
AdafruitRGB*DisplayLibraryDocumentation*

Release 1.0

Michale McWethy

Feb 25, 2021

Contents

1	Dependencies	3
2	Installing from PyPI	5
3	Usage Example	7
3.1	2.2", 2.4", 2.8", 3.2" TFT	7
3.2	1.14" TFT with Raspberry Pi 4	8
4	Contributing	9
5	Documentation	11
6	Table of Contents	13
6.1	Simple test	13
6.2	adafruit_rgb_display.rgb	15
6.3	adafruit_rgb_display.hx8353	16
6.4	adafruit_rgb_display.ili9341	16
6.5	adafruit_rgb_display.s6d02a1	17
6.6	adafruit_rgb_display.ssd1331	17
6.7	adafruit_rgb_display.ssd1351	18
6.8	adafruit_rgb_display.st7735	18
6.9	adafruit_rgb_display.st7789	19
7	Indices and tables	21
	Python Module Index	23
	Index	25

Port of display drivers from <https://github.com/adafruit/micropython-adafruit-rgb-display> to Adafruit CircuitPython for use on Adafruit's SAMD21-based and other CircuitPython boards.

Note: This driver currently won't work on micropython.org firmware, instead you want the micropython-adafruit-rgb-display driver linked above!

This CircuitPython driver currently supports displays that use the following display-driver chips: HX8353, HX8357, ILI9341, S6D02A1, ST7789, SSD1331, SSD1351, and ST7735 (including variants ST7735R and ST7735S).

CHAPTER 1

Dependencies

This driver depends on:

- [Adafruit CircuitPython](#)
- [Bus Device](#)

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

For the Pillow Examples, you will need to be running CPython. This means using a Single Board Computer such as a Raspberry Pi or using a chip such as an FT232H on Linux, Window, or Mac. CircuitPython does not support PIL/pillow (python imaging library)!

For improved performance consider installing NumPy.

CHAPTER 2

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-rgb-display
```

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

```
sudo pip3 install adafruit-circuitpython-rgb-display
```

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-rgb-display
```


CHAPTER 3

Usage Example

3.1 2.2", 2.4", 2.8", 3.2" TFT

```
import time
import busio
import digitalio
from board import SCK, MOSI, MISO, D2, D3

from adafruit_rgb_display import color565
import adafruit_rgb_display.ili9341 as ili9341

# Configuration for CS and DC pins:
CS_PIN = D2
DC_PIN = D3

# Setup SPI bus using hardware SPI:
spi = busio.SPI(clock=SCK, MOSI=MOSI, MISO=MISO)

# Create the ILI9341 display:
display = ili9341.ILI9341(spi, cs=digitalio.DigitalInOut(CS_PIN),
                        dc=digitalio.DigitalInOut(DC_PIN))

# Main loop:
while True:
    # Clear the display
    display.fill(0)
    # Draw a red pixel in the center.
    display.pixel(120, 160, color565(255, 0, 0))
    # Pause 2 seconds.
    time.sleep(2)
    # Clear the screen blue.
    display.fill(color565(0, 0, 255))
    # Pause 2 seconds.
```

(continues on next page)

(continued from previous page)

```
time.sleep(2)
```

3.2 1.14" TFT with Raspberry Pi 4

With 1.14" wiring, here is the working code:

```
import time
import busio
import digitalio
from board import SCK, MOSI, MISO, CE0, D24, D25

from adafruit_rgb_display import color565
from adafruit_rgb_display.st7789 import ST7789

# Configuration for CS and DC pins:
CS_PIN = CE0
DC_PIN = D25
RESET_PIN = D24
BAUDRATE = 24000000

# Setup SPI bus using hardware SPI:
spi = busio.SPI(clock=SCK, MOSI=MOSI, MISO=MISO)

# Create the ST7789 display:
display = ST7789(
    spi,
    rotation=90,
    width=135,
    height=240,
    x_offset=53,
    y_offset=40,
    baudrate=BAUDRATE,
    cs=digitalio.DigitalInOut(CS_PIN),
    dc=digitalio.DigitalInOut(DC_PIN),
    rst=digitalio.DigitalInOut(RESET_PIN))

# Main loop: same as above
while True:
    # Clear the display
    display.fill(0)
    # Draw a red pixel in the center.
    display.pixel(120, 160, color565(255, 0, 0))
    # Pause 2 seconds.
    time.sleep(2)
    # Clear the screen blue.
    display.fill(color565(0, 0, 255))
    # Pause 2 seconds.
    time.sleep(2)
```

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/rgb_display_simpletest.py

```
1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  # Quick test of TFT FeatherWing (ST7789) with Feather M0 or M4
5  # This will work even on a device running displayio
6  # Will fill the TFT black and put a red pixel in the center, wait 2 seconds,
7  # then fill the screen blue (with no pixel), wait 2 seconds, and repeat.
8  import time
9  import random
10 import digitalio
11 import board
12
13 from adafruit_rgb_display.rgb import color565
14 import adafruit_rgb_display.st7789 as st7789
15
16 # Configuratin for CS and DC pins (these are FeatherWing defaults on M0/M4):
17 cs_pin = digitalio.DigitalInOut(board.D5)
18 dc_pin = digitalio.DigitalInOut(board.D6)
19 reset_pin = digitalio.DigitalInOut(board.D9)
20
21 # Config for display baudrate (default max is 24mhz):
22 BAUDRATE = 24000000
23
24 # Setup SPI bus using hardware SPI:
25 spi = board.SPI()
26
27 # Create the ST7789 display:
```

(continues on next page)

(continued from previous page)

```

28 display = st7789.ST7789(spi, cs=cs_pin, dc=dc_pin, rst=reset_pin, baudrate=BAUDRATE)
29
30 # Main loop:
31 while True:
32     # Fill the screen red, green, blue, then black:
33     for color in ((255, 0, 0), (0, 255, 0), (0, 0, 255)):
34         display.fill(color565(color))
35     # Clear the display
36     display.fill(0)
37     # Draw a red pixel in the center.
38     display.pixel(display.width // 2, display.height // 2, color565(255, 0, 0))
39     # Pause 2 seconds.
40     time.sleep(2)
41     # Clear the screen a random color
42     display.fill(
43         color565(random.randint(0, 255), random.randint(0, 255), random.randint(0,
44         ↪ 255))
45     )
46     # Pause 2 seconds.
47     time.sleep(2)

```

Listing 2: examples/rgb_display_ili9341test.py

```

1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  # Quick test of TFT FeatherWing (ILI9341) with Feather M0 or M4
5  # Will fill the TFT black and put a red pixel in the center, wait 2 seconds,
6  # then fill the screen blue (with no pixel), wait 2 seconds, and repeat.
7  import time
8  import random
9  import busio
10 import digitalio
11 import board
12
13 from adafruit_rgb_display.rgb import color565
14 import adafruit_rgb_display.ili9341 as ili9341
15
16
17 # Configuratioin for CS and DC pins (these are FeatherWing defaults on M0/M4):
18 cs_pin = digitalio.DigitalInOut(board.D9)
19 dc_pin = digitalio.DigitalInOut(board.D10)
20
21 # Config for display baudrate (default max is 24mhz):
22 BAUDRATE = 24000000
23
24 # Setup SPI bus using hardware SPI:
25 spi = busio.SPI(clock=board.SCK, MOSI=board.MOSI, MISO=board.MISO)
26
27 # Create the ILI9341 display:
28 display = ili9341.ILI9341(spi, cs=cs_pin, dc=dc_pin, baudrate=BAUDRATE)
29
30 # Main loop:
31 while True:
32     # Fill the screen red, green, blue, then black:
33     for color in ((255, 0, 0), (0, 255, 0), (0, 0, 255)):

```

(continues on next page)

(continued from previous page)

```

34     display.fill(color565(color))
35     # Clear the display
36     display.fill(0)
37     # Draw a red pixel in the center.
38     display.pixel(display.width // 2, display.height // 2, color565(255, 0, 0))
39     # Pause 2 seconds.
40     time.sleep(2)
41     # Clear the screen a random color
42     display.fill(
43         color565(random.randint(0, 255), random.randint(0, 255), random.randint(0,
44         ↪255))
45     )
46     # Pause 2 seconds.
47     time.sleep(2)

```

6.2 adafruit_rgb_display.rgb

Base class for all RGB Display devices

- Author(s): Radomir Dopieralski, Michael McWethy

class adafruit_rgb_display.rgb.**Display** (*width, height, rotation*)

Base class for all RGB display devices :param width: number of pixels wide :param height: number of pixels high

fill (*color=0*)

Fill the whole display with the specified color.

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

Draw a rectangle at specified position with specified width and height, and fill it with the specified color.

hline (*x, y, width, color*)

Draw a horizontal line.

image (*img, rotation=None, x=0, y=0*)

Set buffer to value of Python Imaging Library image. The image should be in 1 bit mode and a size not exceeding the display size when drawn at the supplied origin.

init ()

Run the initialization commands.

pixel (*x, y, color=None*)

Read or write a pixel at a given position.

rotation

Set the default rotation

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

Draw a vertical line.

class adafruit_rgb_display.rgb.**DisplaySPI** (*spi, dc, cs, rst=None, width=1, height=1, baudrate=12000000, polarity=0, phase=0, *, x_offset=0, y_offset=0, rotation=0*)

Base class for SPI type devices

read (*command=None, count=0*)

SPI read from device with optional command

reset ()

Reset the device

write (command=None, data=None)

SPI write to the device: commands and data

class adafruit_rgb_display.rgb.DummyPin

Can be used in place of a DigitalInOut () when you don't want to skip it.

deinit ()

Dummy DigitalInOut deinit

direction

Dummy direction DigitalInOut property

pull

Dummy pull DigitalInOut property

switch_to_input (*args, **kwargs)

Dummy switch_to_input method

switch_to_output (*args, **kwargs)

Dummy switch_to_output method

value

Dummy value DigitalInOut property

adafruit_rgb_display.rgb.color565 (r, g=0, b=0)

Convert red, green and blue values (0-255) into a 16-bit 565 encoding. As a convenience this is also available in the parent adafruit_rgb_display package namespace.

adafruit_rgb_display.rgb.image_to_data (image)

Generator function to convert a PIL image to 16-bit 565 RGB bytes.

6.3 adafruit_rgb_display.hx8353

A simple driver for the HX8353-based displays.

- Author(s): Radomir Dopieralski, Michael McWethy

class adafruit_rgb_display.hx8353.HX8353 (spi, dc, cs, rst=None, width=128, height=128, rotation=0)

A simple driver for the HX8353-based displays.

```
>>> import busio
>>> import digitalio
>>> import board
>>> from adafruit_rgb_display import color565
>>> import adafruit_rgb_display.hx8353 as hx8353
>>> spi = busio.SPI(clock=board.SCK, MOSI=board.MOSI, MISO=board.MISO)
>>> display = hx8353.HX8383(spi, cs=digitalio.DigitalInOut(board.GPIO0),
...   dc=digitalio.DigitalInOut(board.GPIO15))
>>> display.fill(0x7521)
>>> display.pixel(64, 64, 0)
```

6.4 adafruit_rgb_display.ili9341

A simple driver for the ILI9341/ILI9340-based displays.

- Author(s): Radomir Dopieralski, Michael McWethy

class adafruit_rgb_display.ili9341.**ILI9341** (*spi, dc, cs, rst=None, width=240, height=320, baudrate=16000000, polarity=0, phase=0, rotation=0*)

A simple driver for the ILI9341/ILI9340-based displays.

```
>>> import busio
>>> import digitalio
>>> import board
>>> from adafruit_rgb_display import color565
>>> import adafruit_rgb_display.ili9341 as ili9341
>>> spi = busio.SPI(clock=board.SCK, MOSI=board.MOSI, MISO=board.MISO)
>>> display = ili9341.ILI9341(spi, cs=digitalio.DigitalInOut(board.GPIO0),
...     dc=digitalio.DigitalInOut(board.GPIO15))
>>> display.fill(color565(0xff, 0x11, 0x22))
>>> display.pixel(120, 160, 0)
```

scroll (*dy=None*)

Scroll the display by delta y

6.5 adafruit_rgb_display.s6d02a1

A simple driver for the S6D02A1-based displays.

- Author(s): Radomir Dopieralski, Michael McWethy

class adafruit_rgb_display.s6d02a1.**S6D02A1** (*spi, dc, cs, rst=None, width=128, height=160, rotation=0*)

A simple driver for the S6D02A1-based displays.

```
>>> import busio
>>> import digitalio
>>> import board
>>> from adafruit_rgb_display import color565
>>> import adafruit_rgb_display.s6d02a1 as s6d02a1
>>> spi = busio.SPI(clock=board.SCK, MOSI=board.MOSI, MISO=board.MISO)
>>> display = s6d02a1.S6D02A1(spi, cs=digitalio.DigitalInOut(board.GPIO0),
...     dc=digitalio.DigitalInOut(board.GPIO15), rst=digitalio.DigitalInOut(board.
↪GPIO16))
>>> display.fill(0x7521)
>>> display.pixel(64, 64, 0)
```

6.6 adafruit_rgb_display.ssd1331

A simple driver for the SSD1331-based displays.

- Author(s): Radomir Dopieralski, Michael McWethy

class adafruit_rgb_display.ssd1331.**SSD1331** (*spi, dc, cs, rst=None, width=96, height=64, baudrate=16000000, polarity=0, phase=0, *, rotation=0*)

A simple driver for the SSD1331-based displays.

```
import busio
import digitalio
import board
from adafruit_rgb_display import color565
import adafruit_rgb_display.ssd1331 as ssd1331
spi = busio.SPI(clock=board.SCK, MOSI=board.MOSI, MISO=board.MISO)
display = ssd1331.SSD1331(spi, cs=digitalio.DigitalInOut(board.GPIO0),
                        dc=digitalio.DigitalInOut(board.GPIO15),
                        rst=digitalio.DigitalInOut(board.GPIO16))

display.fill(0x7521)
display.pixel(32, 32, 0)
```

```
write(command=None, data=None)
    write procedure specific to SSD1331
```

6.7 adafruit_rgb_display.ssd1351

A simple driver for the SSD1351-based displays.

- Author(s): Radomir Dopieralski, Michael McWethy

```
class adafruit_rgb_display.ssd1351.SSD1351(spi, dc, cs, rst=None, width=128, height=128,
                                             baudrate=16000000, polarity=0, phase=0, *,
                                             x_offset=0, y_offset=0, rotation=0)
```

A simple driver for the SSD1351-based displays.

```
>>> import busio
>>> import digitalio
>>> import board
>>> from adafruit_rgb_display import color565
>>> import adafruit_rgb_display.ssd1351 as ssd1351
>>> spi = busio.SPI(clock=board.SCK, MOSI=board.MOSI, MISO=board.MISO)
>>> display = ssd1351.SSD1351(spi, cs=digitalio.DigitalInOut(board.GPIO0),
...     dc=digitalio.DigitalInOut(board.GPIO15), rst=digitalio.DigitalInOut(board.
->GPIO16))
>>> display.fill(0x7521)
>>> display.pixel(32, 32, 0)
```

6.8 adafruit_rgb_display.st7735

A simple driver for the ST7735-based displays.

- Author(s): Radomir Dopieralski, Michael McWethy

```
class adafruit_rgb_display.st7735.ST7735(spi, dc, cs, rst=None, width=128, height=128,
                                           baudrate=16000000, polarity=0, phase=0, *,
                                           x_offset=0, y_offset=0, rotation=0)
```

A simple driver for the ST7735-based displays.

```
>>> import busio
>>> import digitalio
>>> import board
>>> from adafruit_rgb_display import color565
```

(continues on next page)

(continued from previous page)

```
>>> import adafruit_rgb_display.st7735 as st7735
>>> spi = busio.SPI(clock=board.SCK, MOSI=board.MOSI, MISO=board.MISO)
>>> display = st7735.ST7735(spi, cs=digitalio.DigitalInOut(board.GPIO0),
...     dc=digitalio.DigitalInOut(board.GPIO15), rst=digitalio.DigitalInOut(board.
↪GPIO16))
>>> display.fill(0x7521)
>>> display.pixel(64, 64, 0)
```

class adafruit_rgb_display.st7735.**ST7735R**(*spi, dc, cs, rst=None, width=128, height=160, baudrate=16000000, polarity=0, phase=0, *, x_offset=0, y_offset=0, rotation=0, bgr=False*)

A simple driver for the ST7735R-based displays.

init ()

Run the initialization commands.

class adafruit_rgb_display.st7735.**ST7735S**(*spi, dc, cs, bl, rst=None, width=128, height=160, baudrate=16000000, polarity=0, phase=0, *, x_offset=2, y_offset=1, rotation=0*)

A simple driver for the ST7735S-based displays.

6.9 adafruit_rgb_display.st7789

A simple driver for the ST7789-based displays.

- Author(s): Melissa LeBlanc-Williams

class adafruit_rgb_display.st7789.**ST7789**(*spi, dc, cs, rst=None, width=240, height=320, baudrate=16000000, polarity=0, phase=0, *, x_offset=0, y_offset=0, rotation=0*)

A simple driver for the ST7789-based displays.

```
>>> import busio
>>> import digitalio
>>> import board
>>> from adafruit_rgb_display import color565
>>> import adafruit_rgb_display.st7789 as st7789
>>> spi = busio.SPI(clock=board.SCK, MOSI=board.MOSI, MISO=board.MISO)
>>> display = st7789.ST7789(spi, cs=digitalio.DigitalInOut(board.GPIO0),
...     dc=digitalio.DigitalInOut(board.GPIO15), rst=digitalio.DigitalInOut(board.
↪GPIO16))
>>> display.fill(0x7521)
>>> display.pixel(64, 64, 0)
```

init ()

Run the initialization commands.

CHAPTER 7

Indices and tables

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

a

`adafruit_rgb_display.hx8353`, [16](#)
`adafruit_rgb_display.ili9341`, [16](#)
`adafruit_rgb_display.rgb`, [15](#)
`adafruit_rgb_display.s6d02a1`, [17](#)
`adafruit_rgb_display.ssd1331`, [17](#)
`adafruit_rgb_display.ssd1351`, [18](#)
`adafruit_rgb_display.st7735`, [18](#)
`adafruit_rgb_display.st7789`, [19](#)

A

`adafruit_rgb_display.hx8353` (module), 16
`adafruit_rgb_display.ili9341` (module), 16
`adafruit_rgb_display.rgb` (module), 15
`adafruit_rgb_display.s6d02a1` (module), 17
`adafruit_rgb_display.ssd1331` (module), 17
`adafruit_rgb_display.ssd1351` (module), 18
`adafruit_rgb_display.st7735` (module), 18
`adafruit_rgb_display.st7789` (module), 19

C

`color565()` (in module `adafruit_rgb_display.rgb`), 16

D

`deinit()` (`adafruit_rgb_display.rgb.DummyPin` method), 16
`direction` (`adafruit_rgb_display.rgb.DummyPin` attribute), 16
`Display` (class in `adafruit_rgb_display.rgb`), 15
`DisplaySPI` (class in `adafruit_rgb_display.rgb`), 15
`DummyPin` (class in `adafruit_rgb_display.rgb`), 16

F

`fill()` (`adafruit_rgb_display.rgb.Display` method), 15
`fill_rectangle()` (`adafruit_rgb_display.rgb.Display` method), 15

H

`hline()` (`adafruit_rgb_display.rgb.Display` method), 15
`HX8353` (class in `adafruit_rgb_display.hx8353`), 16

I

`ILI9341` (class in `adafruit_rgb_display.ili9341`), 17
`image()` (`adafruit_rgb_display.rgb.Display` method), 15
`image_to_data()` (in module `adafruit_rgb_display.rgb`), 16
`init()` (`adafruit_rgb_display.rgb.Display` method), 15

`init()` (`adafruit_rgb_display.st7735.ST7735R` method), 19
`init()` (`adafruit_rgb_display.st7789.ST7789` method), 19

P

`pixel()` (`adafruit_rgb_display.rgb.Display` method), 15
`pull` (`adafruit_rgb_display.rgb.DummyPin` attribute), 16

R

`read()` (`adafruit_rgb_display.rgb.DisplaySPI` method), 15
`reset()` (`adafruit_rgb_display.rgb.DisplaySPI` method), 15
`rotation` (`adafruit_rgb_display.rgb.Display` attribute), 15

S

`S6D02A1` (class in `adafruit_rgb_display.s6d02a1`), 17
`scroll()` (`adafruit_rgb_display.ili9341.ILI9341` method), 17
`SSD1331` (class in `adafruit_rgb_display.ssd1331`), 17
`SSD1351` (class in `adafruit_rgb_display.ssd1351`), 18
`ST7735` (class in `adafruit_rgb_display.st7735`), 18
`ST7735R` (class in `adafruit_rgb_display.st7735`), 19
`ST7735S` (class in `adafruit_rgb_display.st7735`), 19
`ST7789` (class in `adafruit_rgb_display.st7789`), 19
`switch_to_input()` (`adafruit_rgb_display.rgb.DummyPin` method), 16
`switch_to_output()` (`adafruit_rgb_display.rgb.DummyPin` method), 16

V

`value` (`adafruit_rgb_display.rgb.DummyPin` attribute), 16

`vline()` (*adafruit_rgb_display.rgb.Display* method),
[15](#)

W

`write()` (*adafruit_rgb_display.rgb.DisplaySPI*
method), [16](#)

`write()` (*adafruit_rgb_display.ssd1331.SSD1331*
method), [18](#)