<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Installation</td>
<td>7</td>
</tr>
<tr>
<td>2.1</td>
<td>Pre-requisites</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>Connecting the display</td>
<td>8</td>
</tr>
<tr>
<td>2.3</td>
<td>Installing from PyPI</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>Python usage</td>
<td>11</td>
</tr>
<tr>
<td>3.1</td>
<td>Pixel Drivers</td>
<td>11</td>
</tr>
<tr>
<td>3.2</td>
<td>Seven-Segment Drivers</td>
<td>12</td>
</tr>
<tr>
<td>3.3</td>
<td>Backlight Control</td>
<td>13</td>
</tr>
<tr>
<td>3.4</td>
<td>Examples</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>API Documentation</td>
<td>15</td>
</tr>
<tr>
<td>4.1</td>
<td>Upgrading</td>
<td>15</td>
</tr>
<tr>
<td>4.2</td>
<td><code>luma.lcd.device</code></td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>References</td>
<td>25</td>
</tr>
<tr>
<td>6</td>
<td>Contributing</td>
<td>27</td>
</tr>
<tr>
<td>6.1</td>
<td>GitHub</td>
<td>27</td>
</tr>
<tr>
<td>6.2</td>
<td>Contributors</td>
<td>27</td>
</tr>
<tr>
<td>7</td>
<td>ChangeLog</td>
<td>29</td>
</tr>
<tr>
<td>8</td>
<td>The MIT License (MIT)</td>
<td>31</td>
</tr>
<tr>
<td>Python Module Index</td>
<td></td>
<td>33</td>
</tr>
<tr>
<td>Index</td>
<td></td>
<td>35</td>
</tr>
</tbody>
</table>
Interfacing small LCD displays with the PCD8544, ST7735, HT1621 and UC1701X 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:
The HT1621 display (as purchased) supports six 7-segment characters with a switchable backlight:

The UC1701X display supports a resolution of 128 x 64 monochrome pixels with a switchable backlight:
The ST7567 display supports a resolution of 128 x 64 monochrome pixels:
See also:

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

- PCD8544
- ST7735
- HT1621
- UC1701X
- ILI9341

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.
Note: The library has been tested against Python 2.7, 3.4, 3.5 and 3.6.

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 (both Raspbian Jessie and Stretch) models A, B2, 3B, Zero, Zero W and OrangePi Zero (Armbian Jessie).

### 2.1 Pre-requisites

Enable the SPI port:

```bash
$ 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:

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

or:

```bash
$ lsmod | grep spi
  spi_bcm2835  6678  0
```

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.
2.2 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 `gpio_DC` and/or `gpio_RST` argument specifying the new GPIO pin numbers in your serial interface create call (this applies to PCD8544, ST7567 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.

### 2.2.1 PCD8544

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

### 2.2.2 ST7735

Depending on the board you bought, there may be different names for the same pins, as detailed below.
2.2.3 ILI9341

No support for the touch-screen, leave the MISO and Touch pins disconnected. Depending on the board you bought, there may be different names for the same pins, as detailed below.

<table>
<thead>
<tr>
<th>LCD Pin</th>
<th>Remarks</th>
<th>RPi Pin</th>
<th>RPi Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCC</td>
<td>+3.3V Power</td>
<td>P01-01</td>
<td>3V3</td>
</tr>
<tr>
<td>GND</td>
<td>Ground</td>
<td>P01-06</td>
<td>GND</td>
</tr>
<tr>
<td>CS</td>
<td>SPI chip select</td>
<td>P01-24</td>
<td>GPIO 8 (CE0)</td>
</tr>
<tr>
<td>RESET or RST</td>
<td>Reset</td>
<td>P01-18</td>
<td>GPIO 24</td>
</tr>
<tr>
<td>DC</td>
<td>Data/command</td>
<td>P01-16</td>
<td>GPIO 23</td>
</tr>
<tr>
<td>SDA (MOSI)</td>
<td>SPI data</td>
<td>P01-19</td>
<td>GPIO 10 (MOSI)</td>
</tr>
<tr>
<td>SCK or CLK</td>
<td>SPI clock</td>
<td>P01-23</td>
<td>GPIO 11 (SCLK)</td>
</tr>
<tr>
<td>LED</td>
<td>Backlight control</td>
<td>P01-12</td>
<td>GPIO 18 (PCM_CLK)</td>
</tr>
<tr>
<td>LED-</td>
<td>Backlight ground</td>
<td>P01-06</td>
<td>GND</td>
</tr>
</tbody>
</table>

2.2.4 ST7567

This driver is designed for the ST7567 in 4-line SPI mode and does not include parallel bus support.
Pin names may differ across different breakouts, but will generally be something like the below.

<table>
<thead>
<tr>
<th>LCD Pin</th>
<th>Remarks</th>
<th>RPi Pin</th>
<th>RPi Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>GND</td>
<td>Ground</td>
<td>P01-06</td>
<td>GND</td>
</tr>
<tr>
<td>3v3</td>
<td>+3.3V Power</td>
<td>P01-01</td>
<td>3V3</td>
</tr>
<tr>
<td>RESET or RST</td>
<td>Reset</td>
<td>P01-18</td>
<td>GPIO 24</td>
</tr>
<tr>
<td>SA0 or D/C</td>
<td>Data/command</td>
<td>P01-16</td>
<td>GPIO 23</td>
</tr>
<tr>
<td>SDA or DATA</td>
<td>SPI data</td>
<td>P01-19</td>
<td>GPIO 10 (MOSI)</td>
</tr>
<tr>
<td>SCK or CLK</td>
<td>SPI clock</td>
<td>P01-23</td>
<td>GPIO 11 (SCLK)</td>
</tr>
<tr>
<td>CS</td>
<td>SPI chip select</td>
<td>P01-24</td>
<td>GPIO 8 (CE0)</td>
</tr>
</tbody>
</table>
2.2.5 HT1621

<table>
<thead>
<tr>
<th>LCD Pin</th>
<th>Remarks</th>
<th>RPi Pin</th>
<th>RPi Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>GND</td>
<td>Ground</td>
<td>P01-06</td>
<td>GND</td>
</tr>
<tr>
<td>VCC</td>
<td>+3.3V Power</td>
<td>P01-01</td>
<td>3V3</td>
</tr>
<tr>
<td>DAT</td>
<td>SPI data</td>
<td>P01-19</td>
<td>GPIO 10 (MOSI)</td>
</tr>
<tr>
<td>WR</td>
<td>SPI clock</td>
<td>P01-23</td>
<td>GPIO 11 (SCLK)</td>
</tr>
<tr>
<td>CS</td>
<td>SPI chip select</td>
<td>P01-24</td>
<td>GPIO 8 (CE0)</td>
</tr>
<tr>
<td>LED</td>
<td>Backlight control</td>
<td>P01-12</td>
<td>GPIO 18 (PCM_CLK)</td>
</tr>
</tbody>
</table>

2.2.6 UC1701X

The UC1701X doesn’t appear to work from 3.3V, but does on the 5.0V rail.

<table>
<thead>
<tr>
<th>LCD Pin</th>
<th>Remarks</th>
<th>RPi Pin</th>
<th>RPi Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROM_IN</td>
<td>Unused</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROM_OUT</td>
<td>Unused</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROM_SCK</td>
<td>Unused</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROM_CS</td>
<td>Unused</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LED A</td>
<td>Backlight control</td>
<td>P01-12</td>
<td>GPIO 18 (PCM_CLK)</td>
</tr>
<tr>
<td>VSS</td>
<td>Ground</td>
<td>P01-06</td>
<td>GND</td>
</tr>
<tr>
<td>VDD</td>
<td>+5.0V</td>
<td>P01-02</td>
<td>5V0</td>
</tr>
<tr>
<td>SCK</td>
<td>SPI clock</td>
<td>P01-23</td>
<td>GPIO 11 (SCLK)</td>
</tr>
<tr>
<td>SDA</td>
<td>SPI data</td>
<td>P01-19</td>
<td>GPIO 10 (MOSI)</td>
</tr>
<tr>
<td>RS</td>
<td>Data/command</td>
<td>P01-16</td>
<td>GPIO 23</td>
</tr>
<tr>
<td>RST</td>
<td>Reset</td>
<td>P01-18</td>
<td>GPIO 24</td>
</tr>
<tr>
<td>CS</td>
<td>SPI chip select</td>
<td>P01-24</td>
<td>GPIO 8 (CE0) Chip Select</td>
</tr>
</tbody>
</table>

2.3 Installing from PyPI

First, install the dependencies for the library with:

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

And finally, install the latest version of the library directly from PyPI with:

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

**Warning:** The default pip bundled with apt on Raspbian Jessie 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)
```
3.1 Pixel Drivers

The PCD8544 is driven with python using the implementation in the `luma.lcd.device.pcd8544` class. Likewise, to drive the ST7735, ST7567 or UC1701X, use the `luma.lcd.device.st7735`, `luma.lcd.device.st7567` or `luma.lcd.device.uc1701x` class respectively. For the ILI9341, use `luma.lcd.device.ili9341`. Usage is very simple if you have ever used Pillow or PIL.

First, import and initialise the device:

```python
from luma.core.interface.serial import spi
from luma.core.render import canvas
from luma.lcd.device import pcd8544, st7735, st7567, uc1701x, ili9341

serial = spi(port=0, device=0, gpio_DC=23, gpio_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.

The `pcd8544`, `st7735`, `st7567`, `uc1701x` and `luma.lcd.device.ili9341` classes all 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.
3.1.1 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 or ILI9341 devices which supports 262K colour RGB images, whereby 24-bit RGB images are downscaled to 18-bit RGB.

3.1.2 Landscape / Portrait Orientation

By default the PCD8544, ST7735, UC1701X and ILI9341 displays will all be oriented in landscape mode (84x48, 160x128, 128x64 and 320x240 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.interface.serial import spi
from luma.core.render import canvas
from luma.lcd.device import pcd8544

serial = spi(port=0, device=0, gpio_DC=23, gpio_RST=24)
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.

3.2 Seven-Segment Drivers

The HT1621 is driven with the luma.lcd.device.ht1621 class, but is not accessed directly: it should be wrapped with the luma.core.virtual.sevensegment wrapper, as follows:

```python
from luma.core.virtual import sevensegment
from luma.lcd.device import ht1621

device = ht1621()
seg = sevensegment(device)
```

The seg instance now has a text property which may be assigned, and when it does will update all digits according to the limited alphabet the 7-segment displays support. For example, assuming there are 2 cascaded modules, we have 16 character available, and so can write:

```python
seg.text = "HELLO"
```
Rather than updating the whole display buffer, it is possible to update ‘slices’, as per the below example:

```
seg.text[0:5] = "BYE"
```

This replaces `HELLO` in the previous example, replacing it with `BYE`. The usual python idioms for slicing (inserting / replacing / deleting) can be used here, but note if inserted text exceeds the underlying buffer size, a `ValueError` is raised.

Floating point numbers (or text with '.') are handled slightly differently - the decimal-place is fused in place on the character immediately preceding it. This means that it is technically possible to get more characters displayed than the buffer allows, but only because dots are folded into their host character.

### 3.3 Backlight Control

These displays typically require a backlight to illuminate the liquid crystal display: by default GPIO 18 (PWM_CL0) is used as the backlight control pin. This can be changed by specifying `gpio_LIGHT=n` when initializing the device. The backlight can be programmatically switched on and off by calling `device.backlight(True)` or `device.backlight(False)` respectively.

### 3.4 Examples

After installing the library, head over to the luma.examples repository. Details of how to run the examples is shown in the example repo’s README.
API DOCUMENTATION

LCD display drivers.

4.1 Upgrading

**Warning:** Version 2.0.0 was released on 2 June 2019: this came with the removal of the `luma.lcd.aux.backlight` class. The equivalent functionality has now been subsumed into the device classes that have a backlight capability.

4.2 `luma.lcd.device`

Collection of serial interfaces to LCD devices.

```python
class luma.lcd.device.ht1621(gpio=None, width=6, rotate=0, WR=11, DAT=10, CS=8, **kwargs):
    Bases: luma.lcd.device.backlit_device

    Serial interface to a seven segment HT1621 monochrome LCD display.

    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.
```

15
Parameters

- **gpio** – The GPIO library to use (usually RPi.GPIO) to delegate sending data and commands through.
- **width** (*int*) – The number of 7 segment characters laid out horizontally.
- **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.
- **WR** (*int*) – The write (SPI clock) pin to connect to, default BCM 11.
- **DAT** (*int*) – The data pin to connect to, default BCM 10.
- **CS** (*int*) – The chip select pin to connect to, default BCM 8.

New in version 0.4.0.

**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 reset the device & switching it off prior to exiting the python process.

**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**(level)

Switches the display contrast to the desired level, in the range 0-255. Note that setting the level to a low (or zero) value will not necessarily dim the display to nearly off. In other words, this method is **NOT** suitable for fade-in/out animation.

**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)

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 method.

**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.deviceILI9341(serial_interface=None, width=320, height=240, rotate=0, framebuffer='diff_to_previous', h_offset=0, v_offset=0, bgr=False, **kwargs)
```

Bases: luma.lcd.device.backlit_device

Serial interface to a 262k color (6-6-6 RGB) ILI9341 LCD display.

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 luma.core.interface.serial.spi instance) to delegate sending data and commands through.
- `width (int)` – The number of pixels laid out horizontally.
- `height` – The number of pixels laid out vertically.
- `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).
- `h_offset (int)` – Horizontal offset (in pixels) of screen to device memory (default: 0).
- `v_offset (int)` – Vertical offset (in pixels) of screen to device memory (default: 0).

New in version 2.2.0.

```python
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.

```python
cleanup()
```
Attempt to reset the device & switching it off prior to exiting the python process.

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

```python
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.

```python
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 ILI9341 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** method.

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.

#### luma.lcd.device.pcd8544

**class** luma.lcd.device.pcd8544 *(serial_interface=None, rotate=0, **kwargs)*

**Bases:** luma.lcd.device.backlit_device

Serial interface to a monochrome PCD8544 LCD display.

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** *(Serial Interface)* – The serial interface (usually a luma.core.interface.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 reset the device & switching it off prior to exiting the python process.

**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 method.

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.

class luma.lcd.device.st7567 \(\) (serial_interface=None, rotate=0, **kwargs)
Bases: luma.lcd.device.backlit_device
Serial interface to a monochrome ST7567 128x64 pixel LCD display.
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 luma.core.interface.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.

New in version 1.1.0.

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 reset the device & switching it off prior to exiting the python process.

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 ST7567 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 method.

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.

class luma.lcd.device.st7735 (serial_interface=None, width=160, height=128, rotate=0, framebuffer='diff_to_previous', h_offset=0, v_offset=0, bgr=False, **kwargs)

Bases: luma.lcd.device.backlit_device

Serial interface to a 262K color (6-6-6 RGB) ST7735 LCD display.

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 luma.core.interface.serial.spi instance) to delegate sending data and commands through.
- width (int) – The number of pixels laid out horizontally.
- height – The number of pixels laid out vertically.
- 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).
- h_offset (int) – Horizontal offset (in pixels) of screen to device memory (default: 0).
- v_offset (int) – Vertical offset (in pixels) of screen to device memory (default: 0).

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 reset the device & switching it off prior to exiting the python process.

`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` method.

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.st7920 (serial_interface=None, width=128, height=64, rotate=0, framebuffer='diff_to_previous', **kwargs)
```

Bases: `luma.core.device.device`

`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.

If `persist` is True, the device will not be switched off.

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**(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)

Should be overridden in sub-classed implementations.

Parameters

  **image** *(PIL.Image.Image)* – An image to display.

Raises **NotImplementedError** –

**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` method.

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.

**class** luma.lcd.device.uc1701x *(serial_interface=None, rotate=0, **kwargs)*

Bases: luma.lcd.device.backlit_device

Serial interface to a monochrome UC1701X LCD display.

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 *luma.core.interface.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.

New in version 0.5.0.

**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 reset the device & switching it off prior to exiting the python process.

**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 UC1701X 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 method.

**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.
REFERENCES

- http://elinux.org/Rpi_Low-level_peripherals#General_Purpose_Input/Output_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/
CONTRIBUTING

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.

6.1 GitHub

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

6.2 Contributors

- Thijs Triemstra (@thijstriemstra)
- Dougie Lawson (@dougielawson)
- WsMithril (@WsMithril)
- Peter Martin (@pe7er)
- Saumyakanta Sahoo (@somu1795)
- Philip Howard (@Gadgetoid)
- Ricardo Amendoeira (@ric2b)
- Kevin Stone (@kevinastone)
## Chapter Seven

### ChangeLog

<table>
<thead>
<tr>
<th>Version</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4.0</td>
<td>• Drop support for Python 2.7, only 3.5 or newer is supported now</td>
<td>2020/07/04</td>
</tr>
<tr>
<td>2.3.0</td>
<td>• Add PWM backlight control</td>
<td>2020/01/08</td>
</tr>
<tr>
<td>2.2.0</td>
<td>• Add ILI9341 Colour LCD display driver</td>
<td>2019/11/25</td>
</tr>
<tr>
<td>2.1.0</td>
<td>• Rework namespace handling for luma sub-projects</td>
<td>2019/06/16</td>
</tr>
</tbody>
</table>
| 2.0.0   | • **BREAKING CHANGES:** Removal of `luma.lcd.aux.backlight` class  
|         | • Device classes now incorporate backlight capability | 2019/06/02 |
| 1.1.1   | • Add support for 160x80 display size for ST7735  
|         | • Minor documentation updates | 2019/03/30 |
| 1.1.0   | • Add ST7567 Monochrome LCD display driver (courtesy of @Gadgetoid)  
|         | • Change HT1621 tests  
|         | • Update dependencies | 2018/09/07 |
| 1.0.3   | • Changed version number to inside `luma/lcd/__init__.py` | 2017/11/23 |
| 1.0.2   | • Documentation and dependencies updates | 2017/10/30 |
| 1.0.1   | • Update dependencies | 2017/09/14 |
| 1.0.0   | • Stable version  
|         | • Remove deprecated methods | 2017/09/09 |
| 0.5.0   | • Add UC1701X Monochrome LCD display driver | 2017/06/11 |
| 0.4.1   | • luma.core 0.9.0 or newer is required now | 2017/04/22 |
| 0.4.0   | • Add HT1621 seven-segment driver | 2017/04/22 |
| 0.3.3   | • Add deprecation warning for `bcm_LIGHT` | 2017/03/14 |
| 0.3.2   | • Raise `error.UnsupportedPlatform` if RPi.GPIO is not available | 2017/03/08 |
| 0.3.0   | • Add ST7735 Color TFT LCD display driver  
|         | • Removed width and height parameters from device constructors  
|         | • **BREAKING CHANGES:** Move backlight class to different package | 2017/03/05 |
| 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 |
| 0.1.0   | • Bit-bang version using wiringPi | 2013/01/28 |
CHAPTER
EIGHT

THE MIT LICENSE (MIT)

Copyright (c) 2013-2020 Richard Hull & Contributors

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documenta-
tion 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 PAR-
TICULAR 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 SOFT-
WARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
luma.lcd, 15
luma.lcd.device, 15
INDEX

C
capabilities() (luma.lcd.device.ht1621 method), 16
capabilities() (luma.lcd.device.ili9341 method), 17
capabilities() (luma.lcd.device.pcd8544 method), 18
capabilities() (luma.lcd.device.st7567 method), 19
capabilities() (luma.lcd.device.st7735 method), 20
capabilities() (luma.lcd.device.st7920 method), 21
capabilities() (luma.lcd.device.uc1701x method), 23
cleanup() (luma.lcd.device.ht1621 method), 16
cleanup() (luma.lcd.device.ili9341 method), 17
cleanup() (luma.lcd.device.pcd8544 method), 18
cleanup() (luma.lcd.device.uc1701x method), 16
cleanup() (luma.lcd.device.st7567 method), 19
cleanup() (luma.lcd.device.st7735 method), 21
cleanup() (luma.lcd.device.uc1701x method), 23
clear() (luma.lcd.device.ht1621 method), 16
clear() (luma.lcd.device.ili9341 method), 17
clear() (luma.lcd.device.pcd8544 method), 18
clear() (luma.lcd.device.uc1701x method), 23
clear() (luma.lcd.device.uc1701x method), 22
clear() (luma.lcd.device.uc1701x method), 23
command() (luma.lcd.device.ht1621 method), 16
command() (luma.lcd.device.ili9341 method), 17
command() (luma.lcd.device.pcd8544 method), 18
command() (luma.lcd.device.uc1701x method), 23
contrast() (luma.lcd.device.uc1701x method), 23
data() (luma.lcd.device.ht1621 method), 16
data() (luma.lcd.device.ili9341 method), 17
data() (luma.lcd.device.pcd8544 method), 19
data() (luma.lcd.device.st7567 method), 20
data() (luma.lcd.device.st7735 method), 21
data() (luma.lcd.device.st7920 method), 22
data() (luma.lcd.device.ili9341 method), 18
display() (luma.lcd.device.ht1621 method), 16
display() (luma.lcd.device.pcd8544 method), 19
display() (luma.lcd.device.st7567 method), 20
display() (luma.lcd.device.st7735 method), 21
display() (luma.lcd.device.st7920 method), 22
display() (luma.lcd.device.uc1701x method), 23
display() (luma.lcd.device.uc1701x method), 23
D
hide() (luma.lcd.device.ht1621 method), 16
hide() (luma.lcd.device.ili9341 method), 17
hide() (luma.lcd.device.pcd8544 method), 19
hide() (luma.lcd.device.st7567 method), 20
hide() (luma.lcd.device.st7735 method), 21
hide() (luma.lcd.device.st7920 method), 22
hide() (luma.lcd.device.uc1701x method), 23
ht1621 (class in luma.lcd.device), 15
I
ili9341 (class in luma.lcd.device), 17
L
luma.lcd module, 15
luma.lcd.device module, 15
M
module
  luma.lcd, 15
  luma.lcd.device, 15
Luma.LCD Documentation, Release 2.4.0

P

pcd8544 (class in luma.lcd.device), 18
preprocess() (luma.lcd.device.ht1621 method), 16
preprocess() (luma.lcd.device.ili9341 method), 18
preprocess() (luma.lcd.device.pcd8544 method), 19
preprocess() (luma.lcd.device.st7567 method), 20
preprocess() (luma.lcd.device.st7735 method), 21
preprocess() (luma.lcd.device.st7920 method), 22
preprocess() (luma.lcd.device.uc1701x method), 23

S

show() (luma.lcd.device.ht1621 method), 17
show() (luma.lcd.device.ili9341 method), 18
show() (luma.lcd.device.pcd8544 method), 19
show() (luma.lcd.device.st7567 method), 20
show() (luma.lcd.device.st7735 method), 21
show() (luma.lcd.device.st7920 method), 22
show() (luma.lcd.device.uc1701x method), 23
st7567 (class in luma.lcd.device), 19
st7735 (class in luma.lcd.device), 20
st7920 (class in luma.lcd.device), 21

U

uc1701x (class in luma.lcd.device), 22