

---

# **AdafruitRGB***DisplayLibraryDocumentation*

## ***Release 1.0***

**Michale McWethy**

**Feb 17, 2020**



---

## Contents

---

<b>1</b>	<b>Dependencies</b>	<b>3</b>
<b>2</b>	<b>Installing from PyPI</b>	<b>5</b>
<b>3</b>	<b>Usage Example</b>	<b>7</b>
<b>4</b>	<b>Contributing</b>	<b>9</b>
<b>5</b>	<b>Documentation</b>	<b>11</b>
<b>6</b>	<b>Table of Contents</b>	<b>13</b>
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
<b>7</b>	<b>Indices and tables</b>	<b>21</b>
	<b>Python Module Index</b>	<b>23</b>
	<b>Index</b>	<b>25</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 (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 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

---

```
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.
    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  # 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/rgb\_display\_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)

```

## 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.

**definit** ()

Dummy `DigitalInOut` `definit`

**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.HX8353(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, ro-  
tation=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
```

(continues on next page)

(continued from previous page)

```

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
>>> import adafruit_rgb_display.st7735 as st7735
>>> spi = busio.SPI(clock=board.SCK, MOSI=board.MOSI, MISO=board.MISO)

```

(continues on next page)

(continued from previous page)

```
>>> 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, polar-
                                         ity=0, phase=0, *, x_offset=2, y_offset=1,
                                         rotation=0)
```

A simple driver for the ST7735S-based displays.





## 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](#)



## 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

## 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`), 15

## 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`), 16  
`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

## 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  
`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*), [15](#)

`write()` (*adafruit\_rgb\_display.ssd1331.SSD1331*  
*method*), [18](#)