AdafruitEPD Library Documentation

Release 1.0

Dean Miller

Jun 14, 2019

Contents

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

CHAPTER 3

Contributing

Contributions are welcome! Please read our Code of Conduct before contributing to help this project stay welcoming.

CHAPTER 4

Building locally

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.

CHAPTER 5

Table of Contents

5.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/epd_simpletest.py

```python
import digitalio
import busio
import board
from adafruit_epd.epd import Adafruit_EPD
from adafruit_epd.il0373 import Adafruit_IL0373
from adafruit_epd.il91874 import Adafruit_IL91874  # pylint: disable=unused-import
from adafruit_epd.il0398 import Adafruit_IL0398  # pylint: disable=unused-import
from adafruit_epd.ssd1608 import Adafruit_SSD1608  # pylint: disable=unused-import
from adafruit_epd.ssd1675 import Adafruit_SSD1675  # pylint: disable=unused-import

# 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_SSD1608(200, 200, spi,        # 1.54" HD mono display
#display = Adafruit_SSD1675(250, 122, spi,        # 2.13" HD mono display
#display = Adafruit_IL91874(176, 264, spi,        # 2.7" Tri-color display
#display = Adafruit_IL0373(152, 152, spi,         # 1.54" Tri-color display
#display = Adafruit_IL0373(128, 296, spi,         # 2.9" Tri-color display
#display = Adafruit_IL0398(400, 300, spi,         # 4.2" Tri-color display
display = Adafruit_IL0373(104, 212, spi,          # 2.13" Tri-color display
```

(continues on next page)

(continued from previous page)

```
                          cs_pin=ecs, dc_pin=dc, sramcs_pin=srcs,
                          rst_pin=rst, busy_pin=busy)

# IF YOU HAVE A FLEXIBLE DISPLAY (2.13" or 2.9") uncomment these lines!
#display.set_black_buffer(1, False)
#display.set_color_buffer(1, False)

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)

print("Draw text")
display.text('hello world', 25, 10, Adafruit_EPD.BLACK)
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_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'*)
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

Python Module Index

Index