproc distribution, can display things like CPU load, system load, memory usage, uptime, and a lot more.

The LCDproc Server - LCDd

LCDd is one of those well known *NIX daemons. BUT it's not just *one* daemon. It's the one that is supposed to drive your LCD ;)

LCDd can either be run from the command line or automatically by the init scripts shipped with the distribution.

As other daemons, LCDd has to be configured. In this respect a lot has changed since LCDproc 0.4.1. While LCDd retrieved all its configuration settings from the command line in 0.4.1, it now has a configuration file, which is normally `/etc/LCDd.conf`.

The LCDproc "Main" Client - lcdproc

While LCDd only offer the functionality of displaying text on a display, lcdproc actually retrieves data worth displaying.

lcdproc gets its information from the `/proc` filesystem.

lcdproc can connect to an LCDproc server either on the local system or on a remote system as long as it is reachable. It extracts the same statistics regardless of where it sends this information. The statistics it gathers include CPU utilization, memory utilization, disk utilization, network utilization, system uptime, time, and date, and so on. It displays this information in assorted ways, and can be tailored to taste.

Other LCDproc Clients

As it is rather simple to write an LCDproc client, you can find various clients on the Internet.

Unfortunately we cannot provide a list of LCDproc clients here. So, take a look at the Clients page on LCDproc's web site or have fun searching Google or freshmeat. Simply use `lcdproc` as the search pattern.

ChapterÂ 2.Â How to Obtain LCDproc

Table of Contents

Versioning
Download Git Version of LCDproc as a Tarball
Download The Latest Version of LCDproc from GitHub

Versioning

At the time of writing the following majors versions of LCDproc are floating around on the internet:

LCDproc 0.5.7
: LCDproc 0.5.7 is the current stable version of LCDproc. This is the version recommended to use.

Older releases
: Older releases of LCDproc can be found on the GitHub project page at https://github.com/lcdproc/lcdproc.

Download Git Version of LCDproc as a Tarball

The snapshot of the current source code repository may be downloaded from the web. Goto https://github.com/lcdproc/lcdproc/archive/master.tar.gz.

To extract the files run

```
$ tar xvfz lcdproc.tar.gz
```

Download The Latest Version of LCDproc from GitHub

Of course you can download the latest stuff from GitHub. For more information on how to use GitHub see https://guides.github.com/.

```
$ git clone https://github.com/lcdproc/lcdproc.git
```

Once you've done that and want to update the downloaded files to the latest stuff you can use the "pull" command of git (make sure to be in the lcdproc directory!):

```
$ git pull
```

ChapterÂ 3.Â Installation

Table of Contents

Build LCDproc
Install LCDproc Directly From The Sources
Generate And Install Packages of LCDproc

Build LCDproc

Now that you have downloaded the LCDproc distribution you can start building it.

Note

If you have installed the Debian package with apt-get (or another Debian package management tool), you can skip this this chapter.

If you're building this version from Git, you'll need autoconf, automake, aclocal and autoheader installed.

If you have autoconf and friends, run:

```
$ sh autogen.sh
```

This produces the configure script and supporting files. It has already been run if you are using the tarball distribution.

Once the above command has run, the rest is pretty standard:

```
$ ./configure --help
```

Read about the options, figure out what to use.

```
$ ./configure --prefix=/usr/local --enable-drivers=curses,CFontz
```

Be sure to replace `/usr/local` with the prefixdir you want (e.g. `/usr` for RedHat) and `curses,CFontz` with the comma-separated list of drivers you want to have compiled.

```
$ make
```

Congratulations: You have just compiled your version of LCDproc ;)

Install LCDproc Directly From The Sources

If you want to install LCDproc more or less permanently you can run:

```
$ su
Password: top secret
# make install
```

Note

make install is absolutely OPTIONAL You can also run LCDproc directly from the source directory. See below for details.

Generate And Install Packages of LCDproc

As an alternative (which is actually better ;) to installing directly from the sources you can generate packages using the packaging tool EPM.

First of all you may need to download EPM from http://www.epmhome.org/ and install it according to the instructions that are included in its source distribution.

Debian users (who do not want to download the official lcdproc debs via apt-get) can of course use Debian's epm package:

```
# apt-get install epm
```

Note

There are of course other and maybe better ways to generate packages for your system. The reason for us to choose EPM was that it provide the developers with a tool that makes it possible to write one list file for all platforms defining what the resulting package is meant to look like. This way we do not have to learn all the package managing tools of the different platforms that are supported by LCDproc.

To generate an LCDproc package follow these instructions:

Tip

It is of certain importance that you have run ./configure with the correct pathname settings for your system. Otherwise the resulting package will install the files in the wrong directories.

```
$ epm -v -f native LCDproc
```

Note

Generating an RPM package as a non-root user will fail, RPM wants to generate the files from the tree under `/usr/src/RPM`, which you do not have write access to as a non-root user. If you want to generate the package as a non-root user anyway, you may want to follow these instructions.

A workaround for the described problem is creating a file named `~/.rpmmacros` which contains:

```
%_topdir ~/rpm
```

Important

`~/rpm` must contain the same tree usually found under `/usr/src/RPM`

Unfortunately epm does not read `~/.rpmmacros` and of course returns warnings. Don't worry! That's OK ;)

In order to actually install the generated package follow the instructions in your system's manual.

ChapterÂ 4.Â LCDproc Configuration

Table of Contents

Configure LCDd
    `LCDd.conf`: The `[Server]` Section
    `LCDd.conf`: The `[Menu]` Section
    `LCDd.conf`: The Driver Section
The LCDproc Init Scripts
    init-LCDd
    init-lcdproc
    init-lcdexec
    init-lcdvc

Configure LCDd

As mentioned in the introduction LCDd, the LCDproc server, has its own configuration file, which is usually `/etc/LCDd.conf`.

Note

If you have not installed LCDproc from the sources the configuration file might have a different location. You should be able to find it by making your system's package manager list all the files in the LCDproc package.

The format of the `/etc/LCDd.conf` is INI-file like.

It is divided into sections that start at declarations that look like `[`*`section`*`]`; i.e. an opening square bracket, followed by the section name, and terminated by a closing square bracket, on a line by itself. Section names are case insensitive.

Parameters are grouped into sections and have the form *`key`*`=`*`value`*; i.e. a key, also known as the configuration option, followed by an equality sign and finally the value for the option. All three elements must occur together on one line. The *`key`*, which is case insensitive, may be surrounded by spaces, but is must be one word (i.e. a sequence of non-space characters) not containing the equality sign. A similar rule applies to the *`value`*: it may be surrounded by spaces, but it must be either one word or enclosed within double quotes (`"`), which are not considered as part of *`value`*. When quoted, the following character sequences are evaluated as in literal C strings:

escape sequence	character
`\a`	alert (bell) character
`\b`	backspace
`\f`	formfeed
`\n`	newline
`\r`	carriage return
`\t`	horizontal tab
`\v`	vertical tab
`\\`	backslash

All other occurrences of \ within quoted values will be ignored.

Comments are all line-based, and may start with '`#`' or '`;`'. Everything including and behind the character starting the comment up to the end of the line is ignored.

The server has a 'central' section named `[Server]`. Further each driver has a section which defines how the driver acts. Those sections start with `[`*`drivername`*`]`.

The drivers are activated by specifying them in a `Driver=` line in the server section, like:

ExampleÂ 4.1.Â `LCDd.conf`: Specify which driver to use

```
[Server]
Driver=curses
```

This tells LCDd to use the `curses` driver.

The drivers read their own options from the config file. For this purpose they use the config sections that are named like the driver.

`LCDd.conf`: The `[Server]` Section

The `[Server]` section of the `LCDd.conf` contains the settings for the LCDproc server LCDd.

DriverPath = *`DRIVERPATH`*
: Tells the server where to look for the driver files. See above for details. If not specified *`DRIVERPATH`* defaults to the empty string, resulting in drivers being searched in the directory LCDd is started in.

Driver = *`DRIVERNAME`*
: Tells the server which driver(s) to use. The first driver specified here that is capable of output functionality will be used as the *master* output driver, defining display properties and capabilities. All other drivers specified can only serve as input drivers or slave output drivers. If not specified *`DRIVERNAME`* defaults to `curses`, a driver that is supposed to work on any half-way decent UNIX console.

  This setting can be overridden on LCDd's command line using the `-d` *`DRIVER`* option. When the command line option is used, only the one driver given there will be loaded, and all drivers specified in the configuration file are ignored.

Bind = *`ADDRESS`*
: Tells the server to bind to the given local IP address and listen for incoming client connections. The default value for *`ADDRESS`* is `127.0.0.1`, which is actually the safest variant, as it allows connections only from the local machine and forbids connections from remote systems.

  This setting can be overridden on LCDd's command line using the `-a` *`ADDRESS`* option.

Port = *`PORTNUMBER`*
: Tells the server to listen to this specified port. If not specified *`PORTNUMBER`* defaults to `13666`.

  This setting can be overridden on LCDd's command line using the `-p` *`PORTNUMBER`* option.

ReportLevel = *`LEVEL`*

Sets the reporting level. Legal values for *`LEVEL`* range from `0` (only critical errors) to `5` (everything including debugging information). If not specified it defaults to `2` (warnings and errors only).

This setting can be overridden on LCDd's command line using the `-r` *`LEVEL`* option.

ReportToSyslog = { *`yes`* | *`no`* }

Should we report to `syslog` (`yes`) instead of `stderr` (`no`)? Default value is `no`.

This setting can be overridden on LCDd's command line using the `-s` *`NUMBER`* option. Passing `-s 1` on the command line enables reporting to `syslog` while `-s 0` disables it.

Warning

If LCDd is started automatically by an init script using the `curses` driver, it will lock `/dev/tty1`! So, be careful about what you are doing here.

User = *`USER`*

User to run as. When started as root LCDd will drop its privileges, and run as *`USER`* instead. Defaults to `nobody`.

This setting can be overridden on LCDd's command line using the `-u` *`USER`* option.

Foreground = { *`yes`* | *`no`* }

The server will stay in the foreground if set to true. Otherwise the server will fork to background and report to syslog. Defaults to `no`.

This setting can be overridden on LCDd's command line with the `-f` option that forces foreground mode.

Hello = *`HELLOMSG`*

Define the startup message printed on the screen when LCDd starts. This message will stay on the screen until the first client connects. If not given, it defaults to the built-in server screen that tells how many clients are connected and how many screens these clients are using in total. If it is given, each `Hello=` directive represents a line on the display.

The *`HELLOMSG`*s will be printed on the display one after each other starting on the beginning of each line. So, the definition of

```
Hello="   Welcome to"
Hello="    LCDproc!"
```

prints a nice 2-line welcome message to the display.

To simply disable the default built-in server screen on startup, and start with a blank screen a single `Hello=""` is sufficient.

GoodBye = *`GOODBYEMSG`*

Define the message left on the screen when LCDd exits. If not given, it defaults to the built-in `Thanks for using LCDproc!`. If it is given, each `GoodBye=` directive represents a line on the display.

The *`GOODBYEMSG`*s will be printed on the display one after each other starting on the beginning of each line. So, the definition of

```
GoodBye="       So Long,"
```

```
GoodBye="        and"
GoodBye="Thanks for All the Fish!"
```

prints the well known dolphin's message on the first 3 lines of the display (which obviously needs to be 24 columns wide to show the full last line).

To simply disable the default built-in message, and leave the screen blank a single `GoodBye=""` suffices.

FrameInterval = *`MICROSECONDS`*
: Sets the interval in microseconds for updating the display. If not specified the default value for *`MICROSECONDS`* is `125000`.

WaitTime = *`SECONDS`*
: Sets the default time in seconds to display a screen. If not specified the default value for *`SECONDS`* is `4`.

  This setting can be overridden on LCDd's command line with the `-w` *`SECONDS`* option.

AutoRotate = { *`yes`* | *`no`* }
: If set to `no`, LCDd will start with screen rotation disabled. This has the same effect as if the ToggleRotateKey had been pressed. Rotation will start if the ToggleRotateKey is pressed.

Note

This setting does not turn off priority sorting of screens. Therefore the client or LCDd may still show a different screen if it assigns it a higher priority than any other screen. Due to the way priority sorting works the screen shown when the first client connects may not be that clients first screen. If the client sets up more than two screens it will be the next to last one (this is not considered a bug).

ServerScreen = { *`yes`* | *`no`* | *`blank`* }
: Control the behaviour of the server screen, that usually shows the number of active clients and screens. When set to its default value `yes`, the server screen is included into the screen rotation scheme when other screens exist. Whet set to `no`, the server screen only shows up when no other screen exists. The special value `blank` is similar to `no`, but instead of displaying the current number of clients and screens, only a blank screen is displayed.

  This setting can be partially overridden on LCDd's command line using the `-i` *`NUMBER`* option. Passing `-i 1` on the command line enables server screen rotation, while `-i 0` disables it.

Note

Using the command line, it is not possible to set the server screen to `blank` mode.

Backlight = { *`off`* | *`open`* | *`on`* }
: Set the master backlight setting. If set to the default value `open`, then the backlight setting of the display can be influenced by the clients. When set to `off` or `on`, the backlight is set to the appropriate value without the clients being able to change the value.

Heartbeat = { *`off`* | *`open`* | *`on`* }
: Set the master heartbeat, the oscillating icon in the top right corner of the display, setting. If set to the default value `open`, then the heartbeat setting of the display can be influenced by the clients. When set to `off` or `on`, the heartbeat is turned on or off without the clients being able to change the value.

TitleSpeed = *`SPEED`*
: Set the speed how fast over-long title lines shall scroll. Legal values are `0` to `10`, where `0` means that no scrolling takes place and `10` stands for fastest scrolling. Default is `10`, where no artificial delay is

inserted.

The **â ¦Key** lines define what the server does with keypresses that don't go to any client.

ToggleRotateKey = *`KEY`*
: Defaults to `Enter`.

PrevScreenKey = *`KEY`*
: Defaults to `Left`.

NextScreenKey = *`KEY`*
: Defaults to `Right`.

ScrollUpKey = *`KEY`*
: Defaults to `Up`.

ScrollDownKey = *`KEY`*
: Defaults to `Down`.

`LCDd.conf`: The `[Menu]` Section

The `[Menu]` section enables you to set some general ("global") options related to the way LCDd handles input "events".

The menu is a special LCDproc client built into LCDd that allows changing server and display settings as well as extending it with entries from client applications.

You can configure what keys the menu should use.

PermissiveGoto = *`BOOL`*
: The flag controls whether transitions between the menus of different clients are allowed. The default `false` is to allow only menu gotos local to the client.

MenuKey = *`KEY`*
: The key that switches into menu mode (=open the main menu). In menu mode it cancels any operation. Cancelling the main menu means returning to the regular display mode. It has no default, but a natural candidate is `Menu`.

Note

The `MenuKey` will be reserved exclusively, while the others work in shared mode and can thus be used by a client application when not in the menu.

EnterKey = *`KEY`*
: The key to enter a sub menu, to select an entry and/or to confirm the value of an input field. If the *`RightKey`* is not defined, it is also used to move right in input fields. In this case the value of the input field is not confirmed, until the right end of the input has been reached. It is not set by default, but a natural candidate is `Enter`.

UpKey = *`KEY`*
: The key to move to the previous item in a menu and/or to select the previous value in input fields (e.g. the previous character available for the current position). If the *`DownKey`* is not set, moving up before the first entry automatically wraps around to the last entry. It is not set by default, but a natural candidate is `Up`.

DownKey = *`KEY`*
: The key to move to the next item in a menu and/or to select the next value in input fields (e.g. the next character available for the current position). If the *`UpKey`* is not set, moving down below the last

entry automatically wraps around to the first entry. It has no default, but a natural candidate is `Down`.

LeftKey = *`KEY`*
: If defined, this optional key is used to to move left in input fields and to select submenu entries. It is not set by default, but if you have more than 4 keys, a natural candidate is `Left`.

RightKey = *`KEY`*
: If defined, this optional key is used to to move right in input fields. It is not set by default, but if you have more than 4 keys, a natural candidate is `Right`.

The minimal keys required for most menus to work correctly are the **MenuKey**, the **EnterKey** and one of **UpKey** or **DownKey**. With these 3 keys the menus can usually be operated. Of course with only 3 keys the navigation gets a bit awkward. So if you have 4 or more keys, you better use them. Especially the **LeftKey** and **RightKey** make a big difference in user experience. Menus using checkbox or ring widgets within configuration wizards additionally need one of **LeftKey** or **RightKey** to work correctly, because **EnterKey** is used to jump to the next option then.

`LCDd.conf`: The Driver Section

As mentioned earlier, each driver has its own section in the `LCDd.conf`.

Although the settings are more or less self-explanatory, they are explained in the next chapter in the section for each driver. So, read through the section of your driver and change everything necessary.

The LCDproc Init Scripts

The LCDproc distribution contains init scripts for LSB 3.1 (Linux Standard Base 3.1) conforming GNU/Linux distributions. In addition to those it contains init scripts for older RedHat- and Debian-based distributions that do not adhere to LSB 3.1. You can find all of them in the `scripts/` directory of the LCDproc sources.

Note

The init scripts are generated using autoconf. So, again it is important that you have run **./configure** with the correct options for your system.

Refer to your system's manual on how to install the scripts.

init-LCDd

The file `scripts/init-LCDd.*` is the init script for the LCDproc server LCDd. It does not require modification.

init-lcdproc

The file `scripts/init-lcdproc.*` is the init script for the LCDproc "main" client lcdproc.

Note

You can retrieve a listing of all options of lcdproc running **lcdproc --help**.

init-lcdexec

The file `scripts/init-lcdexec.*` is the init script for the LCDproc lcdexec client, which can execute predefined commands via the menu feature.

init-lcdvc

The file `scripts/init-lcdvc.*` is the init script for the LCDproc lcdvc client, a simple terminal.

ChapterÂ 5.Â LCDproc Drivers

Table of Contents

The bayrad Driver
- Features
- Configuration in LCDd.conf

The CFontz Driver
- Configuration in LCDd.conf

The CFontzPacket Driver
- Configuration in LCDd.conf

The curses Driver
- Configuration in LCDd.conf

The CwLnx Driver
- Features
- Configuration in LCDd.conf

The ea65 driver
- EA65 front panel VFD
- Configuration in LCDd.conf

The Eyebox One Driver (EyeboxOne)
- Eyebox One LCD Module
- Eyebox One Driver and lcdproc client
- Copyright
- Configuration in LCDd.conf

The Futaba Driver
- Displays
- Requirements
- Configuration in LCDd.conf
- futaba driver status
- Copyright

The G15 Driver
- Features
- Configuration in LCDd.conf

The glcd Driver
- Connections
- Configuration in LCDd.conf

The glcdlib Driver
- Connections
- Configuration in LCDd.conf

The glk Driver
- Supported devices
- Configuration in LCDd.conf

The HD44780 Driver
- Connections
- Compiling
- Configuration in LCDd.conf
- Miscellanea

The i2500vfd Driver
- Features
- Options

The icp_a106 Driver
- Configuration in LCDd.conf

The imon Driver
- General
- Configuration in LCDd.conf

The imonlcd Driver
- General
- Configuration in LCDd.conf

The IOWarrior Driver
- General
- Requirements
- Configuration in LCDd.conf

The irman Driver
- Configuration in LCDd.conf

The irtrans Driver
- General
- Configuration in LCDd.conf

The Joystick Input Driver
- Configuration in LCDd.conf

The lb216 Driver
- Configuration in LCDd.conf

The lcdm001 Driver
- Configuration in LCDd.conf

The lcterm Driver
- Configuration in LCDd.conf

The Linux Event Device Input Driver
- Configuration in LCDd.conf

The lirc Driver
- Checking Your LIRC Setup
- Build LCDd with the lirc Driver
- Configure LCDd to Use the lirc Driver
- Configuration in LCDd.conf
- Modify Your `~/.lircrc`

The lis Driver
- Features
- Requirements
- Configuration in LCDd.conf

The MD8800 Driver
- Features
- Configuration in LCDd.conf

The mdm166a driver
- Configuration in LCDd.conf
- Using the icons
- Special device hints

The ms6931 Driver
- Configuration in LCDd.conf

The mtc_s16209x Driver
- Configuration in LCDd.conf

The MtxOrb Driver
- Matrix Orbital LCD Modules
- Matrix Orbital Hardware Installation

- Configuration in LCDd.conf

The MX5000 Driver
- Features
- Configuration in LCDd.conf

The NoritakeVFD Driver
- Configuration in LCDd.conf

The Olimex_MOD_LCD1x9 Driver
- Configuration in LCDd.conf

The *Really* Simple Serial interface (rawserial)
- Features
- Requirements
- Configuration in LCDd.conf

The Mini-Box.com USB LCD picoLCD Driver
- Displays
- Requirements
- Configuration in LCDd.conf
- picolcd driver status
- Copyright

The pyramid Driver
- Features
- Connector
- Requirements
- LED output
- Configuration in LCDd.conf

The SDEC LCD Driver
- Features

The sed1330 Driver
- Connections
- Keypad
- Configuration in LCDd.conf

The sed1520 Driver
- Connections
- Configuration in LCDd.conf

The serialPOS Driver
- Connecting The Display
- Configuration in LCDd.conf

The serialVFD Driver
- Connecting The Display
- Configuration in LCDd.conf

The shuttleVFD Driver
- Features
- Configuration in LCDd.conf

The sli Driver
- Configuration in LCDd.conf

The stv5730 Driver
- Configuration in LCDd.conf

The SureElec Driver
- Configuration in LCDd.conf

The svga Driver
- Configuration in LCDd.conf

The Toshiba T6963 Driver

- Connections
- Compiling
- Configuration in LCDd.conf

The text Driver
- Configuration in LCDd.conf

The tyan Driver
- Features
- Configuration in LCDd.conf

The ula200 Driver
- General
- Requirements
- Known problems
- Implementation note
- Configuration in LCDd.conf

The vlsys_m428 driver
- General
- Configuration

The xosd Driver
- Configuration in LCDd.conf

The yard2 driver
- General
- Configuration

Parallel Port Troubleshooting
- Check The Wiring
- Power Source Unregulated / Noisy
- Ground Lift
- Latchup
- Contrast
- Parallel Port Voltage
- Enable Signal Rise Time
- EMI
- One or Two Black Lines
- Software Too Fast
- LED Backlight
- HD44780 Compatible
- Miscellanea

This chapter contains the documentation of each LCDproc driver, which may include the installation process of the hardware as well as the configuration of LCDd.

The bayrad Driver

This section talks about using LCDproc with the BayRAD LCD modules by EMAC, Inc.

Features

The BayRAD LCD modules are designed to fit into 5,25" drive bays. They contain an LCD display that is 20 characters wide and 2 lines high surrounded by 4 buttons labeled `Menu`, `Select`, `+/Yes`, and `-/No`.

BayRAD modules are connected to the PC using a serial RS232 connection getting operating power using the standard floppy drive power connector.

For more information see the BayRAD home page

Configuration in LCDd.conf

[bayrad]

Device = `DEVICE`
: Select the serial output device to use. If not given, default is `/dev/lcd`.

Speed = { *1200* | *2400* | *9600* | *19200* }
: Set the the baud rate to use when communicating with the LCD. `9600` is the default, if not specified.

The CFontz Driver

This section talks about using LCDproc with the serial LCD displays of the CFA632 and CFA634 series by CrystalFontz, Inc.

Configuration in LCDd.conf

[CFontz]

Device = *DEVICE*
: Select the serial output device to use. If not given, default is `/dev/lcd`.

Size = *WIDTH x HEIGHT*
: Set the LCD's dimensions in terms of characters per line and lines. If not given, it defaults to `20x4`.

Contrast = *CONTRAST*
: Set the initial contrast. Legal values for *CONTRAST* are in the range between `0` and `1000`. If not given, it defaults to `560`.

Brightness = *BRIGHTNESS*
: Set the initial brightness. Legal values for *BRIGHTNESS* range from `0` to `1000`. If not given, it defaults to `1000`.

OffBrightness = *BRIGHTNESS*
: Set the initial off-brightness. This value is used when the display is normally switched off in case LCDd is inactive. Legal values *BRIGHTNESS* are in the range from `0` to `1000`. The default is `0`.

Speed = { *1200* | *2400* | *9600* | *19200* | *115200* }
: Set the the baud rate to use when communicating with the LCD. It defaults to `9600` if not specified.

NewFirmware = { *yes* | *no* }
: Set the firmware version (*New* means >= 2.0) [default: `no`; legal: `yes`, `no`].

Reboot = { *yes* | *no* }
: Reinitialize the LCD's BIOS [default: `no`; legal: `yes`, `no`] normally you shouldn't need this.

The CFontzPacket Driver

CrystalFontz offers a wide range of character and graphical LCD modules. The `CFontzPacket` driver supports the modules that communicate with the host computer using a packet-based communications protocol with 16-bit CRC (hence the driver name).

Currently this line of modules comprises the models:

CFA-631

- ◊ 20x2 character LCD with backlight
- ◊ keypad with 4 keys: Up, Down, enter & Escape
- ◊ USB connection for data and power
- ◊ mounting bracket to fit into 3,5" drive bays
- ◊ Optional (via add-on board) temperature sensor, fan and ATX power control connectors (all unsupported by LCDproc)

CFA-633

- ◊ 16x2 character LCD with backlight
- ◊ keypad with 6 keys: 4 directions, Enter & Escape
- ◊ serial or USB connection
- ◊ mounting bracket to fit into 5,25" drive bays
- ◊ Temperature sensor and fan control connectors (both unsupported by LCDproc)

CFA-533

- ◊ This is the same as CFA-633 except it misses the fan control capabilities. The temperature monitoring is not supported in LCDproc though.

CFA-635

- ◊ 20x4 characters LCD with backlight
- ◊ keypad with 6 keys: 4 directions, Enter & Escape
- ◊ USB connection for data and power
- ◊ four bi-color LEDs to show status information
- ◊ optional mounting bracket to fit into 5,25" drive bays

Configuration in LCDd.conf

[CFontzPacket]

Model = { *`533`* | *`631`* | *`633`* | *`635`* }

Select the LCD model that is connected. Legal values for this option are `631`, `533`, `633`, or `635`, with the default being `633`.

Device = *`DEVICE`*

Select the output device to use. It may be a serial device or a USB device in serial emulation mode. If not given, it defaults to `/dev/lcd`.

Note

Mac OS X users may need to use one of the `/dev/cu` devices instead of the `/dev/tty` ones.

USB = { *`yes`* | *`no`* }

Enable this flag if the device is connected to an USB port. For serial ports leave it disabled. [default: `no`; legal: `yes`, `no`]

Size = *`WIDTH x HEIGHT`*

Select the LCD size. This overrides the size the driver uses for the selected model (631: `20x2`, 533/633: `16x2`, 635: `20x4`).

Note

You should usually not need to set this value!

Contrast = *`CONTRAST`*

Set the initial contrast. Legal values for *`CONTRAST`* are `0` - `1000`. If not specified, it defaults to `560`.

Brightness = *`BRIGHTNESS`*
: Set the initial brightness [default: `1000`; legal: `0` - `1000`]

OffBrightness = *`BRIGHTNESS`*
: Set the initial off-brightness [default: `0`; legal: `0` - `1000`] This value is used when the display is normally switched off in case LCDd is inactive

Speed = { *`19200`* | *`115200`* }
: Override the default baud rate the driver uses for communication with the selected LCD model. Allowed values are `19200` (default for CFA-533 and CFA-633) and `115200` (default for the CFA631 and CFA635).

Note

You should usually not need to set this value!

OldFirmware = { *`yes`* | *`no`* }
: Very old 633 firmware versions do not support partial screen updates using 'Send Data to LCD' command (31). For those devices it may be necessary to enable this flag. [default: `no`; legal: `yes`, `no`]

Reboot = { *`yes`* | *`no`* }
: Reinitialize the LCD's BIOS [default: `no`; legal: `yes`, `no`].

The curses Driver

This section talks about using LCDproc with the (n)curses library. This diver displays an emulated LCD display of configurable size at a configurable position of the terminal screen using (n)curses.

Configuration in LCDd.conf

[curses]

Foreground = *`COLOUR`*
: Set the foreground color. If not given, it defaults to `blue`.

  Legal values for *`COLOUR`* are `red`, `black`, `green`, `yellow`, `blue`, `magenta`, `cyan` and `white`.

Background = *`COLOUR`*
: Set the background color. The default is `cyan`.

  The legal values for *`COLOUR`* are the same as for the Foreground setting.

Backlight = *`COLOUR`*
: Set the background color that is to be used when backlight is set on. backlight color. If not given, the default is `red`.

  The legal values for *`COLOUR`* are the same as for the Foreground setting.

Size = *`WIDTH x HEIGHT`*
: display size [default: `20x4`]

TopLeftx = *`X-OFFSET`* , TopLefty = *`Y-OFFSET`*
: What position (X,Y) to start the left top corner at. Default: (7,7)

UseACS = { *`yes`* | *`no`* }
: Tell whether to use ACS (alternative character set) symbols for icons and bars instead of simple ASCII characters.

DrawBorder = { `yes` | `no` }
: Tell whether to draw a border around the screen.

The CwLnx Driver

This section talks about using LCDproc with the serial / USB LCDs CW12232, CW12832 and CW1602 by CwLinux.

Features

The CwLinux CW12232 LCDs are graphical LCDs with 122 x 32 dots that also have a text mode with 20 x 4 characters, the CW12832 are graphical displays with 128 x 32 dots and a 21 x 4 character text mode, the CW1602 LCDs are character LCDs that are 16 characters wide and 2 lines high.

The modules can be ordered bare or as part of a kit mounted on brackets that fit in half-height 5.25" (CW12232 and CW1608) or 3,5" (CW12832) drive bays. The mounting brackets optionally feature a 6 button keypad that makes use of the keypad connector on the display modules.

The kits allow to programmatically switch on/off their backlight. Newer revisions of the kits also have programmable brighness as well as 4 general purpose IO ports.

The displays come in 2 variants that differ how they communicate with the host: The serial modules are connected to the PC using a serial RS232 connection getting operating power using the standard floppy drive power connector, while the USB modules only require an USB connection.

For more information see the CwLinux web site

Configuration in LCDd.conf

[CwLnx]

Model = { *12232* | *12832* | *1602* }
: Select the LCD model [default: `12232`; legal: `12232`, `12832`, `1602`]

Device = *DEVICE*
: Select the output device to use [default: `/dev/lcd`] May be serial device or USB device in serial emulation mode.

Size = *WIDTH x HEIGHT*
: Select the LCD size [default: depending on model: 12232: `20x4`, 12832: `21x4`] 1602: `16x2`]

Speed = { *9600* | *19200* }
: Set the the baud rate for communication with the LCD. If not given, the default is `19200`.

Reboot = { *yes* | *no* }
: Reinitialize the LCD's BIOS [default: `no`; legal: `yes`, `no`] normally you shouldn't need this

Keypad = { *yes* | *no* }
: Tells if you have a keypad connected. Keypad layout is currently not configurable from the config file.

KeyMap_A = *KEY* , KeyMap_B = *KEY* , KeyMap_C = *KEY* , KeyMap_D = *KEY* , KeyMap_E = *KEY* , KeyMap_F = *KEY*
: If you have a non standard keypad you can associate any keystrings to keys. There are 6 input key in the CwLnx hardware that generate characters from '`A`' to '`F`'. Legal values for *KEY* are `Up`, `Down`, `Left`, `Right`, `Enter` and `Escape`.

The following is the built-in default mapping hardcoded in the driver.

KeyMap_A	`Up`
KeyMap_B	`Down`
KeyMap_C	`Left`
KeyMap_D	`Right`
KeyMap_E	`Enter`
KeyMap_F	`Escape`

You may leave it unchanged if you have a standard keypad. You can change it if you want to report other keystrings or have a non standard keypad.

keypad_test_mode = { *`yes`* | *`no`* }

keypad_test_mode permit to test keypad assignment Default value is no

The ea65 driver

This section describes the ea65 driver which works with the front panel VFD display on the AOpen XC Cube-AV EA65 media barebone.

EA65 front panel VFD

The AOpen XC Cube-AV is a barebone designed for using as a media center. It comes with a front panel display which is capable of displaying one line of 9 characters.

The display is internally connected to the serial port (`/dev/ttyS1`) with a fixed rate of 9600 baud.

The display uses 13 segments per character. That's why the driver provides no custom characters like the ones for dot matrix displays do.

The front panel furthermore has 9 keys which are illuminated by blue LEDs. The LEDs can be controlled with the backlight functions. The keys are not supported by this driver. The red LED (RECORD) can be controlled with the `output` command of LCDd.

Configuration in LCDd.conf

[ea65]

Brightness = *`BRIGHTNESS`*

Set the brightness for the front LEDs if backlight is switched on. Legal values for *`BRIGHTNESS`* are in the range between `0` and `1000`. Values under `300` set the LEDs off. Values between `300` and `700` turn on the LEDs with half brightness. Values above `700` turn on the LEDs with full brightness. If not given, it defaults to `500`.

OffBrightness = *`OFFBRIGHTNESS`*

Set the brightness for the front LEDs if backlight is switched off. Legal values for *`OFFBRIGHTNESS`* are in the range between `0` and `1000`. Values under `300` set the LEDs off. Values between `300` and `700` turn on the LEDs with half brightness. Values above `700` turn on the LEDs with full brightness. If not given, it defaults to `0`.

The Eyebox One Driver (EyeboxOne)

This section describes the Eyebox One.

Eyebox One LCD Module

Eyebox One is a small rackmounted server marketed by Rightvision (http://www.alcateleyebox.rightvision.com/). This server has an LCD module, a keypad, two graphbars and some leds.

The LCD is a 20x4 alphanumeric module connected via standard DB-9 cabling and connector.

I couldn't find any documentation about it. All I know has been obtained with some reverse engineering. It seems that it can run only at 19.200 baud. Sending ASCII to the module will make it simply display that text at its current cursor position. The module has a built-in BIOS that recognizes commands (sent by transmitting a single-byte "marker" signifying that a command is on the way, followed by the single-byte command character itself along with any parameters, if needed) allowing the programmer to clear the screen, position the cursor anywhere, hide/show the cursor, on/off the backlight, and so on.

This module is fast. If updating less than the whole screen, the LCD can update faster than can be seen by the human eye. This, of course, more than meets LCDproc's needs.

Eyebox One Driver and lcdproc client

You can use the two Eyebox One graphbars, one as a free CPU meter, and one as a free RAM meter with lcdproc client (see eyebox.c in lcdproc client sources).

In order to use it, you must execute ./configure with a special parameter:
CPPFLAGS=-DLCDPROC_EYEBOXONE ./configure --enable-drivers=EyeboxOne

This is only a BETA version modification, take it as a demo...

Copyright

This section was originally part of the mtxorb.docbook file by Rene Wagner <`reenoo@gmx.de`>

This section has been modified by CÃ©dric TESSIER (http://www.nezetic.info)

Configuration in LCDd.conf

[EyeboxOne]

Device = *`DEVICE`*
    Select the output device to use [default: `/dev/ttyS1`]
Size = *`WIDTH x HEIGHT`*
    Set the display size [default: `20x4`]
Backlight = { *`yes`* | *`no`* }
    Switch on the backlight [default: `yes`; legal: `yes`, `no`]

Note

If you choose yes, you can switch on/off the backlight in real time using the LCDproc server menu with the keypad.

Cursor = { `yes` | `no` }
: Switch on the cursor? [default: `no`; legal: `yes`, `no`]

Speed = { `1200` | `2400` | `9600` | `19200` }
: Set the the baud rate to use when communicating with the LCD. If not specified, it defaults to `19200`.

Note

As I said, I think only `19200` is a good choice.

LeftKey = `D` , RightKey = `C` , UpKey = `A` , DownKey = `B` , EscapeKey = `P` , EnterKey =
: The following table translate from EyeboxOne Key to Logical Key. EyeboxOne Enter Key is a \r character, so it's hardcoded in the driver.

keypad_test_mode = { `yes` | `no` }
: You can find out which key of your display sends which character by setting keypad_test_mode to yes and running LCDd. LCDd will output all characters it receives. Afterwards you can modify the settings above and set keypad_set_mode to no again.

The Futaba Driver

This section covers the use of the `Futaba` TOSD-5711BB USB VFD display. This as commonly used on Elonex Artisan, Fujitsu Scaleo E and FIC Spectra Media Centre PCs.

Displays

The Futaba TOSD-5711BB is a VFD based USB 2.0 display, very much in the style of VHS Recorders/Players from the 1990s. It contains a single line, 7 Character 14-segment display and multiple icons to show the media type, codec, etc; current state of the PC; Volume knob and indicator. Each segment will display one character (Letters can only be in Capitals), and has a ':' (colon) and '.' (dot) between each character. There is no contrast control or back-light. The volume knob is lit using a blue LED and spins eternally. An image of the VFD circuit board and Volume knob can be seen `here`.

The IR attached is a standard MCE eHome remote which does not need any new drivers to work with lirc. It also works with scripts.xbmc.lcdproc on Kodi.

It comes pre-installed on Elonex Artisan, Fujitsu Scaleo E and FIC Spectra Media Centre PCs FIC Spectra Manual.

Note

- If the display crashes on start-up, this diff will solve the problem: Linux Diff. I have submitted this patch for inclusion into the standard Linux Kernel.
- The patched version of scripts.xbmc.lcproc can be found here, this should be added upstream soon too.

Requirements

The driver is based on the `libusb` USB library (Versions 1.0 and 0.1), which should make it work with Linux, the different BSD variants as well as Darwin/MacOS X.

Note

When using a `libusb` based driver like `futaba`, LCDd needs to be started as root. However LCDd may then drop down to a less privlidged user, such as nobody, after it has started.

On Linux, the only kernel module required is the USB host controller driver to fire up the USB bus to which the LCD is attached. For other operating systems, analogous requirements apply.

Configuration in LCDd.conf

[futaba]

Note

There are no Brightness, Contrast or Back-light controls as the display does not support these functions

futaba driver status

All icons/indicators function well. Currently the ':' (colon) or '.' (dot) in between characters do NOT operate. In a future update I hope to get these to work, with three settings: one to enable/disable flashing; one to set the flash rate; and one to choose between ':' (colon) or '.' (dot).

Copyright

The lcdproc futaba driver originally was written by Blackeagle email: gm(dot)blackeagle(at)gmail(dot)com Additions by Alex Wood (`2015/6`) email: thetewood(at)gmail(dot)com.

The G15 Driver

This section talks about using LCDproc with LCD displays on Logitech G15 gaming keyboards.

Features

If libg15daemon_client is present during build time, the g15 driver will include g15daemon support. If build with g15daemon support, the driver first tries to connect to the g15daemon. If the g15 is build without g15daemon support, or cannot connect to the g15daemon, it will use direct /dev/hidraw? device access. The g15 driver uses text and other rendering functions from libg15render. The g15daemon client-lib and libg15render are available from the g15daemon and g15tools projects at sourceforge.net.

If g15daemon is used, then input is provided by g15daemon, enabling use of the L1-L5 and G1 keys. When hidraw access is used the linux_input driver can be used for input.

Configuration in LCDd.conf

Currently there aren't really any configuration options to be set. The width and height are hardcoded based on the font currently used. In the future, now that libg15render has FreeType2 support, there may be options to adjust the font used and the display size.

The glcd Driver

Markus Dolze

The glcd driver (graphic lcd) driver is a "meta driver" that renders text for display on graphic displays. It uses either a built-in 5x8 font (the same as used in elsewhere in LCDproc) or Freetype 2 to draw the characters and icons into an internal frame buffer. That frame buffer is then copied to the display by a small sub-driver called connection type driver (CT-driver).

Note

LCDproc is compiled with FreeType support by default if it is installed on your system. It is enabled by default, too. However a font is not configured by default. Thus the driver will not start unless a font is configured!

Connections

Connection type t6963

Support for displays using a Toshiba T6863 controller connected to the parallel port. Wiring is the same as for the t6963 driver. Refer to the section called â   Connectionsâ   for details.

If used without FreeType this connection type uses the same font as the t6963 driver but draws the characters into the graphic memory of the T6963.

Important

You must configure the display for 8x8 font to make this connection type work!

Connection type png

This connection type writes out the frame buffer into files in `/tmp`. The files are named `lcdproc######.png` where `######` is a number starting at 0.

Tip

As a new file is written on any change to the screen it is best to turn off the heartbeat.

Connection type serdisplib

Use the serdisplib library (http://serdisplib.sourceforge.net/) for output. This enables use of a number of graphical displays connected to a parallel, serial, or USB port. See the serdisplib web page for details, especially for available options for your display.

Connection type glcd2usb

Support for the glcd2usb device (http://www.harbaum.org/till/glcd2usb/index.shtml, a graphic LCD to USB converter based on Atmel ATmega16 and the V-USB stack. This device features an adjustable backlight and 4 keys. Right now only displays with the KS0108 controller are supported.

The device has 4 keys which may be configured using the `KeyMap_A` (leftmost key) to `KeyMap_D` (rightmost key) settings.

Note that any size setting configured in `LCDd.conf` is ignored as the size reported by the device is used.

Connection type x11

This connection type draws the frame buffer to a X window. It features adjustable LCD pixel size, pixel color, backlight color and simulates contrast and brightness. PC keyboard is used to simulate buttons.

Connection type picolcdgfx

Support for the picoLCD Graphics 256x64 display from mini-box.com.

Configuration in LCDd.conf

[glcd]

Settings affecting all connection type drivers

ConnectionType = { *`t6963`* | *`glcd2usb`* | *`png`* | *`picolcdgfx`* | *`serdisplib`* | *`x11`* }
: Specify which connection type to use. See above for details.

Size = *`WIDTH x HEIGHT`*
: Specifies the size of the LCD in pixels. Default: `128x64`. Maximum value supported: `640x480`.

  The size in characters is automatically calculated from this value and the CellSize value (see below) and cannot be configured.

Contrast = *`CONTRAST`*
: Set the initial contrast (if supported by the connection type driver). Legal values for *`CONTRAST`* are `0` - `1000`. If not given, it defaults to `600`.

Brightness = *`BRIGHTNESS`*
: Set the initial brightness when the backlight is "on" (if supported by the connection type driver). Legal values are `0` - `1000`. If not given, it defaults to `800`.

OffBrightness = *`BRIGHTNESS`*
: Set the initial brightness when the backlight is set "off" (if supported by the connection type driver). Legal values are `0` - `1000`. If not given, it defaults to `100`.

Available parameters if compiled with FreeType support

useFT2 = { *`yes`* | *`no`* }
: Tell whether to use FreeType2 or not. It is set to `yes` by default. If turned off (set to `no`), the fixed internal 5x8 font is used to draw characters and icons.

normal_font = *`FONTFILE`*
: Set path to the font file to use, e.g. `/usr/local/lib/X11/fonts/TTF/andalemo.ttf`. This option is required if FreeType 2 support is enabled, but it is not set to any default value. A font with a fixed character width (monotype) is strongly recommended.

fontHasIcons = { *`yes`* | *`no`* }
: Many fonts to not include the Unicode glyphs used for drawing icons. If this option is set to `no` then the internal 5x8 font is used even if FreeType is enabled. This option is on (`yes`) by default.

CellSize = *`WIDTH x HEIGHT`*
: Specifies the size of the character cell in pixels. Characters will be drawn within this cell. Default:

`6x8`; minimum value `4x6`; maximum value: `24x32`.

Keypad settings

KeyRepeatDelay = *`DELAY`*

*`DELAY`*is the time in milliseconds from first key report to first repeat. Set to `0` to disable repeated key reports. Allowed values are between `0` and `3000`, the default is `500`.

KeyRepeatInterval = *`DELAY`*

*`DELAY`*is the time in milliseconds between repeated key strokes. Allowed values are between `0` and `3000`, the default is `300`. This setting is ignored, if KeyRepeatDelay is disabled (set to zero).

KeyMap_A = *`KEY`* , KeyMap_B = *`KEY`* , KeyMap_C = *`KEY`* , KeyMap_D = *`KEY`* , KeyMap_E = *`KEY`* , KeyMap_F = *`KEY`*

These settings allow to assign arbitrary key strings to key strokes. Up to 26 keys may be configured ('`A`' to '`Z`').

The following is the built-in default mapping hard-coded in the driver. Only keys '`A`' to '`F`' are used by default.

KeyMap_A	`Up`
KeyMap_B	`Down`
KeyMap_C	`Left`
KeyMap_D	`Right`
KeyMap_E	`Enter`
KeyMap_F	`Escape`

Settings for the t6963 connection type

Port = *`PORT`*

Specify the address of the parallel port the LCD is connected to. Common values for *`PORT`* are `0x278`, `0x378` and `0x3BC`. If not given, the default is `0x378`.

bidirectional = { *`yes`* | *`no`* }

Use parallel port in bi-directional mode. [default: `yes`; legal: `yes`, `no`]

Most LPT ports can be used in bi-directional mode. It is required for proper timing of the display.

delayBus = { *`yes`* | *`no`* }

Use additional delay in read / write operations. [default: `no`; legal: `yes`, `no`]. As the driver implements busy checking usually no additional delays are required.

Settings for the serdisplib connection type

serdisp_name = *`NAME`*

This is the name or alias to select a display driver in the serdisplib library. As there is not default value setting this option is mandatory.

serdisp_device = *`DEVICE`*

Set the device to use. Unlike elsewhere in `LCDd.conf` this is a device description understood by serdisplib. As there is not default value setting this option is mandatory.

serdisp_options = *`OPTIONS`*

`OPTIONS` is a serdisplib option string, which is a list of semicolon separated key / value pairs. Use this to pass any additional options to serdisplib, e.g. wiring or rotation settings.

Note

As the value will contain equal signs the whole `OPTIONS` value must be enclosed in double quotes.

The display width and height are always set using the values from the glcd driver (see above). Any width / height values set in the option string will be ignored.

Settings for the x11 connection type

x11_PixelSize = *`PIXELS + GAP`*
: Each LCD dot is drawn in the X window as a filled rectangle of this size plus a gap between each filled rectangle. A PixelSize of `3+1` would draw a 3x3 filled rectangle with a gap of 1 pixel to the right and bottom, effectively using a 4x4 area of the window. Default is `3+1`.

x11_PixelColor = *`0xRRGGBB`*
: The color used to draw each LCD dot at full contrast and full brightness. Default is `0x000000`.

x11_BacklightColor = *`0xRRGGBB`*
: The color used for the backlight at full brightness. Default is `0x80FF80`.

x11_Border = *`PIXELS`*
: Extra space in pixels around the LCD portion of the X window. Default is `20`.

x11_Inverted = { *`yes`* | *`no`* }
: Swap the colors for the pixel and backlight within the LCD portion of the X window. This results in an inverted display.

Settings for the picolcdgfx connection type

picolcdgfx_KeyTimeout = *`TIMEOUT`*
: The time in milliseconds to wait for a key press before assuming no key was pressed. This settings is the usb_read timeout. Default 125.

picolcdgfx_Inverted = { *`yes`* | *`no`* }
: When set to yes, the pixel will be inverted. Default no.

The glcdlib Driver

Lucian Muresan

This section talks about using LCDproc with LCD displays supported by graphlcd-base.

Connections

The so-called "meta-driver" glcdlib extends LCDproc's supported drivers by all the drivers supported by graphlcd-base, which you can get from http://projects.vdr-developer.org/projects/graphlcd/.

In order to be able to use it, you have to get and install the glcdprocdriver from http://lucianm.github.com/GLCDprocDriver/ before configuring the LCDproc build process --enable-drivers=glcdlib.

Configuration in LCDd.conf

[glcdlib]

Mandatory settings

Driver = *`GRAPHLCD-DRIVER`*
: Specify which graphical display supported by graphlcd-base to use. Legal values for *`GRAPHLCD-DRIVER`* are specified in graphlcd's configuration file `/etc/graphlcd.conf`. For graphlcd 0.13 they comprise `avrctl`, `framebuffer`, `gu140x32f`, `gu256x64-372`, `gu256x64C-3xx0`, `hd61830`, `image`, `ks0108`, `noritake800`, `sed1330`, `sed1520`, `serdisp`, `simlcd`, and `t6963c`. If not specified it defaults to `image`.

UseFT2 = { *`yes`* | *`no`* }
: Tell whether to use FreeType2 or not. If set to `no` use graphlcd's bitmap fonts, which is only one size/font file. If set to to the default value `yes` use the fonts that FreeType2 provides.

Note

Setting it to `yes` requires Freetype2 support in libglcdprocdriver and its dependants.

TextResolution = *`WIDTH x HEIGHT`*
: Give text resolution in fixed width characters. If it won't fit according to the available physical pixel resolution and the minimum available font face size in pixels, 'DebugBorder' will automatically be turned on. If not specified, it defaults to `16x4`.

FontFile = *`FILENAME`*
: Set path to font file to use, e.g. `/usr/share/fonts/corefonts/courbd.ttf`.

Availalble parameters if UseFT2 = `yes`

CharEncoding = *`CHARSET`*
: Specify character encoding to use, e.g. `iso8859-2`. If not given, use the default `ISO8859-1`.

MinFontFaceSize = *`COLUMNS x ROWS`*
: minimum size in pixels in which fonts should be rendered

Optional settings

Brightness = *`BRIGHTNESS`*
: Brightness (in %) if applicable Legal values are `0` - `100`. If not specified, the default is `50`.

Contrast = *`CONTRAST`*
: Set the contrast (in %) if applicable. Legal values are `0` - `100`, with `50` being the default when not specified.

Backlight = { *`yes`* | *`no`* }
: Backlight if applicable

UpsideDown = { *`yes`* | *`no`* }
: flip image upside down

Invert = { *`yes`* | *`no`* }
: invert light/dark pixels

ShowDebugFrame = { *`yes`* | *`no`* }
: turns on/off 1 pixel thick debugging border within the usable text area, for setting up TextResolution and MinFontFaceSize (if using FT2);

ShowBigBorder = { *`yes`* | *`no`* }

border around the unused area

ShowThinBorder = { *yes* | *no* }
: border around the unused area

PixelShiftX = *SHIFTX* , PixelShiftY = *SHIFTY*
: Shifts the content of the display by *SHIFTX* (default: `0`) and *SHIFTY* (default: `0`) pixels.

The glk Driver

This section talks about using LCDproc with LCD displays that use the Matrix Orbital GLK and GLC chipset.

Supported devices

Currently the drivers supports the following devices:

- GLC12232 (20x4)
- GLC12864 (20x8)
- GLC128128 (20x16)
- GLC24064 (40x8)
- GLK12232-25 (20x4)
- GLK12232-25-SM (20x4)
- GLK12864-25 (20x8)
- GLK128128-25 (20x16)
- GLK24064-25 (40x8)
- GLK19264-7T-1U-USB (32x8)

Note

Modules not in the list above are not recognized and the driver will not load if it encounteres an unrecognized display.

Configuration in LCDd.conf

[glk]

Device = *DEVICE*
: select the serial device to use [default: `/dev/lcd`]

Contrast = *CONTRAST*
: Set the initial contrast. Legal values for *CONTRAST* are `0` - `1000`. If not given, it defaults to `500`.

Speed = { *9600* | *19200* | *38400* | *57600* | *115200* }
: Set the the baud rate for communication with the LCD. The default is `19200`.

The HD44780 Driver

The HD44780 has become the de-facto standard of alphanumeric character displays. Although the original Hitachi HD44780 is long out of production, its command set has survived in a variety of (fully or nearly) compatible LCD, VFD and even OLED displays sold by a broad range of manufacturers all over the world. To name only a few: KS0066, KS0070, KS0076, LC7985, NT3881, SED1278, ST7066 ...

The command set of these displays, which sometimes are advertised as being "industry standards compatible", is thus the workhorse of controllers for displays ranging from 8x1 to 20x4 or 40x2 characters. There are even displays with larger dimensions sporting two controllers, one for each half of the display.

The HD44780 driver supports various ways of connecting HD44780 devices to your system. Each of these different ways is called a *connection type* of the driver.

Depending on the connection type and the display connected, the driver supports various special features.

- Input keys
- Software controllable brightness / backlight
- Software controllable contrast
- multiple displays / multi-controller displays

The following table lists the available connection types, sorted by the kind of connection (parallel port, serial port, USB, I2C, SPI or others).

TableÂ 5.1.Â HD44780: Connection Types

ConnectionType	**Wiring / Display Type**
Parallel port connection types:	
`4bit`	4-bit wiring. This is the default.
`8bit`	8-bit wiring ("lcdtime" style)
`winamp`	8-bit wiring ("winamp" style)
`lcm162`	8-bit wiring ("lcm162" style)
`serialLpt`	Serial LPT wiring
Serial (RS-232) connection types:	
`picanlcd`	PIC-an-LCD serial device "picanlcd"
`lcdserializer`	LCD serializer "lcdserializer"
`los-panel`	LCD on Serial panel device "los-panel" (http://mlf.home.xs4all.nl/los/)
`vdr-lcd`	VDR LCD serial device "vdr-lcd"
`vdr-wakeup`	VDR-Wakeup module "vdr-wakeup"
`ezio`	EZIO-100 and EZIO-300 by Portwell
USB connection types:	
`pertelian`	Pertelian X2040 LCD display (http://pertelian.com/joomla/index.php?option=com_content&task=view&id=43&Itemid=48)
`lis2`	LIS2 from VLSystem (http://www.vlsys.co.kr)
`mplay`	MPlay Blast from VLSystem (http://www.vlsys.co.kr)
`usblcd`	USBLCD adapter from Adams IT Services (http://www.usblcd.de/)
`bwctusb`	BWCT USB device "bwctusb" (http://www.bwct.de/lcd.html)
`lcd2usb`	Till Harbaum's LCD2USB (http://www.harbaum.org/till/lcd2usb/)
`usbtiny`	Dick Streefland's USBtiny (http://www.xs4all.nl/~dicks/avr/usbtiny/)
`uss720`	Belkin USS-720 USB-to-IEEE 1284 Bridge (Belkin F5U002)
`USB-4-all`	The USB-4-all controller board from Sprut (http://www.sprut.de/electronic/pic/projekte/usb4all/usb4all.htm)

ConnectionType	Wiring / Display Type
`ftdi`	Display connected to a FTDI RS2232 (dual channel) or FTDI RS232 (single channel) USB chip
I^{2}C connection types:	
`i2c`	LCD driven by PCF8574(A)/PCA9554(A) connected via I^{2}C
`piplate`	Adafruit RGB Positive 16x2 LCD+Keypad for Raspberry Pi
SPI (Serial Peripheral Interface):	
`spi`	Display using a KS0073 or similar in serial mode accessed via Linux SPI device
`pifacecad`	PiFace Control and Display for Raspberry Pi
Other connection types:	
`ethlcd`	Display connected via TCP to PoE powered ethlcd device (http://manio.skyboo.net/ethlcd/)
`raspberrypi`	LCD connected to the GPIO header of a Raspberry Pi
`gpio`	LCD connected to GPIO lines (linux sysfs interface)

Tip

If you suspect the table above to be outdated, you might want to have a look at `server/drivers/hd44780-drivers.h` in LCDproc's source directory which contains the actual translation code.

Connections

Common connections for all connection types

No matter what connection type you choose, you will always need some connections. They are explained here.

Power

All variants use the same method of obtaining power. i.e., for each LCD:

TableÂ 5.2.Â HD44780: Power Connections

LCD		Signal
name	pin	
V_{EE}	1	GND (connect to any of pins 18 - 25 of you parallel port)
V_{CC}	2	+5V
V_{LC}	3	(contrast adjustment)

Warning

Always double check your power connection, your display will probably *not* survive a reversely connected supply !

There are several ways to get 5V:

- Connect to a 5V line intended for disk drives (the red wire is 5V, black is GND).
- Get it from the V_{CC} and GND pins of an USB connector. For the USB connection types this is done automatically, as the circuits used there automatically power the LCD.
- Get it from a joystick port (pin 1 and 9 are 5V, 4, 5 and 12 are GND). It seems that some soundcards can use these lines for communication, so if you want to use this first check whether it really gives a 'clean' 5V.
- If you don't have a backlight, you can sometimes get the needed mA's from the LPT port itself. Connect a few diodes from the data pins to a capacitor and you have the 5V. If it's strong enough is another question...
- Get it from the keyboard connector. I do not recommend to use this with a backlight, as the keyboard connector is often protected with a fuse of 100mA or 200mA.

FigureÂ 5.1.Â HD44780: Connecting the contrast adjusting pin (V_{LC})

```
      (variable resistor)
         .------.
 Vcc ---|  10k |--- GND
         `---^--'
            /|\
             |
            Vlc
```

Keypad

You can connect a keypad with most connection types. The maximum supported number of keys differs per type. There are several ways to connect the keys to the input pins.

Direct Keys

If you connect a key like sketched below, then you can only connect one key per input pin. It is a simple solution if you need only few keys.

FigureÂ 5.2.Â HD44780: Direct Keys

```
 O 5V
 |
 |
 -
| | 10k
| |
 -
 |
 +-----------o input  (X)
 |
 |
 o
  \
 o
 |
 |
   === GND
```

By default, the following keystrokes are generated by the different keys:

TableÂ 5.3.Â HD44780: Direct Key Mapping

key index	mapped string
X_1	`A`
X_2	`B`
X_3	`C`
X_4	`D`
X_5	`E`

You can change the mapping using the KeyDirect_*`NUM`* configuration option, where *`NUM`* is the subscript to the `X` in the table above.

Matrix Keys

Using a matrix, we can connect much more keys. To simplify the drawing here, we replace all switches with an @ symbol:

FigureÂ 5.3.Â HD44780: Single Matrix Key

```
           X line
              |
              |
Y line ---+---------
          |   |                    |
          o   |               =  --@--
           \  |                    |
          o   |
          |   |
          +---+
              |
              |
```

We connect the matrix of keys like this:

FigureÂ 5.4.Â HD44780: Complete Key Matrix

```
Y1 o---|<---@--@--@
            |  |  |
Y2 o---|<---@--@--@
            |  |  |
Y3 o---|<---@--@--@
            |  |  |
Y4 o---|<---@--@--@            O 5V
            |  |  |            |
    diodes  |  |  |     ___    |
    1N4148  +----------|___|---+
            |  |  |     ___    |
            |  +-------|___|---+
            |  |  |     ___    |
            |  |  +----|___|---+     resistors 22k
            |  |  |
            o  o  o
            X1 X2 X3
```

As you can see, you need 1 resistor per X line, and 1 diode per Y line. By default, lcdproc will presume that you have a keypad with a layout like a telephone connected, with X and Y lines connected as show. To be more precise, it assumes this mapping:

TableÂ 5.4.Â HD44780: Matrix Keypad Layout

Â	X_1	X_2	X_3	X_4	X_5
Y_1	1	2	3	A	E
Y_2	4	5	6	B	F
Y_3	7	8	9	C	G
Y_4	*	0	#	D	H

This mapping can be changed using the KeyMatrix_`X`_`Y` configuration option, where `X` and `Y` are the subscripts to the respective axes above.

If you only need e.g. 10 keys, leave the rest away. You should modify and recompile the driver to get an other keypad layout.

You can buy arrays of keys that are connected like this in the electronics shop. They usually call it a matrix keypad. To hook it to lcdproc, you would only need to add the resistors and diodes.

If you want to use just one return line, for example with the serialLpt wiring, it looks (completely drawn) like this:

FigureÂ 5.5.Â HD44780: One Return Line

```
                  O 5V
                  |
                 .-.
                 | | 4k7 or 22k
     diodes      | |
     1N4148      '-'
            ___   |
Y1 o---|<---o o---+
            ___   |
Y2 o---|<---o o---+
            ___   |
Y3 o---|<---o o---+
            ___   |
Y4 o---|<---o o---+----o return line
```

Tip

If the driver generates keypresses without that you actually press a key, it might be that the unconnected input lines are picking up electromagnetic waves from the air. In that case connect the unconnected input lines (pin 10, 11, 12, 13 and 15 of the LPT) to V_{CC} = 5V.

Backlight

A small extension allows you to switch the backlight of the display on and off. At the moment only the `4bit` and `winamp` connection types support this. The extension uses one output pin, you cannot use that pin for other functions anymore. The wiring looks like this:

FigureÂ 5.6.Â HD44780: Backlight Wiring

```
                                             O 5V
                                      ___    |
                                 +---|___|---+
LPT Sub-D connector              |   4k7     |
                                 |           |e
                          ___    |       b |/
     BL pin o------------|___|---+---------|
                          1k               |\
                                       BC327 |c
                                             |        LCD connector
                                             |
                                             +---------o 15 backlight

                                             +---------o 16 GND backlight
                                             |
                                            === GND

Note: 4k7 means 4,7 kOhm.
The BC327 transistor has the following connections:

     _____
    |     |
    |BC327|
    |_____|
     | | |
     | | |
     | | |
     c b e
```

Caution

Sometimes the backlight connections are not on the 'main' connector, but on the side. If that is the case, there is usually NO RESISTOR present to limit the current through the LEDs. Therefore you should then add a resistor after the transistor of about 10 ohm (see display documentation).

Tip

If you want the backlight to light a bit while it is "switched off", you can add a resistor bypassing the transistor from e to c, with a value of, say 47ohm or 22ohm. (My 4x20 has an internal resistor of 6ohm, so with 47 ohm extra it lights at only 1/9th. I like this. Joris.)

4bit

This wiring is originally based on "lcdtext" (by Matthias Prinke).

TableÂ 5.5.Â HD44780: 4bit Pinouts (1)

printer port		<->	LCD	
name	pin		name	pin
Â	Â	GND	V_{EE}	1
Â	Â	+5V	V_{CC}	2
Â	Â	(contrast adjustment)	V_{LC}	3
D4	6	Â	RS	4
Â	Â	GND	RW	5
D6	8	Â	EN	6
D0	2	Â	D4	11
D1	3	Â	D5	12
D2	4	Â	D6	13
D3	5	Â	D7	14

Note

The `RW` (pin 5) line of the display decides whether the display receives data from the LPT port, or whether it sends data to the LPT port: if grounded it receives, if High or connected to nothing at all it "sends" (i.e., will not work as intended). So, if you are not sure that you need it otherwise, then connect it to GND. This certainly applies if you have only one display.

Theoretically this wiring sends the data over twice as slow as the winamp or ext8bit wirings, because it only sends 4 bits at a time.

The 4bit connection type supports more than one display connected to the same parallel port. If you want to connect more than one display, then wire the all the displays to the parallel port according to the scheme above with the exception of the EN (pin 6) line of the LCDs.

For the second and further displays, you can find the wiring for the EN (pin 6) line in the table below.

TableÂ 5.6.Â HD44780: 4bit Pinouts (2)

printer port		<->	LCD	
name	**pin**		**name**	**pin**
D7	9	Â	EN_2	6 of 2nd display
D5	7	Â	EN_3	6 of 3rd display
STR	1	Â	EN_4	6 of 4th display
LF	14	Â	EN_5	6 of 5th display
INIT	16	Â	EN_6	6 of 6th display
SEL	17	Â	EN_7	6 of 7th display

The optional keypad can be connected as follows:

TableÂ 5.7.Â HD44780: 4bit Keypad Pinouts

printer port		<->	keypad	remarks
name	**pin**		**pin**	
D0	2	Â	Y_1	Â
D1	3	Â	Y_2	Â
D2	4	Â	Y_3	Â
D3	5	Â	Y_4	Â
D4	6	Â	Y_5	Â
D5	7	Â	Y_6	Only if not used for backlight or 3rd controller.
nSTRB	1	Â	Y_7	Only if not used for additional controllers.
nLF	14	Â	Y_8	
INIT	16	Â	Y_9	
nSEL	17	Â	Y_{10}	
nACK	10	Â	X_1	Â
BUSY	11	Â	X_2	Â
PAPEREND	12	Â	X_3	Â
SELIN	13	Â	X_4	Â
nFAULT	15	Â	X_5	Â

The optional backlight wiring should be connected to D5, pin 7.

8bit "Winamp"

This type of connection should work with winamp.

TableÂ 5.8.Â HD44780: "Winamp" wiring

printer port		<->	LCD	
name	pin		name	pin
D0	2	Â	D0	7
D1	3	Â	D1	8
D2	4	Â	D2	9
D3	5	Â	D3	10
D4	6	Â	D4	11
D5	7	Â	D5	12
D6	8	Â	D6	13
D7	9	Â	D7	14
nSTRB	1	Â	EN	6
nLF	14	Â	RW	5 (EN3 6 - LCD 3) (optional (*))
INIT	16	Â	RS	4
nSEL	17	Â	EN2	(6 - LCD 2) (optional)

(*) on the RW line of the display: this line decides whether the display receives data from the LPT port, or whether it sends data to the LPT port: if grounded it receives, if High or connected to nothing at all it "sends" (i.e., will not work as intended). So, if you are not sure that you need it otherwise, then connect it to GND. This certainly applies if you have only one display.

If you want the display to work with the Winamp plugin, wire nLF (pin 14) to RW of your LCD. You can then use the plugin in bidirectional mode (which is much faster). With 3 connected LCDs this is not possible. Note from Benjamin: I haven't tried using winamp while having the third LCD connected to this line.

The optional keypad can be connected as follows:

TableÂ 5.9.Â HD44780: "Winamp" wiring - Keypad

printer port		<->	keypad
name	pin		pin
D0	2	Â	Y_1
D1	3	Â	Y_2
D2	4	Â	Y_3
D3	5	Â	Y_4
D4	6	Â	Y_5
D5	7	Â	Y_6
D6	8	Â	Y_7
D7	9	Â	Y_8

printer port		<->	keypad
nACK	10	Â	X_1
BUSY	11	Â	X_2
PAPEREND	12	Â	X_3
SELIN	13	Â	X_4
nFAULT	15	Â	X_5

The optional backlight wiring should be connected to nSEL, pin 17.

8bit "lcdtime"

This is originally based on "lcdtime" (by Benjamin Tse <blt@ComPorts.com>) and allows you to combine the LCD with a LED bargraph. The LCD is driven by LCDproc and the LEDs by another program such as portato. Further details can be obtained from:

http://www.ibiblio.org/pub/linux/system/status/lcdtime-0.2.tar.gz
http://www.ibiblio.org/pub/linux/system/status/meter-0.2.tar.gz
http://www.ibiblio.org/pub/linux/system/status/portato-1.2.tar.gz
The LCD connections are:

TableÂ 5.10.Â HD44780: "lcdtime" wiring

printer port		<->	LCD	
name	pin		name	pin
D0	2	Â	D0	7
D1	3	Â	D1	8
D2	4	Â	D2	9
D3	5	Â	D3	10
D4	6	Â	D4	11
D5	7	Â	D5	12
D6	8	Â	D6	13
D7	9	Â	D7	14
nSEL	17	Â	-	Â
nSTRB	1	Â	RS	4
nLF	14	Â	RW	5 (optional - pull LCD RW low (*)
INIT	16	Â	EN	6

(*) on the RW line of the display: this line decides whether the display receives data from the LPT port, or whether it sends data to the LPT port: if grounded it receives, if High or connected to nothing at all it "sends" (i.e., will not work as intended). So, if you are not sure that you need it otherwise, then connect it to GND.

See the lcdtime tar-ball (above) for full details of the bargraph connections.

The optional keypad can be connected as follows:

TableÂ 5.11.Â HD44780: "lcdtime" wiring - keypad

printer port		<->	keypad
name	**pin**		**pin**
D0	2	Â	Y_1
D1	3	Â	Y_2
D2	4	Â	Y_3
D3	5	Â	Y_4
D4	6	Â	Y_5
D5	7	Â	Y_6
D6	8	Â	Y_7
D7	9	Â	Y_8
nSTRB	1	Â	Y_9
nSEL	17	Â	Y_{10} (only if not used for backlight)
nACK	10	Â	X_1
BUSY	11	Â	X_2
PAPEREND	12	Â	X_3
SELIN	13	Â	X_4
nFAULT	15	Â	X_5

The backlight wiring should be attached to nSEL, pin 17. Because the portato program (mentioned above) also uses this pin to control the bargraph, you cannot use the backlight control together with the bargraph.

8bit ("LCM-162" style)

This interface is pretty similar to the 8bit connection type except that the RS, RW and ENABLE signals are wired differently.

This device is usually found directly wired in the Nextgate NSA network appliances, so no soldering is necessary.

Serial LPT

This interface uses a handful of wires to interface to the HD44780. Suitable for high noise, long connections. Designed by Andrew McMeikan <`andrewm@engineer.com`>.

I (Joris) have extended this driver and the wiring a bit. It now supports keys again (it had earlier supported keys, but some time did not).

Further I have extended the driver and the wiring to be able to run using 2 instead of 3 output pins. That's even one less pin ! :)

Of course the use of fewer lines than the other wirings can not stay without drawbacks. In this case the simplicity of the long feeding wires is compensated by some intelligence in the decoding of the data. If you have no experience with the soldering iron, I do not recommend to build this wiring.

OK, so here is the wiring. First of the 'simple' 3 wires version. IC1 is the shift register, a 4094. Do not forget to connect the 5V to pin 16 and GND to pin 8 of the IC.

FigureÂ 5.7.Â HD44780: Serial LPT wiring ('simple')

```
                                     IC1
                                  -----------
                                 |   4094    |
                           5V    | shift reg |                    display
                            O    |           |                            keys
                            |   1|           |4
                            +----|STR      Q0|--------------------o 11 D4   Y1
                            |    |           |5
          Data              |   2|         Q1|--------------------o 12 D5   Y2
  D3 5 o-------------------------|D          |6
                            |    |         Q2|--------------------o 13 D6   Y3
                            |   3|           |7
  D4 6 o-------------------------|CK       Q3|--------------------o 14 D7   Y4
                            |    |           |14
                            |  15|         Q4|--------------------o         Y5
                            +----|OE         |13
                                 |         Q5|--------------------o 4  RS   Y6
                                 |           |12
                                 |         Q6|--------------------o         Y7
                                 |           |11
                                 |         Q7|--------------------o         Y8
                                 |           |9
                                 |         QS|--               +--o 5  RW
                                 |         __|10               |
                                 |         QS|--              ===
                                 |           |
                                  -----------

  D2 4 o----------------------------------------------------------o 6 EN

  D7 9 o----------------------------------------------------------o 6 EN2
                                                                    (2nd LCD)


         5V  O-----+--------+-------------------------------+-----o 2 VCC
                   |        |                               |
                   |        |                               |
                   |100n    O 16                           .-.
                  ---      IC1                             | |<---o 3 Vlcd
                  ---       O 8                            | |10k
                   |        |                              '-'
  GND              |        |                               |
18..25 o-----------+--------+-----------------------+-------+-----o 1 GND
                                                    |
                                                   === GND
```

The second possible wiring is with 2 output lines. This one is a bit more complex. If you do not understand the schematic, do not build it.

FigureÂ 5.8.Â HD44780: Serial LPT wiring ('complex')

```
                                     IC2
                                  -----------
                                 |  74HCT164 |
                                 | shift reg |                    display
                                 |           |                            keys
          Data                  1|           |3
  D3 5 o---------------------+---|D        Q0|--------------------o 11 D4   Y1
                             |   |           |4
                             |  2|         Q1|--------------------o 12 D5   Y2
                             +---|D          |5
                                 |         Q2|--------------------o 13 D6   Y3
                                 |           |6
                                 |         Q3|--------------------o 14 D7   Y4
                                 |           |10
          Clock                 8|         Q4|--------------------o         Y5
  D4 6 o-------------------------|CK         |11
                                 |         Q5|--------------------o 4  RS   Y6
              ___       9|\ 8   9|_          |12
          +--|___|--+----| >o----|R        Q6|--------------------o         Y7
          |   22k   |    |/      |           |13
          |        ---   IC1     |         Q7|---+             +--o 5  RW
          |        ---           |           |   |     5V      |
          |         |100p         -----------    |     O      ===
          |         |                            |     |
          |        ===                           |    .-.
          |                                      |    | |22k
          +--------------------------------------+    | |
          |                                           '-'
          |   ___      11|\ 10                         |   5|\ 6
          +--|___|--+----| >o-------------------||-----+----| >o--o 6 EN
              22k   |    |/                    22p          |/
                   ---   IC1                                IC1
                   ---
                    |22p
                    |            IC1=74HCT14 (6x Schmitt trigger inverter)
                   ===


       5V  O--+-------+------+------+-----------------------+-----o 2 VCC
              |       |      |      | 13|\ 12               |
              |       |      |      +---| >o-               |
              |100n   O 14   O 14       |/                 .-.
             ---     IC1    IC2                            | |<---o 3 Vlcd
             ---      O 7    O 7       1|\ 2      3|\ 4    | |10k
              |       |      |       +--| >o-   +--| >o-   '-'
  GND         |       |      |       |  |/      |  |/       |
18..25 o------+-------+------+-------+----------+-----+-----+-----o 1 GND
                                                      |
                                                     === GND
```

Serial LPT Keypad

Note

To understand this part of the serialLpt documentation, you also need to read the keypad section in this document.

serialLpt wiring supports a keypad. The 3 wires version supports 8 keys, or if you use multiple return lines up to 8 x 5 = 40 lines. The 2 wires version supports 7 keys, or with multiple return lines 7 x 5 = 35 keys.

TableÂ 5.12.Â HD44780: Serial LPT - Keypad return lines

printer port		<->	keypad
name	pin		pin
nACK	10	Â	X_1
BUSY	11	Â	X_2
PAPEREND	12	Â	X_3
SELIN	13	Â	X_4
nFAULT	15	Â	X_5

On lines longer than, say a meter, you should buffer the return line(s). If you only have 1 return line, you can buffer it with two remaining buffers from the 74HCT14:

FigureÂ 5.9.Â HD44780: Serial LPT - Keypad return lines buffered

```
               1|\ 2    13|\ 12    ___
keypad o-----| >o------| >o---|___|---+---o input pin on LPT port
return        |/        |/      220E   |
              IC1       IC1           ---
                                      --- 1nF
                                       |
                                      ===
```

Serial LPT Backlight

Also a backlight is supported. You will also need a port from the 74HCT14 for that. The BL output below should be connected to the BL input in the backlight section

FigureÂ 5.10.Â HD44780: Serial LPT - Backlight extra circuit

```
                ___        3|\ 4
  Data o-----|___|--+----| >o----o BL output
LPT-D3       470k   |     |/
                   ---   IC1
                   ---
```

```
                     |100nF
                     |
                    ===
```

spi

This connection type drives a LCD in serial mode (supported on Hitachi HD66712, Samsung KS0073 and KS0074 (and clones) controllers) connected to some Linux SPI device.

This connection type is currently only supported on Linux. It is strongly recommended to use a SPI implementation with hardware support, e.g. on the Raspberry Pi.

As the SPI communication to the display cannot drive a backlight switch, this connection type features a `BacklightDevice` option that can use a GPIO pin exposed via sysfs (see eLinux GPIO).

FTDI FT2232D USB chip "ftdi"

You can use a FTDI FT2232D dual channel USB <-> parallel FIFO chip to connect a display via the USB bus. The chip is switched to bitbang mode and drives both channels as outputs to control the display in 8bit mode.

TableÂ 5.13.Â HD44780: 8bit FTDI

FTDI chip		<->	LCD	
name	pin		name	pin
ADBUS0	24	Â	D0	7
ADBUS1	23	Â	D1	8
ADBUS2	22	Â	D2	9
ADBUS3	21	Â	D3	10
ADBUS4	20	Â	D4	11
ADBUS5	19	Â	D5	12
ADBUS6	18	Â	D6	13
ADBUS7	17	Â	D7	14
BDBUS0	40	Â	RS	4
BDBUS1	39	Â	RW	5
BDBUS2	38	Â	EN	6
BDBUS3	37	Â	BL	Backlight (optional)

You can configure the USB vendor/product ID in `LCDd.conf`. The wiring of the control lines can optionally be reconfigured, please look at the driver source if you really need that.

Note

The backlight line is driven high when the backlight is `on`. Therefore the standard backlight circuit (FigureÂ 5.6, â HD44780: Backlight Wiringâ) will not work. Use the following instead.

FigureÂ 5.11.Â hd44780/ftdi: Backlight Wiring

```
                                        O 5V
                                        |
                                        +--------o 15 backlight

                                        +--------o 16 GND backlight
                                        |
                      small resistor   .-.
                      10 - 47 ohm      | |
                      depending on     | |
                      display          '-'
                                        |
                                        |c
                     ___            b |/
BL pin o------------|___|-------------|
                     4k7              |\
                                  BC547 |e
                                        |
                                       === GND
```

Alternatively you can use a single channel FTDI FT245BM USB <-> parallel FIFO chip and use the display in its 4 bit mode. If the LCD display you are using has two chip selects, then also configure CE2, and set RW to 0x00. Don't forget to wire RW to ground.

TableÂ 5.14.Â HD44780: 4bit FTDI

FTDI chip		<->	LCD	
name	pin		name	pin
D0	25	Â	D4	11
D1	24	Â	D5	12
D2	23	Â	D6	13
D3	22	Â	D7	14
D4	21	Â	EN	6
D5	20	Â	RS	4
D6	19	Â	RW	5
D6	19	Â	EN2	(alternative)
D7	18	Â	BL	Backlight (optional)

The following special configuration settings are required to use a single channel FTDI FIFO chip:

ExampleÂ 5.1.Â HD44780: Configuration for FTDI 4bit

Note

```
[hd44780]
ConnectionType=ftdi
ftdi_mode=4
ftdi_line_EN=0x10
ftdi_line_RS=0x20
ftdi_line_RW=0x40
ftdi_line_backlight=0x80
#ftdi_line_EN2=0x40
```

LIS2 USB device "lis2"

LIS2 from VLSystem (http://www.vlsys.co.kr) is a full featured USB VFD module with four channel fan controls. This device can be accessed as a serial device with the help of the kernel module `ftdi_sio.ko` that maps the USB port to a serial port (e.g. `/dev/ttyUSB`*x*).

MPlay Blast USB device "mplay"

MPlay Blast from VLSystem (http://www.vlsys.co.kr) is a full featured USB VFD module with two channel fan controls and two channel temperature sensors. This device can be accessed as a serial device with the help of the kernel module `ftdi_sio.ko` that maps the USB port to a serial port (e.g. `/dev/ttyUSB`*x*).

LCD on Serial panel device "los-panel"

The LoS-Panel is a DIY device built using an Atmel ATtiny2313 and supports the following features:

- Drives displays with one controller.
- Switchable backlight.
- One 4x4 matrix keypad and 4 direct keys.

Note

The direct keys are reported as a fifth column of a matrix keypad to LCDd.

Column and rows are reported reverse (column 1 / row 1 is in the lower right corner) to LCDd which expects (1/1) to be the upper left corner. You have to take this into account when configuring keys.

See http://mlf.home.xs4all.nl/los/ for more information on this device.

VDR LCD serial device "vdr-lcd"

â ¦ to be documented â ¦

Please address Matteo Pillon for further information.

VDR-Wakeup module "vdr-wakeup"

The VDR-Wake module by Frank Jepsen is a serial IO extension module for the famous Linux-based VDR that allows to connect an LCD to it and supports LCDproc.

See http://www.jepsennet.de/vdr/ (German) for more information on VDR-Wakeup.

EZIO-100 and EZIO-300 by Portwell

Portwell EZIO-100 and EZIO-300 are 20x2 HD44780-compatible devices featuring a serial port interface to the host on "COM2", and a 4-key keypad. These devices are typically found in firewall appliances by Portwell, Caswell, Check Point and others.

The backlight is always on. The serial port is operating at 2400 bps (2400-N-8-1), which the driver defaults to with this connection type. The keypad keys map to the default KeyMatrix_4_x settings in the [HD44780] section of LCDd.conf.

See http://drivers.portwell.com/CA_Manual/EZIO/EZIO-FINAL.PDF for the EZIO-100 technical document. See http://drivers.portwell.com/CA_Manual/EZIO/EZIO-300.pdf for the EZIO-300 technical document.

Pertelian X2040 "pertelian"

The Pertelian X2040 includes an HD44780 display with enclosure and USB connection. In order to work with LCDproc in Linux you need the `usbserial.ko` and `ftdi_sio.ko` kernel modules loaded. The display will then be available on a serial port `/dev/ttyUSB`*`x`*.

See the X4020 product page for more information.

PIC-an-LCD serial device "picanlcd"

The PIC-an-LCD module is also supported. It is not connected to the LPT port but to a serial port, which saves you from a lot of potential problems. To use it, specify the device to which you have connected the module in the config file with the Device setting. The default is `/dev/lcd`. It does not support a keypad nor backlight switching.

Note

As of 2012 these devices are not sold anymore. Search the Internet Archive for a copy of http://dalewheat.com/pdf/PIC-an-LCD.pdf if you need the manual.

LCD serializer device "lcdserializer"

LCD serializer connection is technically the same as PIC-an-LCD with the same advantages, it uses the serial port making things really simple. Unlike PIC-an-LCD LCD serializer is not a commercial product. it's just a project found digging on the net and freely available. You have all the tools and the code to build it yourself and to customize the behaviour of the device.

What you need

- Some electronic knowledge and familiarity with the soldering iron
- A PIC16F84 (I used PIC16F84A) or PIC16C54
- JDM PIC programmer
- gputils and picprog installed on your GNU/Linux box

Burning the PIC

First, you need to download the ASM source for your PIC and then make the hex:

```
$ gpasm lcd16f84_custom.asm
```

Now the binary is ready to be flashed to the PIC. Connect the programmer with the PIC installed and issue the following command to see it burning ;-):

```
$ picprog --erase --burn --input lcd16f84.hex --pic /dev/ttyS0
```

Running lcdproc

It's time to build the operating circuit, remember this driver uses a baud rate of 9600, so JP2 need to be closed.

Now power on the board. You should see `OK` on the LCD screen. Otherwise double-check all the connections.

Change `LCDd.conf` to include the following statements in the `[hd44780]` section:

```
ConnectionType=lcdserializer
Device=/dev/ttyS0
```

Finally, start the daemon and relax watching lcdproc running.

BWCT USB LCD module "bwctusb"

The BWCT USB LCD module, sold by Bernd Walter Computing Technology (http://www.bwct.de/lcd.html) is a little board that can be piggy-packed to a HD44780 display and connects that to USB.

The board, driven by the `bwctusb` connection type, does neither support a keypad for input, nor more than one single-controller display, nor does it allow setting the backlight or brightness. But you can set the display's contrast using software (see the Contrast configuration parameter).

Special configuration options

If there is more than one BWCT USB module connected to the system, the SerialNumber configuration parameter allows selecting which display is used in LCDd.

Till Harbaum's "lcd2usb"

LCD2USB is a cheap but powerful do-it-yourself interface to connect HD44780 based displays via USB, consisting of easily available parts only. The device supports software adjustable contrast and backlight as well as dual controller displays (required for 4*27 and 4*40).

It is based upon an Atmel AVR Mega8 CPU with a pure software implementation of the USB protocol for the Atmel AVR microcontroller series.

The whole interface incl. the hardware layout is under a GPL like license. This means that you can take these schematics and use it as a basis for your own interface e.g. for a graphic LCD.

Two keys can be connected to the LCD2USB interface board. They can generate three key events that can be mapped to key names using the DirectKey_1 to DirectKey_3 commands: one for each key and the third if the keys are pressed simultaneously. With this 3-key setup, menus can be used (see example below).

This driver supports the original LCD2USB interface board as described above as well as compatible devices like those sold by Lcdmod Kit or those developed by Malte PÃ¶ggel.

Special configuration options

Besides the standard configuration options for `hd44780` displays, the `lcd2usb` connection type supports three additional options: Contrast to set the display's contrast, Brightness to set the display's brightness when the backlight is switched on and OffBrightness to set the display's brightness when is backlight is switched off. All three options expect a number in the range from `0` to `1000`.

ExampleÂ 5.2.Â HD44780: Configuration for LCD2USB

```
[menu]
MenuKey=Escape
EnterKey=Enter
DownKey=Down

[hd44780]
ConnectionType=lcd2usb
Contrast=850
Brightness=800
OffBrightness=0
Keypad=yes
Backlight=yes
Size=20x2
KeyDirect_1=Enter
KeyDirect_2=Down
KeyDirect_3=Escape
```

Tip

In order to make the `lcd2usb` connection type work with a 2-controller display you may need to set the vSpan config option accordingly.

Dick Streefland's "USBtiny"

â USBtiny is a software implementation of the USB low-speed protocol for the Atmel ATtiny microcontrollers.â It is also the name of a 'reference circuit' using the ATtiny2313.

The reference circuit features an IR receiver for remote controls and a LCD. Due to hardware limitations of the ATtiny2313 the LCD does not have switchable backlight, adjustable contrast, any keys, nor does it support displays with more than one controller. If you want these features and do not require the IR receiver we recommend to take a look at the LCD2USB device.

Note

LCDproc does not make use of the IR receiver. 3rd party software is required to make it do anything, e.g. LIRC.

USS-720 USB-to-IEEE 1284 Bridge (Belkin F5U002) "uss720"

The USS-720 USB-to-IEEE 1284 Bridge is a fully featured USB to parallel chip that is used in most (but not all) Belkin F5U002 USB Parallel Printer Adapters. Because these adapters are inexpensive and readily available on the second-hand market, they provide an excellent solution for users who want to experiment with a parallel port but only have USB ports on their computers.

Because the chip acts as a parallel port, the driver maintains the same features and wiring as the 8-bit "winamp" driver. However, because most USB Parallel Printer Adapters use a centronics printer connector, be sure to convert the pin numbering of the parallel port pins in the "winamp" wiring to the pin numbering of the centronics port. Many tables are available on the internet that illustrate how the pin numbering differs between the two.

Special configuration options

Because several manufacturers used the USS720 chip in their USB Parallel Printer Adapters, the VendorID and ProductID options are configurable in the `LCDd.conf` file.

Note

Not all Belkin F5U002 USB Parallel Printer Adapters used the USS720 chip. Look for the dark grey adapters with the removable USB cable for best results.

I^{2}C with Port-Expander

If you have an I^{2}C port available that is supported by your kernel (through `/dev/i2c*`), you can add a I^{2}C port expander there (PCF8574P in this example).

FigureÂ 5.12.Â HD44780: PCF8574P port expander on I^{2}C bus

```
                                   IC1
                              ------------
                              | PCF8574P  |
                              | I2C-Port- |                            HD44780
                              | Expander  |                            display
                              |           |4
                              |         P0|----------------------------o 11 D4
                              |           |5
       I2C-Bus            14|         P1|----------------------------o 12 D5
SCL o-------------------------|SCL        |6
                              |         P2|----------------------------o 13 D6
                            15|           |7
SDA o-------------------------|SDA      P3|----------------------------o 14 D7
                              |           |9
                              |1        P4|----------------------------o 4  RS
       Set I2C-Address  +-----|A0         |10
       here:            |     |2        P5|----------------------------o 5  RW
       GND: Bit:=0      +-----|A1         |11
       VCC: Bit:=1      |     |3        P6|----------------------------o 6  EN
                        +-----|A1         |                      ___
       Here: 0x00       |     |           |                 +---|___|--o 15 backlight
                       ===    |           |                 |c   10R
                              |           |12    ___       b|/
                              |13       P7|-----|___|-----|
                             -|INT        |      1k       |\
                              |           |          bc557 |e
                               -----------                  |
                                                            |
+5V o-----------+------------------+---------------------+--+----------o 2  VCC
                |                  |                     |
                |                  |                     |
                |10uF              O 16                 .-.
               ---                IC1                   | |<-----------o 3  Vlcd
               ---                 O 8                  | |10k
                |                  |                    '-'
                |                  |                     |
GND o-----------+-------+----------+---------------------+-------+-----o 1  GND
                        |                                        |
                       === GND                                   +-----o 16 GND Backli
```

Configuration

ExampleÂ 5.3.Â HD44780: Configuration for I²C with port expander

```
[HD44780]
ConnectionType=i2c
Device=/dev/i2c-0
Port=0x20
Backlight=yes
Size=40x2
DelayBus=false
DelayMult=1
Keypad=no
#i2c_line_RS=0x10
#i2c_line_RW=0x20
#i2c_line_EN=0x40
#i2c_line_BL=0x80
#i2c_line_D4=0x01
#i2c_line_D5=0x02
#i2c_line_D6=0x04
#i2c_line_D7=0x08
# BacklightInvert=yes
# The Backlight Invert is used if a 0 turns the backlight on, and 1 turns it off, i.e. npn
```

If your port expander has different wiring you can re-assign the pins in the config file

ExampleÂ 5.4.Â HD44780: Configuration for I²C with port expander Alternative Wiring

```
-   PCF8574AP: P0 P1 P2 P3 P4 P5 P6 P7
-              |  |  |  |  |  |  |  |
-   HD44780:   RS RW EN BL D4 D5 D6 D7
```

ExampleÂ 5.5.Â HD44780: Configuration for I²C with port expander alternative config

```
[HD44780]
ConnectionType=i2c
Device=/dev/i2c-0
Port=0x20
Size=40x2
DelayBus=false
DelayMult=1
Keypad=no
i2c_line_RS=0x01
i2c_line_RW=0x02
i2c_line_EN=0x04
i2c_line_BL=0x08
i2c_line_D4=0x10
i2c_line_D5=0x20
i2c_line_D6=0x40
i2c_line_D7=0x80
Backlight=yes
BacklightInvert=yes
#The Backlight Invert is used if a 0 turns the backlight on, and 1 turns it off,
i.e. PNP transistor
```

The Device configuration setting denotes the device file of your I²C bus (here `/dev/i2c-0`). You have to load the kernel standard module `i2c-dev.ko` and the bus driver, but *no* I²C chip modules (e.g. `pcf8574.ko`)!

The Port config option contains the I²C address of the I²C port expander (here 0x20, the PCF8574 from the example above, with all address bits set to `0`). Bit 8 of the address (normally `0` in I²C addresses) has a special meaning: It tells the driver to treat the device as PCA9554 or similar, a device that needs a 2-byte command, and it will be stripped off the address.

TableÂ 5.15.Â HD44780: Examples of I²C port expander addresses

Port value	Meaning
`0x20â ¦0x27`	PCF8574 with A[012]=`0â ¦7`
`0x38â ¦0x3f`	PCF8574A with A[012]=`0â ¦7`
`0xa0â ¦0xa7`	PCA9554 with A[012]=`0â ¦7`
`0xa0â ¦0xa7`	PCA9554A with A[012]=`0â ¦7`

ethlcd Device

The ethlcd (http://manio.skyboo.net/ethlcd/) device is simply an LCD display driven by an ATmega microcontroller, controlled and powered by "home-made" Power over Ethernet. The hardware and software are open source.

Features

- ethernet connection using ENC28J60 ethernet controller
- Power over Ethernet (data and power using single UTP cable)

- Atmel's ATmega168 microcontroller
- 6 buttons (MENU, UP, DOWN, LEFT, RIGHT, ENTER)
- backlight control in three states: ON, Night-Mode (partialy ON) and OFF
- beeper
- device act as a TCP server - LCDproc driver is connecting to device just by creating a TCP socket - no need to PC-side additional hardware besides NIC

FigureÂ 5.13.Â HD44780: ethlcd - block diagram

```
 _____________________________________________________
|                    buttons                          |
|                       ^                             |
|                       |                             |
|  _________       ___________        ____________    |
| |         |     |           |      |            |   |
| |         |     |           |      |            |   |
| |   LCD   |     |           | SPI  |  ENC28J60  |   |  UTP cable
| | HD44780 |<--- | ATmega168 | <---> |  ethernet  |<========+------> PC running
| |         |     |           |      | controller |   |       |         LCDproc
| |         |     |           |      |            |   |       |
| |_________|     |___________|      |____________|   |       |
|                       |                             |       |
|                       |                             |       |
|                     beeper                          |       |
|_____________________________________________________|       |
 ethlcd                                                  AC Adapter
```

The device is "visible" to LCDproc just like any other HD44780 device. The difference is the wiring - instead of connecting the display directly to PC (via serial/parallel/usb port), it is connected via ethernet and the communication is done over TCP connection. The main feature is - that to power and control the LCD is needed single UTP cable. To use the driver, specify the device IP address or hostname, on which the ethlcd device is accessible by setting in config file the Device value. The default is `ethlcd`.

USBLCD adapter

The USBLCD adapter from Adams IT Services (http://www.usblcd.de/) is a small interface board which allows you to connect an alphanumerical display module based on the HD44780 or compatible controller to the USB. The display will be powered by the USB. It features a switchable backlight (on or off) and can be used with 16x2, 16x4 or 20x4 displays.

The `usblcd` connection type communicates with a kernel driver by using a device file `/dev/usb/lcdx`. The kernel driver providing this device currently only exists for Linux kernels newer than 2.4.20-pre7.

Note

As of 2007 these device are not sold anymore. This driver has been ported from lcdproc 0.4.5 to support existing users.

USB-4-all controller

The USB-4-all controller board from Sprut (http://www.sprut.de/electronic/pic/projekte/usb4all/usb4all.htm) is a small interface board which allows you to connect different hardware to the USB. The subdriver for LCDproc supports dual HD44780 displays and a 4x4 keypad as well as software adjustable contrast and brightness (backlight).

TableÂ 5.16.Â HD44780: USB-4-all - Display connection

USB-4-all		LCD1	LCD2
name	pin	name	name
RB7	28	D7	D7
RB6	27	D6	D6
RB5	26	D5	D5
RB4	25	D4	D4
RB3	24	R/W	R/W
RB2	23	RS	RS
RB0	21	EN	
RC0	11		EN

TableÂ 5.17.Â HD44780: USB-4-all - Keypad connection

USB-4-all		keypad
name	pin	name
RA0	2	X1
RA1	3	X2
RA2	4	X3
RA3	5	X4
RA4	6	Y1
RA5	7	Y2
RC6	17	Y3
RC7	18	Y4

TableÂ 5.18.Â HD44780: USB-4-all - Brightness and contrast connection

USB-4-all		Desc
name	pin	
RC1	12	brightness
RC2	13	contrast

FigureÂ 5.14.Â HD44780: USB-4-all brightness control

```
           o 5V (PIC Pin 20)
           |
           |e
___     b |/
```

```
RC1 o------|___|-----|    pnp
            10k      |\
                       |c
                       |
                       -
                      | | 10R
                      | |
                       -
                       |
                       +---> LCD 15 (+ Backlight)

                       +---> LCD 16 (- Backlight)
                       |
                      === GND
```

FigureÂ 5.15.Â HD44780: USB-4-all contrast control

```
                        o 5V (PIC Pin 20)
                        |
                        -
                       | |4,7k
                       | |
                        -
                        |     ____
                        +----|____|-----+----> LCD 3 (Contrast)
                        |c    680R      |
            ___     b |/                |+
RC2 o------|___|------|  npn           --- 10uF
            22k       |\               ---
                        |e              |
                        |               |
                        |               |
                       ===             === GND
```

Raspberry Pi

Warning

This sub-driver is obsolete and probably won't get bug and security fixes. Use the gpio connection type instead.

This connection type supports a LCD connected to the GPIO header P1 on a Raspberry Pi. Displays with one or two controllers are supported.

It supports a switchable backlight connected to pin P1-11 by default. Use the switch circuit as described in FigureÂ 5.11, â hd44780/ftdi: Backlight Wiringâ .

Warning

The signal level on the GPIO pins is 3.3 V and they are said not to be 5 V tolerant. Therefore the R/W pin of the display *must* be wired to GND to prevent 5V logic appearing on the GPIO pins. For the same reason the backlight cicuit from FigureÂ 5.6, â   HD44780: Backlight Wiringâ   *must not* be used.

Powering a LCD that uses 5 V from pin P1-02 will work. Powering a LCD from 3.3 V (pin P1-01) will only work if it is designed for 3.3 V operation.

TableÂ 5.19.Â HD44780: Default pin mapping for Raspberry Pi

Connector		<->	LCD	
Pin Name	**Pin Number**		**Name**	**Pin**
GPIO18	P1-12	Â	D7	14
GPIO23	P1-16	Â	D6	13
GPIO24	P1-18	Â	D5	12
GPIO25	P1-22	Â	D4	11
GPIO08	P1-24	Â	EN	6
GPIO07	P1-26	Â	RS	4
GND	P1-06	Â	RW	5
GPIO22	P1-15	Â	EN2	Second controller (optional)
GPIO17	P1-11	Â	BL	Backlight (optional)

GPIO07 and GPIO08 used for RS and EN signal are also used by the SPI bus. If you loaded the SPI driver you will need to assign them to a different GPIO, e.g. GPIO04 and GPIO22.

When using a display with two controllers the `vspan` option has to be configured as well (e.g. `vspan=2,2` for a 40x4 display).

Special Raspberry Pi configuration options

This connection driver can be configured to use other pins than the one described in the table above. Using the `pin_<LCD pin name>` configuration option, any LCD pin can be assigned to any GPIO pin.

Important

The values for the `pin_<LCD pin name>` configuration option are the number part from the GPIO signal name, not the pin number from the header connector! To find out which signal is routed to which connector pin refer to the RPi Low-level peripherals description.

The connection driver contains a check for possible signal names. You will receive an error message if you try to assign a pin to a signal that is not available on connector P1 or P5.

ExampleÂ 5.6.Â HD44780: Alternate configuration for Raspberry Pi GPIO pins

```
[hd44780]
ConnectionType=raspberrypi
```

```
        Backlight=yes
        Size=16x2

        pin_D4=25
        pin_D5=24
        pin_D6=23
        pin_D7=18
        pin_EN=8
        pin_RS=7
        pin_BL=17
```

Adafruit Pi Plate

Warning

This sub-driver is obsolete and probably won't get bug and security fixes. Use the gpio connection type together with the gpio-mcp23s08 kernel module.

The Adafruit RGB Positive 16x2 LCD+Keypad for Raspberry Pi (aka Pi Plate, see (http://www.adafruit.com/products/1109)) combines a 16x2 Character LCD, up to 3 backlight pins AND 5 keypad pins using only the two I^{2}C pins on the Rasperry Pi.

Note that this connection type drives all RGB pins of the backlight at the same time, resulting in a white backlight. You will need to modify the source code if you want to have some other color.

ExampleÂ 5.7.Â HD44780: Configuration example for Pi Plate connection type

```
        [hd44780]
        ConnectionType=piplate
        Device=/dev/i2c-1
        Port=0x20
        Size=16x2
        Backlight=yes

        Keypad=yes
        KeyDirect_1=Enter
        KeyDirect_2=Up
        KeyDirect_3=Down
        KeyDirect_4=Left
        KeyDirect_5=Right
```

PiFace Control and Display

Warning

This sub-driver is obsolete and probably won't get bug and security fixes. Use the gpio connection type together with the gpio-mcp23s08 kernel module.

The PiFace Control and Display for Raspberry Pi from OpenLX SP Ltd combines a 16x2 Character LCD and navigation buttons using the SPI bus on the Rasperry Pi.

Note

The PiFace Control and Display features a IR receiver, which LCDproc does not make use of. You have to setup and use LIRC for that. See Setting up PiFace Control And Display to use a remote.

ExampleÂ 5.8.Â HD44780: Configuration example for PiFace Control and Display connection type

```
[hd44780]
ConnectionType=pifacecad
Device=/dev/spidev0.1
Size=16x2
Backlight=yes

Keypad=yes
KeyMatrix_1_1=Left
KeyMatrix_1_2=Down
KeyMatrix_1_3=Up
KeyMatrix_1_4=Right
KeyMatrix_1_5=Escape
KeyMatrix_1_6=Enter
KeyMatrix_1_7=Left
KeyMatrix_1_8=Right
```

GPIO, UGPIO, GPIOD

This connection types support an LCD connected to GPIO lines that can be controlled via the linux kernel interface. This is common for System-On-Chip (SOC) processors and i2c port expanders. Currently it supports displays with one or two controllers.

The UGPIO connection type uses the deprecated sysfs interface via libugpio. This is only kept around to support legacy systems, where the kernel can't be updated easily.

The GPIOD connection type uses the official gpio-device API via libgpiod. Currently all GPIO lines must belong to the same chip. You should use this if possible.

The GPIO connection type has been removed from the HD44780 driver. For backwards compatibility the driver tries to use UGPIO instead. Don't use this when writing new configuration files.

A GPIO pin can be configured for backlight switch. Use the switch circuit as described in FigureÂ 5.11, â hd44780/ftdi: Backlight Wiringâ .

Warning

The configuration option for R/W pin of the display is optional. If the pin is configured, it is always held in low state since the driver always performs write operations and keeps the direction of data pins as output. This is what you need for i2c port expanders. If you omit configuration of R/W pin (maybe you are manually wiring the display to a SoC), the R/W pin *must* be wired to GND. This is also necessary to prevent 5V logic appearing on I/O pins which are not 5V tolerant.

Note

The UGPIO driver exports the configured GPIOs to sysfs automatically, before using them. At exit it tries to unexport them again, but this only succeeds if the daemon is still running with root permissions. If the daemon was configured to drop root privileges after initialization, the GPIOs will remain accessible from sysfs and unloading the respective kernel module will fail.

TableÂ 5.20.Â HD44780: Example pin mapping for Beaglebone Black

Connector		<->	LCD	
Pin Name	GPIO Number		Name	Pin
P8_36	80	Â	D7	14
P8_34	81	Â	D6	13
P8_40	77	Â	D5	12
P8_39	76	Â	D4	11
P8_26	61	Â	EN	6
P8_18	65	Â	RS	4
GND	Â	Â	RW	5
P8_14	26	Â	EN2	Second controller (optional)
P8_15	47	Â	BL	Backlight (optional)

Special GPIO configuration options

The GPIO connection driver can be configured to assign any LCD pin to any GPIO pin using the `pin_`*`<LCD pin name>`* configuration option.

When using a display with two controllers the `vspan` option has to be configured as well (e.g. `vspan=2,2` for a 40x4 display).

ExampleÂ 5.9.Â HD44780: Example configuration for Beaglebone Black GPIO pins with libugpio

```
[hd44780]
ConnectionType=ugpio
Backlight=yes
Size=16x2

pin_D4=76
pin_D5=77
pin_D6=81
pin_D7=80
pin_EN=61
pin_RS=65
pin_BL=47
```

ExampleÂ 5.10.Â HD44780: Example configuration for Raspberry Pi 4 GPIO pins with libgpiod

```
[hd44780]
ConnectionType=gpio

gpiochip=0
pin_D4=25
pin_D5=24
pin_D6=23
pin_D7=18
pin_EN=8
pin_RS=7
#pin_BL=xx
```

Compiling

Make sure that the HD44780 files are built when you run **./configure**. This can be done by specifying `--enable-drivers=all` or by including `hd44780` in the list of enabled drivers (e.g. `--enable-drivers=hd44780`).

Configuration in LCDd.conf

[hd44780]

Port = *`PORT`*
: For parallel connections, specify the address of the parallel port the LCD is connected to. Common values for *`PORT`* are `0x278`, `0x378` and `0x3BC`. If not given, the default is `0x378`.

  For I²C connection types this sets the slave address.

Device = *`DEVICE`*
: If you are using a serial or I²C connection, you need to set this parameter to the device your LCD is connected to. For example, if the display is connected to the first serial port, you have to set it to `/dev/ttyS0`. The default value is `/dev/lcd`.

ConnectionType = { *`4bit`* | *`8bit`* | *`serialLpt`* | *`winamp`* | *`lcm162`* | *`picanlcd`* | *`lcdserializer`* | *`los-panel`* | *`vdr-lcd`* | *`vdr-wakeup`* | *`pertelian`* | *`lis2`* | *`mplay`* | *`usblcd`* | *`bwctusb`* | *`lcd2usb`* | *`usbtiny`* | *`uss720`* | *`usb4all`* | *`ftdi`* | *`i2c`* | *`piplate`* | *`spi`* | *`pifacecad`* | *`ethlcd`* | *`raspberrypi`* | *`gpio`* | *`ezio`* }
: Select the type of the wiring / display connection. See also (TableÂ 5.1, â HD44780: Connection Typesâ).

Model = { *`standard`* | *`extended`* | *`winstar_oled`* | *`pt6314_vfd`* }
: Some devices (ie. WINSTAR WEH001602A OLED or PTC PT6314 VFD) require additional initialization or configuration incompatible with "classic" LCD or just have extra features, such as internal commands for brightness handling. If you have such display setting this option may help in problems with initialization or adds extra functionality.

  ◊ The default, `standard` or `default`, is for "classic" HD44780 displays.
  ◊ `extended`, `hd66712` or `ks0073` is for displays which have "extended mode" turned on with special instruction. If you have a Samsung KS0073, PowerTip Corp. PC2004LRU, other based on HD66710/HD66712 chip or an other 'almost HD44780-compatible', set `Model` to this value to get into extended, 4-line linear addressing mode.

Use this instead of deprecated option `ExtendedMode`.

◊ `winstar_oled` or `weh00xxyya` is for WINSTAR WEH00xxyyA OLED displays and allows performing proper initialization of display device, especially after display reset without powering it off.

It also allows handling backlight setting using internal commands for such displays.

◊ `pt6314_vfd` is for some Princeton Technology Corp.'s PT6314 VFDs and allows handling brightness of display.

You only may need to set this parameter if you have a non-standard HD44780 display such as specified above and have problems with it and/or want reveal extra functionality with these displays.

This option should be independent of connection type.

Speed = *`BITRATE`*

For a serial connection, set to the serial port bitrate. To use the default value for the chosen interface, just set to 0.

CharMap = { *`hd44780_default`* | *`hd44780_euro`* | *`ea_ks0073`* | *`sed12780f_0b`* | *`hd44780_koi8_r`* | *`hd44780_cp1251`* | *`hd44780_8859_5`* | *`upd16314`* | *`weh001602a_1`* | *`none`* }

Set the character mapping depending on the display you have:

◊ The default, `hd44780_default`, is for "classic" HD44780 displays.

◊ `hd44780_euro` is for displays with a ROM mask supporting the european charset (ROM code A02).

◊ `ea_ks0073` is the charmap for Electronic Assembly's KS0073 based displays. These devices have a richer charset, including many icons and many more characters of the ISO-8859-1 than standard HD44780s.

◊ `sed12780f_0b` is for some SED 1278 displays.

◊ The `none` charmap does not translate any characters. It displays the characters the display controllers actually has stored in its CGROM for that position instead. This setting is intended for debugging purpose.

You only need to set this parameter if you have a non-standard HD44780 display or charmap.

If LCDproc was configured with '--enable-extra-charmaps' option the following character mappings are available, too:

◊ `hd44780_koi8_r` maps input from a client in Russian KOI8-R to displays with a ROM mask supporting the european charset (ROM code A02).

◊ `hd44780_cp1251` maps input from a client in Russian CP1251 (Windows-1251) to displays with a ROM mask supporting the european charset (ROM code A02).

◊ `hd44780_8859_5` maps input from a client in Russian ISO 8859-5 to displays with a ROM mask supporting the european charset (ROM code A02).

◊ `upd16314` is for displays with a Nec uPD16314 vacuum fluorescent display (VFD) controller with ROM code 002 character set. If your display has ROM code 001 character set you may use the `hd44780_euro` charmap instead.

◊ `weh001602a_1` is for displays such as the WINSTAR WEH001602A OLED display with bank font 1 (Western Europe I) selected. Clients should use ISO 8859-1. See `FontBank` to select the font bank.

Tip

See `server/drivers/hd44780-charset.h` in LCDproc's source directory for the actual mappings.

FontBank = { *0* | *1* | *2* | *3* }
: For some displays such as the WINSTAR WEH001602A, set the font bank to be used.

  ◊ `0` (default) is the English/Japanese font bank.
  ◊ `1` is the Western Europe I font bank. You can use the following `CharMap` with this font bank: `weh001602a_1`.
  ◊ `2` is the English/Rusian font bank.
  ◊ `3` is the Western Europe II font bank.

  You only need to set this parameter if you have a HD44780 display such as the WINSTAR WEH001602A, which allows selecting the font bank during initialization. Additionally you may need to set correct `CharMap`.

Note

Setting this to nonzero value on standard HD44780 is usually harmless, but some non-standard displays may use bits used for `FontBank` selection, so it is safer to leave this option at default value for displays not supporting Font bank.

Keypad = { *yes* | *no* }
: Tell whether you have a keypad connected. You may also need to configure the keypad layout further on in this file.

Brightness = *BRIGHTNESS*
: Set the initial brightness when the backlight is on for the `lcd2usb` connection type. Legal values are `0` - `1000`, with `800` being the default.

OffBrightness = *BRIGHTNESS*
: Set the initial off-brightness, i.e. the brightness when the backlight is off, for the `lcd2usb` connection type. The legal range is `0` - `1000`. If not given, it defaults to `300`.

Contrast = *CONTRAST*
: Set the initial contrast for the `bwctusb` and `lcd2usb` connection types. Legal values for *CONTRAST* are `0` - `1000`. If not given, it defaults to `500` which may be too low or too high for the selected connection type. So, if the screen is blank or dark, please try playing with the contrast a bit.

Backlight = { *none* | *external* | *internal* | *internalCmds* }
: Specify if you have a switchable backlight and if yes, can select method for turning it on/off:

  ◊ `none` - no switchable backlight is available. For compability also boolean `0`, `n`, `no`, `off` and `false` are aliases.
  ◊ `external` - use external pin or any other method defined with `ConnectionType` backlight handling. For backward compability also this value is chosen for boolean TRUE values: `1`, `y`, `yes`, `on` and `true`.
  ◊ `internal` means that backlight is handled using internal commands according to selected display model (with `Model` option). Depending on model, `Brightness` and `OffBrightness` options can be taken into account.
  ◊ `internalCmds` means that commands for turning on and off backlight are given with extra options `BacklightOnCmd` and `BacklightOffCmd`, which would be treated as catch up (last resort) for other types of displays which have similar features.

  You can provide multiple occurences of this option to use more than one method. This can be useful for example for glowing buttons with external pin when backlight is on and using internal command to set high brightness of display.

Default is model specific (depends on `Model` option): Winstar OLED and PT6314 VFD enables `internal` backlight mode, for others it is set to `none`.

BacklightCmdOn = *`COMMAND(s)`*
: Commands for enabling internal backlight for use with `Backlight=internalCmds`. Up to 4 bytes can be encoded, as integer number (hex) in big-endian order. Ignored if `Backlight` does not specify `internalCmds`, required otherwise.

Note

This is advanced option, if command contains bits other than only brighness handling, they must be set accordingly to not disrupt display state. If for example 'FUNCTION SET' command is used for this purpose, bits of interface length (4-bit / 8-bit) must be set according to selected ConnectionType.

BacklightCmdOff = *`COMMAND(s)`*
: Commands for disabling internal backlight for use with `Backlight=internalCmds`. Up to 4 bytes can be encoded, as integer number (hex) in big-endian order. Ignored if `Backlight` does not specify `internalCmds`, required otherwise. See above note about `BacklightCmdOn`.

OutputPort = { *`yes`* | *`no`* }
: Tell if you have the additional output port ("bargraph") and you want to be able to control it with the lcdproc OUTPUT command.

Lastline = { *`yes`* | *`no`* }
: Specifies whether the lowest pixel line of a character is pixel addressable or if it controls an underline effect. The default is `yes`, meaning a pixel addressable last pixel line.

Size = *`WIDTH x HEIGHT`*
: Specifies the size of the LCD. Default: `20x4` In case of multiple combined displays, this should be the total size.

vSpan = *`HEIGHT ,â ¦`*
: The "vertical span" when using the driver with multi-controller displays or with multiple displays that are treated as a single virtual display. It is a comma separated list of the heights of each display. In multi-controller displays it lists the number of lines each controller is responsible for.

E.g. `vSpan=2,2,1` means you have three physical displays, the first two having two lines each, and the third having one line, that together form a virtual display that is 5 lines high.

The sum of the *`HEIGHT`*s must match the total height given in `Size=`.

ExtendedMode = { *`yes`* | *`no`* }
: If you have a KS0073 or an other 'almost HD44780-compatible', set this flag to get into extended,4-line linear addressing mode.

Note

Deprecated, use `Model=extended` for such displays.

LineAddress = *`ADDR`*
: If the next line of your display doesn't start `0x20` higher in DDRAM you can override the default value of the ExtendedMode with this parameter.

DelayMult = *`DELAY`*
: If your display is slow and cannot keep up with the flow of data from LCDd, garbage can appear on the LCDd. Set this delay multiplier to `2` or `4` to increase the delays. The default is `1` for a non-multiplied delay.

DelayBus = { *`yes`* | *`no`* }

You can reduce the inserted delays by setting this to `no`. On fast PCs it is possible your LCD does not respond correctly. Default: `yes`.

KeepAliveDisplay = *`SECONDS`*
: Some displays (e.g. `vdr-wakeup`) need a message from the driver to indicate that it is still alive. When set to a value greater than `0` the character in the upper left corner is updated every *`SECONDS`* seconds. The default `0` does not cause any extra updates.

RefreshDisplay = *`SECONDS`*
: If you experience occasional garbage on your display you can use this option as workaround. If set to a value greater than `0` it forces a full screen refresh every *`SECONDS`* seconds. Default: `0`.

KeyDirect_*`NUM`* = *`KEY`* , KeyMatrix_*`X`*_*`Y`* = *`KEY`*
: If you have a keypad you can assign keystrings to the keys. See the keypad section for used terms and the section on the specific connection type how to wire it.

  To map, for example, the directly connected key `4` to the string `Enter`, use `KeyDirect_4=Enter`. For matrix keys use the *`X`* and *`Y`* coordinates of the key; e.g. `KeyMatrix_1_3=Enter`.

VendorID = *`VENDORID`*
: USB vendor ID to look for in certain USB connection types. When using an FTDI chip with connection type `ftdi`, the default value is `0x4003`. When using a USS720 chip with connection type `uss720`, the default value is `0x1293`.

ProductID = *`PRODUCTID`*
: USB product ID to look for in certain USB connection types. When using an FTDI chip with connection type `ftdi`, the default value is `0x6001`. When using a USS720 chip with connection type `uss720`, the default value is `0x0002`.

UsbDescription = *`DESCRIPTION`*
: USB Description string to look for in FTDI connection types. If not given, the first USB device will be used. If given, the first matching device will be used.

SerialNumber = *`SERIALNO`*
: Serial number of the USB device to look for with `bwctusb` and `FTDI` connection types. If not given, the first BWCT/FTDI USB device found will be used.

Miscellanea

This text has originally been taken from a message by Bill Farrow <`bfarrow@arrow.bsee.swin.edu.au`>.

Updated February 2000, Benjamin Tse <`blt@ComPorts.com`>

Updated October 2001, Joris Robijn <`joris@robijn.net`>

Converted to Docbook March 2002, Rene Wagner <`reenoo@gmx.de`>

Updated April 2002, Rene Wagner <`reenoo@gmx.de`>

Updated and extended April 2006 to November 2007, Peter Marschall <`peter@adpm.de`>

The i2500vfd Driver

This section talks about using LCDproc with an Intra2net Intranator 2500 VFD display. The displays are part of a custom high grade steel chassis and not sold separately. More information can be found here: Intra2net

AG

It connects to USB using a FTDI FT245BM chip and therefore needs libftdi as requirement. libftdi can be found here: libftdi

Features

The display features a 140x32 pixel Noritake VFD tube which gets driven by an Atmel ATmega128 processor and used as a 23x4 character display with a 6x8 pixel font. It features B/W colors and two additional adjustable grayscale colors. It does hardware double buffering with up to 27 FPS and shows an animated boot logo until data is received from USB.

Options

There are currently no options for this driver.

The icp_a106 Driver

This section talks about using LCDproc with the ICP Peripheral Communication Protocol used by A125 and A106 LCD displays

Both LCD and alarm functions are accessed via one serial port, using separate commands. Unfortunately, the device runs at slow 1200bps and the LCD does not allow user-defined characters, so the bargraphs do not look very nice. The A125 does include two buttons marked ENTER and SELECT. Short press of ENTER: ENTER Long press of ENTER: ESC Short press of SELECT: DOWN Long press of SELECT: UP

Configuration in LCDd.conf

[icp_a106]

Device = `DEVICE`
: Sets the device to use. Defaults to `/dev/lcd`.

Size = `SIZE`
: Sets the display resolution to use. Defaults to 20x2

The imon Driver

General

The `imon` driver controls Soundgraph iMON VFD devices, that are either preinstalled or available as optional accessories for a variety of Home Theater PC (HTPC) cases from Ahanix, Silverstone, Cooler Master and others. They can also be bought separately and then fit into a 5,25" disk drive bay of any regular PC.

The iMON VFD sports a vacuum fluorescent display with 16x2 characters that connects to the computer using USB. Although the device is shipped with an IR remote control and some versions even have a volume knob, LCDproc's driver currently only supports the display part of the device.

In order to be able to use it, you have to get and install one of the following Linux kernel modules:

- standalone iMON VFD driver from http://venky.ws/projects/imon/
- the iMON module included with LIRC ver. 0.7.1 or newer from the LIRC project

For further details, please consult the page and the forum at http://venky.ws/projects/imon/.

Configuration in LCDd.conf

[imon]

Size = *WIDTH x HEIGHT*
: Set the display size. The default `16x2` should be safe for most if not all users, since the device seems to be made only with this one size. But who knows â ¦

Device = *DEVICE*
: Select the output device to use. Change this from the default `/dev/lcd` to the device file that gets created when the kernel module (see above) is loaded.

CharMap = { *hd44780_euro* | *hd44780_koi8_r* | *hd44780_cp1251* | *hd44780_8859_5* | *upd16314* | *none* }
: Set the character mapping depending on the display you have:

  ◊ The default, `hd44780_euro` is for displays with a ROM mask supporting the european charset (ROM code A02).
  ◊ The `none` charmap does not translate any characters and is only useful for debugging.

  You only need to set this parameter if you have a non-standard charmap.

  If LCDproc was configured with '--enable-extra-charmaps' option the following character mappings are available, too:

  ◊ `hd44780_koi8_r` maps input from a client in Russian KOI8-R to displays with a ROM mask supporting the european charset (ROM code A02).
  ◊ `hd44780_cp1251` maps input from a client in Russian CP1251 (Windows-1251) to displays with a ROM mask supporting the european charset (ROM code A02).
  ◊ `hd44780_8859_5` maps input from a client in Russian ISO 8859-5 to displays with a ROM mask supporting the european charset (ROM code A02).
  ◊ `upd16314` is for displays with a Nec uPD16314 vacuum fluorescent display (VFD) controller with ROM code 002 character set. If your display has ROM code 001 character set you may use the `hd44780_euro` charmap instead.

Tip

See `server/drivers/hd44780-charset.h` in LCDproc's source directory for the actual mappings.

The imonlcd Driver

General

This section talks about using LCDproc with LCD devices manufactured by SoundGraph. For example, the iMON OEM LCD.

This driver currently supports versions 15c2:ffdc and 15c2:0038 of the device. (You can find the version of your LCD via the lsusb command).

In many systems, the LCD backlight will remain on after the system is shutdown. This behavior remains a mystery - somehow the LCD receives a reset command (or similar) AFTER LCDd is stopped.

This driver requires the iMON module included with LIRC v0.8.4a or newer, available from the LIRC project. The 15c2:0038 device may require LIRC v0.8.5 or newer.

Configuration in LCDd.conf

[imonlcd]

Protocol = *`PROTOCOL`*
: Specify which version of iMON LCD is installed. The default, `0` specifies the :ffdc device. `1` should be used for the :0038 device.

OnExit = *`ONEXIT`*
: Specify the exit behavior. The default is `1`, which turns on the big ugly clock upon shutdown. `0` leaves the shutdown message on the screen. `2` turns the LCD off.

Device = *`DEVICE`*
: Select the output device to use. Change this from the default `/dev/lcd0` to the device file that gets created when the kernel module (see above) is loaded.

Contrast = *`CONTRAST`*
: Select the display's contrast `200` is the default. Permissible values are in the range of `0-1000`.

Size = *`WIDTH x HEIGHT`*
: Set the display size in pixels. The default `96x16` should be safe for most if not all users, since the device seeems to be made only with this one size.

Backlight = *`BACKLIGHT`*
: Set the backlight state. The default is `1`, which turns the backlight on. `0` turns the backlight off.

DiscMode = *`DISCMODE`*
: Sets the disc mode. The default is `0` which spins the "slim" disc. `1` spins their complement.

The IOWarrior Driver

General

IOWarrior is the name of a series of multi-purpose USB controller chips produced and sold by Code Mercenaries. This series currently consists of three main types, that - among other features - support controlling LCD displays:

IOWarrior24
- ◊ USB 1.1 Low Speed
- ◊ 16 generic I/O Pins, typ. 125Hz read rate
- ◊ I^{2}C master function, 100kHz, throughput typ. 750 bytes/sec
- ◊ SPI master interface, up to 2MHz, throughput typ. 750 bytes/sec
- ◊ control an HD44780 compatible LCD
- ◊ drive a matrix of up to 8x32 LEDs
- ◊ decode RC5 compatible infrared remote controls

IOWarrior40
- ◊ USB 1.1 Low Speed

- ◊ 32 generic I/O Pins, typ. 125Hz read rate
- ◊ I^{2}C master function, 100kHz, throughput typ. 750 bytes/sec
- ◊ control an HD44780 compatible LCD
- ◊ drive a matrix of up to 8x32 LEDs
- ◊ drive a 8x8 switch or button matrix

IOWarrior56
- ◊ USB 1.1 Full Speed
- ◊ 50 generic I/O Pins, typ. 1000Hz read rate
- ◊ I^{2}C master function, 50, 100, or 400kHz
- ◊ SPI master interface, up to 12MHz, throughput up to 62Kbytes/sec
- ◊ control various display modules, including most graphic modules
- ◊ drive a matrix of up to 8x64 LEDs
- ◊ drive a 8x8 switch or button matrix

The `IOWarrior` driver currently only supports writing to a single-controller HD44780-type display. LED output using the **output()** function is also implemented, although not tested very well. The hardware's support for input using keys or IR and dual-controller displays is not implemented yet. Please note that the latter requires extra circuitry with IOWarrior24 and IOWarrior40.

The driver was developed and tested with IOWarrior24 and the IOWarrior40. Although there are good chances for it to work with an IOWarrior56, the current state regarding support of this chip is unknown due to the lack of the required hardware.

Requirements

The driver is based on the `libusb` USB library, which should make it work with Linux, the different BSD variants as well as Darwin/MacOS X.

Note

When using a `libusb` based driver like `IOWarrior`, LCDd needs to be started as root.

Note

Newer Linux kernels (2.6.20 and higher) provide a kernel module `iowarrior.ko` that allows controlling IOWarrior chips using device files. LCDd tries to unload this kernel module for `libusb` to be able to control IOWarrior devices. If this fails, this may hinder LCDd from starting using the `IOWarrior` driver. In this case, simply unload the kernel module by hand.

Configuration in LCDd.conf

[IOWarrior]

Size = *`WIDTH x HEIGHT`*
: Set the display dimensions. If not given, it defaults to `20x4`.

SerialNumber = *`SERIALNO`*
: Use the IOWarrior module with the serial number *`SERIALNO`*. If this parameter is missing, the default is to use the first IOWarrior module found.

ExtendedMode = { *`yes`* | *`no`* }

If you have a KS0073 or an other 'almost HD44780-compatible' display connected to the IOWarrior, set this flag to get into extended, 4-line linear addressing mode

Lastline = { `yes` | `no` }
: Specifies if the last line is pixel addressable or it controls an underline effect. The default `yes` means, it is pixel addressable.

The irman Driver

The `irman` driver allows you to use the IrMan IR remote control to control the LCDproc server LCDd and/or clients that can handle input.

The keys are mapped according to the following table:

TableÂ 5.21.Â Mapping between LCDproc keys and IrMan commands

LCDproc key	IrMan command
`Up`	lcdproc-Up
`Down`	lcdproc-Down
`Left`	lcdproc-Left
`Right`	lcdproc-Right
`Enter`	lcdproc-Enter
`Escape`	lcdproc-Escape

Tip

If you have trouble using the `irman` driver, you might try the `lirc` driver. lirc supports IrMan as well.

Configuration in LCDd.conf

[IrMan]

Device = *`DEVICE`*
: Select the input device to use, e.g. `/dev/irman`.

Config = *`FILENAME`*
: Select the IrMan configuration file to use, e.g. `/etc/irman.cfg`.

The irtrans Driver

General

The `irtrans` driver controls IRTrans VFD devices, that are preinstalled in cases such as the Ahanix MCE303.

The IRTrans VFD sports a vacuum fluorescent display with 16x2 characters that connects to the computer using USB. Although the device is shipped with an IR remote control, LCDproc's driver currently only supports the display part of the device.

In order to be able to use it, you have to get and install the IRTrans irserver package from http://www.irtrans.de/en/download/linux.php.

Configuration in LCDd.conf

[irtrans]

Backlight = { *yes* | *no* }
: Tell whether the device has a backlight, or whether the backlight shall be used. If not given, it defaults to `no`.

Hostname = *HOSTNAME_OR_IP-ADDRESS*
: Set the hostname or IP address of the IRTrans device to connect to. If not set or set, the default is `localhost`.

Size = *WIDTH x HEIGHT*
: Select the display size [default: `16x2`]

The Joystick Input Driver

This section covers the joystick input driver for LCDd.

Configuration in LCDd.conf

[joy]

Device = *DEVICE*
: Select the input device to use [default: `/dev/js0`]

Map_Axis*NUM*neg = *KEY* , Map_Axis*NUM*pos = *KEY*
: Set the axis map.

  NUM is an integer starting with `1` that represents each axis with the affixes `neg` and `pos` determining the direction. The exact numbering of the axes depends on the hardware used.

  KEY can be one of the keys that LCDd recognizes (`Left`, `Right`, `Up`, `Down`, `Enter` or `Escape`) or any other string that a client can parse.

Map_Button*NUM* = *KEY*
: Set the button map.

  NUM is an integer starting with `1` that represents each button. The exact numbering of the buttons depends on the hardware used.

  KEY can be one of the keys that LCDd recognizes (`Left`, `Right`, `Up`, `Down`, `Enter` or `Escape`) or any other string that a client can parse.

The lb216 Driver

Chris Debenham

<`chris.debenham@aus.sun.com`>

This section talks about using LCDproc with LCD displays that use the lb216 chipset.

```
Heres a bit more info on the display.
It is the LB216 and is made by R.T.N. Australia
The web page for it is http://www.nollet.com.au/
It is a serial 16x2 LCD with software controllable backlight.
They also make 40x4 displays (which I'll be getting one of soon :-) )
3 wire connection (5V,0V and serial), 2400 or 9600 bps.
8 custom characters
40*83.5MM size
made in australia :-)
```

Configuration in LCDd.conf

[lb216]

Device = *`DEVICE`*
: Select the output device to use [default: `/dev/lcd`]

Brightness = *`BRIGHTNESS`*
: Set the initial brightness [default: `255`; legal: `0` - `255`]

Speed = { *`2400`* | *`9600`* }
: Set the the baud rate to use when communicating with the LCD. If not given, the default is `9600`.

Reboot = { *`yes`* | *`no`* }
: Reinitialize the LCD's BIOS [default: `no`; legal: `yes`, `no`]

The lcdm001 Driver

This section talks about using LCDproc with serial LCD displays from kernel concepts.

Configuration in LCDd.conf

[lcdm001]

Device = *`DEVICE`*
: Default: `/dev/lcd`

PauseKey = *`KEY`* , BackKey = *`KEY`* , ForwardKey = *`KEY`* , MainMenuKey = *`KEY`*
: keypad settings

key name	function (normal context)	function (menu context)
PauseKey	Pause/Continue	Enter/select
BackKey	Back(Go to previous screen)	Up/Left
ForwardKey	Forward(Go to next screen)	Down/Right
MainMenuKey	Open main menu	Exit/Cancel

You can rearrange the settings here.

If your device is broken, have a look at server/drivers/lcdm001.h

The lcterm Driver

This section talks about using LCDproc with serial LCD displays from Helmut Neumark Elektronik.

Configuration in LCDd.conf

[lcterm]

Device = `DEVICE`
: Default: `/dev/lcd`

Size = `WIDTH x HEIGHT`
: Default: `16x2`.

The Linux Event Device Input Driver

This section covers the linux event device input driver for LCDd.

Configuration in LCDd.conf

[linux_input]

Device = `DEVICE`
: Select the input device to use [default: `/dev/input/event0`]. This may be either an absolute path to the input node, starting with '/', or an input device name, e.g. "Logitech Gaming Keyboard Gaming Keys".

key = `KEYCODE,KEY`
: Set an alternate key map, e.g. to use custom buttons instead of the standard codes for Escape, Enter, Left, Right, Up and Down. This entry typically repeaded for any non-standard key code.

: `KEYCODE` is an integer like the ones defined in `/usr/include/linux/input.h`. This may be specified as either a decimal number, or as a hexadecimal number prefixed with 0x. You can also find the key code in the log output of LCDd when the `ReportLevel` is at least 4.

: `KEY` can be one of the keys that LCDd recognizes (`Left`, `Right`, `Up`, `Down`, `Enter` or `Escape`) or any other string that a client can parse.

The lirc Driver

The `lirc` driver enables you to use any IR remote control that works with LIRC to control the LCDproc server LCDd and/or clients that can handle input.

Of course you need a working LIRC setup. Refer to the LIRC project for more information on LIRC itself.

Checking Your LIRC Setup

Basically all you need is a running lircd. And of course you have to start lircd as root.

Also, make sure that the permission of `/dev/lircd` are correct.

Build LCDd with the lirc Driver

You need to add lirc to the --enable-drivers=... list.

Then simply run make.

Configure LCDd to Use the lirc Driver

First of all you need to activate the driver by adding a `Driver=lirc` line to your `LCDd.conf`

ExampleÂ 5.11.Â `LCDd.conf`: Activate the lirc driver

```
Driver=mtxorb
Driver=lirc
```

This activates the `mtxorb` driver as the output driver and the `lirc` driver as the input driver.

Then you have to modify the `[lirc]` section of your `LCDd.conf`.

Configuration in LCDd.conf

[lirc]

lircrc = *`FILENAME`*
: Normally all LIRC clients scan the file `~/.lircrc`. However, you might want to have a separate file to configure the LCDproc lirc driver only.

  This option enables you to specify the file you want the lirc driver to scan. If not given it defaults to `~/.lircrc`.

Prog = *`PROGRAM`*
: All LIRC keys are assigned to a program using the `prog=...` directive in the lirc configuration file.

  `PROGRAM` must be the same as in your lirc configuration file.

Modify Your `~/.lircrc`

As mentioned above you can either modify the `~/.lircrc` or use a separate file for the lirc LCDproc driver (See above for details).

No matter which file you use, you have to add at least the following lines to the file:

ExampleÂ 5.12.Â `~/.lircrc`: Specify the associations from buttons to keys for the lirc driver

```
begin
        prog = lcdd
        button = 2
        config = Up
end

begin
        prog = lcdd
        button = 4
        config = Left
end
```

```
begin
        prog = lcdd
        button = 6
        config = Right
end

begin
        prog = lcdd
        button = 8
        config = Down
end

begin
        prog = lcdd
        button = 1
        config = Escape
end

begin
        prog = lcdd
        button = 0
        config = Enter
end
```

Which buttons you specify here depends on your remote control and your LIRC configuration.

The `config` values need to be one of `Up`, `Down`, `Left`, `Right`, `Escape` or `Enter`. For LCDd's server menu at least the keys `Up`, `Escape` and `Enter` are necessary.

Of course you can define other keys. Those keys will not be handled by the server but sent to a client. Refer to the documentation of the client you want to use, to find out which keys are necessary for that client.

The lis Driver

This section talks about using LCDproc with the VLSystem L.I.S MCE 2005 Vacuum Fluorescent Display (VFD) based on the FTDI USB-to-serial converter, the Microchip PIC-16F716 microcontroller, and the NEC UPD16314 display driver manufactured by VLSystem.

Features

This device uses a vacuum fluorescent display of 20 characters by 2 lines. Each each character is 5 pixels wide by 8 pixels high. The device is connected by USB. The FTDI chip translates the USB protocol to serial expected by the VFD driver chip, an NEC UPD16314. A programmable interrupt controller (PIC), the PIC16F716 by Microchip, provides the glue between the FTDI and the NEC chips.

Requirements

The driver depends on `libftdi`, version 0.8 or higher, from http://www.intra2net.com/en/developer/libftdi/.

`libftdi` itself depends on the `libusb` USB library.

Note

When using a `libusb` based driver like `lis`, LCDd needs to be started as root.

Configuration in LCDd.conf

[lis]

Size = *`WIDTH x HEIGHT`*
: Set the display size. The default `20x2` should be safe for most if not all users, since the device seems to be made only with this one size. But who knows â ¦

VendorID = *`VENDORID`*
: The USB Vendor ID of the device to use. If not given, it defaults to `0x0403` for a VLSystems L.I.S. MCE 2005 VFD based on a FT232BL USB-to-RS232 converter by FTDI, which was produced before March 2007.

  It is usually not necessary to specify a *`VENDORID`*. Please do so only if you want to test a compatible device.

ProductID = *`PRODUCTID`*
: The USB Product ID of the device to use. If not given, it defaults to `0x6001` for a VLSystems L.I.S. MCE 2005 VFD based on a FT232BL USB-to-RS232 converter by FTDI, which was produced before March 2007.

  It is usually not necessary to specify a *`PRODUCTID`*. Please do so only if you want to test a compatible device.

Brightness = *`BRIGHTNESS`*
: Set the initial brightness [default: `1000`; legal: `0` - `1000`. Values between 0 and 250 give 25% brightness, 251 to 500 give 50% brightness, 501 and 750, give 75% brightness, and values higher than 751 give 100% brightness.

The MD8800 Driver

This section talks about using LCDproc with VFD displays in Medion MD8800 PCs.

Features

You may find more information about the LCD on Martin Moeller's homepage.

Configuration in LCDd.conf

[MD8800]

Device = *`DEVICE`*
: device to use [default: `/dev/ttyS1`]

Size = *`WIDTH x HEIGHT`*
: display size [default: `16x2`]

Brightness = *`BRIGHTNESS`*
: Set the initial brightness [default: `1000`; legal: `0` - `1000`]

OffBrightness = *`BRIGHTNESS`*

Set the initial off-brightness [default: `0`; legal: `0 - 1000`] This value is used when the display is normally switched off in case LCDd is inactive

The mdm166a driver

Markus Dolze

This section talks about using LCDproc with the Futaba / Targa USB Graphic Vacuum Fluorescent Display (MDM166A; USB VID=0x19c2, PID=0x6a11).

The MDM166A is a graphical VFD with a 96x16 pixel dot matrix area which is used for 16x2 characters with a 6x8 pixel font. It features several icons, volume level and WLAN strength indicator which are all software controllable using the `output` function.

The mdm166a driver builds on top of `libhid` which in turn uses `libusb`.

The driver was developed by Christoph Rasim (http://www.rasim.net/) without any available documentation from the vendor, but with a good protocol description from Thomas Koos (http://www.muetze1.de) created by reverse engineering the protocol.

Configuration in LCDd.conf

[mdm166a]

Clock = *`TYPE`*
: Show self-running clock after LCDd shutdown. Possible values for *`TYPE`* are `no`, `small` and `big`. If not given no clock is shown.

Dimming = { *`yes`* | *`no`* }
: Dim display, no dimming gives full brightness.

OffDimming = { *`yes`* | *`no`* }
: Dim display in case LCDd is inactive.

Using the icons

The icons, the volume bar and the WLAN indicator supported by the VFD can be controlled by use of the `output` function.

The *`on`* parameter is used as a bitmask to control which elements to display. Setting an icon bit to `1` enables the icon, setting the bit to `0` disables an icon. Volume and WLAN strength indicator accept an numeric value in the given bit mask.

TableÂ 5.22.Â mdm166a_output bitmask

bit(s)	Icon
0	Play
1	Pause
2	Record
3	Message

bit(s)	Icon
4	Mail (at-symbol)
5	Mute
6	WLAN tower
7	Volume (the word)
8â ¦12	Volume (decimal 0â ¦28)
13â ¦14	WLAN strength (0â ¦3)

Special device hints

This display may appear as a HID device in your system which may prevent `libhid` from being able to open the USB device. In this case you have to create some OS-specific configuration to prevent the HID driver to take control of this display.

Configuring FreeBSD 7.x to exclude this device from uhid

To make the uhid driver ignore this device you have to apply the following patch to your kernel source and recompile and install your kernel:

```
--- sys/dev/usb/usbdevs.orig    2010-11-24 02:06:30.000000000 +0100
+++ sys/dev/usb/usbdevs 2010-11-24 07:36:38.000000000 +0100
@@ -623,6 +623,7 @@
 vendor AMIT            0x18c5  AMIT
 vendor QCOM            0x18e8  Qcom
 vendor LINKSYS3                0x1915  Linksys
+vendor TARGA           0x19c2  Targa Corporation
 vendor QUALCOMMINC     0x19d2  Qualcomm, Incorporated
 vendor STELERA         0x1a8d  Stelera Wireless
 vendor DRESDENELEKTRONIK 0x1cf1 dresden elektronik
@@ -2343,6 +2344,9 @@
 /* System TALKS, Inc. */
 product SYSTEMTALKS SGCX2UL    0x1920  SGC-X2UL

+/* Targa Corporation */
+product TARGA VFD              0x6a11  Targa USB Graphic VFD
+
 /* Tapwave products */
 product TAPWAVE ZODIAC         0x0100  Zodiac

--- sys/dev/usb/usb_quirks.c.orig       2010-11-24 02:07:03.000000000 +0100
+++ sys/dev/usb/usb_quirks.c    2010-11-24 02:16:30.000000000 +0100
@@ -112,6 +112,8 @@
        ANY, { UQ_HID_IGNORE }},
  { USB_VENDOR_APPLE, USB_PRODUCT_APPLE_IPHONE_3G,
        ANY, { UQ_HID_IGNORE }},
+ { USB_VENDOR_TARGA, USB_PRODUCT_TARGA_VFD,
+       ANY, { UQ_HID_IGNORE }},

  /* Devices which should be ignored by both ukbd and uhid */
  { USB_VENDOR_CYPRESS, USB_PRODUCT_CYPRESS_WISPY1A,
```

The ms6931 Driver

This section talks about using LCDproc with LCD displays that use the ms6931 chipset.

Configuration in LCDd.conf

[ms6931]

Device = *`DEVICE`*
: device to use [default: `/dev/ttyS1`]

Size = *`WIDTH x HEIGHT`*
: display size [default: `16x2`]

The mtc_s16209x Driver

This section talks about using LCDproc with LCD displays that use the mtc_s16209x chipset.

Configuration in LCDd.conf

[mtc_s16209x]

Device = *`DEVICE`*
: Select the output device to use [default: `/dev/lcd`]

Brightness = *`BRIGHTNESS`*
: Set the initial brightness [default: `255`; legal: `0` - `255`]

Reboot = { *`yes`* | *`no`* }
: Reinitialize the LCD's BIOS [default: `no`; legal: `yes`, `no`]

The MtxOrb Driver

This section covers the installation process for the Matrix Orbital LCD module intended for use with LCDproc.

We will examine the installation process of the hardware in small steps, as it is vitally important to pay close attention to detail during hardware installation to avoid damaging equipment.

Matrix Orbital LCD Modules

LCDproc was born out of original tinkering by William Ferrell with one of these LCD modules. Their ease of installation and use (as well as the amazing amount of patience demonstrated by the folks at Matrix Orbital whilst William figured things out) meant one less thing to worry about during the early stages of LCDproc's life.

These 20x4 alphanumeric modules are connected via standard DB-9 cabling and connectors. They draw either 5V or 12V, depending on the module purchased, and are attached with a standard floppy cable connector (with a slightly modified wire configuration).

Once connected, using them is a breeze. They can operate at any number of different baud rates and serial configurations, but normally they run at 19,200 baud, 8-N-1, making them quite quick. Sending ASCII to the module will make it simply display that text at its current cursor position. The module has a built-in BIOS that recognizes commands (sent by transmitting a single-byte "marker" signifying that a command is on the way, followed by the single-byte command character itself along with any parameters, if needed) allowing the programmer to clear the screen, position the cursor anywhere, define custom characters (up to 8 at a time), draw bar graphs and large numbers, change the LCD's contrast, and so on.

The BIOS included also implements line-wrapping (i.e. writing past the twentieth character on the first row will automatically move the cursor to the first character on the second row), and screen scrolling (i.e. writing past the twentieth character on the fourth row causes the whole screen to scroll up one row, clearing the fourth line and positioning the cursor at the first character on that line).

These modules are fast. Using the auto-line-wrap feature and disabling the auto-scrolling feature, the screen can be updated thirty times per second if *every* character on the screen is changed. If updating less than the whole screen, the LCD can update faster than can be seen by the human eye. This, of course, more than meets LCDproc's needs.

Matrix Orbital Hardware Installation

Regardless of what specific type of hardware you intend to use with LCDproc, installation is usually straightforward, and requires only a few steps. Regardless, you must use caution while working inside your computer system or with any hardware attachments.

Warning

Installing new hardware inside a computer system can be dangerous to both system components and the installer. Use caution whenever adding a component to the inside of your system, altering a power cable, or physically mounting a device inside a computer system.

When installing hardware inside a computer, make sure it's turned off and that its power is disconnected. This is especially important when making changes to power cables (as some LCD modules require).

Matrix Orbital LCD/VFD Module Installation

The LCD and VFD modules from Matrix Orbital are relatively straightforward to install. With a small, regular (flat-head) screwdriver, a spare floppy drive power cable, and a bit of luck, installation will take less than an hour.

These installation instructions assume that you are installing the module into a PC or PC-style system (one with AT- or ATX-compliant power cabling) and that you have some idea of where you intend to permanently mount the module.

Before you start

Your Matrix Orbital LCD or VFD module should be clearly marked with an indication of the module's power requirements. It should be either a 5 volt or 12 volt unit. You should have this information available before proceeding.

Power Cable Modification

The first step in installing the module is making the necessary modifications to a floppy drive power cable in order to provide power to the module. The modifications must be made based on the module's power requirements -- either 5V or 12V -- depending on which module you purchased.

A standard floppy drive power cable has a smaller connection than a "normal" PC power connector. However, like a "normal" power connector, it has four wires: one yellow, one red, and two black. The red wire provides +5V power, and is "hot" or live when the system is powered up. The yellow wire provides +12V power, and is also hot when the system is powered up. Both black wires are ground. [TODO: INCLUDE A FIGURE HERE SHOWING A "STANDARD" FLOPPY CONNECTOR]

One of the hot wires and one of the black wires will not be needed for your module's power connection; they will be completely removed when the power cable modification is complete.

Warning

Do NOT make this modification to a power cable attached to a running system! Electrocution resulting in personal injury and/or damage to the system can result.

Using a regular screwdriver, press down the small metal locking flap of one of the two black wires on the small end of the cable, and pull the black wire from the connector. Using a pair of needle-nose pliers, squeeze the other end of the same black wire, and pull it out of the large end of the cable. This black wire can be set aside; it will not be used for the module's power connection. Either wire can be safely removed; you may safely remove either wire. [TODO: INCLUDE A FIGURE HERE SHOWING THIS PROCESS]

Next, using the same procedure, remove the unneeded hot wire. If your module is 5V, you do not need the yellow (+12V) wire. Conversely, if your module is 12V, you do not need the red (+5V) wire. The removed wire can be set aside; it will not be used for the module's power connection. [TODO: INCLUDE A FIGURE HERE]

The floppy power connector should now have only two wires attached to it. Leave the larger end alone from now on; these connections are correct (the larger end connects to your system's power mains). Move the two remaining wires to the outside connectors on the small end of the cable. Orientation does not particularly matter here; the connector will fit on the module's receptacle in either orientation. [TODO: A FIGURE HERE]

You should now have a properly modified power connector. When physically attaching this connector to the module, the black (ground) lead should be connected to the pin labelled GND, while the colored (+5V/+12V) lead should be connected to the pin labelled +5V/+12V.

Test the power connection before connecting the data line or mounting the module. Connect the module to the power connector, and the connector to your system's power mains. Turn the system on.

Caution

If the module does not immediately display its initial BIOS screen and light up its backlight (or light up the screen if a VFD module is being used), *immediately* power down the system, disconnect the module and connector, and double-check the modification before trying again. Do NOT leave the system on if the module does not immediately respond; module or system damage could result.

When the LCD powers up and displays its initial BIOS screen, you've gotten the power connection wired properly and can now properly mount the module and make its final connections. Matrix Orbital Corporation sells a PC bay insert mount for the 20x4 and 20x2 modules (LCDproc, however, only supports the 20x4 at present). The inserts provide an easy means of mounting the LCD modules inside a PC using one (for the 20x2) or two (for the 20x4) 5 1/4" bays.

Note

Describing how to physically mount the module in a PC case is beyond the scope of this document; LCDproc's website contains more detailed mounting information and examples.

Serial Connection

The LCD module uses a standard DB9 serial connector. You can attach the module to your system using a direct cable to the motherboard, or by removing one of your system's serial ports from the back of the case, then connecting it to a standard serial cable to the module.

While connecting the serial cable to the module, be sure to configure the module's serial interface settings. Typically, setting the module to its fastest setting (19,200 baud, 8-N-1) is recommended. The speed settings can be configured from the config file `/etc/LCDd.conf`. If not specified in the config file, the Matrix Orbital module driver in LCDproc default to use these settings.

Configuration in LCDd.conf

[MtxOrb]

Device = *`DEVICE`*
: Select the output device to use [default: `/dev/lcd`]

Size = *`WIDTH x HEIGHT`*
: Set the display size [default: `20x4`]

Type = { *`lcd`* | *`lkd`* | *`vfd`* | *`vkd`* }
: Set the display type [default: `lcd`; legal: `lcd`, `lkd`, `vfd`, `vkd`]

Contrast = *`CONTRAST`*
: Set the initial contrast. Legal values for *`CONTRAST`* are `0 - 1000`. If not given, the default value is `480`.

Note

The driver will ignore this setting if the display is a VFD or VKD as they do not support this feature.

hasAdjustableBacklight = { *`yes`* | *`no`* }
: Some old firmware versions of Matrix Orbital modules do not support an adjustable backlight but only can switch the backlight on/off. If you own such a module and experience randomly appearing block characters and backlight cannot be switched on or off, use this to `no` [default: `yes`].

Brightness = *`BRIGHTNESS`*
: Set the initial brightness [default: `1000`; legal: `0 - 1000`]

OffBrightness = *`BRIGHTNESS`*
: Set the initial off-brightness [default: `0`; legal: `0 - 1000`] This value is used when the display is normally switched off in case LCDd is inactive

Speed = { *`1200`* | *`2400`* | *`9600`* | *`19200`* }
: Set the the baud rate to use when communicating with the LCD. If not specified, it defaults to `19200`.

KeyMap_*LETTER* = *KEY*

Matrix Orbital displays support keypads with up to 25 keys, which return one of the letters `A` - `Y` for each pressed key.

These settings allow to map the letter *LETTER*, that is generated by the display when a key is pressed, to be mapped to a key name *KEY* that LCDd can understand (see `LCDd.conf`: The `[Menu]` Section for more information).

There is no default key mapping; if no keys are mapped in the `LCDd.conf` config file, the display is treated as if it had no keys attached.

ExampleÂ 5.13.Â Matrix Orbital: keymap config

```
KeyMap_A=Left
KeyMap_B=Right
KeyMap_C=Up
KeyMap_D=Down
KeyMap_E=Enter
KeyMap_F=Escape
```

keypad_test_mode = { *yes* | *no* }

You can find out which key of your display sends which character by setting keypad_test_mode to yes and running LCDd. LCDd will output all characters it receives. Afterwards you can modify the settings above and set keypad_set_mode to no again.

The MX5000 Driver

This section talks about using LCDproc with LCD displays on Logitech MX5000 keyboards.

Features

This drivers uses the library from mx5000tools.

Configuration in LCDd.conf

[mx5000]

Device = *DEVICE*

Select the output device to use [default: `/dev/hiddev0`]

WaitAfterRefresh = *MS*

Set the time (in ms) to wait after each screen sent to the keyboard. Default is 1000ms.

The NoritakeVFD Driver

This section talks about using LCDproc with text mode VFD displays from Noritake Itron.

Configuration in LCDd.conf

[NoritakeVFD]

Device = *`DEVICE`*
: Port where the VFD is. Usual values are `/dev/ttyS0` and `/dev/ttyS1` Default: `/dev/lcd`

Size = *`WIDTH x HEIGHT`*
: Specifies the size of the LCD. Default: `20x4`

Brightness = *`BRIGHTNESS`*
: Set the initial brightness [default: `1000`; legal: `0` - `1000`]

OffBrightness = *`BRIGHTNESS`*
: Set the initial off-brightness [default: `0`; legal: `0` - `1000`] This value is used when the display is normally switched off in case LCDd is inactive

Speed = { *`1200`* | *`2400`* | *`9600`* | *`19200`* | *`115200`* }
: Set the the baud rate to use when communicating with the VFD. If not specified, it defaults to `9600`.

Parity = { *`0`* | *`1`* | *`2`* }
: Set the parity for communication with the device to even parity (`2`), odd parity (`1`) or no parity (`0`). If not given, it defaults to `0`.

Reboot = { *`yes`* | *`no`* }
: Reinitialize the VFD [default: `no`; legal: `yes`, `no`]

The Olimex_MOD_LCD1x9 Driver

The Olimex_MOD_LCD1x9 driver supports the popular MOD-LCD1x9 14 segment display on I2C lines. Currently this driver only supports Linux.

Configuration in LCDd.conf

[Olimex_MOD_LCD1x9]

Device = *`DEVICE`*
: Set the I2C device [default: `/dev/i2c-0`]

The *Really* Simple Serial interface (rawserial)

Connor Wolf

This section talks about using LCDproc with custom hardware.

This driver is intended for using LCDproc with custom hardware. In the original implementation, this was for a PFsense router with a homemade arduino-style circuit that controlled analog panel meters to display network traffic. It is intended for this sort of application, where the LCD control characters emitted normally, as well as the partial screen drawing, make parsing the serial stream a considerably complex affair, particularly on small (generally 8 bit) microprocessors.

Features

The driver emulates a 40x4 display, and simply dumps the entire framebuffer out the serial port at a rate configured in the config file (default of 1 Hz), post-pended with a single "\n" character. At 1 Hz, it therefore generates 161 serial characters per second.

The maximum update-rate is limited by the internal update-rate of LCDproc, which is currently 8 Hz, so speeds greater then 8 Hz will be simply limited to 8 Hz. The update-rate is also granular at 1 second/8 time-intervals (125 ms), so interesting fractional update-rates will behave oddly. The event-loop is rigidly deterministic, so fractional rates will *average* out to the desired rate, but that will be achieved by dithering between nearby available intervals.

The baud rate is configurable, with a default of 9600 baud. Baud rate will have to be chosen with the required data-per-second in mind, since at 1 Hz, there are 161 characters (40x4+1) per second, which requires that's a theoretical minimum baud of 161*10 = 1610 baud. Higher frame-rates will require higher baud rates.

Requirements

None. The only requirement is a serial port of some sort.

Configuration in LCDd.conf

[rawserial]

Device = *`DEVICE`*
: Serial port to use. Default: `/dev/cuaU0`

Speed = *`NUMBER`*
: Desired baud-rate. Possible values: `1200`, `2400`, `9600` (default), `19200`, and `115200`.

Size = *`WIDTH x HEIGHT`*
: Specifies the size of the LCD. If this driver is loaded as a secondary driver it always adopts to the size of the primary driver. If loaded as the only (or primary) driver, the size can be set. Default: `40x4`.

UpdateRate = *`NUMBER`*
: Desired update-rate in Hertz (e.g. updates per second). Fractional values and values less than one are valid (e.g. 0.5). Legal values are `0.0005` (equals 2000 seconds) - `10`, with `1.0` being the default.

The Mini-Box.com USB LCD picoLCD Driver

This section covers the use of the Mini-Box USB LCD displays.

Displays

Mini-Box.com offers two types of USB LCD displays:

PicoLCD 4x20-Sideshow
: PicoLCD 4x20-Sideshow is the desktop variant targeted at end users. It is an external USB 2.0 full speed device that comes in a stylish case and sports a 4 line by 20 character display with white letters on a blue background, a built-in InfraRed receiver as well as a keypad with 8 keys labeled `Escape`, `F1`, `F2`, `F3`, `Home`, `Up`, `Down` and `Enter`.

picoLCD 20x2 (OEM)
: picoLCD-20x2-OEM is the OEM version. It is a 2 line by 20 character display with black letters on a yellow-green background, that can be connected to the system via USB, I^{2}C or USART (the latter two are not supported by this driver). It has connectors for an InfraRed receiver, keypad and LEDs.

When pre-installed in enclosures like the Mini-Box M300 LCD it comes equipped with an InfraRed receiver as well as key pad with 12 keys labeled `Plus`, `Minus`, `F1`, `F2`, `F3`, `F4`, `F5`, `Up`, `Down`, `Left`, `Right`, and `Enter`.

Finally, the picoLCD 20x2 (OEM) supports 8 general purpose outputs and 10 custom splash screens. When the keypad is connected the outputs control the key LEDs. The output command and KeyLight settings below can be used to control the outputs. Although splash screens are not supported by this driver, the splash screens can be changed using the **usblcd** tool, that can be built from the Linux SDK available on the picoLCD web page.

Requirements

The driver is based on the `libusb` USB library, which should make it work with Linux, the different BSB variants as well as Darwin/MacOS X.

Note

When using a `libusb` based driver like `picolcd`, LCDd needs to be started as root.

On Linux, the only kernel module required is the USB host controller driver (`uhci_hcd` on the M300) to fire up the USB bus to which the LCD is attached. For other operating systems, analogous requirements apply.

Lastly, for `libusb` to work correctly, the `usbfs` file system must be mounted on `/proc/bus/usb`, e.g. using the command `mount -t usbfs usbfs /proc/bus/usb` or by your system's default configuration.

Configuration in LCDd.conf

[picolcd]

Note

The Brightness and OffBrightness settings only have an effect on the 20x4 device. With the 20x2 device the backlight can only be set on (any value `1` or greater) or off (`0`).

Backlight = { *`yes`* | *`no`* }
: Turns the backlight on or off on start-up, default `yes`.

Brightness = *`BRIGHTNESS`*
: Set the initial brightness if the backlight is on. Legal values are: `0` - `1000`. If not given, it defaults to `1000`.

OffBrightness = *`OFFBRIGHTNESS`*
: Set the initial value for the backlight if it is off. Legal values are: `0` - `1000`. If not given, it defaults to `0`.

Contrast = *`CONTRAST`*
: Contrast: `0`-`1000`, default to `1000` (full contrast).

LinkLights = { *`yes`* | *`no`* }
: Allow key LEDs to be turned on or off with the backlight. Default is `yes`.

KeyLights = { *`yes`* | *`no`* }
: Allow key LEDs to be turned on or off. Default is `yes`. This setting affects all keys. If set to `on` each key can be disabled independently by setting `KeyXLight` below.

Key0Light = { `yes` | `no` }
: If Keylights is set, you can disable the directional pad LED by setting this value to `no`. Default is `yes`.

Key1Light = { `yes` | `no` }
: If Keylights is set, you can disable the F1 LED by setting this value to `no`. Default is `yes`.

Key2Light = { `yes` | `no` }
: If Keylights is set, you can disable the F2 LED by setting this value to `no`. Default is `yes`.

Key3Light = { `yes` | `no` }
: If Keylights is set, you can disable the F3 LED by setting this value to `no`. Default is `yes`.

Key4Light = { `yes` | `no` }
: If Keylights is set, you can disable the F4 LED by setting this value to `no`. Default is `yes`.

Key5Light = { `yes` | `no` }
: If Keylights is set, you can disable the F5 LED by setting this value to `no`. Default is `yes`.

KeyTimeout = *`DURATION`*
: KeyTimeout is only used if the picoLCD driver is built with libusb-0.1, when built with libusb-1.0 key and IR data is input asynchronously so there is no need to wait for the USB data thus allowing LCDd to process other inputs at the correct rate.

  This value controls how long LCDd waits for a key press when get_key() is called. The value represents milliseconds and the default is `500` or .5 seconds. Lowering this value will make LCDd more responsive but also causes LCDd to use more CPU time and, as the timeout grows shorter, key presses become harder to detect. Large values make key presses more reliable but may slow down LCDd. Values in the range `0-1000` (1s) are allowed.

KeyRepeatDelay = *`DURATION`*
: KeyRepeatDelay is only used if the picoLCD driver is built with libusb-1.0, when built with libusb-0.1 key input blocks all other processing until the key is released.

  This value controls how long LCDd waits from when a key is pressed and reported before generating the first repeat. The value represents milliseconds and the default is `300` (0.3 second). Use zero to disable auto repeat. Values in the range `0-3000` (3s) are allowed.

KeyRepeatInterval = *`DURATION`*
: KeyRepeatInterval is only used if the picoLCD driver is built with libusb-1.0, when built with libusb-0.1 key input blocks all other processing until the key is released.

  This value controls how long LCDd waits between key reports after generating the first repeat. The value represents milliseconds and the default is `200` (0.2 second). Use zero to disable auto repeat. Values in the range `0-3000` (3s) are allowed.

LircHost = *`HOSTNAME_OR_IP-ADDRESS`*
: Set the hostname or IP address to which the driver will send IR data from the sensor. If not set or set to an empty value, IR support for LIRC will be disabled.

  LIRC should be configured to use the driver "udp", which will cause it to listen on some UDP port for packets containing a series of integers, representing mark and space intervals from the sensor. It doesn't matter whether LCDd or LIRC is started first; if LIRC isn't listening, the packets from LCDd will be discarded. When LIRC comes back, it will start picking up the packets. Similarly, LCDd can be stopped and restarted without affecting anything, because UDP is a connectionless protocol.

LircPort = *`PORTNUM`*
: This value determines the UDP port to which the driver will send IR data from the sensor. It defaults to `8765`, which is also the default port on which LIRC will listen.

LircTime_us = { `yes` | `no` }

If LircTime_us is set to on mark and space times are sent to LIRC in microseconds (requires a LIRC UDP driver that accepts this (LIRC 0.9.3 onwards)).

If LircTime_us is set to off mark and space times are sent to LIRC in 'jiffies' (1/16384s) (supported by the standard LIRC UDP driver).

Default is `off`.

Note

One 'jiffy' is approximately 61 microseconds about a tenth of typical IR mark and space times. LIRC configuration program `irrecord` cannot reliably detect the IR data timing when measured in 'jiffies' it works better with microseconds.

I have submitted a patch that modifies the LIRC udp driver to support timing data in microseconds. This was included in LIRC 0.9.3. The default (61Âµs) UDP driver settings will work but I recommend using 1 microsecond. The LIRC upd plugin timing resolution can be set using the `--driver-option=clocktick:value` or `-A clocktick:value` command line switch.

LircFlushThreshold = *`DURATION`*
: This value is the length in microseconds of the gap that will trigger sending the queued IR data to LIRC. Values greater than 1000 (1ms) are permitted, lower values will set the default value 8000 (8ms). The maximum depends on the setting of LircTime_us; if LircTime_us is on values greater than 32.767ms will disable the flush, if LircTime_us is off values greater than 1.999938s will disable the flush. The value should be less than the gap times specified in `lircd.conf` and greater than any space time specified in any header, one, zero, etc. field.

picolcd driver status

The hardware also reports key-up events. Normally this would be of no issue (they are usually a 0 or 'no key') except that when keys are used in combination, the key-up event may actually come back as multiple events depending on how the user released the keys. If the key-up event for a multiple key press comes back as two events, the first up event will actually look like a new key press. The algorithm in get_key tries to deal with this in a sane way and toss out all key-up events for now. The hardware is touchy and both combo key-down and key-up actions may be reported as multiple events if the user is more than a tenth of a second (maybe less?) off in motions. The hardware is *not* "touchy" it reports what it receives. Two key presses or releases may appear simultaneous to a human but they are always some time apart. The hardware probably samples the keys for every USB transfer cycle; that is every 10ms, significantly faster than the typical human response time of a few hundred milliseconds!

Infrared sensor status

LIRC expects sensor data that starts with a longish 'sync' space, denoting the start of a command; followed by the code data, a sequence of mark/space pairs; sometimes followed by a 'gap', which should be a space long enough to make the entire command up to a preset duration in milliseconds. The 'sync' and the 'gap' are absent from the data emitted by the picolcd hardware. I found that LIRC configuration files for remotes similar to the ones I tested all used such a fixed-duration encoding, and as that was the only way I could get it working, this driver by default adds the gap as well as the sync. However I have *still* had trouble getting `irrecord` to work; you need at least to feed it a template configuration containing sync and gap data. The LIRC configuration program `irrecord` cannot reliably detect the IR data timing when measured in 'jiffies' it works better with microseconds, see `LircTime_us` above.

Note

The current libusb-1.0 implementation polls at 32Hz to see if any USB processing is required, this is needed when a key has been pressed or some IR data has been received. 32Hz has a period of 31.25ms this is not really fast enough the picoLCD USB transfers can occur every 10ms. This may cause buffer overrun problems for long bursts of IR data. The best solution would be to include USB processing in the main loop `select` statement this has not been done to avoid major changes to the core code, to work-round this extra USB transfer buffers are allocated. It may also be worth building with `PROCESS_FREQ` set to 100Hz (in `server/main.h`).

Copyright

The lcdproc picolcd driver originally was written by Gatewood Green (woody@nitrosecurity.com) or (woody@linif.org) and paid for by NitroSecurity, Inc (nitrosecurity.com), but has been extended with code from various contributors since then.

The pyramid Driver

Stefan Reinauer

This section talks about using LCDproc with programmable LC displays sold by Pyramid Computer.

Pyramid Computer builds these LC displays into its thriving server products to show system data, and to allow the user to change important parameters or shut down the appliance in a controlled manner. The LCD module, accessible via USB, can be integrated by Pyramid's BTO server and appliance manufacturing division at the customer's request or it can be made available separately for self-integration, e.g. as a 5.25" module.

Features

The displays are 16 characters wide and 2 lines high. They have 4 programmable buttons as well as 3 (or 9) LEDs which can also be software controlled.

Connector

The display is connected to the host system using USB, with the usual USB pinout as shown below. The +5V V_{CC} pin is marked with red colour.

TableÂ 5.23.Â USB Pinout

GND	D+	D-	V_{CC}
GND	P_0+	P_0-	+5V

Requirements

For Linux the driver depends on features of the `ftdi_sio.ko` that have only been added to version 2.6.15 version of the 2.6 kernel series. For older Linux kernels of the 2.6 series these patches to `ftdi_sio.c` may help:

- Linux 2.6.9-11.
- Linux 2.6.12.x.
- Linux 2.6.13-14.

LED output

I've added support for the LEDs on the Pyramid LC-Display to the "pyramid" driver of lcdproc.

Since it seems that LEDs on an LCD are not directly supported by the lcdproc API I've used the "output" command of the server to trigger the LEDs, similar to what the IOWarrior driver does.

The Pyramid LC displays come in two different versions, with 3 and with 9 LEDs. Two of these LEDs can not be controlled by software but are usually hard wired to power and HDD. The other 1 or 7 LEDs can now be controlled by sending an `output` command to the server.

The argument to the `output` command is a bitmask that controls the LEDs according to the table below:

TableÂ 5.24.Â Mapping of `output` bits to LEDs

Bit	LED
0	3
1	4
2	5
3	6
4	7
5	8
6	9

For the LCDd server there is no way to find out whether 3 or 9 LEDs are available, so it is up to the client software to do the right thing.

ExampleÂ 5.14.Â Pyramid: How to use the LED output from the client

```
telnet localhost 13666
hello
output 67
```

will light up LEDs 3, 4 and 9.

```
output 0
```

will clear all LEDs.

More information on the Pyramid LC-Display can be found here:
http://www.pyramid.de/en/products/specialities.php

Configuration in LCDd.conf

[pyramid]

Device = *DEVICE*
Device to connect to. Default: `/dev/lcd`

The SDEC LCD Driver

The SDEC LCD modules are fitted to the Watchguard Firebox firewall appliances. Watchguard identifies these units by the X-Core, X-e and X-Peak product lines. The model number indicates a software license level, and has no bearing on the hardware configuration.

This driver implements the same protocol as the HD44780 with lcm162 connection type.

Features

The SDEC LCDs are interfaced to the appliance using a parallel port. All lines are hard-wired by Watchguard. The LCD displays are 20 characters wide and 2 lines high, and have 4 buttons: `Up`, `Down`, `Left`, and `Right`. The SDEC model number is `LMC-S2D20-01`. There is a back light that can be controlled by software. Due to its low half-life, the driver tries to keep the light off a short while after buttons have been pressed.

For more detailed information about the SDEC LCD, locate the technical and programming guide LMC-S2D20-01.pdf. The manufacturer web site is: SDEC home page

The sed1330 Driver

This section talks about using LCDproc with graphical LCD displays driven by the EPSON/SMOS SED1330 or SED1335 LCD controllers, which may also be known by their new names: S1D13300 (= SED1330) and S1D13305 (= SED1335).

The displays are driven in text mode using their built-in character generator and the graphic mode for bars and the heartbeat icon (a bouncing ball).

Connections

Warning

Displays using SED1330/1335 chipset come in a variety of pin-outs and power configurations. They usually require negative voltage for contrast and some displays have a negative voltage generator onboard.

Connections below are for the G242C, G121C and G321D displays. Always consult documentation about the specific display before assuming the connections given here are also correct for your display!

TableÂ 5.25.Â ConnectionType `classic` ordered by LPT port pins

LPT port		<->	LCD	
pin	name		name	pin

LPT port		<->	LCD	
1	^STROBE	Â	^RESET	1
2-9	D0-D7	Â	D0-D7	8-15
16	^INIT	Â	^WR	3
17	^SELECT_IN	Â	A0	7
18-25	GND	GND	V_{EE}	17
Â	Â	+5V	^RD	2
Â	Â	GND	SEL1	4
Â	Â	GND	SEL2	5
Â	Â	GND	^CS	6
Â	Â	+5V	V_{CC}	16
Â	Â	potentiometer	V0	18
Â	Â	-24V (not required for G242C)	V_{LC}	19
Â	Â	GND	F_{GND}	20

TableÂ 5.26.Â ConnectionType `bitshaker` ordered by LPT port pins

LPT port		<->	LCD	
pin	**name**		**name**	**pin**
1	^STROBE	Â	^WR	3
2-9	D0-D7	Â	D0-D7	8-15
14	^LF	Â	A0	7
16	^INIT	Â	^RESET	1
18-25	GND	GND	V_{EE}	17
Â	Â	+5V	^RD	2
Â	Â	GND	SEL1	4
Â	Â	GND	SEL2	5
Â	Â	GND	^CS	6
Â	Â	+5V	V_{CC}	16
Â	Â	potentiometer	V0	18
Â	Â	-24V (not required for G242C)	V_{LC}	19
Â	Â	GND	F_{GND}	20

The potentiometer should be connected like this on these display modules:

```
=== GND
 |
.-.
| |
| |5k
'-'
 |
 |
.-.10k potentiometer
| |
| |<----------------o V0
| |
'-'
 |
 O Vlc (= -24V)
```

The G242C generates -24V internally. It is available on Vlc.

To generate -24 from the +5V without an external power source, you can use the following circuit.

```
 5V O------+----------+                                 pinout:
           |          |                                  _____
           |         --- 100uF                          |  _  |
           |         --- 10V                            | (_) | <-3
           |          |                                 |_____|
           |          +--------+--------+--------+      | max |
           |5         |        |        |        |      | 724 |
       ---------     === GND   C        -        |      |_____|
      |         |              C coil  | |       |       |||||
      |         |              C 47uH  | |10k    |       |||||.
      |         |4             |        -        |       | | |
      | MAX724  |--------------+        |        |       12345
      |   or    |              |        |        |
      | MAX726  |1             |        |        |+
      |         |-----------------------+       --- 47uF
      |         |              |        |       --- 50V
      |         |              |        |        |
       ---------               |        -        |
        |2    |3               |       | |       |
        |     |              '---,     | |1k     |
       ---    |        SB160  / \       -        |
  100nF---    |               ^T^       |        |
        |     |                |        |        |
        +-----+----------------+--------+--------+----O -24V out
```

Keypad

The sed1330 driver supports the same direct keys and matrix keypad connections as the hd44780 driver. See the section called â Keypadâ .

Configuration in LCDd.conf

[sed1330]

CellSize = *WIDTH x HEIGHT*

Specify the size of a character in pixels. *WIDTH* may vary between `6` and `8`; legal values for *HEIGHT* range from `7` to `16`. If not given CellSize defaults to `6x10`.

Contrary to other drivers the character size of an LCD using the sed1330 driver is not given directly. Instead it is determined by the pixel size of the display, which is derived from the display type setting and the character cell size specified with this setting.

ConnectionType = { *classic* | *bitshaker* }

Select the type of the wiring, see the section called â Connectionsâ .

If not given, it defaults to `classic`.

Port = *PORT*
: Specify the address of the parallel port the LCD is connected to. Common values for *PORT* are `0x278`, `0x378` and `0x3BC`. If not given, it defaults to `0x278`.

Type = { *G321D* | *G121C* | *G242C* | *G191D* | *G2446* | *SP14Q002* | *HG25504* }
: Type of LCD module. Besides other things (internal setup) this configuration setting determines the size of the display in pixels.

Type	Size (in pixels)
`G321D`	320 x 200
`G121C`	128 x 128
`G242C`	240 x 128
`G191D`	192 x 192
`G2446`	240 x 64
`SP14Q002`	320 x 240
`HG25504`	320 x 240

Note

Currently the G321D and SP14Q002 are the only ones that this driver is tested with.

The sed1520 Driver

This section talks about using LCDproc with LCD displays that use the SED1520 chipset.

Currently the driver supports 122x32 pixel graphic displays based on the SED1520 controller connected to the parallel port.

Note

Those SED1520 displays are the most troublesome I have ever used. You should probably avoid using them and get a 20x4 text display instead.

Connections

Warning

Displays using SED1520 come in a wide variety of configurations. It is possible to cause harm to your display (e.g. connecting negative voltage incorrectly. Be sure to check your datasheet! Do not try to use a display without having a datasheet to check against!

Here here some of the options I encountered:

- Negative voltage: The chip requires negative voltage for driving the display. Some display modules have a negative voltage converter on-board. On those that don't you have to supply about -7 V (will not show how to do this).
- Reset circuit: Some display modules have an R/C-combination on-board selecting an MCU interface if the pin is not connected.

- Frequency generator: The SED1520 is manufactured in several versions. Some contain an on-chip frequency generator, others require an external clock of 2 kHz (won't show this here). Note that display modules with an on-chip generator do not have a /CS (chip select) line.
- Incorrect datasheets: I have seen display datasheets incorrectly naming pins, missing some of the commands, missing pages from the chip description and other odds. Be warned!
- No pin numbers are given in the tables below. You have to figure out those yourself from the datasheet for your display!

That said here some wirings:

80-style connection style

This mode of operation is selected if the RESET line is wired to ground. The wiring used by this driver in 80-style mode assumes you have /CS1 and /CS2 lines available (thus you must have an external clock generator) and toggles the /RW line. This is the original wiring by Robin Adams (SED1520 LPT Port).

TableÂ 5.27.Â SED1520 80-style wiring schematic

Parallel port		<->	LCD	
name	**pin**		**name**	**pin**
D0-D7	2-9	Â	DB0-DB7	*
nSTRB	1	Â	/WR	*
nLF	14	Â	/CS1	*
INIT	16	Â	/CS2	*
nSEL	17	Â	A0	*
Â	Â	VDD	/RD	*
Â	Â	GND	RESET/IF	*
Â	Â	2 kHz clock	CL	*

68-style connection style

This mode of operation is selected if the RESET line is wired to VDD. For this wiring the display is required to have an on-board frequency generator (something you really want to have) and the display module has an E1 and E2 line. Writing is controlled by toggling the E1 and E2 line while /RW is low.

TableÂ 5.28.Â SED1520 68-style wiring schematic

Parallel port		<->	LCD	
name	**pin**		**name**	**pin**
D0-D7	2-9	Â	DB0-DB7	*
nSTRB	1	Â	R/-W	*
nLF	14	Â	E1	*
INIT	16	Â	E2	*
nSEL	17	Â	A0	*
Â	Â	VDD	RESET/IF	*

Configuration in LCDd.conf

[sed1520]

Port = *PORT*
: Specify the address of the parallel port the LCD is connected to. Common values for *PORT* are `0x278`, `0x378` and `0x3BC`. The default value is `0x378`.

InterfaceType = *INTERFACE*
: The SED1520 chip can be driven with one of two interface types selected by the level of the RESET line: 68-style MCU interface (high level) or 80-style MCU interface (low level).

  Use value `68` if your display is connected using "68 family MPU" style. In this mode E1 and E2 lines are cycled to write the data. If you use `80` (the default), "80 family MPU" style is selected and the /WR line is cycled and /CS1 and /CS2 lines are required.

DelayMult = *DELAY*
: This value adds an additional delay to each write, in microseconds. For the 80-style connection type actual two delays are added each. The default value of `1` already slows down communication a lot, larger value should likely be avoided.

haveInverter = { *yes* | *no* }
: The original wiring by Robin used an inverter to drive the control lines. If you do not use an inverter set this to `no`.

The serialPOS Driver

This section talks about using LCDproc with a point of sale ("POS") character-display.

The `serialPOS` driver is currently working with the AEDEX emulation protocol, and may support the CD5220, Epson ESC/POS, Logic Controls, EMAX, and Ultimate (UTC/S) protocols. It can be extended to work with various other protocol displays.

Brightness adjustment, custom characters, and cursor control are available for protocols that support them. At this moment, custom character support is limited to only devices with protocols that are able to load custom characters into the "extended"-ASCII range, characters from `0x80` onwards. This is to avoid clashing with the usual ASCII characters.

Custom characters are only used for the rendering of horizontal bars and vertical bars. For displays whose cell character cell widths are lower than the number of custom characters supported, then custom characters will be used to render the horizontal bars.

For displays with sufficient custom character support to include another set of custom characters, equal in size to the character cell height minus one, on top of the characters already used to support rendering custom horizontal bars, then custom characters will be used to render the vertical bars as well.

Note that some displays use EEPROM to contain their custom characters, and, upon receiving custom character commands, directly write the received characters to EEPROM. Excessive writes to EEPROM on initialization may then cause fast EEPROM wear. Users who do not want this may like to set the number of custom characters supported by their display to zero. Displays that store their custom characters in RAM should work fine, as the driver does not attempt to send commands to permanently store the custom characters.

The driver should operate most character POS displays with a serial (RS-232) input. Because these displays use a standardized protocol, if the protocol is supported by your display, it should work as expected. Feedback is welcome.

TableÂ 5.29.Â serialPOS: Emulation Protocol Status

Protocol	Display tested	Currently Supported	Remark
AEDEX	Emax, KPD220V7	Yes	Max size: 40x2
IEE		No	
CD5220	KPD220V7	Yes	Max size: 20x2, Cell size must be 5x7 for custom characters
Epson	KPD220V7	Yes	Max size: 20x4, Cell size must be 5x7 for custom characters
Emax		Yes	
IBM		No	
Logic Controls	KPD220V7	Yes	Max size: 20x2
Ultimate	KPD220V7	Yes	Max size: 20x2

(â ¦) : Feature not tested.

Connecting The Display

Connecting the display should consist of simply plugging it into your computer's RS-232 serial port. Because these displays typically support full RS-232, no additional wiring is needed. If your computer does not have such a port (many newer computers don't), you can use a USB to serial adapter with a driver provided by the adapter manufacturer.

If your display supports a *pass-through* function, you can connect an RS-232 keyboard or terminal to the pass-through port. This will allow you to input keystrokes to LCDproc and control features and menus. Use the pass-through keyboard's arrow, delete, and return keys by default.

Note

If your display supports a *pass-through* function, but you do not have another RS-232 device connected to the pass-through port, you may experience hangs if an improperly formatted command sneaks through. This is because the display is waiting for the pass-through device to accept the data and a blocking state is created within the display. You can either connect another RS-232 device or use a wire to jumper the CTS and RTS pins together within the display.

Configuration in LCDd.conf

[serialPOS]

Device = `DEVICE`

Device to use in serial mode. Usual values are `/dev/ttyS0` or `/dev/cu.usbserial`. Default is `/dev/ttyS0`.

Size = *`WIDTH x HEIGHT`*
: Specifies the size of the VFD in characters. If not given, it defaults to `16x2`.

Cellsize =
: Specifies the cell size of each character cell on the display, in pixels. If not given, it defaults to `5x8`.

Custom_chars =
: Specifies the number of custom characters supported by the display. Maximum number of custom characters supported by the driver is `32`. If the protocol supports custom characters but your display is not of the right cell size, set this to zero so the protocol driver skips initializing custom character support and succeeds initialization. If not given, it defaults to `0`.

Type = { *`AEDEX`* | *`CD5220`* | *`Epson`* | *`Emax`* | *`LogicControls`* | *`Ultimate`* }
: Set the communication protocol to use with the POS display. If not specified it defaults to `AEDEX`.

Speed = { *`1200`* | *`2400`* | *`4800`* | *`9600`* | *`19200`* | *`115200`* }
: Set the the baud rate communication with the POS display. If not given the default of `9600` is used.

The serialVFD Driver

This section talks about using LCDproc with VFD character-displays build by NEC, FUTABA and NORITAKE.

The serialVFD-driver is working with NEC FIPC8367 based VFDs and the "KD Rev 2.1" (an ATMEL AT90S.... based FM20X2KB-AB replacement). It is also known to work on FUTABA VFDs.

The driver should operate most of NEC, Futaba and Noritake 7x5 dot VFDs with serial(RS-232) and/or parallel interface. See the following table for testing-status. Feedback is welcome.

TableÂ 5.30.Â serialVFD: Display Status

Display	Controller	Serial	Parallel	Display tested	Type	Remark
NEC FM20X2KB-AB	NEC FIPC8367	Ok	Ok	Yes	0	
NEC FC20X2JA	NEC FIPC8367	(Ok)[1]	(Ok)[1]	No	0	Same Controller as on FM20X2KB-AB
NEC FC20X1SA-AB/AA	NEC FIPC8367	(Ok)[1]	(Ok)[1]	No	0	Same Controller as on FM20X2KB-AB
KD Rev 2.1	AT90S.... microcontroller	Ok	-	Yes	1, 0[2]	
FUTABA M402SD06GJ	?	(?)[1]	Ok	Yes	3	The display has no user-characters. Serial interface with PC compatible baudrate is optional only, feature not tested.
FUTABA M204SD01AA	FUTABA 5P00A016	(Ok)[1]	(Ok)[1]	No	3	
Futaba NA202SD08FA	?	Ok	Ok	Yes	6	almost IEE compatible, no Custom-Characters
	?	Ok	Ok	Yes		

Display	Controller	Serial	Parallel	Display tested	Type	Remark
Samsung 20S204DA2 and 20S207DA1					3[2], 7[2][1]	The display is FUTABA compatible (hard- and software). Custom-Characters not supported(?).
Samsung 20S207DA4 and 20S207DA6	?	Ok	Ok	Yes	7, 3[2]	almost Futaba compatible
Noritake CU20026SCPB-T	microcontroller	(Ok)[1]	(Ok)[1]	No	2	
Noritake CU20045SCPB-T28A	?	(Ok)[1]	(Ok)[1]	No	2	
IEE 36657-01 (= 02S-93290-VFD 36657-01)	?	Ok	Ok	Yes	4	
IEE S03601-95B	?	(Ok)[1]	(Ok)[1]	No	4	
IEE S03601-96-080	?	(Ok)[1]	(Ok)[1]	No	5	
Siemens/Wincor Nixdorf BA63/66	?	Ok	-	Yes	8	Display needs different connection, see below! no Custom-Characters, no brightness-control

[1] Should work, but feature not tested yet.

[2] Custom-Characters are not supported with this **Type**, set **Custom-Characters=0** in `LCDd.conf`.

If your display isn't working 100% satisfactorily you can add a new device with modified hardware commands to the driver if you want. To do that you have to add a new section to the display **Type**-switch-case in `./server/drivers/serialVFD-displays.c` and to write a new "load" function with the correct commands for the display. (Try which display **Type** works best with your display, then copy, rename and modify this function to your needs - that is the easiest way I guess.)

On Malte Poeggel's page you may find pictures and datasheets of the VFDs: http://www.maltepoeggel.de/index.php?site=vfd

It is possible to switch the display off and back on while the server is running. It may take a few minutes until the next full refresh makes the display show everything correctly.

Connecting The Display

The Connections shown have been tested successfully.

Warning

Always cross-check with your datasheet, before connecting your display! Different displays of even the same manufacturer may have different pin assignments!

With this example connections it will be easy to connect displays with different connector pin-layouts, the pins are commonly named equal in the datasheet.

Serial Connections

It is *not* possible to connect most of the displays directly to the serial port. The signal has to be inverted. I use the following circuit to do that job.

FigureÂ 5.16.Â serialVFD: Serial Inverter

```
Computer                                          Display
                                                  (signal)
HDD Powerconnector
color(Voltage)

red(+5V)    ----------------------------------o----- +5V
                                              |
black(GND)  ----------------------o          .-.
                                  |          | | 10k
Serial(SUB-D  9Pin female)        |          | |
pin(signal)                       V*         '-'
           ___                                |
3(TxD)  --|___|--o---o      o-----------------o----- RxD
           10k   |   |      |
                 |   |     C|
                .-.  |   -----  BC547c (or similar NPN)
                | |  | B|  /  |
            10k | |  o--|-|   |
                '-'     |  \  |
                 |       --->-    A*
                 |          E|    |
                 |           |    |
5(GND)  ---------o-----------o----o--------------o-- GND
                                                 |*
Shield  -----------------------------------------o
optional

*connect near display
```

The pins on the different displays vary.

NEC Displays (FM20X2KB-AB):

CN1:

```
Pin 33 ------- RxD
(Testmode: connect pins 25 and 26, then power up)
```

CN2:

```
Pin 1  ------- +5V
Pin 2  ------- GND
```

KD Rev 2.1:

The blue connector (6pin in a row) (the important one!):

```
--------------------------------
| +5V  +5V  RxD  GND  GND  GND |
--------------------------------
```

Tip

Hold the display in that position where you can read the `KD Rev 2.1` marking normally!

The gray connector (10pin 2 rows):

```
Do not use. (the ATMEL ISP connector I guess)
```

The two jumpers next to the gray connector:

```
Normally not used.
You can activate two different testmodes with them.
```

FUTABA Displays (M402SD06GJ):

```
Pin 2   ------ +5V
Pin 4   ------ +5V
Pin 6   ------ +5V

Pin 10  ------ GND
Pin 12  ------ GND
Pin 14  ------ GND

Pin 19  ------ RxD
```

Siemens/Wincor Nixdorf BA63/66:

This display doesn't need the inverter! It must be connected directly with the serial port.

Check the serial port setup of the display, it has to be "9600 8N1". In most cases JP3 needs to be modified (closed) by the user!

```
Suggested Jumper Setting:
JP1 open, JP2 open, JP3 closed, JP4 open, JP5 open
```

More detailed information can be found in the users manual of the display.

Parallel Connections

NEC Displays (FM20X2KB-AB):

CN1:

```
parallel Port:                  Display (NEC):
Pin:    Name:                   Name:   Pin:

1       /STROBE --------------  /WR     17
2       D0      --------------  D0      15
3       D1      --------------  D1      13
4       D2      --------------  D2      11
5       D3      --------------  D3      9
6       D4      --------------  D4      7
7       D5      --------------  D5      5
8       D6      --------------  D6      3
9       D7      --------------  D7      1
10      /ACK    --o
                  |
11      BUSY    --o-----------  BUSY    27

                           o--  /CS     23
                           |
18 - 25 GND    ------------o--  GND     2,4,6,8,10,12,14,16,18,20,22,24,26,28,20,32,34
                           |*
Shield  -------------------o
optional

*connect near display
```

CN2:

```
Pin 1  ------ +5V
Pin 2  ------ GND
```

FUTABA Displays (M402SD06GJ):

```
parallel Port:                  Display (FUTABA):
Pin:    Name:                   Name:   Pin:

1       /STROBE --------------  /WR     17
2       D0      --------------  D0      15
3       D1      --------------  D1      13
4       D2      --------------  D2      11
5       D3      --------------  D3      9
6       D4      --------------  D4      7
7       D5      --------------  D5      5
8       D6      --------------  D6      3
9       D7      --------------  D7      1
10      /ACK    --o
                  |
11      BUSY    --o-----------  BUSY    20

                           o--  /SEL    18
                           |
18 - 25 GND    ------------o--  GND     10,12,14
                           |*
Shield  -------------------o
optional
                           0--  TEST    16
                           |
                +5V -------o--  +5V     2,4,6

*connect near display
```

Configuration in LCDd.conf

[serialVFD]

use_parallel = { *yes* | *no* }
: "no" if display connected serial, "yes" if connected parallel [default: no(serial)].

Port = *PORT*
: Portaddress where the LPT is. Used in parallel mode only. Usual values are 0x278, 0x378 and 0x3BC [default: 0x278].

PortWait = *DELAY*
: Set parallel port timing delay (us). Used in parallel mode only. [default: 2; legal: 0 - 255].

Device = *DEVICE*
: Device to use in serial mode. Usual values are `/dev/ttyS0` and `/dev/ttyS1` [default: `/dev/lcd`].

Custom-Characters = *CUSTOM-CHARACTERS*
: Number of Custom-Characters [default: Display-Type dependent].

Size = *WIDTH x HEIGHT*
: Specifies the size of the VFD. [default: 20x2]

Type = *CODE*
: Specifies the display type.[default: 0] The following type codes are available:

CODE	VFD model
0	NEC (FIPC8367 based) VFDs
1	KD Rev 2.1
2	Noritake VFDs
3	Futaba VFDs
4	IEE S03601-95B
5	IEE S03601-96-080
6	Futaba NA202SD08FA (allmost IEE compatible)
7	Samsung 20S207DA?
8	Siemens/Wincor Nixdorf BA63/66

Note

Noritake VFDs have not been tested yet. Feedback is welcome.

Brightness = *BRIGHTNESS*
: Set the initial brightness [default: 1000; legal: 0 - 1000] (4 steps 0-250, 251-500, 501-750, 751-1000)

OffBrightness = *OFFBRIGHTNESS*
: Set the initial off-brightness [default: 0; legal: 0 - 1000]. This value is used when the display is normally switched off in case LCDd is inactive. (4 steps 0-250, 251-500, 501-750, 751-1000)

Speed = { *1200* | *2400* | *9600* | *19200* | *115200* }
: Set the the baud rate communication with the VFD. If not given [default 9600].

ISO_8859_1 = { *yes* | *no* }
: Enable ISO-8859-1 compatibility [default is yes].

The shuttleVFD Driver

Features

The shuttleVFD drivers works with Shuttle Computer Group VFDs. These VFDs are found on various Shuttle XPC models. A partial list include Shuttle M1000, Shuttle M2000, Shuttle G5 3300m and Shuttle SG33G5M.

The display itself is a 20x1 character display. Each character cell is 5x8 pixels. It also has smaller row of specialized icons: clock, radio, music, CD/DVD, television, camera, rewind, record, play, pause, stop, fast-forward, reverse, repeat, mute, and a series of volume bars. Some or all of the icons can be displayed.

The display is driven by Princeton Technologies PT6314 VFD Controller according to the sources cited in the driver source. The PT6314 is probably driven off the Serial Peripheral Interface of a Cypress CY7C63723C low-speed USB controller that is connected to the mainboard via USB. Data is written to the Cypress CY7C63723C in bytes. The specifications for both the PT6314 and CY7C63723C are available, but not used when writing this driver. It seems that much more advanced uses are available if the specifications were to be used, specifically pixel addressing.

A current limitation of the driver is that it sleeps for some number of microseconds after writing to the display. If data is written to the display too quickly, it is simply discarded therefore the driver must sleep. The sleep time was experimentally found on a Shuttle M1000 machine and is hard-coded into the driver. This maybe different for other machines and configurations. This may occasionally cause the display to flicker or refresh unevenly across the display. It seems that the PT6314 can signal when it has read the data from the buffer, but this functionality is not used.

Configuration in LCDd.conf

[shuttleVFD]

There are no configuration options available for this display.

The sli Driver

This section talks about using LCDproc with Serial LCD Interface (SLI-OEM) boards from Wirz Electronics.

Note

This driver is intended for use with the original SLI-OEM boards from Wirz Electronics. As of 2010 Parallax, Inc. sells serial displays designed by Element Products (formerly Wirz Electronics). Due to the lack of documentation it is unclear if these devices do work with this driver.

Configuration in LCDd.conf

[sli]

Device = `DEVICE`
: Select the output device to use [default: `/dev/lcd`]

Speed = { *1200* | *2400* | *9600* | *19200* | *38400* | *57600* | *115200* }
: Set the the baud rate communication with the LCD. If not specified, the default is `19200`.

The stv5730 Driver

This section talks about using LCDproc with LCD displays that use the stv5730 chipset.

Configuration in LCDd.conf

[stv5730]

Port = *PORT*
: Specify the address of the parallel port the LCD is connected to. Common values for *PORT* are `0x278`, `0x378` and `0x3BC`. If not given, it defaults to `0x378`.

The SureElec Driver

Driver for the LCD modules (actually the controller board) available from the 'SURE electronics' shop (http://www.sureelectronics.net/).

These devices are PIC based controlled, using a serial communication protocol with the host. The actual connection to host is done through USB through a serial-to-USB converter (CP2102 USB to UART Bridge) integrated on the board.

Configuration in LCDd.conf

[SureElec]

Device = *DEVICE*
: Name of the device the display appears as. By default first USB serial device `/dev/ttyUSB0` is used.

Edition = *EDITION*
: Edition level of the device (can be 1, 2 or 3). The default is `2`.

Size = *WIDTH x HEIGHT*
: Set the display size in characters. This is required for edition 1 devices. For edition 2 & 3 devices this value, if defined, overrides the size read directly from the device.

Contrast = *CONTRAST*
: Select the display's contrast, `480` is the default. Permissible values are in the range of `0-1000`.

Brightness = *BRIGHTNESS*
: Select the display's brightness, `480` is the default. Permissible values are in the range of `0-1000`.

OffBrightness = *OFFBRIGHTNESS*
: Select the display's when the display is normally switched off in case LCDd is inactive, `100` is the default. Permissible values are in the range of `0-1000`.

The svga Driver

This section talks about using LCDproc with LCD displays that use the svga library.

Configuration in LCDd.conf

[svga]

Mode = *SCGALIB-MODE*
: svgalib mode to use [default: `G320x240x256`] *SCGALIB-MODE* can be any legal mode string for svgalib. See the svgalib(7) manual page for legal mode strings.

Size = *WIDTH x HEIGHT*
: set display size [default: `20x4`]

Contrast = *CONTRAST*
: Set the initial contrast. Legal values for *CONTRAST* are `0` - `1000`. If not given, the default value is `500`.

Note

This parameter can be set but it does not change anything in the driver.

Brightness = *BRIGHTNESS*
: Set the initial brightness [default: `1000`; legal: `1` - `1000`]

OffBrightness = *BRIGHTNESS*
: Set the initial off-brightness [default: `500`; legal: `1` - `1000`] This value is used when the display is normally switched off in case LCDd is inactive

The Toshiba T6963 Driver

This section talks about using LCDproc with LCD displays that use the T6963 chipset from Toshiba. Usually, this chipset is used on big graphical LCD displays that can often act as a screen.

The driver uses the text mode of the chipset with a custom font loaded which resembles the characters from HD44780 ROM 002 character set (European character set). If your display has a 'Font Select' pin it must be wired to use the 6x8 font.

Only displays in 'Single Scan' configurations are supported. Displays configured as 'Dual Scan' are currently not supported. Those use on-board memory differently. Check the datasheet of your display!

Connections

Warning

Displays using T6963 chipset come in a variety of pin-outs and power configurations. They usually require negative voltage for contrast and some displays have a negative voltage generator on-board. Therefore the wiring table below does only list signal names, not LCD pins. Be sure to get the correct datasheet for your display and identify the pins to use!

We do not give wiring examples for power, contrast, and other control lines. You have to figure out this from your display's datasheet!

TableÂ 5.31.Â T6963 wiring schematic

Parallel port		<->	LCD	
name	pin		name	pin

Parallel port		<->	LCD	
nSTRB	1		/WR	*
D0-D7	2-9	Â	DB0-DB7	*
nLF	14	Â	/CE	*
INIT	16	Â	C/D	*
nSEL	17	Â	/RD	*
Â	Â	VDD	FS	*

Compiling

Make sure that the T6963 files are built when you run configure. This can be done by specifying "--enable-drivers=all" or by "--enable-drivers=t6963".

Configuration in LCDd.conf

[t6963]

Size = *`WIDTH x HEIGHT`*
: Set display size in pixels [default: `128x64`]. The size in characters is automatically calculated assuming the font size is set to `6x8` for the display.

Port = *`PORT`*
: Specify the address of the parallel port the LCD is connected to. Common values for *`PORT`* are `0x278`, `0x378` and `0x3BC`. Legal values are `0x200` - `0x400`. If not given the default value is `0x378`.

bidirectional = { *`yes`* | *`no`* }
: Use LPT port in bi-directional mode. [default: `yes`; legal: `yes`, `no`]

  Most LPT ports can be used in bi-directional mode. It is required for proper timing of the display. Leave this on unless you experience problems.

ClearGraphic = { *`yes`* | *`no`* }
: Clear graphic memory on start-up [default: `no`; legal: `yes`, `no`]. The T6963 has a graphic and a text area which can be combined using several modes. This driver uses default OR-mode. Usually the graphic area is empty after power-on but if you see random garbage overlaying the text this option may be enabled.

delayBus = { *`yes`* | *`no`* }
: Use additional delay in read / write operations. [default: `no`; legal: `yes`, `no`]. The display can execute operations very fast. As the driver implements busy checking no additional delays are required. But if you experience problems you may try to slow down communication by enabling this setting.

The text Driver

The text driver simply outputs the content of the internal framebuffer to the current console using printf().

Configuration in LCDd.conf

[text]

Size = `WIDTH x HEIGHT`
: Set the display size [default: `20x4`]

The tyan Driver

This section talks about using LCDproc with LCD modules used in Tyan GS10 and GS12 barebones.

Features

The LCD modules used on the front side of the Tyan GS10 and GS12 series barebones consist of an LCD display by Winstar Display Co. LTD that is 16 characters wide and 2 lines high. To the right of the display there is a 6 button keypad: 4 array buttons and two buttons labeled `C` and `S`.

For more information see the LCD pack from the Tyan support page. Besides a useless old version of LCDproc it contains some documentation about the panel itself including a PDF specification of the LCD display by its manufacturer.

Configuration in LCDd.conf

[tyan]

Device = `DEVICE`
: Select the output device to use [default: `/dev/lcd`]

Speed = { `4800` | `9600` }
: Set the the baud rate communication with the LCD. If not given, the default is `9600`.

Size = `WIDTH x HEIGHT`
: set display size [default: `16x2`]

The ula200 Driver

General

The ULA-200 (short for German *USB-LCD-Ansteuerung*), manufactured and sold by ELV, is a small board that connects a HD44780-compatible display to the computer using the USB interface. Additionally it provides 6 digital inputs that can be used for keys.

The `ula200` driver controls this board supporting the features:

- display on a single-controller HD44780 display
- standard icons (heart, checkbox)
- backlight control
- input buttons
- *no* horizontal or vertical bars

Requirements

The driver uses `libftdi`, which again uses `libusb` for communication with the device, so no kernel driver is needed on Linux, and the driver can be used on other operating systems as well.

Note

When using a `libusb` based driver like `IOWarrior`, LCDd needs to be started as root.

Note

On Linux, you have to take care that the `ftdi_sio.ko` kernel module doesn't claim the ELV device. If you didn't change the IDs in the kernel driver (`ftdi_sio.c`), this should not matter.

Known problems

Sometimes the display hangs (the ACK response is not received) on shutdown. Reconnect the display in that case. Please do the same if it hangs while starting up. The latter only happens if it was not the first time LCDd talked to the display.

Implementation note

(by the driver's author Bernhard Walle)

The ULA-200 talks a text protocol which allows to display text using a high-level language, i.e. `STX s` *`len char0 char1`* ... `ETX`. It also allows low-level register access to the HD44780. So in theory, it would be possible to write a connection type for the `hd44780` driver and let the `hd44780` core do the rest. I tried this. It was slow and didn't work with user-specific characters (the hd44780 frequently changes this characters which seems to confuse the microcontroller, at least I cannot explain why it didn't work, there was garbage).

So I wrote a separate driver, the `ula200`, which uses the high-level language and should work for displays with all sizes. I only tested 20x4, so maybe for other sizes the positioning code may be adapted.

As I mentioned, there were problems with frequently changing the user-definable characters. I also tried to implement bar code in the `ula200` driver with similar effects. I gave it up because I don't need it personally and it can be done later. However, standard icons are implemented. The user-definable characters are set in startup and are not changed. This works like a charm. It is not possible to use character 0 with the high-level language (or at least it isn't documented how to escape it). It could be done with hd44780 code, but I replaced the character with a standard character which looks good.

Configuration in LCDd.conf

[ula200]

Size = *`WIDTH x HEIGHT`*

Select the LCD size [default: `20x4`]

KeyMap_A = *`KEY`* , KeyMap_B = *`KEY`* , KeyMap_C = *`KEY`* , KeyMap_D = *`KEY`* , KeyMap_E = *`KEY`* , KeyMap_F = *`KEY`*

If you have a non standard keypad you can associate any keystrings to keys. There are 6 input keys in the ULA-200 hardware that generate characters from '`A`' to '`F`'. Legal values for *`KEY`* are `Up`, `Down`, `Left`, `Right`, `Enter` and `Escape`.

The following table lists the built-in default mapping hardcoded in the driver.

KeyMap_A	`Up`
KeyMap_B	`Down`
KeyMap_C	`Left`
KeyMap_D	`Right`
KeyMap_E	`Enter`
KeyMap_F	`Escape`

You may leave it unchanged if you have a standard keypad. You can change it if you want to report other keystrings or have a non-standard keypad.

The vlsys_m428 driver

General

The vlsys_m428 driver handles the VFD/IR combination from VLSystem built into the case MonCaso 320 from the manufacturer Moneual. The driver functionality is rudimentary as it is derived from a reverse engineering from the Windows driver behaviour.

This driver writes only to the serial port, and it triggers no transmission activity of the combination. Thus, it cooperates with drivers reading only from the port. Other drivers are allowed to initialize the combination for IR operation.

The vlsys_m428 driver assumes the character encoding ISO 8859-1. Note: The mapping from the character encoding to the display encoding is not complete; furthermore some characters are approximated by similar ones.

Configuration

The only configuration parameter is Device, the path name of the serial device (a mapper between USB and serial line emulation). By default it is `/dev/ttyUSB0`.

The xosd Driver

This section talks about using LCDproc with libxosd.

libxosd is a system that displays text on top of your X-Windows screen, much like the on-screen display (OSD) used by most modern televisions and video-players.

Configuration in LCDd.conf

[xosd]

Size = *`WIDTH x HEIGHT`*
: set display size [default: `20x4`]

Offset = *`X-OFFSET x Y-OFFSET`*
: Offset (in pixels) of the top-left corner of LCDproc's xosd window from the top-left corner of the monitor. If not given, it defaults to `0x0`.

Font = *`FONT`*
: X font to use, in XLFD (X Logical Font Description) format, as given by the xfontsel property. E.g. `-*-terminus-*-r-*-*-*-320-*-*-*-*-*`.

  For best results it is recommended to use a mono-spaced font to mimic the aspect of a physical LCD display that most clients expect with regard to their screen layouts.

Contrast = *`CONTRAST`*
: Set the initial contrast. Legal values for *`CONTRAST`* are `0` - `1000`. If not given, the default value is `500`.

  #### Note

  This parameter can be set but it does not change anything in the driver.

Brightness = *`BRIGHTNESS`*
: Set the initial brightness [default: `1000`; legal: `1` - `1000`]

OffBrightness = *`BRIGHTNESS`*
: Set the initial off-brightness [default: `500`; legal: `1` - `1000`] This value is used when the display is normally switched off in case LCDd is inactive

The yard2 driver

General

The yard2 driver handles the yard2 LCD device. The driver functionality is currently limited to character LCD.

This driver writes via socket to yard2srvd deamon.

The yard2 driver assumes the character encoding ISO 8859-1.

Configuration

The only configuration parameter is Size. All other settings are done in yard2srvd daemon.

Parallel Port Troubleshooting

Unfortunately attaching an LCD module to a parallel port is not trivial.

In most cases it requires soldering abilities and basic knowledge of electronics.

The following hints might be helpful:

Check The Wiring

Wiring errors can easily be made. If you are inexperienced with the soldering iron better have someone solder it for you. Display modules are sensitive to electrostatic discharges, so touch an earthed surface (computer case, water pipes...) before you handle these.

Power Source Unregulated / Noisy

Make sure your power supply delivers steady 5 Volts without noise or interruptions. The bare wall plug-in transformer is often not suitable, though you can make it stabilized by adding an 7805 and a few capacitors. Some noise induced in the supply lines my be tricky to track, even if you have an oscilloscope.

Ground Lift

The power supply wires and especially the GND wires should be a little thicker than the other wires. If GND is not thick enough (or not existent, see 1) the resistance of the wire may cause differing GND potentials in the circuit. This may lead to strange display behaviour. It may also be wise to solder a 100nF capacitor directly to the GND and V_{CC} pins of the display.

Latchup

Never let the supply voltage get much below the io signal voltage. It may lead to a latchup condition which will destroy the controller chip on the display.

Contrast

If you don't see anything on your display it may be that your contrast voltage is set wrong. Turn your contrast potentiometer all the way to the end connected to GND. Contrast is highest then.

Beware

The module you got so ultra cheap may be an enhanced temperature model which needs a negative contrast voltage for sufficient contrast - see chapter 99 on how to make negative voltage.

Parallel Port Voltage

Many modern mainboards and especially notebooks will not nearly output 5V for a logic H as the older parallel ports did, because the operating voltage of computers is lower than 5V these days. I have measured voltages between 2.5V and 4V for logic H, which is barely within specification of the HD44780. If you account R_{CL} of your cable, this may not be enough and can cause unreliable operation.

Enable Signal Rise Time

If you ever read the HD44780 datasheet you will notice that somewhere in the 'signal timing' table is written: 'Enable Signal Rise Time max. 20nS'. That means the Voltage on the HD44780 pin called 'Enable' has to rise from 0 Volts to 5 Volts within 20 Nanoseconds and the other way round. They should better print that in big fat red letters, because most HD44780s are really picky about the enable signal rise time.

That is a Problem: If you count together the bad driving characteristics of the parallel port combined with the capacitance of flat ribbon cable you may easily get an order of magnitude slower rise time. Therefore you should only use really short cable (shorter than 50cm) for connecting the display to the parallel port. It may also be useful to use pull-up resistors on the display module or a schmitt-trigger.

Note

The rise time of a digital output can (usually) not be altered by software.

EMI

The cable from the parallel port to the display may be sensible to electromagnetic interference and may emit electromagnetic radiation. If you place your cellphone near the cable, you may get unexpected display readings, on the other hand your house neighbour may not be able to listen to his/her favourite radio station any more - so better use shielded cable and put the display in a metallic case, perhaps a computer case.

One or Two Black Lines

If you see one or two black lines on the display it means nothing more than that the display is powered and contrast voltage is present. If one or two black lines appear the controller has not been reset properly by the on chip power on reset generator. No need to worry - it will be reset by the LCDd software. But if the black line will not disappear although the wiring is working, the controller on the display may be defective.

Software Too Fast

If you have a super GHz computer it may happen that the signal timing generated by LCDd is too fast. Adjust DELAYMULT in the source file to a bigger value. Parallel port wirings usually don't permit to read back the busy flag of the controller chip, so timing must be adjust so that the controller never is busy.

LED Backlight

Check whether you need a resistor for your LED Backlight and which value it should have. If you forget the required resistor the backlighting LEDs might become hot and draw excessive current.

HD44780 Compatible

The original HD44780 controller that we advertise to support has become the industry standard for alphanumeric character displays. The original HD44780 is out of production. It has many successors from many manufactures, which sometimes won't tell you that their chips are 'compatible'.

To name a few: KS0066, KS0070, KS0076, LC7985, NT3881, SED1278, ST7066 ...

Miscellanea

This text has originally been taken from a message by Robin Adams <`robin@adams-online.de`>

Converted to Docbook and slightly modified May 2002, Rene Wagner <`reenoo@gmx.de`>

ChapterÂ 6.Â Running LCDproc

Table of Contents

Running LCDd
    Running LCDd from the command line
    The Command Line Options of LCDd
Running lcdproc
    The Command Line Options of lcdproc

Running LCDd

If you have installed the init-scripts you can simply start, stop and restart LCDd with the init-script.

Running LCDd from the command line

There are several reasons for running LCDd from the command line

- You don't want to install LCDd but run it from the source directory.
- You want to do some debugging.
- You want to get the output directly on stderr.
- ...

Note

If you run LCDd as a "normal" user, it will not change to the user specified in the config file. For some devices, mostly parallel port devices but also some USB devices, you will need root privileges anyway ;)

The simplest command that will run LCDd is the following. It is useful for running LCDd from the source directory, e.g. after building.

```
$ server/LCDd -c LCDd.conf
```

The Command Line Options of LCDd

Running **LCDd -h** gives you an overview of the currently available command line options.

ExampleÂ 6.1.Â LCDd -h

```
LCDd - LCDproc Server Daemon, 0.5dev

Copyright (c) 1999-2006 Selene Scriven, William Ferrell, and misc. contributors.
This program is released under the terms of the GNU General Public License.

Usage: LCDd [<options>]
  where <options> are:
    -h                  Display this help screen
    -c <config>         Use a configuration file other than /etc/lcdproc/LCDd.conf
    -d <driver>         Add a driver to use (overrides drivers in config file) [curses]
    -f                  Run in the foreground
    -a <addr>           Network (IP) address to bind to [127.0.0.1]
    -p <port>           Network port to listen for connections on [13666]
    -u <user>           User to run as [nobody]
    -w <waittime>       Time to pause at each screen (in seconds) [4]
    -s <bool>           If set, reporting will be done using syslog
    -r <level>          Report level [2]
    -i <bool>           Whether to rotate the server info screen
```

Running lcdproc

You will probably more often run **lcdproc** from the command line than you will run LCDd.

The Command Line Options of lcdproc

Running **lcdproc -h** gives you an overview of the currently available command line options.

ExampleÂ 6.2.Â lcdproc -h

```
lcdproc - LCDproc system status information viewer

Copyright (c) 1999-2006 Selene Scriven, William Ferrell, and misc. contributors.
This program is released under the terms of the GNU General Public License.

Usage: lcdproc [<options>] [<screens> ...]
  where <options> are
    -s <host>           connect to LCDd daemon on <host>
    -p <port>           connect to LCDd daemon using <port>
    -f                  run in foreground
    -e <delay>          slow down initial announcement of screens (in 1/100s)
    -c <config>         use a configuration file other than /etc/lcdproc/lcdproc.conf
    -h                  show this help screen
    -v                  display program version
  and <screens> are
    C CPU               detailed CPU usage
    P SMP-CPU           CPU usage overview (one line per CPU)
    G CPUGraph          CPU usage histogram
    L Load              load histogram
    M Memory            memory & swap usage
    S ProcSize          biggest processes size
    D Disk              filling level of mounted file systems
    I Iface             network interface usage
    B Battery           battery status
    T TimeDate          time & date information
    O OldTime           old time screen
    U Uptime            uptime screen
    K BigClock          big clock
    N MiniClock         minimal clock
    A About             credits page

Example:
    lcdproc -s my.lcdproc.server.com -p 13666 C M X
```

Note

You will not be able to connect to a remote server, unless it listens to the correct interface and port! See `LCDd.conf`: The [server] Section for details on the server setup.

ChapterÂ 7.Â Contact Us

Table of Contents

Errata
The LCDproc Mailing List
Reporting Bugs

Errata

Of course LCDproc is not perfect (yet). We do our very best to improve it, but in some cases we are very much restricted in our efforts.

The main reason for that is the fact that we do *not* have all the hardware people have written drivers for. Unfortunately some developers have kind of vanished and don't react to mails from the mailing list any more.

So, as far as drivers are concerned we rely on *you* as testers. We have developed elaborate "coding in the dark" skills over the time. E.g. the CFontz driver has been updated and ported to 0.4.3 without the developers having the hardware.

The LCDproc Mailing List

We are a bit lazy about the bug-tracking and whatever stuff on github. So, please contact us directly through the mailing list.

For details on how to subscribe to the list see https://lists.lcdproc.org/wws/info/lcdproc.

We like people to subscribe to the list with their real names. Of course we cannot and do not want to force you to do so. Anyway, we need to know your name, if you want to contribute code to LCDproc (legal issues of copyrights).

Reporting Bugs

Possibly you will find a bug in the LCDproc distribution. Before reporting this to the mailing list, please respect the following:

- Your system should be more or less up to date. This does not mean that you have to update from GNU/Linux kernel 2.2.x to 2.4.x or from 2.4.x to 2.6.x. But we would like to make sure that your problem is not related to a known bug in the kernel or maybe your compiler.
- Especially LCDd might need certain privileges to access a device. Make sure LCDd *has* the required rights to do so.
- When experiencing problems with LCDd, make sure that your hardware is OK. E.g. you should make sure that the wiring for your (in this case most likely parallel) device is correct.
- Make sure that you use the correct speed settings for your device. Incorrect speed settings (baud rate) are most likely to produce garbage scrolling on your display. Refer to the specifications of your device. If your device needs a speed setting that is not supported by LCDd send us a mail.
- Make sure that you have modified the configuration file according to your needs and that LCDd actually uses the configuration file. I.e. you might have to run LCDd with the `-c` option

Anyway, no question is too stupid to ask ;) Feel free to ask whatever you want. Unfortunately replying to mails takes time as well (a damn lot of time). So, if you want LCDproc to develop faster, please try to solve a problem yourself first.

BUT if you have actually FOUND A BUG we will be quite happy if you let us know. We NEED YOU as testers and appreciate any feedback.

AppendixÂ A.Â GNU Free Documentation License

Table of Contents

PREAMBLE
APPLICABILITY AND DEFINITIONS
VERBATIM COPYING
COPYING IN QUANTITY
MODIFICATIONS
COMBINING DOCUMENTS
COLLECTIONS OF DOCUMENTS
AGGREGATION WITH INDEPENDENT WORKS
TRANSLATION
TERMINATION
FUTURE REVISIONS OF THIS LICENSE
How to use this License for your documents

Version 1.1, March 2000

PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of

the version it refers to gives permission.

K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.
N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software