
AdafruitEPD Library Documentation

Release 1.0

Dean Miller

May 20, 2019

Contents

1	Dependencies	3
2	Usage Example	5
3	Contributing	7
4	Building locally	9
4.1	Zip release files	9
4.2	Sphinx documentation	9
5	Table of Contents	11
5.1	Simple test	11
5.2	adafruit_epd.epd - Adafruit EPD - ePaper display driver	12
6	Indices and tables	15
	Python Module Index	17

This library is for using CircuitPython with e-ink displays with built in SRAM.

CHAPTER 1

Dependencies

This driver depends on:

- [Adafruit CircuitPython](#)
- [Bus Device](#)
- [font5x8.bin](#) found in the examples bundle

Please ensure all dependencies are available on the CircuitPython filesystem. This is easily achieved by downloading the [Adafruit library and driver bundle](#).

CHAPTER 2

Usage Example

```
import digitalio
import busio
import board
from adafruit_epd.epd import Adafruit_EPD
from adafruit_epd.il0373 import Adafruit_IL0373

# create the spi device and pins we will need
spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
ecs = digitalio.DigitalInOut(board.D12)
dc = digitalio.DigitalInOut(board.D11)
srcs = digitalio.DigitalInOut(board.D10)    # can be None to use internal memory
rst = digitalio.DigitalInOut(board.D9)     # can be None to not use this pin
busy = digitalio.DigitalInOut(board.D5)    # can be None to not use this pin

# give them all to our driver
print("Creating display")
display = Adafruit_IL0373(104, 212, spi,          # 2.13" Tri-color display
                          cs_pin=ecs, dc_pin=dc, sramcs_pin=srcs,
                          rst_pin=rst, busy_pin=busy)

display.rotation = 1

# clear the buffer
print("Clear buffer")
display.fill(Adafruit_EPD.WHITE)
display.pixel(10, 100, Adafruit_EPD.BLACK)

print("Draw Rectangles")
display.fill_rect(5, 5, 10, 10, Adafruit_EPD.RED)
display.rect(0, 0, 20, 30, Adafruit_EPD.BLACK)

print("Draw lines")
display.line(0, 0, display.width-1, display.height-1, Adafruit_EPD.BLACK)
display.line(0, display.height-1, display.width-1, 0, Adafruit_EPD.RED)
```

(continues on next page)

(continued from previous page)

```
print("Draw text")
display.text('hello world', 25, 10, Adafruit_EPDM.BLACK)
display.display()
```

CHAPTER 3

Contributing

Contributions are welcome! Please read our [Code of Conduct](#) before contributing to help this project stay welcoming.

4.1 Zip release files

To build this library locally you'll need to install the `circuitpython-build-tools` package.

```
python3 -m venv .env
source .env/bin/activate
pip install circuitpython-build-tools
```

Once installed, make sure you are in the virtual environment:

```
source .env/bin/activate
```

Then run the build:

```
circuitpython-build-bundles --filename_prefix adafruit-circuitpython-epd --library_
↪location .
```

4.2 Sphinx documentation

Sphinx is used to build the documentation based on rST files and comments in the code. First, install dependencies (feel free to reuse the virtual environment from above):

```
python3 -m venv .env
source .env/bin/activate
pip install Sphinx sphinx-rtd-theme
```

Now, once you have the virtual environment activated:

```
cd docs
sphinx-build -E -W -b html . _build/html
```

This will output the documentation to `docs/_build/html`. Open the `index.html` in your browser to view them. It will also (due to `-W`) error out on any warning like Travis will. This is a good way to locally verify it will pass.

Table of Contents

5.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/epd_simpletest.py

```

1  import digitalio
2  import busio
3  import board
4  from adafruit_epd.epd import Adafruit_EPd
5  from adafruit_epd.il0373 import Adafruit_IL0373
6  from adafruit_epd.il91874 import Adafruit_IL91874 # pylint: disable=unused-import
7  from adafruit_epd.il0398 import Adafruit_IL0398 # pylint: disable=unused-import
8  from adafruit_epd.ssd1608 import Adafruit_SSD1608 # pylint: disable=unused-import
9  from adafruit_epd.ssd1675 import Adafruit_SSD1675 # pylint: disable=unused-import
10
11 # create the spi device and pins we will need
12 spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
13 ecs = digitalio.DigitalInOut(board.D12)
14 dc = digitalio.DigitalInOut(board.D11)
15 srcs = digitalio.DigitalInOut(board.D10) # can be None to use internal memory
16 rst = digitalio.DigitalInOut(board.D9) # can be None to not use this pin
17 busy = digitalio.DigitalInOut(board.D5) # can be None to not use this pin
18
19 # give them all to our driver
20 print("Creating display")
21 #display = Adafruit_SSD1608(200, 200, spi, # 1.54" HD mono display
22 #display = Adafruit_SSD1675(250, 122, spi, # 2.13" HD mono display
23 #display = Adafruit_IL91874(176, 264, spi, # 2.7" Tri-color display
24 #display = Adafruit_IL0373(152, 152, spi, # 1.54" Tri-color display
25 #display = Adafruit_IL0373(128, 296, spi, # 2.9" Tri-color display
26 #display = Adafruit_IL0398(400, 300, spi, # 4.2" Tri-color display
27 display = Adafruit_IL0373(104, 212, spi, # 2.13" Tri-color display

```

(continues on next page)

(continued from previous page)

```

28         cs_pin=ecs, dc_pin=dc, sramcs_pin=srcs,
29         rst_pin=rst, busy_pin=busy)
30
31 # IF YOU HAVE A FLEXIBLE DISPLAY (2.13" or 2.9") uncomment these lines!
32 #display.set_black_buffer(1, False)
33 #display.set_color_buffer(1, False)
34
35 display.rotation = 1
36
37 # clear the buffer
38 print("Clear buffer")
39 display.fill(Adafruit_EPDM.WHITE)
40 display.pixel(10, 100, Adafruit_EPDM.BLACK)
41
42 print("Draw Rectangles")
43 display.fill_rect(5, 5, 10, 10, Adafruit_EPDM.RED)
44 display.rect(0, 0, 20, 30, Adafruit_EPDM.BLACK)
45
46 print("Draw lines")
47 display.line(0, 0, display.width-1, display.height-1, Adafruit_EPDM.BLACK)
48 display.line(0, display.height-1, display.width-1, 0, Adafruit_EPDM.RED)
49
50 print("Draw text")
51 display.text('hello world', 25, 10, Adafruit_EPDM.BLACK)
52 display.display()

```

5.2 adafruit_epd.epd - Adafruit EPD - ePaper display driver

CircuitPython driver for Adafruit ePaper display breakouts * Author(s): Dean Miller

class `adafruit_epd.epd.Adafruit_EPDM` (*width, height, spi, cs_pin, dc_pin, sramcs_pin, rst_pin, busy_pin*)

Base class for EPD displays

command (*cmd, data=None, end=True*)
Send command byte to display.

display ()
show the contents of the display buffer

fill (*color*)
fill the screen with the passed color

fill_rect (*x, y, width, height, color*)
fill a rectangle with the passed color

hardware_reset ()
If we have a reset pin, do a hardware reset by toggling it

height
The height of the display, accounting for rotation

hline (*x, y, width, color*)
draw a horizontal line

image (*image*)
Set buffer to value of Python Imaging Library image. The image should be in RGB mode and a size equal

to the display size.

line (*x_0, y_0, x_1, y_1, color*)

Draw a line from (*x_0, y_0*) to (*x_1, y_1*) in passed color

pixel (*x, y, color*)

draw a single pixel in the display buffer

power_down ()

Power down the display, must be implemented in subclass

power_up ()

Power up the display in preparation for writing RAM and updating. must be implemented in subclass

rect (*x, y, width, height, color*)

draw a rectangle

rotation

The rotation of the display, can be one of (0, 1, 2, 3)

set_black_buffer (*index, inverted*)

Set the index for the black buffer data (0 or 1) and whether its inverted

set_color_buffer (*index, inverted*)

Set the index for the color buffer data (0 or 1) and whether its inverted

set_ram_address (*x, y*)

Set the RAM address location, must be implemented in subclass

text (*string, x, y, color, *, font_name='font5x8.bin'*)

Write text string at location (*x, y*) in given color, using font file

update ()

Update the display from internal memory, must be implemented in subclass

vline (*x, y, height, color*)

draw a vertical line

width

The width of the display, accounting for rotation

write_ram (*index*)

Send the one byte command for starting the RAM write process. Returns the byte read at the same time over SPI. *index* is the RAM buffer, can be 0 or 1 for tri-color displays. must be implemented in subclass

CHAPTER 6

Indices and tables

- `genindex`
- `modindex`
- `search`

a

`adafruit_epd.epd`, 12

A

Adafruit_EPDP (class in *adafruit_epd.epd*), 12
adafruit_epd.epd (module), 12

C

command() (*adafruit_epd.epd.Adafruit_EPDP* method), 12

D

display() (*adafruit_epd.epd.Adafruit_EPDP* method), 12

F

fill() (*adafruit_epd.epd.Adafruit_EPDP* method), 12
fill_rect() (*adafruit_epd.epd.Adafruit_EPDP* method), 12

H

hardware_reset() (*adafruit_epd.epd.Adafruit_EPDP* method), 12
height (*adafruit_epd.epd.Adafruit_EPDP* attribute), 12
hline() (*adafruit_epd.epd.Adafruit_EPDP* method), 12

I

image() (*adafruit_epd.epd.Adafruit_EPDP* method), 12

L

line() (*adafruit_epd.epd.Adafruit_EPDP* method), 13

P

pixel() (*adafruit_epd.epd.Adafruit_EPDP* method), 13
power_down() (*adafruit_epd.epd.Adafruit_EPDP* method), 13
power_up() (*adafruit_epd.epd.Adafruit_EPDP* method), 13

R

rect() (*adafruit_epd.epd.Adafruit_EPDP* method), 13

rotation (*adafruit_epd.epd.Adafruit_EPDP* attribute), 13

S

set_black_buffer() (*adafruit_epd.epd.Adafruit_EPDP* method), 13
set_color_buffer() (*adafruit_epd.epd.Adafruit_EPDP* method), 13
set_ram_address() (*adafruit_epd.epd.Adafruit_EPDP* method), 13

T

text() (*adafruit_epd.epd.Adafruit_EPDP* method), 13

U

update() (*adafruit_epd.epd.Adafruit_EPDP* method), 13

V

vline() (*adafruit_epd.epd.Adafruit_EPDP* method), 13

W

width (*adafruit_epd.epd.Adafruit_EPDP* attribute), 13
write_ram() (*adafruit_epd.epd.Adafruit_EPDP* method), 13