
AdafruitEPD Library Documentation

Release 1.0

Dean Miller

Jul 22, 2021

Contents

1	Dependencies	3
2	Installing from PyPI	5
3	Usage Example	7
4	Contributing	9
5	Documentation	11
6	Table of Contents	13
6.1	Simple test	13
6.2	adafruit_epd.epd - Adafruit EPD - ePaper display driver	14
7	Indices and tables	17
	Python Module Index	19
	Index	21

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](#).

Installing from PyPI

On supported GNU/Linux systems like the Raspberry Pi, you can install the driver locally [from PyPI](#). To install for current user:

```
pip3 install adafruit-circuitpython-epd
```

To install system-wide (this may be required in some cases):

```
sudo pip3 install adafruit-circuitpython-epd
```

To install in a virtual environment in your current project:

```
mkdir project-name && cd project-name  
python3 -m venv .env  
source .env/bin/activate  
pip3 install adafruit-circuitpython-epd
```


CHAPTER 3

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_EPD.BLACK)
display.display()
```

CHAPTER 4

Contributing

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

CHAPTER 5

Documentation

For information on building library documentation, please check out [this guide](#).

6.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/epd_simpletest.py

```
1 # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2 # SPDX-License-Identifier: MIT
3
4 import digitalio
5 import busio
6 import board
7 from adafruit_epd.epd import Adafruit_EPD
8 from adafruit_epd.il0373 import Adafruit_IL0373
9 from adafruit_epd.il91874 import Adafruit_IL91874 # pylint: disable=unused-import
10 from adafruit_epd.il0398 import Adafruit_IL0398 # pylint: disable=unused-import
11 from adafruit_epd.ssd1608 import Adafruit_SSD1608 # pylint: disable=unused-import
12 from adafruit_epd.ssd1675 import Adafruit_SSD1675 # pylint: disable=unused-import
13 from adafruit_epd.ssd1680 import Adafruit_SSD1680 # pylint: disable=unused-import
14 from adafruit_epd.ssd1681 import Adafruit_SSD1681 # pylint: disable=unused-import
15
16 # create the spi device and pins we will need
17 spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
18 ecs = digitalio.DigitalInOut(board.D12)
19 dc = digitalio.DigitalInOut(board.D11)
20 srcs = digitalio.DigitalInOut(board.D10) # can be None to use internal memory
21 rst = digitalio.DigitalInOut(board.D9) # can be None to not use this pin
22 busy = digitalio.DigitalInOut(board.D5) # can be None to not use this pin
23
24 # give them all to our drivers
25 print("Creating display")
26 # display = Adafruit_SSD1608(200, 200, # 1.54" HD mono display
27 # display = Adafruit_SSD1675(122, 250, # 2.13" HD mono display
```

(continues on next page)

(continued from previous page)

```

28 # display = Adafruit_SSD1680(122, 250,           # 2.13" HD Tri-color display
29 # display = Adafruit_SSD1681(200, 200,           # 1.54" HD Tri-color display
30 # display = Adafruit_IL91874(176, 264,           # 2.7" Tri-color display
31 # display = Adafruit_IL0373(152, 152,           # 1.54" Tri-color display
32 # display = Adafruit_IL0373(128, 296,           # 2.9" Tri-color display
33 # display = Adafruit_IL0398(400, 300,           # 4.2" Tri-color display
34 display = Adafruit_IL0373(
35     104,
36     212, # 2.13" Tri-color display
37     spi,
38     cs_pin=ecs,
39     dc_pin=dc,
40     sramcs_pin=srscs,
41     rst_pin=rst,
42     busy_pin=busy,
43 )
44
45 # IF YOU HAVE A FLEXIBLE DISPLAY (2.13" or 2.9") uncomment these lines!
46 # display.set_black_buffer(1, False)
47 # display.set_color_buffer(1, False)
48
49 display.rotation = 1
50
51 # clear the buffer
52 print("Clear buffer")
53 display.fill(Adafruit_EPD.WHITE)
54 display.pixel(10, 100, Adafruit_EPD.BLACK)
55
56 print("Draw Rectangles")
57 display.fill_rect(5, 5, 10, 10, Adafruit_EPD.RED)
58 display.rect(0, 0, 20, 30, Adafruit_EPD.BLACK)
59
60 print("Draw lines")
61 display.line(0, 0, display.width - 1, display.height - 1, Adafruit_EPD.BLACK)
62 display.line(0, display.height - 1, display.width - 1, 0, Adafruit_EPD.RED)
63
64 print("Draw text")
65 display.text("hello world", 25, 10, Adafruit_EPD.BLACK)
66 display.display()

```

6.2 adafruit_epd.epd - Adafruit EPD - ePaper display driver

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

```
class adafruit_epd.epd.Adafruit_EPD(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', size=1*)
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 7

Indices and tables

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

a

`adafruit_epd.epd`, 14

A

Adafruit_EPDP (class in *adafruit_epd.epd*), 14
adafruit_epd.epd (module), 14

C

command() (*adafruit_epd.epd.Adafruit_EPDP* method), 14

D

display() (*adafruit_epd.epd.Adafruit_EPDP* method), 14

F

fill() (*adafruit_epd.epd.Adafruit_EPDP* method), 14
fill_rect() (*adafruit_epd.epd.Adafruit_EPDP* method), 14

H

hardware_reset() (*adafruit_epd.epd.Adafruit_EPDP* method), 15
height (*adafruit_epd.epd.Adafruit_EPDP* attribute), 15
hline() (*adafruit_epd.epd.Adafruit_EPDP* method), 15

I

image() (*adafruit_epd.epd.Adafruit_EPDP* method), 15

L

line() (*adafruit_epd.epd.Adafruit_EPDP* method), 15

P

pixel() (*adafruit_epd.epd.Adafruit_EPDP* method), 15
power_down() (*adafruit_epd.epd.Adafruit_EPDP* method), 15
power_up() (*adafruit_epd.epd.Adafruit_EPDP* method), 15

R

rect() (*adafruit_epd.epd.Adafruit_EPDP* method), 15

rotation (*adafruit_epd.epd.Adafruit_EPDP* attribute), 15

S

set_black_buffer() (*adafruit_epd.epd.Adafruit_EPDP* method), 15
set_color_buffer() (*adafruit_epd.epd.Adafruit_EPDP* method), 15
set_ram_address() (*adafruit_epd.epd.Adafruit_EPDP* method), 15

T

text() (*adafruit_epd.epd.Adafruit_EPDP* method), 15

U

update() (*adafruit_epd.epd.Adafruit_EPDP* method), 15

V

vline() (*adafruit_epd.epd.Adafruit_EPDP* method), 15

W

width (*adafruit_epd.epd.Adafruit_EPDP* attribute), 15
write_ram() (*adafruit_epd.epd.Adafruit_EPDP* method), 15