Luma.LCD Documentation

Release 2.4.0

Richard Hull and contributors

Aug 13, 2020

CONTENTS

CHAPTER

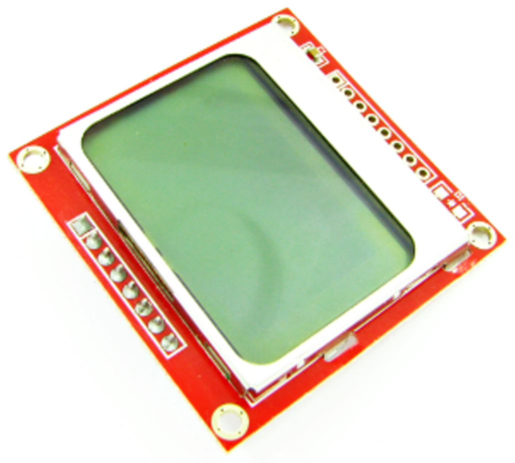
ONE

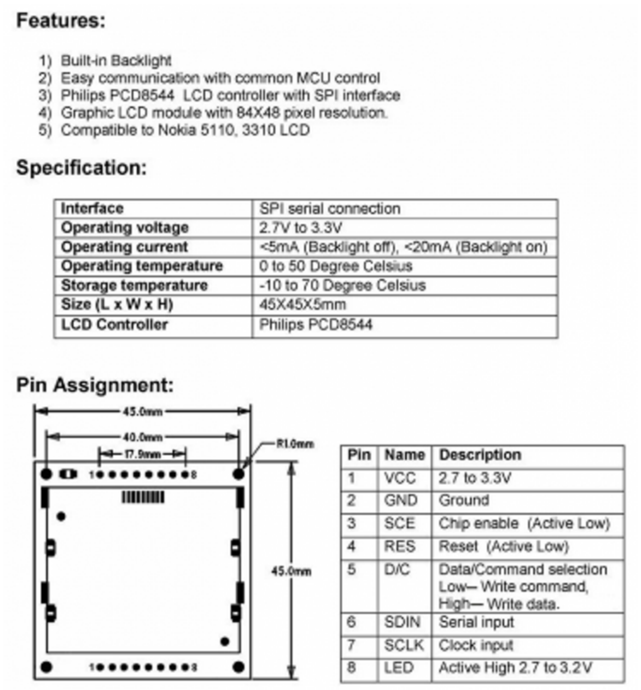
INTRODUCTION

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:





Features:

1) Built-in Backlight
2) Easy communication with common MCU control
3) Philips PCD8544 LCD controller with SPI interface
4) Graphic LCD module with 84X48 pixel resolution.
5) Compatible to Nokia 5110, 3310 LCD

Specification:

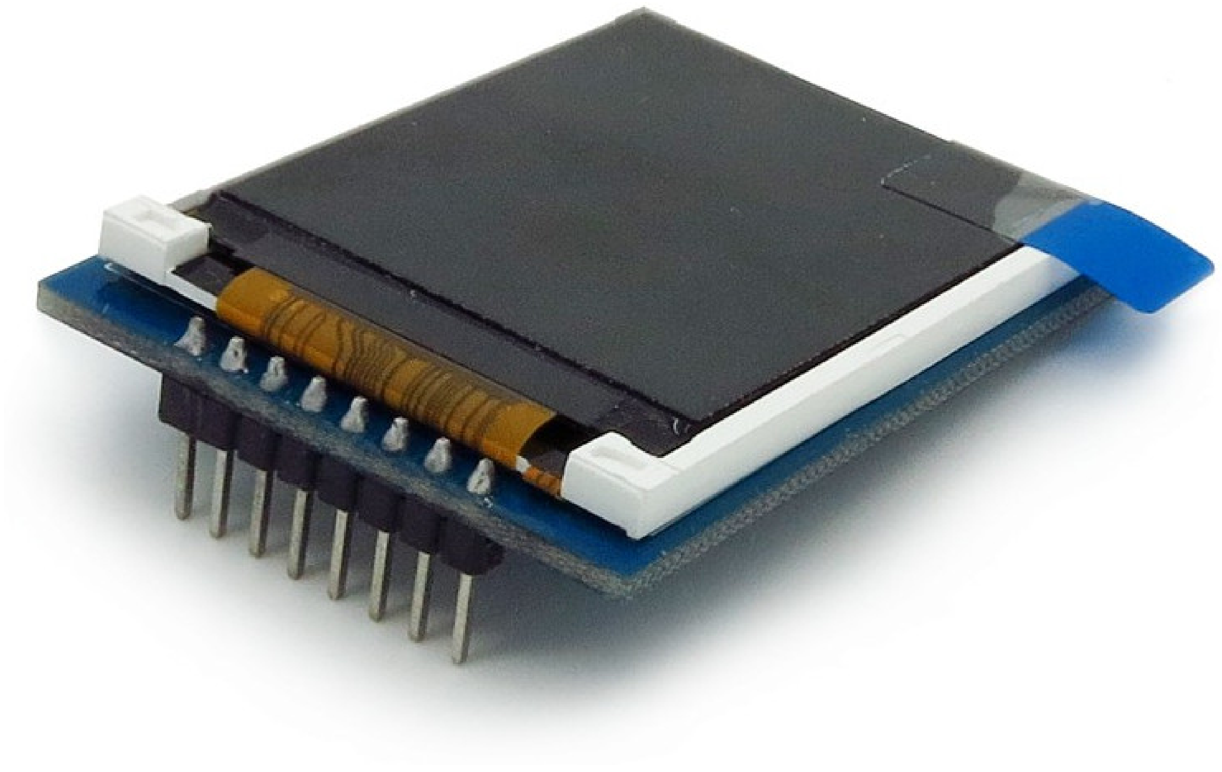
Interface	SPI serial connection
Operating voltage	2.7V to 3.3V
Operating current	<5mA (Backlight off), <20mA (Backlight on)
Operating temperature	0 to 50 Degree Celsius
Storage temperature	-10 to 70 Degree Celsius
Size (L x W x H)	45X45X5mm
LCD Controller	Philips PCD8544

Pin Assignment:

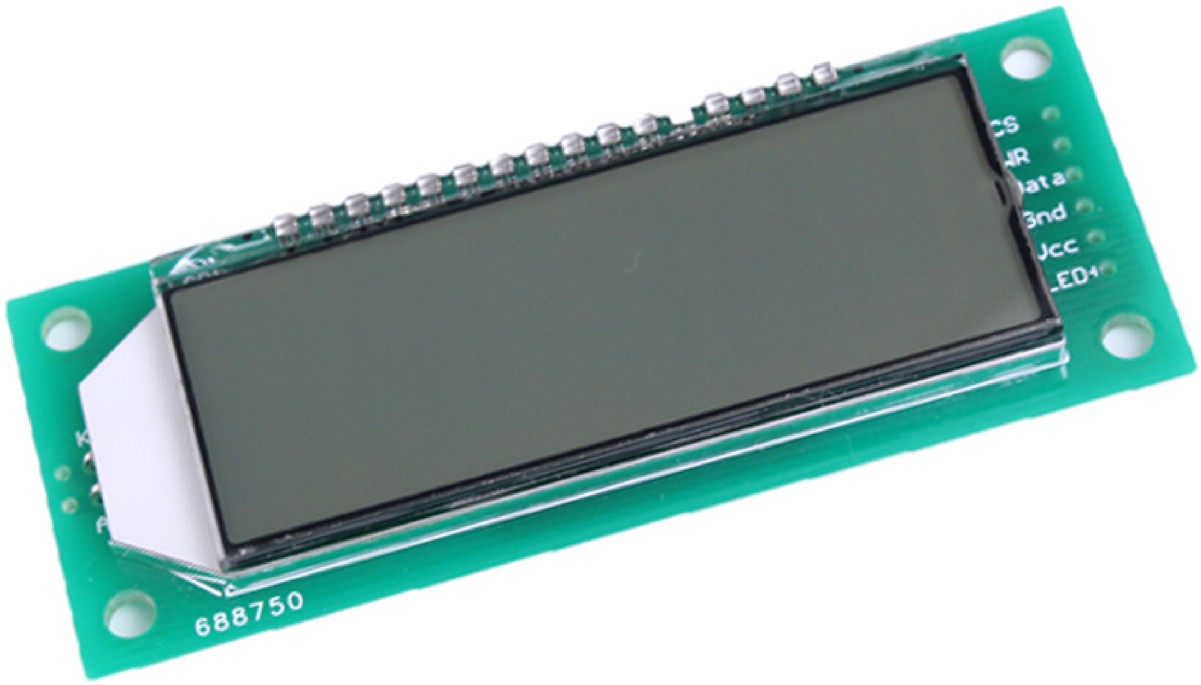
Pin	Name	Description
1	VCC	2.7 to 3.3V
2	GND	Ground
3	SCE	Chip enable (Active Low)
4	RES	Reset (Active Low)
5	D/C	Data/Command selection Low— Write command, High— Write data.
6	SDIN	Serial input
7	SCLK	Clock input
8	LED	Active High 2.7 to 3.2V

They are now commonly recycled, and sold on ebay with a breakout board and SPI interface.

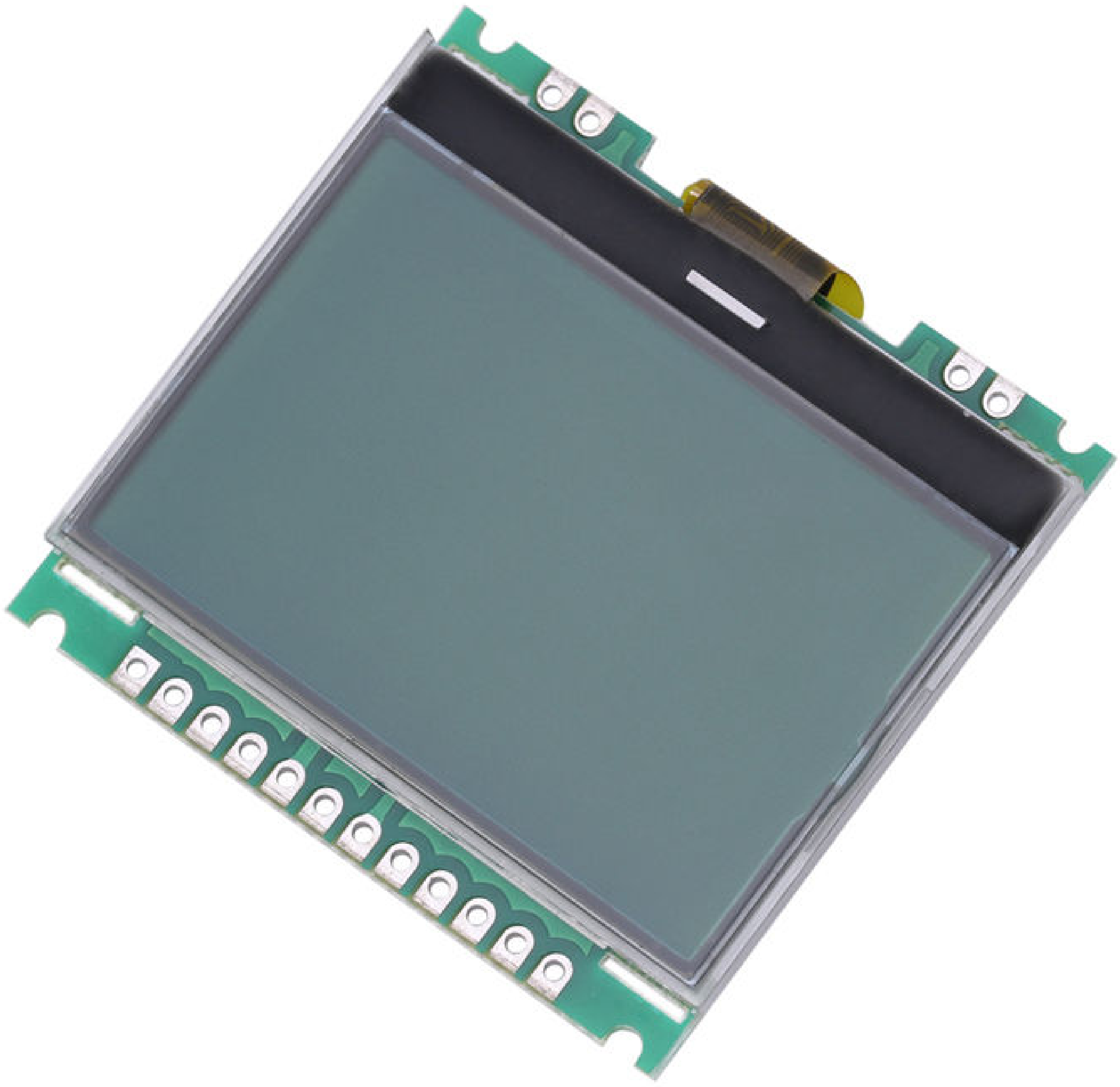
The ST7735 display supports a resoltion of 160 x 128 RGB pixels (18-bit / 262K colors) with a switchable backlight:



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.

CHAPTER

TWO

INSTALLATION

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:

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

or:

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

Then add your user to the *spi* and *gpio* groups:

```
$ 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

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

2.2.2 ST7735

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

LCD Pin	Remarks	RPi Pin	RPi Function
GND	Ground	P01-06	GND
VCC	+3.3V Power	P01-01	3V3
RESET or RST	Reset	P01-18	GPIO 24
A0 or D/C	Data/command	P01-16	GPIO 23
SDA or DIN	SPI data	P01-19	GPIO 10 (MOSI)
SCK or CLK	SPI clock	P01-23	GPIO 11 (SCLK)
CS	SPI chip select	P01-24	GPIO 8 (CE0)
LED+ or BL	Backlight control	P01-12	GPIO 18 (PCM_CLK)
LED-	Backlight ground	P01-06	GND

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.

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

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.

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

2.2.5 HT1621

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

2.2.6 UC1701X

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

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

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)
```

CHAPTER

THREE

PYTHON USAGE

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:

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

```
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).

```
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:

```
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:

```
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:

```
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 / deleteing) 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_CLK0) 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

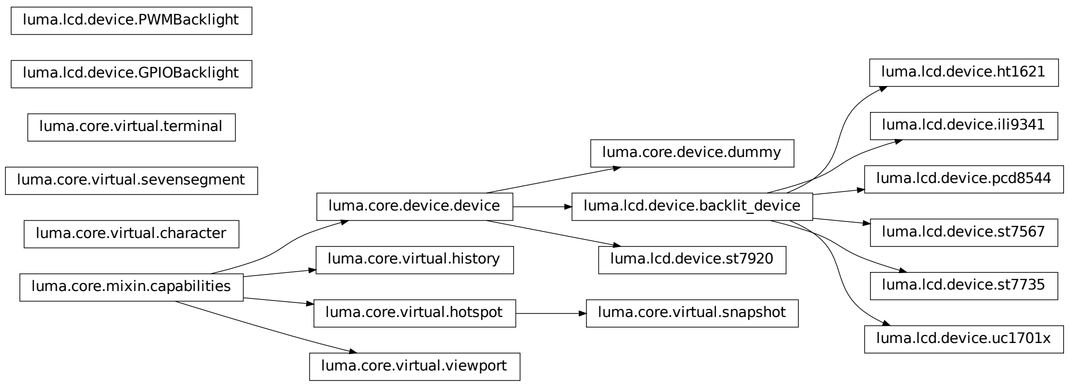
After installing the library, head over to the luma.examples repository. Details of how to run the examples is shown in the example repo's README.

CHAPTER

FOUR

API DOCUMENTATION

LCD display drivers.


luma.lcd.device.PWMBacklight
luma.lcd.device.GPIOBacklight
luma.core.virtual.terminal
luma.core.virtual.sevensegment
luma.core.virtual.character
luma.core.mixin.capabilities
luma.core.device.device
luma.core.virtual.history
luma.core.virtual.hotspot
luma.core.virtual.viewport
luma.core.device.dummy
luma.lcd.device.backlit_device
luma.lcd.device.st7920
luma.core.virtual.snapshot
luma.lcd.device.ht1621
luma.lcd.device.ili9341
luma.lcd.device.pcd8544
luma.lcd.device.st7567
luma.lcd.device.st7735
luma.lcd.device.uc1701x

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.

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.

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.

class luma.lcd.device.**ili9341**(*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.

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 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.

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** – 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.

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 processs 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.

CHAPTER FIVE

REFERENCES

- http://elinux.org/Rpi_Low-level_peripherals#General_Purpose_Input.2FOutput_.28GPIO.29
- http://binerry.de/post/25787954149/pcd8544-library-for-raspberry-pi
- http://www.avdweb.nl/arduino/hardware-interfacing/nokia-5110-lcd.html
- http://www.raspberrypi.org/phpBB3/viewtopic.php?f=32&t=9814&start=100
- 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://www.sitronix.com.tw/sitronix/product.nsf/Doc/ST7735?OpenDocument
- http://learn.adafruit.com/1-8-tft-display
- http://www.raspberrypi.org/phpBB3/viewtopic.php?t=28696&p=262909
- http://elinux.org/images/1/19/Passing_Time_With_SPI_Framebuffer_Driver.pdf
- http://www.flickr.com/photos/ngreatorex/7672743302/
- https://github.com/notro/fbtft
- https://github.com/rm-hull/st7735fb
- http://www.areinhardt.de/news/raspberry-pi-tft-display/
- http://www.whence.com/rpi/
- http://harizanov.com/product/1-8-tft-display-for-raspberry-pi/

CHAPTER

SIX

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

Version	Description	Date
2.4.0	• Drop support for Python 2.7, only 3.5 or newer is supported now	2020/07/04
2.3.0	• Add PWM backlight control	2020/01/08
2.2.0	• Add ILI9341 Colour LCD display driver	2019/11/25
2.1.0	• Rework namespace handling for luma sub-projects	2019/06/16
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

CHAPTER EIGHT

THE MIT LICENSE (MIT)

PYTHON MODULE INDEX

INDEX