## Contents

1 Introduction .......................................................... 1

2 Installation ........................................................... 5
   2.1 Pre-requisites .................................................. 5
   2.2 Connecting the display ....................................... 6
   2.3 Installing from PyPI .......................................... 7

3 Python usage .......................................................... 9
   3.1 Color Model ..................................................... 10
   3.2 Landscape / Portrait Orientation .............................. 10
   3.3 Backlight Control ............................................. 10
   3.4 Examples ......................................................... 10
   3.5 Emulators ......................................................... 11

4 API Documentation .................................................. 13
   4.1 luma.lcd.aux .................................................. 13
   4.2 luma.lcd.device .............................................. 14

5 References ........................................................... 17

6 Contributing ........................................................ 19
   6.1 GitHub .......................................................... 19
   6.2 Contributors .................................................. 19

7 ChangeLog ........................................................... 21

8 The MIT License (MIT) ............................................. 23

Python Module Index ................................................ 25
Interfacing small LCD displays with the PCD8544 and ST7735 driver in Python using SPI on the Raspberry Pi and other linux-based single-board computers: the library provides a Pillow-compatible drawing canvas, and other functionality to support:

- scrolling/panning capability,
- terminal-style printing,
- state management,
- color/greyscale (where supported),
- dithering to monochrome

The PCD8544 display pictured below was used originally as the display for Nokia 5110 mobile phones, supporting a resolution of 84 x 48 monochrome pixels and a switchable backlight:
They are now commonly recycled, and sold on ebay with a breakout board and SPI interface.

The ST7735 display supports a resolution of 160 x 128 RGB pixels (18-bit / 262K colors) with a switchable backlight:

See also:
Further technical information for the specific device can be found in the datasheet below:

- PCD8544
- ST7735

As well as display drivers for the physical device, there are emulators that run in real-time (with pygame) and others that can take screenshots, or assemble animated GIFs, as per the examples below (source code for these is available in the examples repository).
Installation

Note: The library has been tested against Python 2.7, 3.4 and 3.5.
For Python3 installation, substitute the following in the instructions below.

- pip pip3,
- python python3,
- python-dev python3-dev,
- python-pip python3-pip.

It was originally tested with Raspbian on a rev.2 model B, with a vanilla kernel version 4.1.16+, and has subsequently been tested on Raspberry Pi model A, model B2 and 3B (Debian Jessie) and OrangePi Zero (Armbian Jessie).

Pre-requisites

Enable the SPI port:

```
$ sudo raspi-config
> Advanced Options > A6 SPI
```

If raspi-config is not available, enabling the SPI port can be done manually.

Ensure that the SPI kernel driver is enabled:

```
$ ls -l /dev/spi*
crw-rw---- 1 root spi 153, 0 Nov 25 08:32 /dev/spidev0.0
```

or:

```
crw-rw---- 1 root spi 153, 1 Nov 25 08:32 /dev/spidev0.1
```
Then add your user to the `spi` and `gpio` groups:

```bash
$ sudo usermod -a G spi pi
$ sudo usermod -a G gpio pi
```

Log out and back in again to ensure that the group permissions are applied successfully.

### Connecting the display

- If you don’t want to solder directly on the Pi, get 2.54mm 40 pin female single row headers, cut them to length, push them onto the Pi pins, then solder wires to the headers.
- If you need to remove existing pins to connect wires, be careful to heat each pin thoroughly, or circuit board traces may be broken.
- Triple check your connections. In particular, do not reverse VCC and GND.

The GPIO pins used for this SPI connection are the same for all versions of the Raspberry Pi, up to and including the Raspberry Pi 3 B.

**Warning:** There appears to be varying pin-out configurations on different modules - beware!

**Note:**

- If you’re already using the listed GPIO pins for Data/Command and/or Reset, you can select other pins and pass `bcm_DC` and/or `bcm_RST` argument specifying the new BCM pin numbers in your serial interface create call (this applies to both PCD8544 and ST7735).
- Because CE is connected to CE0, the display is available on SPI port 0. You can connect it to CE1 to have it available on port 1. If so, pass `port=1` in your serial interface create call.

### PCD8544

<table>
<thead>
<tr>
<th>LCD Pin</th>
<th>Name</th>
<th>Remarks</th>
<th>RPi Pin</th>
<th>RPi Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RST</td>
<td>Reset</td>
<td>P01-18</td>
<td>GPIO 24</td>
</tr>
<tr>
<td>2</td>
<td>CE</td>
<td>Chip Enable</td>
<td>P01-24</td>
<td>GPIO 8 (CE0)</td>
</tr>
<tr>
<td>3</td>
<td>DC</td>
<td>Data/Command</td>
<td>P01-16</td>
<td>GPIO 23</td>
</tr>
<tr>
<td>4</td>
<td>DIN</td>
<td>Data In</td>
<td>P01-19</td>
<td>GPIO 10 (MOSI)</td>
</tr>
<tr>
<td>5</td>
<td>CLK</td>
<td>Clock</td>
<td>P01-23</td>
<td>GPIO 11 (SCLK)</td>
</tr>
<tr>
<td>6</td>
<td>VCC</td>
<td>+3.3V Power</td>
<td>P01-01</td>
<td>3V3</td>
</tr>
<tr>
<td>7</td>
<td>LIGHT</td>
<td>Backlight</td>
<td>P01-12</td>
<td>GPIO 18 (PCM_CLK)</td>
</tr>
<tr>
<td>8</td>
<td>GND</td>
<td>Ground</td>
<td>P01-06</td>
<td>GND</td>
</tr>
</tbody>
</table>
## ST7735

<table>
<thead>
<tr>
<th>LCD Pin</th>
<th>Name</th>
<th>Remarks</th>
<th>RPi Pin</th>
<th>RPi Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GND</td>
<td>Ground</td>
<td>P01-06</td>
<td>GND</td>
</tr>
<tr>
<td>2</td>
<td>VCC</td>
<td>+3.3V Power</td>
<td>P01-01</td>
<td>3V3</td>
</tr>
<tr>
<td>3</td>
<td>NC</td>
<td>Not connected</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>4</td>
<td>NC</td>
<td>Not connected</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>5</td>
<td>NC</td>
<td>Not connected</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>6</td>
<td>RESET</td>
<td>Reset</td>
<td>P01-18</td>
<td>GPIO 24</td>
</tr>
<tr>
<td>7</td>
<td>A0</td>
<td>Data/command</td>
<td>P01-16</td>
<td>GPIO 23</td>
</tr>
<tr>
<td>8</td>
<td>SDA</td>
<td>SPI data</td>
<td>P01-19</td>
<td>GPIO 10 (MOSI)</td>
</tr>
<tr>
<td>9</td>
<td>SCK</td>
<td>SPI clock</td>
<td>P01-23</td>
<td>GPIO 11 (SCLK)</td>
</tr>
<tr>
<td>10</td>
<td>CS</td>
<td>SPI chip select</td>
<td>P01-24</td>
<td>GPIO 8 (CE0)</td>
</tr>
<tr>
<td>11</td>
<td>SD-SCK</td>
<td>SD serial clock</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>12</td>
<td>SD-MISO</td>
<td>SD data in</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>13</td>
<td>SD-MOSI</td>
<td>SD data out</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>14</td>
<td>SD-CS</td>
<td>SD chip select</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>15</td>
<td>LED+</td>
<td>Backlight control</td>
<td>P01-12</td>
<td>GPIO (PCM_CLK)</td>
</tr>
<tr>
<td>16</td>
<td>LED-</td>
<td>Backlight ground</td>
<td>P01-06</td>
<td>GND</td>
</tr>
</tbody>
</table>

### Installing from PyPI

Install the dependencies for library first with:

```
$ sudo usermod -a -G spi,gpio pi
$ sudo apt-get install python-dev python-pip
$ sudo -i pip install --upgrade pip
$ sudo apt-get purge python-pip
```

**Warning:** The default pip bundled with apt on Raspbian is really old, and can cause components to not be installed properly. Please ensure that **pip 9.0.1** is installed prior to continuing:

```
$ pip --version
pip 9.0.1 from /usr/local/lib/python2.7/dist-packages (python 2.7)
```

Proceed to install latest version of the library directly from **PyPI**:

```
$ sudo -H pip install --upgrade luma.lcd
```
Python usage

The PCD8544 is driven with python using the implementation in the `luma.lcd.device.pcd8544` class. Likewise, to drive the ST7735, use the `luma.lcd.device.st7735` class. Usage is very simple if you have ever used Pillow or PIL.

First, import and initialise the device:

```python
from luma.core.serial import spi
from luma.core.render import canvas
from luma.lcd.device import pcd8544, st7735

serial = spi(port=0, device=0, bcm_DC=23, bcm_RST=24)
device = pcd8544(serial)
```

The display device should now be configured for use. Note, all the example code snippets in this section are interchangeable between PCD8544 and ST7735 devices.

Both the `pcd8544` and `st7735` classes expose a `display()` method which takes an image with attributes consistent with the capabilities of the device. However, for most cases, for drawing text and graphics primitives, the canvas class should be used as follows:

```python
with canvas(device) as draw:
    draw.rectangle(device.bounding_box, outline="white", fill="black")
    draw.text((30, 40), "Hello World", fill="red")
```

The `luma.core.render.canvas` class automatically creates an `PIL.ImageDraw` object of the correct dimensions and bit depth suitable for the device, so you may then call the usual Pillow methods to draw onto the canvas.

As soon as the with scope is ended, the resultant image is automatically flushed to the device’s display memory and the `PIL.ImageDraw` object is garbage collected.
Color Model

Any of the standard PIL.ImageColor color formats may be used, but since the PCD8544 LCD is monochrome, only the HTML color names "black" and "white" values should really be used; in fact, by default, any value other than black is treated as white. The luma.core.render.canvas object does have a dither flag which if set to True, will convert color drawings to a dithered monochrome effect (see the 3d_box.py example, below).

```python
with canvas(device, dither=True) as draw:
    draw.rectangle((10, 10, 30, 30), outline="white", fill="red")
```

Note that there is no such limitation for the ST7735 device which supports 262K colour RGB images, whereby 24-bit RGB images are downscaled to 18-bit RGB.

Landscape / Portrait Orientation

By default the PCD8544 and ST7735 displays will both be oriented in landscape mode (84x48 and 160x128 pixels respectively). Should you have an application that requires the display to be mounted in a portrait aspect, then add a rotate=N parameter when creating the device:

```python
from luma.core.serial import spi
from luma.core.render import canvas
from luma.lcd.device import pcd8544

serial = spi(port=0, device=0, bcm_DC=23, bcm_RST=23)
device = pcd8544(serial, rotate=1)

# Box and text rendered in portrait mode
with canvas(device) as draw:
    draw.rectangle(device.bounding_box, outline="white", fill="black")
    draw.text((10, 40), "Hello World", fill="red")
```

$N$ should be a value of 0, 1, 2 or 3 only, where 0 is no rotation, 1 is rotate 90° clockwise, 2 is 180° rotation and 3 represents 270° rotation.

The device.size, device.width and device.height properties reflect the rotated dimensions rather than the physical dimensions.

Backlight Control

These displays typically require a backlight to illuminate the liquid crystal display: the luma.lcd.aux.backlight class allows a BCM pin to be specified to control the backlight through software.

Examples

After installing the library, download the luma.examples directory and try running the following examples:
<table>
<thead>
<tr>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3d_box.py</td>
<td>Rotating 3D box wireframe &amp; color dithering</td>
</tr>
<tr>
<td>bounce.py</td>
<td>Display a bouncing ball animation and frames per second</td>
</tr>
<tr>
<td>carousel.py</td>
<td>Showcase viewport and hotspot functionality</td>
</tr>
<tr>
<td>clock.py</td>
<td>An analog clockface with date &amp; time</td>
</tr>
<tr>
<td>colors.py</td>
<td>Color rendering demo</td>
</tr>
<tr>
<td>crawl.py</td>
<td>A vertical scrolling demo, which should be familiar</td>
</tr>
<tr>
<td>demo.py</td>
<td>Use misc draw commands to create a simple image</td>
</tr>
<tr>
<td>game_of_life.py</td>
<td>Conway’s game of life</td>
</tr>
<tr>
<td>grayscale.py</td>
<td>Greyscale rendering demo</td>
</tr>
<tr>
<td>invaders.py</td>
<td>Space Invaders demo</td>
</tr>
<tr>
<td>maze.py</td>
<td>Maze generator</td>
</tr>
<tr>
<td>perfloop.py</td>
<td>Simple benchmarking utility to measure performance</td>
</tr>
<tr>
<td>pi_logo.py</td>
<td>Display the Raspberry Pi logo (loads image as .png)</td>
</tr>
<tr>
<td>savepoint.py</td>
<td>Example of savepoint/restore functionality</td>
</tr>
<tr>
<td>starfield.py</td>
<td>3D starfield simulation</td>
</tr>
<tr>
<td>sys_info.py</td>
<td>Display basic system information</td>
</tr>
<tr>
<td>terminal.py</td>
<td>Simple println capabilities</td>
</tr>
<tr>
<td>tv_snow.py</td>
<td>Example image-blitting</td>
</tr>
<tr>
<td>welcome.py</td>
<td>Unicode font rendering &amp; scrolling</td>
</tr>
</tbody>
</table>

See the README in that project for further information on how to run the demos.

**Emulators**

There are various display emulators available for running code against, for debugging and screen capture functionality:

- The `luma.emulator.device.capture` device will persist a numbered PNG file to disk every time its `display()` method is called.
- The `luma.emulator.device.gifanim` device will record every image when its `display()` method is called, and on program exit (or Ctrl-C), will assemble the images into an animated GIF.
- The `luma.emulator.device.pygame` device uses the `pygame` library to render the displayed image to a pygame display surface.

Invoke the demos with:

```
$ python examples/clock.py -d capture
```

or:

```
$ python examples/clock.py -d pygame
```

**Note:** `Pygame` is required to use any of the emulated devices, but it is NOT installed as a dependency by default, and so must be manually installed before using any of these emulation devices (e.g. `pip install pygame`). See the install instructions in `luma.emulator` for further details.
LCD display driver for PCD8544 devices.

**luma.lcd.aux**

```python
class luma.lcd.aux.backlight (gpio=None, bcm_LIGHT=18):
    Bases: object

    Controls a backlight, assumed to be on GPIO 18 (PWM_CLK0) by default.

    Parameters
    ----------
    * gpio – GPIO interface (must be compatible with RPi.GPIO).
    * bcm_LIGHT (int) – the GPIO pin to use for the backlight.
```
enable(value)
Switches on the backlight on and off.

Parameters value (bool) – Switched on when True supplied, else False switches it off.

luma.lcd.device

class luma.lcd.device.pcd8544 (serial_interface=None, rotate=0, **kwargs)
Bases: luma.core.device.device
Encapsulates the serial interface to the monochrome PCD8544 LCD display hardware. On creation, an initialization sequence is pumped to the display to properly configure it. Further control commands can then be called to affect the brightness and other settings.

Parameters
• serial_interface – the serial interface (usually a :py:class:`luma.core.serial.spi` instance) to delegate sending data and commands through.
• rotate (int) – an integer value of 0 (default), 1, 2 or 3 only, where 0 is no rotation, 1 is rotate 90° clockwise, 2 is 180° rotation and 3 represents 270° rotation.

capabilities (width, height, rotate, mode='1')
Assigns attributes such as width, height, size and bounding_box correctly oriented from the supplied parameters.

Parameters
• width (int) – the device width
• height (int) – the device height
• rotate (int) – an integer value of 0 (default), 1, 2 or 3 only, where 0 is no rotation, 1 is rotate 90° clockwise, 2 is 180° rotation and 3 represents 270° rotation.
• mode (str) – the supported color model, one of “1”, “RGB” or “RGBA” only.

cleanup()
Attempt to switch the device off or put into low power mode (this helps prolong the life of the device), clear the screen and close resources associated with the underlying serial interface.

This is a managed function, which is called when the python process is being shutdown, so shouldn’t usually need be called directly in application code.
clear()
Initializes the device memory with an empty (blank) image.

command (*cmd)
Sends a command or sequence of commands through to the delegated serial interface.

contrast (value)
Sets the LCD contrast
data (data)
Sends a data byte or sequence of data bytes through to the delegated serial interface.
display (image)
Takes a 1-bit PIL.Image and dumps it to the PCD8544 LCD display.
hide()
Switches the display mode OFF, putting the device in low-power sleep mode.
**preprocess** (*image*)

Provides a preprocessing facility (which may be overridden) whereby the supplied image is rotated according to the device’s rotate capability. If this method is overridden, it is important to call the super.

**Parameters**
- *image* (*PIL.Image.Image*) – An image to pre-process

**Returns**
A new processed image

**Return type**
PIL.Image.Image

**show()**

Sets the display mode ON, waking the device out of a prior low-power sleep mode.

```python
class luma.lcd.device.st7735(serial_interface=None, rotate=0, framebuffer='diff_to_previous', bgr=False, **kwargs)
```

Encapsulates the serial interface to the 262K color (6-6-6 RGB) ST7735 LCD display hardware. On creation, an initialization sequence is pumped to the display to properly configure it. Further control commands can then be called to affect the brightness and other settings.

**Parameters**
- *serial_interface* – the serial interface (usually a :py:class:`luma.core.serial.spi` instance) to delegate sending data and commands through.
- *rotate* (*int*) – an integer value of 0 (default), 1, 2 or 3 only, where 0 is no rotation, 1 is rotate 90° clockwise, 2 is 180° rotation and 3 represents 270° rotation.
- *framebuffer* (*str*) – Framebuffering strategy, currently values of “diff_to_previous” or “full_frame” are only supported
- *bgr* (*bool*) – set to True if device pixels are BGR order (rather than RGB)

New in version 0.3.0.

**capabilities** (*width, height, rotate, mode='1'*)

Assigns attributes such as *width*, *height*, *size* and *bounding_box* correctly oriented from the supplied parameters.

**Parameters**
- *width* (*int*) – the device width
- *height* (*int*) – the device height
- *rotate* (*int*) – an integer value of 0 (default), 1, 2 or 3 only, where 0 is no rotation, 1 is rotate 90° clockwise, 2 is 180° rotation and 3 represents 270° rotation.
- *mode* (*str*) – the supported color model, one of “1”, “RGB” or “RGBA” only.

**cleanup()**

Attempt to switch the device off or put into low power mode (this helps prolong the life of the device), clear the screen and close resources associated with the underlying serial interface.

This is a managed function, which is called when the python process is being shutdown, so shouldn’t usually need be called directly in application code.

**clear()**

Initializes the device memory with an empty (blank) image.

**command**(cmd, *args)

Sends a command and an (optional) sequence of arguments through to the delegated serial interface. Note that the arguments are passed through as data.
**contrast** *(level)*

NOT SUPPORTED

**Parameters**

- **level** *(int)* – Desired contrast level in the range of 0-255.

**data** *(data)*

Sends a data byte or sequence of data bytes through to the delegated serial interface.

**display** *(image)*

Renders a 24-bit RGB image to the ST7735 LCD display. The 8-bit RGB values are passed directly to the devices internal storage, but only the 6 most-significant bits are used by the display.

**Parameters**

- **image** *(PIL.Image.Image)* – the image to render

**hide** ()

Switches the display mode OFF, putting the device in low-power sleep mode.

**preprocess** *(image)*

Provides a preprocessing facility (which may be overridden) whereby the supplied image is rotated according to the device’s rotate capability. If this method is overridden, it is important to call the super

**Parameters**

- **image** *(PIL.Image.Image)* – An image to pre-process

**Returns**

A new processed image

**Return type**

PIL.Image.Image

**show** ()

Sets the display mode ON, waking the device out of a prior low-power sleep mode.
CHAPTER 5

References

- http://elinux.org/Rpi_Low-level_peripherals#General_Purpose_Input.2FOutput_.28GPIO.29
- http://binerry.de/post/25787954149/pcd8544-library-for-raspberry-pi
- https://projects.drogon.net/raspberry-pi/wiringpi/pins/
- http://www.henningkarlsen.com/electronics/t_imageconverter_mono.php
- https://vimeo.com/41393421
- http://fritzing.org
- http://learn.adafruit.com/1-8-tft-display
- http://elinux.org/images/1/19/Passing_Time_With_SPI_Framebuffer_Driver.pdf
- https://github.com/notro/fbtft
- https://github.com/rm-hull/st7735fb
- http://www.whence.com/rpi/
- http://harizanov.com/product/1-8-tft-display-for-raspberry-pi/
Pull requests (code changes / documentation / typos / feature requests / setup) are gladly accepted. If you are intending to introduce some large-scale changes, please get in touch first to make sure we’re on the same page: try to include a docstring for any new method or class, and keep method bodies small, readable and PEP8-compliant. Add tests and strive to keep the code coverage levels high.

GitHub

The source code is available to clone at: https://github.com/rm-hull/luma.lcd.git

Contributors

• Thijs Triemstra (@thijstriemstra)
• Dougie Lawson (@dougielawson)
• WsMithril (@WsMithril)
• Peter Martin (@pe7er)
## ChangeLog

<table>
<thead>
<tr>
<th>Version</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
</table>
| **Upcoming** | • Add ST7735 Color TFT LCD display driver  
• Removed width and height parameters from device constructors  
• **BREAKING CHANGES:** Move backlight class to different package | |
| 0.2.3   | • Allow PCD8544 driver constructor to accept any args | 2017/03/02 |
| 0.2.2   | • Restrict exported Python symbols from `luma.lcd.device` | 2017/03/02 |
| 0.2.1   | • Bugfix: Backlight didn’t switch off properly  
• Add tests | 2017/01/23 |
| 0.2.0   | • **BREAKING CHANGES:** Package rename to `luma.lcd` | 2017/01/13 |
| 0.0.1   | • Bit-bang version using `wiringPi` | 2013/01/28 |
CHAPTER 8

The MIT License (MIT)

Copyright (c) 2013-17 Richard Hull & Contributors

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
Python Module Index

luma.lcd, 13
luma.lcd.aux, 13
luma.lcd.device, 14
Index

B
backlight (class in luma.lcd.aux), 13

C
capabilities() (luma.lcd.device.pcd8544 method), 14
capabilities() (luma.lcd.device.st7735 method), 15
cleanup() (luma.lcd.device.pcd8544 method), 14
cleanup() (luma.lcd.device.st7735 method), 15
clear() (luma.lcd.device.pcd8544 method), 14
clear() (luma.lcd.device.st7735 method), 15
command() (luma.lcd.device.pcd8544 method), 14
command() (luma.lcd.device.st7735 method), 15
contrast() (luma.lcd.device.pcd8544 method), 14
contrast() (luma.lcd.device.st7735 method), 15

data() (luma.lcd.device.pcd8544 method), 14
data() (luma.lcd.device.st7735 method), 16
display() (luma.lcd.device.pcd8544 method), 14
display() (luma.lcd.device.st7735 method), 16

E
enable() (luma.lcd.aux.backlight method), 13

H
hide() (luma.lcd.device.pcd8544 method), 14
hide() (luma.lcd.device.st7735 method), 16

L
luma.lcd (module), 13
luma.lcd.aux (module), 13
luma.lcd.device (module), 14

P
pcd8544 (class in luma.lcd.device), 14
preprocess() (luma.lcd.device.pcd8544 method), 14
preprocess() (luma.lcd.device.st7735 method), 16

S
show() (luma.lcd.device.pcd8544 method), 15
show() (luma.lcd.device.st7735 method), 16
st7735 (class in luma.lcd.device), 15