

---

# **AdafruitRGB***DisplayLibraryDocumentation*

## ***Release 1.0***

**Michale McWethy**

**May 07, 2019**



---

## Contents

---

<b>1</b>	<b>Dependencies</b>	<b>3</b>
<b>2</b>	<b>Usage Example</b>	<b>5</b>
<b>3</b>	<b>Contributing</b>	<b>7</b>
<b>4</b>	<b>Building locally</b>	<b>9</b>
4.1	Sphinx documentation . . . . .	9
<b>5</b>	<b>Table of Contents</b>	<b>11</b>
5.1	Simple test . . . . .	11
5.2	adafruit_rgb_display.rgb . . . . .	13
5.3	adafruit_rgb_display.hx8353 . . . . .	14
5.4	adafruit_rgb_display.ili9341 . . . . .	14
5.5	adafruit_rgb_display.s6d02a1 . . . . .	15
5.6	adafruit_rgb_display.ssd1331 . . . . .	15
5.7	adafruit_rgb_display.ssd1351 . . . . .	15
5.8	adafruit_rgb_display.st7735 . . . . .	16
<b>6</b>	<b>Indices and tables</b>	<b>17</b>
	<b>Python Module Index</b>	<b>19</b>



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.



# 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](#).





## CHAPTER 2

---

### Usage Example

---

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



## CHAPTER 3

---

### Contributing

---

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



## CHAPTER 4

---

### Building locally

---

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-rgb_display --
↳ library_location .
```

### 4.1 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.



## 5.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/rgbdisplay\_simpletest.py

```
1  # Quick test of TFT FeatherWing (ST7789) with Feather M0 or M4
2  # This will work even on a device running displayio
3  # Will fill the TFT black and put a red pixel in the center, wait 2 seconds,
4  # then fill the screen blue (with no pixel), wait 2 seconds, and repeat.
5  import time
6  import random
7  import digitalio
8  import board
9  import displayio
10
11  from adafruit_rgb_display.rgb import color565
12  import adafruit_rgb_display.st7789 as st7789
13
14  displayio.release_displays()
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(color565(random.randint(0, 255),
43                             random.randint(0, 255),
44                             random.randint(0, 255)))
45     # Pause 2 seconds.
46     time.sleep(2)

```

Listing 2: examples/rgbdisplay\_ili9341test.py

```

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

```

(continues on next page)



(continued from previous page)

```

35 display.pixel(display.width//2, display.height//2, color565(255, 0, 0))
36 # Pause 2 seconds.
37 time.sleep(2)
38 # Clear the screen a random color
39 display.fill(color565(random.randint(0, 255),
40                      random.randint(0, 255),
41                      random.randint(0, 255)))
42 # Pause 2 seconds.
43 time.sleep(2)

```

## 5.2 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)

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.

**init**()

Run the initialization commands.

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

Read or write a pixel at a given position.

**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)

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.

## 5.3 `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)`  
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.HX8353(spi, cs=digitalio.DigitalInOut(board.GPIO0),
...     dc=digitalio.DigitalInOut(board.GPIO15))
>>> display.fill(0x7521)
>>> display.pixel(64, 64, 0)
```

## 5.4 `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)`  
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

## 5.5 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*)

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)
```

## 5.6 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*)

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

## 5.7 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*)

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)
```

## 5.8 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*)

A simple driver for the ST7735-based displays.

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

A simple driver for the ST7735R-based displays.

**init** ()

Run the initialization commands.

## CHAPTER 6

---

### Indices and tables

---

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



### a

`adafruit_rgb_display.hx8353`, [14](#)  
`adafruit_rgb_display.ili9341`, [14](#)  
`adafruit_rgb_display.rgb`, [13](#)  
`adafruit_rgb_display.s6d02a1`, [15](#)  
`adafruit_rgb_display.ssd1331`, [15](#)  
`adafruit_rgb_display.ssd1351`, [15](#)  
`adafruit_rgb_display.st7735`, [16](#)





## A

adafruit\_rgb\_display.hx8353 (module), 14  
adafruit\_rgb\_display.ili9341 (module), 14  
adafruit\_rgb\_display.rgb (module), 13  
adafruit\_rgb\_display.s6d02a1 (module), 15  
adafruit\_rgb\_display.ssd1331 (module), 15  
adafruit\_rgb\_display.ssd1351 (module), 15  
adafruit\_rgb\_display.st7735 (module), 16

## C

color565() (in module adafruit\_rgb\_display.rgb), 14

## D

deinit() (adafruit\_rgb\_display.rgb.DummyPin method), 13  
direction (adafruit\_rgb\_display.rgb.DummyPin attribute), 13  
Display (class in adafruit\_rgb\_display.rgb), 13  
DisplaySPI (class in adafruit\_rgb\_display.rgb), 13  
DummyPin (class in adafruit\_rgb\_display.rgb), 13

## F

fill() (adafruit\_rgb\_display.rgb.Display method), 13  
fill\_rectangle() (adafruit\_rgb\_display.rgb.Display method), 13

## H

hline() (adafruit\_rgb\_display.rgb.Display method), 13  
HX8353 (class in adafruit\_rgb\_display.hx8353), 14

## I

ILI9341 (class in adafruit\_rgb\_display.ili9341), 14  
init() (adafruit\_rgb\_display.rgb.Display method), 13  
init() (adafruit\_rgb\_display.st7735.ST7735R method), 16

## P

pixel() (adafruit\_rgb\_display.rgb.Display method), 13

pull (adafruit\_rgb\_display.rgb.DummyPin attribute), 13

## R

read() (adafruit\_rgb\_display.rgb.DisplaySPI method), 13  
reset() (adafruit\_rgb\_display.rgb.DisplaySPI method), 13

## S

S6D02A1 (class in adafruit\_rgb\_display.s6d02a1), 15  
scroll() (adafruit\_rgb\_display.ili9341.ILI9341 method), 14  
SSD1331 (class in adafruit\_rgb\_display.ssd1331), 15  
SSD1351 (class in adafruit\_rgb\_display.ssd1351), 16  
ST7735 (class in adafruit\_rgb\_display.st7735), 16  
ST7735R (class in adafruit\_rgb\_display.st7735), 16  
switch\_to\_input() (adafruit\_rgb\_display.rgb.DummyPin method), 14  
switch\_to\_output() (adafruit\_rgb\_display.rgb.DummyPin method), 14

## V

value (adafruit\_rgb\_display.rgb.DummyPin attribute), 14  
vline() (adafruit\_rgb\_display.rgb.Display method), 13

## W

write() (adafruit\_rgb\_display.rgb.DisplaySPI method), 13  
write() (adafruit\_rgb\_display.ssd1331.SSD1331 method), 15