

---

**Adafruit**  
**CIRCUITPYTHON***CHARLCDLibraryDocumenta*  
**Release 1.0**

**Brent Rubell**

**Feb 09, 2021**



---

## Contents

---

<b>1</b>	<b>Installing from PyPI</b>	<b>3</b>
<b>2</b>	<b>Dependencies</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_character_lcd.character_lcd .....	19
6.2.1	Implementation Notes .....	19
6.3	adafruit_character_lcd.character_lcd_i2c .....	25
6.3.1	Implementation Notes .....	25
6.4	adafruit_character_lcd.character_lcd_spi .....	25
6.4.1	Implementation Notes .....	25
6.5	adafruit_character_lcd.character_lcd_i2c .....	26
6.5.1	Implementation Notes .....	26
<b>7</b>	<b>Indices and tables</b>	<b>29</b>
	<b>Python Module Index</b>	<b>31</b>
	<b>Index</b>	<b>33</b>



This library is compatible with standard Character LCDs such as: \* Adafruit Standard LCD 16x2 \* Adafruit RGB backlight negative LCD 16x2 \* Adafruit RGB backlight negative LCD 20x4



# CHAPTER 1

---

## 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-charlcd
```

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

```
sudo pip3 install adafruit-circuitpython-charlcd
```

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-charlcd
```





## CHAPTER 2

---

### Dependencies

---

This driver depends on:

- [Adafruit CircuitPython](#)
- [Adafruit CircuitPython BusDevice](#)
- [Adafruit CircuitPython MCP230xx](#)
- [Adafruit CircuitPython 74HC595](#)

I2C & SPI displays also depend on:

- [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 3

---

### Usage Example

---

The `Character_LCD` class interfaces a predefined Character LCD display with CircuitPython.

```
import board
import digitalio
import adafruit_character_lcd.character_lcd as character_lcd
```

You must define the data pins (RS, EN, D4, D5, D6, D7) in your code before using the `Character_LCD` class. If you want to have on/off backlight functionality, you can also define your backlight as `lcd_backlight`. Otherwise, the backlight will always remain on. The following is an example setup.

```
lcd_rs = digitalio.DigitalInOut(board.D7)
lcd_en = digitalio.DigitalInOut(board.D8)
lcd_d7 = digitalio.DigitalInOut(board.D12)
lcd_d6 = digitalio.DigitalInOut(board.D11)
lcd_d5 = digitalio.DigitalInOut(board.D10)
lcd_d4 = digitalio.DigitalInOut(board.D9)
lcd_backlight = digitalio.DigitalInOut(board.D13)
```

You must also define the size of the CharLCD by specifying its `lcd_columns` and `lcd_rows`:

```
lcd_columns = 16
lcd_rows = 2
```

After you have set up your LCD, we can make the device by calling it

```
lcd = character_lcd.Character_LCD_Mono(lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7,
↪ lcd_columns, lcd_rows, lcd_backlight)
```

To verify that your pins are correct, print a hello message to the CharLCD:

```
lcd.message = "Hello\nCircuitPython"
```

Custom character example with `create_char()` is provided within `/examples/`



## 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/charlcd\_mono\_simpletest.py

```
1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  """Simple test for monochromatic character LCD"""
5  import time
6  import board
7  import digitalio
8  import adafruit_character_lcd.character_lcd as characterlcd
9
10 # Modify this if you have a different sized character LCD
11 lcd_columns = 16
12 lcd_rows = 2
13
14 # Metro M0/M4 Pin Config:
15 lcd_rs = digitalio.DigitalInOut(board.D7)
16 lcd_en = digitalio.DigitalInOut(board.D8)
17 lcd_d7 = digitalio.DigitalInOut(board.D12)
18 lcd_d6 = digitalio.DigitalInOut(board.D11)
19 lcd_d5 = digitalio.DigitalInOut(board.D10)
20 lcd_d4 = digitalio.DigitalInOut(board.D9)
21 lcd_backlight = digitalio.DigitalInOut(board.D13)
22
23 # Initialise the LCD class
24 lcd = characterlcd.Character_LCD_Mono(
25     lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7, lcd_columns, lcd_rows, lcd_
26     ↪backlight
27 )
```

(continues on next page)

(continued from previous page)

```

27
28 # Turn backlight on
29 lcd.backlight = True
30 # Print a two line message
31 lcd.message = "Hello\nCircuitPython"
32 # Wait 5s
33 time.sleep(5)
34 lcd.clear()
35 # Print two line message right to left
36 lcd.text_direction = lcd.RIGHT_TO_LEFT
37 lcd.message = "Hello\nCircuitPython"
38 # Wait 5s
39 time.sleep(5)
40 # Return text direction to left to right
41 lcd.text_direction = lcd.LEFT_TO_RIGHT
42 # Display cursor
43 lcd.clear()
44 lcd.cursor = True
45 lcd.message = "Cursor! "
46 # Wait 5s
47 time.sleep(5)
48 # Display blinking cursor
49 lcd.clear()
50 lcd.blink = True
51 lcd.message = "Blinky Cursor!"
52 # Wait 5s
53 time.sleep(5)
54 lcd.blink = False
55 lcd.clear()
56 # Create message to scroll
57 scroll_msg = "<-- Scroll"
58 lcd.message = scroll_msg
59 # Scroll message to the left
60 for i in range(len(scroll_msg)):
61     time.sleep(0.5)
62     lcd.move_left()
63 lcd.clear()
64 lcd.message = "Going to sleep\nCya later!"
65 time.sleep(3)
66 # Turn backlight off
67 lcd.backlight = False
68 time.sleep(2)

```

Listing 2: examples/charlcd\_rgb\_simpletest.py

```

1 # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2 # SPDX-License-Identifier: MIT
3
4 """Simple test for RGB character LCD"""
5 import time
6 import board
7 import digitalio
8 import pwmio
9 import adafruit_character_lcd.character_lcd as characterlcd
10
11 # Modify this if you have a different sized character LCD

```

(continues on next page)

(continued from previous page)

```

12 lcd_columns = 16
13 lcd_rows = 2
14
15 # Metro M0/M4 Pin Config:
16 lcd_rs = digitalio.DigitalInOut(board.D7)
17 lcd_en = digitalio.DigitalInOut(board.D8)
18 lcd_d7 = digitalio.DigitalInOut(board.D12)
19 lcd_d6 = digitalio.DigitalInOut(board.D11)
20 lcd_d5 = digitalio.DigitalInOut(board.D10)
21 lcd_d4 = digitalio.DigitalInOut(board.D9)
22 red = pwmio.PWMOut(board.D3)
23 green = pwmio.PWMOut(board.D5)
24 blue = pwmio.PWMOut(board.D6)
25
26 # Initialise the LCD class
27 lcd = characterlcd.Character_LCD_RGB(
28     lcd_rs,
29     lcd_en,
30     lcd_d4,
31     lcd_d5,
32     lcd_d6,
33     lcd_d7,
34     lcd_columns,
35     lcd_rows,
36     red,
37     green,
38     blue,
39 )
40
41 lcd.clear()
42 # Set LCD color to red
43 lcd.color = [100, 0, 0]
44 time.sleep(1)
45 # Print two line message
46 lcd.message = "Hello\nCircuitPython"
47 # Wait 5s
48 time.sleep(5)
49 # Set LCD color to blue
50 lcd.color = [0, 100, 0]
51 time.sleep(1)
52 # Set LCD color to green
53 lcd.color = [0, 0, 100]
54 time.sleep(1)
55 # Set LCD color to purple
56 lcd.color = [50, 0, 50]
57 time.sleep(1)
58 lcd.clear()
59 # Print two line message right to left
60 lcd.text_direction = lcd.RIGHT_TO_LEFT
61 lcd.message = "Hello\nCircuitPython"
62 # Wait 5s
63 time.sleep(5)
64 # Return text direction to left to right
65 lcd.text_direction = lcd.LEFT_TO_RIGHT
66 # Display cursor
67 lcd.clear()
68 lcd.cursor = True

```

(continues on next page)

(continued from previous page)

```

69 lcd.message = "Cursor! "
70 # Wait 5s
71 time.sleep(5)
72 # Display blinking cursor
73 lcd.clear()
74 lcd.blink = True
75 lcd.message = "Blinky Cursor!"
76 # Wait 5s
77 time.sleep(5)
78 lcd.blink = False
79 lcd.clear()
80 # Create message to scroll
81 scroll_msg = "<-- Scroll"
82 lcd.message = scroll_msg
83 # Scroll to the left
84 for i in range(len(scroll_msg)):
85     time.sleep(0.5)
86     lcd.move_left()
87 lcd.clear()
88 time.sleep(1)
89 lcd.message = "Going to sleep\nCya later!"
90 time.sleep(5)
91 # Turn off LCD backlights and clear text
92 lcd.color = [0, 0, 0]
93 lcd.clear()

```

Listing 3: examples/charlcd\_i2c\_mono\_simpletest.py

```

1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  """Simple test for 16x2 character lcd connected to an MCP23008 I2C LCD backpack."""
5  import time
6  import board
7  import busio
8  import adafruit_character_lcd.character_lcd_i2c as character_lcd
9
10 # Modify this if you have a different sized Character LCD
11 lcd_columns = 16
12 lcd_rows = 2
13
14 # Initialise I2C bus.
15 i2c = busio.I2C(board.SCL, board.SDA)
16
17 # Initialise the lcd class
18 lcd = character_lcd.Character_LCD_I2C(i2c, lcd_columns, lcd_rows)
19
20 # Turn backlight on
21 lcd.backlight = True
22 # Print a two line message
23 lcd.message = "Hello\nCircuitPython"
24 # Wait 5s
25 time.sleep(5)
26 lcd.clear()
27 # Print two line message right to left
28 lcd.text_direction = lcd.RIGHT_TO_LEFT

```

(continues on next page)

(continued from previous page)

```

29 lcd.message = "Hello\nCircuitPython"
30 # Wait 5s
31 time.sleep(5)
32 # Return text direction to left to right
33 lcd.text_direction = lcd.LEFT_TO_RIGHT
34 # Display cursor
35 lcd.clear()
36 lcd.cursor = True
37 lcd.message = "Cursor! "
38 # Wait 5s
39 time.sleep(5)
40 # Display blinking cursor
41 lcd.clear()
42 lcd.blink = True
43 lcd.message = "Blinky Cursor!"
44 # Wait 5s
45 time.sleep(5)
46 lcd.blink = False
47 lcd.clear()
48 # Create message to scroll
49 scroll_msg = "<-- Scroll"
50 lcd.message = scroll_msg
51 # Scroll message to the left
52 for i in range(len(scroll_msg)):
53     time.sleep(0.5)
54     lcd.move_left()
55 lcd.clear()
56 lcd.message = "Going to sleep\nCya later!"
57 time.sleep(5)
58 # Turn backlight off
59 lcd.backlight = False
60 time.sleep(2)

```

Listing 4: examples/charlcd\_spi\_mono\_simpletest.py

```

1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  """Simple test for 16x2 character LCD connected to 74HC595 SPI LCD backpack."""
5  import time
6  import board
7  import busio
8  import digitalio
9  import adafruit_character_lcd.character_lcd_spi as character_lcd
10
11 # Modify this if you have a different sized character LCD
12 lcd_columns = 16
13 lcd_rows = 2
14
15 # Backpack connection configuration:
16 clk = board.SCK # Pin connected to backpack CLK.
17 data = board.MOSI # Pin connected to backpack DAT/data.
18 latch = board.D5 # Pin connected to backpack LAT/latch.
19
20 # Initialise SPI bus.
21 spi = busio.SPI(clk, MOSI=data)

```

(continues on next page)

(continued from previous page)

```

22
23 # Initialise the LCD class
24 latch = digitalio.DigitalInOut(latch)
25 lcd = character_lcd.Character_LCD_SPI(spi, latch, lcd_columns, lcd_rows)
26
27 # Turn backlight on
28 lcd.backlight = True
29 # Print a two line message
30 lcd.message = "Hello\nCircuitPython"
31 # Wait 5s
32 time.sleep(5)
33 lcd.clear()
34 # Print two line message right to left
35 lcd.text_direction = lcd.RIGHT_TO_LEFT
36 lcd.message = "Hello\nCircuitPython"
37 # Wait 5s
38 time.sleep(5)
39 # Return text direction to left to right
40 lcd.text_direction = lcd.LEFT_TO_RIGHT
41 # Display cursor
42 lcd.clear()
43 lcd.cursor = True
44 lcd.message = "Cursor! "
45 # Wait 5s
46 time.sleep(5)
47 # Display blinking cursor
48 lcd.clear()
49 lcd.blink = True
50 lcd.message = "Blinky Cursor!"
51 # Wait 5s
52 time.sleep(5)
53 lcd.blink = False
54 lcd.clear()
55 # Create message to scroll
56 scroll_msg = "<-- Scroll"
57 lcd.message = scroll_msg
58 # Scroll message to the left
59 for i in range(len(scroll_msg)):
60     time.sleep(0.5)
61     lcd.move_left()
62 lcd.clear()
63 lcd.message = "Going to sleep\nCya later!"
64 # Turn backlight off
65 lcd.backlight = False
66 time.sleep(2)

```

Listing 5: examples/charlcd\_keypad\_simpletest.py

```

1 # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2 # SPDX-License-Identifier: MIT
3
4 """Simple test for keypad on I2C RGB character LCD Shield or Pi Plate kits"""
5 import time
6 import board
7 import busio
8 import adafruit_character_lcd.character_lcd_rgb_i2c as character_lcd

```

(continues on next page)

(continued from previous page)

```

9
10 # Modify this if you have a different sized Character LCD
11 lcd_columns = 16
12 lcd_rows = 2
13
14 # Initialise I2C bus.
15 i2c = busio.I2C(board.SCL, board.SDA)
16
17 # Initialise the LCD class
18 lcd = character_lcd.Character_LCD_RGB_I2C(i2c, lcd_columns, lcd_rows)
19
20 lcd.clear()
21 lcd.color = [100, 0, 0]
22 while True:
23     if lcd.left_button:
24         print("Left!")
25         lcd.message = "Left!"
26
27     elif lcd.up_button:
28         print("Up!")
29         lcd.message = "Up!"
30
31     elif lcd.down_button:
32         print("Down!")
33         lcd.message = "Down!"
34
35     elif lcd.right_button:
36         print("Right!")
37         lcd.message = "Right!"
38
39     elif lcd.select_button:
40         print("Select!")
41         lcd.message = "Select!"
42
43     else:
44         time.sleep(0.1)
45         lcd.clear()

```

## 6.2 adafruit\_character\_lcd.character\_lcd

Module for interfacing with monochromatic character LCDs

- Author(s): Kattni Rembor, Brent Rubell, Asher Lieber, Tony DiCola (original python charLCD library)

### 6.2.1 Implementation Notes

#### Hardware:

“\* Adafruit Character LCDs”

#### Software and Dependencies:

- Adafruit CircuitPython firmware: <https://github.com/adafruit/circuitpython/releases>

- Adafruit’s Bus Device library (when using I2C/SPI): [https://github.com/adafruit/Adafruit\\_CircuitPython\\_BusDevice](https://github.com/adafruit/Adafruit_CircuitPython_BusDevice)

**class** adafruit\_character\_lcd.character\_lcd.**Character\_LCD**(rs, en, d4, d5, d6, d7, columns, lines)

Base class for character LCD.

#### Parameters

- **rs** (*DigitalInOut*) – The reset data line
- **en** (*DigitalInOut*) – The enable data line
- **d4** (*DigitalInOut*) – The data line 4
- **d5** (*DigitalInOut*) – The data line 5
- **d6** (*DigitalInOut*) – The data line 6
- **d7** (*DigitalInOut*) – The data line 7
- **columns** – The columns on the charLCD
- **lines** – The lines on the charLCD

#### blink

Blink the cursor. True to blink the cursor. False to stop blinking.

The following example shows a message followed by a blinking cursor for five seconds.

```
import time
import board
import busio
import adafruit_character_lcd.character_lcd_i2c as character_lcd

i2c = busio.I2C(board.SCL, board.SDA)
lcd = character_lcd.Character_LCD_I2C(i2c, 16, 2)

lcd.blink = True
lcd.message = "Blinky cursor!"
time.sleep(5)
lcd.blink = False
```

#### clear()

Clears everything displayed on the LCD.

The following example displays, “Hello, world!”, then clears the LCD.

```
import time
import board
import busio
import adafruit_character_lcd.character_lcd_i2c as character_lcd

i2c = busio.I2C(board.SCL, board.SDA)
lcd = character_lcd.Character_LCD_I2C(i2c, 16, 2)

lcd.message = "Hello, world!"
time.sleep(5)
lcd.clear()
```

#### column\_align

If True, message text after ‘n’ starts directly below start of first character in message. If False, text after ‘n’ starts at column zero.



### **create\_char** (*location, pattern*)

Fill one of the first 8 CGRAM locations with custom characters. The location parameter should be between 0 and 7 and pattern should provide an array of 8 bytes containing the pattern. E.g. you can easily design your custom character at <http://www.quinapalus.com/hd44780udg.html> To show your custom character use, for example, `lcd.message = ""`

#### **Parameters**

- **location** – integer in range(8) to store the created character
- **pattern** (~*bytes*) – len(8) describes created character

### **cursor**

True if cursor is visible. False to stop displaying the cursor.

The following example shows the cursor after a displayed message:

```
import time
import board
import busio
import adafruit_character_lcd.character_lcd_i2c as character_lcd

i2c = busio.I2C(board.SCL, board.SDA)
lcd = character_lcd.Character_LCD_I2C(i2c, 16, 2)

lcd.cursor = True
lcd.message = "Cursor! "
time.sleep(5)
```

### **cursor\_position** (*column, row*)

Move the cursor to position `column, row` for the next message only. Displaying a message resets the cursor position to (0, 0).

**param column** column location

**param row** row location

### **display**

Enable or disable the display. True to enable the display. False to disable the display.

The following example displays, “Hello, world!” on the LCD and then turns the display off.

```
import time
import board
import busio
import adafruit_character_lcd.character_lcd_i2c as character_lcd

i2c = busio.I2C(board.SCL, board.SDA)
lcd = character_lcd.Character_LCD_I2C(i2c, 16, 2)

lcd.message = "Hello, world!"
time.sleep(5)
lcd.display = False
```

### **home** ()

Moves the cursor “home” to position (0, 0).

### **message**

Display a string of text on the character LCD. Start position is (0,0) if `cursor_position` is not set. If `cursor_position` is set, message starts at the set position from the left for left to right text and from the right for right to left text. Resets cursor column and row to (0,0) after displaying the message.

The following example displays, “Hello, world!” on the LCD.

```
import time
import board
import busio
import adafruit_character_lcd.character_lcd_i2c as character_lcd

i2c = busio.I2C(board.SCL, board.SDA)
lcd = character_lcd.Character_LCD_I2C(i2c, 16, 2)

lcd.message = "Hello, world!"
time.sleep(5)
```

#### **move\_left()**

Moves displayed text left one column.

The following example scrolls a message to the left off the screen.

```
import time
import board
import busio
import adafruit_character_lcd.character_lcd_i2c as character_lcd

i2c = busio.I2C(board.SCL, board.SDA)
lcd = character_lcd.Character_LCD_I2C(i2c, 16, 2)

scroll_message = "<-- Scroll"
lcd.message = scroll_message
time.sleep(2)
for i in range(len(scroll_message)):
    lcd.move_left()
    time.sleep(0.5)
```

#### **move\_right()**

Moves displayed text right one column.

The following example scrolls a message to the right off the screen.

```
import time
import board
import busio
import adafruit_character_lcd.character_lcd_i2c as character_lcd

i2c = busio.I2C(board.SCL, board.SDA)
lcd = character_lcd.Character_LCD_I2C(i2c, 16, 2)

scroll_message = "Scroll -->"
lcd.message = scroll_message
time.sleep(2)
for i in range(len(scroll_message) + 16):
    lcd.move_right()
    time.sleep(0.5)
```

#### **text\_direction**

The direction the text is displayed. To display the text left to right beginning on the left side of the LCD, set `text_direction = LEFT_TO_RIGHT`. To display the text right to left beginning on the right side of the LCD, set `text_direction = RIGHT_TO_LEFT`. Text defaults to displaying from left to right.

The following example displays “Hello, world!” from right to left.

```
import time
import board
import busio
import adafruit_character_lcd.character_lcd_i2c as character_lcd

i2c = busio.I2C(board.SCL, board.SDA)
lcd = character_lcd.Character_LCD_I2C(i2c, 16, 2)

lcd.text_direction = lcd.RIGHT_TO_LEFT
lcd.message = "Hello, world!"
time.sleep(5)
```

```
class adafruit_character_lcd.character_lcd.Character_LCD_Mono(rs, en, db4,
                                                             db5, db6,
                                                             db7, columns,
                                                             lines, back-
                                                             light_pin=None,
                                                             back-
                                                             light_inverted=False)
```

Interfaces with monochromatic character LCDs.

#### Parameters

- **rs** (*DigitalInOut*) – The reset data line
- **en** (*DigitalInOut*) – The enable data line
- **d4** (*DigitalInOut*) – The data line 4
- **d5** (*DigitalInOut*) – The data line 5
- **d6** (*DigitalInOut*) – The data line 6
- **d7** (*DigitalInOut*) – The data line 7
- **columns** – The columns on the charLCD
- **lines** – The lines on the charLCD
- **backlight\_pin** (*DigitalInOut*) – The backlight pin
- **backlight\_inverted** (*bool*) – False if LCD is not inverted, i.e. backlight pin is connected to common anode. True if LCD is inverted i.e. backlight pin is connected to common cathode.

#### backlight

Enable or disable backlight. True if backlight is on. False if backlight is off.

The following example turns the backlight off, then displays, “Hello, world?”, then turns the backlight on and displays, “Hello, world!”

```
import time
import board
import busio
import adafruit_character_lcd.character_lcd_i2c as character_lcd

i2c = busio.I2C(board.SCL, board.SDA)

lcd = character_lcd.Character_LCD_I2C(i2c, 16, 2)

lcd.backlight = False
```

(continues on next page)

(continued from previous page)

```
lcd.message = "Hello, world?"
time.sleep(5)
lcd.backlight = True
lcd.message = "Hello, world!"
time.sleep(5)
```

```
class adafruit_character_lcd.character_lcd.Character_LCD_RGB(rs,    en,    db4,
                                                             db5,    db6,    db7,
                                                             columns,    lines,
                                                             red,    green,    blue,
                                                             read_write=None)
```

Interfaces with RGB character LCDs.

#### Parameters

- **rs** (*DigitalInOut*) – The reset data line
- **en** (*DigitalInOut*) – The enable data line
- **db4** (*DigitalInOut*) – The data line 4
- **db5** (*DigitalInOut*) – The data line 5
- **db6** (*DigitalInOut*) – The data line 6
- **db7** (*DigitalInOut*) – The data line 7
- **columns** – The columns on the charLCD
- **lines** – The lines on the charLCD
- **DigitalInOut red** (*PWMOut*,) – Red RGB Anode
- **DigitalInOut green** (*PWMOut*,) – Green RGB Anode
- **DigitalInOut blue** (*PWMOut*,) – Blue RGB Anode
- **read\_write** (*DigitalInOut*) – The rw pin. Determines whether to read to or write from the display. Not necessary if only writing to the display. Used on shield.

#### color

The color of the display. Provide a list of three integers ranging 0 - 100, [R, G, B]. 0 is no color, or “off”. 100 is maximum color. For example, the brightest red would be [100, 0, 0], and a half-bright purple would be, [50, 0, 50].

If PWM is unavailable, 0 is off, and non-zero is on. For example, [1, 0, 0] would be red.

The following example turns the LCD red and displays, “Hello, world!”.

```
import time
import board
import busio
import adafruit_character_lcd.character_lcd_rgb_i2c as character_lcd

i2c = busio.I2C(board.SCL, board.SDA)

lcd = character_lcd.Character_LCD_RGB_I2C(i2c, 16, 2)

lcd.color = [100, 0, 0]
lcd.message = "Hello, world!"
time.sleep(5)
```

## 6.3 adafruit\_character\_lcd.character\_lcd\_i2c

Module for using I2C with I2C/SPI character LCD backpack

- Author(s): Kattni Rembor

### 6.3.1 Implementation Notes

#### Hardware:

“\* I2C / SPI character LCD backpack”

#### Software and Dependencies:

- Adafruit CircuitPython firmware: <https://github.com/adafruit/circuitpython/releases>
- Adafruit’s Bus Device library (when using I2C/SPI): [https://github.com/adafruit/Adafruit\\_CircuitPython\\_BusDevice](https://github.com/adafruit/Adafruit_CircuitPython_BusDevice)

**class** adafruit\_character\_lcd.character\_lcd\_i2c.**Character\_LCD\_I2C** (*i2c, columns, lines, address=None, backlight\_inverted=False*)

Character LCD connected to I2C/SPI backpack using its I2C connection. This is a subclass of Character\_LCD and implements all of the same functions and functionality.

To use, import and initialise as follows:

```
import board
import busio
from adafruit_character_lcd.character_lcd_i2c import Character_LCD_I2C

i2c = busio.I2C(board.SCL, board.SDA)
lcd = Character_LCD_I2C(i2c, 16, 2)
```

## 6.4 adafruit\_character\_lcd.character\_lcd\_spi

Module for using SPI with I2C/SPI character LCD backpack

- Author(s): Kattni Rembor

### 6.4.1 Implementation Notes

#### Hardware:

“\* I2C / SPI character LCD backpack”

#### Software and Dependencies:

- Adafruit CircuitPython firmware: <https://github.com/adafruit/circuitpython/releases>
- Adafruit’s Bus Device library (when using I2C/SPI): [https://github.com/adafruit/Adafruit\\_CircuitPython\\_BusDevice](https://github.com/adafruit/Adafruit_CircuitPython_BusDevice)

```
class adafruit_character_lcd.character_lcd_spi.Character_LCD_SPI (spi,    latch,
                                                                columns,
                                                                lines,  back-
                                                                light_inverted=False)
```

Character LCD connected to I2C/SPI backpack using its SPI connection. This is a subclass of Character\_LCD and implements all of the same functions and functionality.

To use, import and initialise as follows:

```
import board
import busio
import digitalio
import adafruit_character_lcd.character_lcd_mono as character_lcd

spi = busio.SPI(board.SCK, MOSI=board.MOSI)
latch = digitalio.DigitalInOut(board.D5)
lcd = character_lcd.Character_LCD_SPI(spi, latch, 16, 2)
```

## 6.5 adafruit\_character\_lcd.character\_lcd\_i2c

Module for using I2C with I2C RGB LCD Shield or I2C RGB LCD Pi Plate

- Author(s): Kattni Rembor

### 6.5.1 Implementation Notes

#### Hardware:

- “\* RGB LCD Shield Kit w/ 16x2 Character Display - Negative Display”
- “\* RGB LCD Shield Kit w/ 16x2 Character Display - Positive Display”
- “\* Adafruit RGB Negative 16x2 LCD+Keypad Kit for Raspberry Pi”
- “\* Adafruit RGB Positive 16x2 LCD+Keypad Kit for Raspberry Pi”

#### Software and Dependencies:

- Adafruit CircuitPython firmware: <https://github.com/adafruit/circuitpython/releases>
- Adafruit’s Bus Device library (when using I2C/SPI): [https://github.com/adafruit/Adafruit\\_CircuitPython\\_BusDevice](https://github.com/adafruit/Adafruit_CircuitPython_BusDevice)

```
class adafruit_character_lcd.character_lcd_rgb_i2c.Character_LCD_RGB_I2C (i2c,
                                                                columns,
                                                                lines,
                                                                ad-
                                                                dress=None)
```

RGB Character LCD connected to I2C shield or Pi plate using I2C connection. This is a subclass of Character\_LCD\_RGB and implements all of the same functions and functionality.

To use, import and initialise as follows:

```
import board
import busio
from adafruit_character_lcd.character_lcd_rgb_i2c import Character_LCD_RGB_I2C
```

(continues on next page)

(continued from previous page)

```
i2c = busio.I2C(board.SCL, board.SDA)
lcd = Character_LCD_RGB_I2C(i2c, 16, 2)
```

#### down\_button

The down button on the RGB Character LCD I2C Shield or Pi plate.

The following example prints “Down!” to the LCD when the down button is pressed:

```
import board
import busio
from adafruit_character_lcd.character_lcd_rgb_i2c import Character_LCD_RGB_I2C

i2c = busio.I2C(board.SCL, board.SDA)
lcd = Character_LCD_RGB_I2C(i2c, 16, 2)

while True:
    if lcd.down_button:
        lcd.message = "Down!"
```

#### left\_button

The left button on the RGB Character LCD I2C Shield or Pi plate.

The following example prints “Left!” to the LCD when the left button is pressed:

```
import board
import busio
from adafruit_character_lcd.character_lcd_rgb_i2c import Character_LCD_RGB_I2C

i2c = busio.I2C(board.SCL, board.SDA)
lcd = Character_LCD_RGB_I2C(i2c, 16, 2)

while True:
    if lcd.left_button:
        lcd.message = "Left!"
```

#### right\_button

The right button on the RGB Character LCD I2C Shield or Pi plate.

The following example prints “Right!” to the LCD when the right button is pressed:

```
import board
import busio
from adafruit_character_lcd.character_lcd_rgb_i2c import Character_LCD_RGB_I2C

i2c = busio.I2C(board.SCL, board.SDA)
lcd = Character_LCD_RGB_I2C(i2c, 16, 2)

while True:
    if lcd.right_button:
        lcd.message = "Right!"
```

#### select\_button

The select button on the RGB Character LCD I2C Shield or Pi plate.

The following example prints “Select!” to the LCD when the select button is pressed:

```
import board
import busio
```

(continues on next page)

(continued from previous page)

```
from adafruit_character_lcd.character_lcd_rgb_i2c import Character_LCD_RGB_I2C

i2c = busio.I2C(board.SCL, board.SDA)
lcd = Character_LCD_RGB_I2C(i2c, 16, 2)

while True:
    if lcd.select_button:
        lcd.message = "Select!"
```

### up\_button

The up button on the RGB Character LCD I2C Shield or Pi plate.

The following example prints “Up!” to the LCD when the up button is pressed:

```
import board
import busio
from adafruit_character_lcd.character_lcd_rgb_i2c import Character_LCD_RGB_I2C

i2c = busio.I2C(board.SCL, board.SDA)
lcd = Character_LCD_RGB_I2C(i2c, 16, 2)

while True:
    if lcd.up_button:
        lcd.message = "Up!"
```



## CHAPTER 7

---

### Indices and tables

---

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



### a

`adafruit_character_lcd.character_lcd`,  
19  
`adafruit_character_lcd.character_lcd_i2c`,  
24  
`adafruit_character_lcd.character_lcd_rgb_i2c`,  
26  
`adafruit_character_lcd.character_lcd_spi`,  
25



## A

adafruit\_character\_lcd.character\_lcd  
(*module*), 19

adafruit\_character\_lcd.character\_lcd\_i2c  
(*module*), 24

adafruit\_character\_lcd.character\_lcd\_rgb\_i2c  
(*module*), 26

adafruit\_character\_lcd.character\_lcd\_spi  
(*module*), 25

## B

backlight (adafruit\_character\_lcd.character\_lcd.Character\_LCD\_Mono  
attribute), 23

blink (adafruit\_character\_lcd.character\_lcd.Character\_LCD  
attribute), 20

## C

Character\_LCD (class in  
adafruit\_character\_lcd.character\_lcd), 20

Character\_LCD\_I2C (class in  
adafruit\_character\_lcd.character\_lcd\_i2c),  
25

Character\_LCD\_Mono (class in  
adafruit\_character\_lcd.character\_lcd), 23

Character\_LCD\_RGB (class in  
adafruit\_character\_lcd.character\_lcd), 24

Character\_LCD\_RGB\_I2C (class in  
adafruit\_character\_lcd.character\_lcd\_rgb\_i2c),  
26

Character\_LCD\_SPI (class in  
adafruit\_character\_lcd.character\_lcd\_spi),  
25

clear () (adafruit\_character\_lcd.character\_lcd.Character\_LCD  
method), 20

color (adafruit\_character\_lcd.character\_lcd.Character\_LCD\_RGB  
attribute), 24

column\_align (adafruit\_character\_lcd.character\_lcd.Character\_LCD  
attribute), 20

create\_char () (adafruit\_character\_lcd.character\_lcd.Character\_LCD  
method), 20

cursor (adafruit\_character\_lcd.character\_lcd.Character\_LCD  
attribute), 21

cursor\_position ()  
(adafruit\_character\_lcd.character\_lcd.Character\_LCD  
method), 21

## D

display (adafruit\_character\_lcd.character\_lcd.Character\_LCD  
attribute), 21

down\_button (adafruit\_character\_lcd.character\_lcd\_rgb\_i2c.Character\_LCD\_RGB\_I2C  
attribute), 27

## H

home () (adafruit\_character\_lcd.character\_lcd.Character\_LCD  
method), 21

## L

left\_button (adafruit\_character\_lcd.character\_lcd\_rgb\_i2c.Character\_LCD\_RGB\_I2C  
attribute), 27

## M

message (adafruit\_character\_lcd.character\_lcd.Character\_LCD  
attribute), 21

move\_left () (adafruit\_character\_lcd.character\_lcd.Character\_LCD  
method), 22

move\_right () (adafruit\_character\_lcd.character\_lcd.Character\_LCD  
method), 22

## R

right\_button (adafruit\_character\_lcd.character\_lcd\_rgb\_i2c.Character\_LCD\_RGB\_I2C  
attribute), 27

## S

select\_button (adafruit\_character\_lcd.character\_lcd\_rgb\_i2c.Character\_LCD\_RGB\_I2C  
attribute), 27

## T

`text_direction(adafruit_character_lcd.character_lcd.Character_LCD  
attribute)`, [22](#)

## U

`up_button(adafruit_character_lcd.character_lcd_rgb_i2c.Character_LCD_RGB_I2C  
attribute)`, [28](#)