
Adafruit
CIRCUITPYTHON*CHARLCDLibraryDocumenta*
Release 1.0

Brent Rubell

Feb 02, 2019

Contents

1	Installing from PyPI	3
2	Dependencies	5
3	Usage Example	7
4	Contributing	9
5	Installation	11
6	Building locally	13
6.1	Sphinx documentation	13
7	Table of Contents	15
7.1	Simple test	15
7.2	adafruit_character_lcd.character_lcd	21
7.2.1	Implementation Notes	21
7.3	adafruit_character_lcd.character_lcd_i2c	26
7.3.1	Implementation Notes	26
7.4	adafruit_character_lcd.character_lcd_spi	27
7.4.1	Implementation Notes	27
7.5	adafruit_character_lcd.character_lcd_i2c	27
7.5.1	Implementation Notes	28
8	Indices and tables	31
	Python Module Index	33

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](#)

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

Installation

This library is **NOT** built into CircuitPython to make it easy to update. To install it either follow the directions below or [install the library bundle](#).

To install:

1. Download and unzip the [latest release zip](#).
2. Copy the unzipped `adafruit_character_lcd` to the `lib` directory on the `CIRCUITPY` or `MICROPYTHON` drive.

CHAPTER 6

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-charlcd --
↳library_location .
```

6.1 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.

7.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/charlcd_mono_simpletest.py

```
1  """Simple test for monochromatic character LCD"""
2  import time
3  import board
4  import digitalio
5  import adafruit_character_lcd.character_lcd as characterlcd
6
7  # Modify this if you have a different sized character LCD
8  lcd_columns = 16
9  lcd_rows = 2
10
11  # Metro M0/M4 Pin Config:
12  lcd_rs = digitalio.DigitalInOut(board.D7)
13  lcd_en = digitalio.DigitalInOut(board.D8)
14  lcd_d7 = digitalio.DigitalInOut(board.D12)
15  lcd_d6 = digitalio.DigitalInOut(board.D11)
16  lcd_d5 = digitalio.DigitalInOut(board.D10)
17  lcd_d4 = digitalio.DigitalInOut(board.D9)
18  lcd_backlight = digitalio.DigitalInOut(board.D13)
19
20  # Initialise the LCD class
21  lcd = characterlcd.Character_LCD_Mono(lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6,
22                                       lcd_d7, lcd_columns, lcd_rows, lcd_backlight)
23
24  # Turn backlight on
25  lcd.backlight = True
26  # Print a two line message
27  lcd.message = "Hello\nCircuitPython"
```

(continues on next page)

(continued from previous page)

```

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

```

Listing 2: examples/charlcd_rgb_simpletest.py

```

1  """Simple test for RGB character LCD"""
2  import time
3  import board
4  import digitalio
5  import pulseio
6  import adafruit_character_lcd.character_lcd as characterlcd
7
8  # Modify this if you have a different sized character LCD
9  lcd_columns = 16
10 lcd_rows = 2
11
12 # Metro M0/M4 Pin Config:
13 lcd_rs = digitalio.DigitalInOut(board.D7)
14 lcd_en = digitalio.DigitalInOut(board.D8)
15 lcd_d7 = digitalio.DigitalInOut(board.D12)
16 lcd_d6 = digitalio.DigitalInOut(board.D11)

```

(continues on next page)

(continued from previous page)

```

17 lcd_d5 = digitalio.DigitalInOut(board.D10)
18 lcd_d4 = digitalio.DigitalInOut(board.D9)
19 red = pulseio.PWMOut(board.D3)
20 green = pulseio.PWMOut(board.D5)
21 blue = pulseio.PWMOut(board.D6)
22
23 # Initialise the LCD class
24 lcd = characterlcd.Character_LCD_RGB(lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7, ↵
↵ lcd_columns,
25                                     lcd_rows, red, green, blue)
26
27 lcd.clear()
28 # Set LCD color to red
29 lcd.color = [100, 0, 0]
30 time.sleep(1)
31 # Print two line message
32 lcd.message = "Hello\nCircuitPython"
33 # Wait 5s
34 time.sleep(5)
35 # Set LCD color to blue
36 lcd.color = [0, 100, 0]
37 time.sleep(1)
38 # Set LCD color to green
39 lcd.color = [0, 0, 100]
40 time.sleep(1)
41 # Set LCD color to purple
42 lcd.color = [50, 0, 50]
43 time.sleep(1)
44 lcd.clear()
45 # Print two line message right to left
46 lcd.text_direction = lcd.RIGHT_TO_LEFT
47 lcd.message = "Hello\nCircuitPython"
48 # Wait 5s
49 time.sleep(5)
50 # Return text direction to left to right
51 lcd.text_direction = lcd.LEFT_TO_RIGHT
52 # Display cursor
53 lcd.clear()
54 lcd.cursor = True
55 lcd.message = "Cursor! "
56 # Wait 5s
57 time.sleep(5)
58 # Display blinking cursor
59 lcd.clear()
60 lcd.blink = True
61 lcd.message = "Blinky Cursor!"
62 # Wait 5s
63 time.sleep(5)
64 lcd.blink = False
65 lcd.clear()
66 # Create message to scroll
67 scroll_msg = '<-- Scroll'
68 lcd.message = scroll_msg
69 # Scroll to the left
70 for i in range(len(scroll_msg)):
71     time.sleep(0.5)
72     lcd.move_left()
```

(continues on next page)

(continued from previous page)

```

73 lcd.clear()
74 time.sleep(1)
75 lcd.message = "Going to sleep\nCya later!"
76 time.sleep(5)
77 # Turn off LCD backlights and clear text
78 lcd.color = [0, 0, 0]
79 lcd.clear()

```

Listing 3: examples/charlcd_i2c_mono_simpletest.py

```

1  """Simple test for 16x2 character lcd connected to an MCP23008 I2C LCD backpack."""
2  import time
3  import board
4  import busio
5  import adafruit_character_lcd.character_lcd_i2c as character_lcd
6
7  # Modify this if you have a different sized Character LCD
8  lcd_columns = 16
9  lcd_rows = 2
10
11 # Initialise I2C bus.
12 i2c = busio.I2C(board.SCL, board.SDA)
13
14 # Initialise the lcd class
15 lcd = character_lcd.Character_LCD_I2C(i2c, lcd_columns, lcd_rows)
16
17 # Turn backlight on
18 lcd.backlight = True
19 # Print a two line message
20 lcd.message = "Hello\nCircuitPython"
21 # Wait 5s
22 time.sleep(5)
23 lcd.clear()
24 # Print two line message right to left
25 lcd.text_direction = lcd.RIGHT_TO_LEFT
26 lcd.message = "Hello\nCircuitPython"
27 # Wait 5s
28 time.sleep(5)
29 # Return text direction to left to right
30 lcd.text_direction = lcd.LEFT_TO_RIGHT
31 # Display cursor
32 lcd.clear()
33 lcd.cursor = True
34 lcd.message = "Cursor! "
35 # Wait 5s
36 time.sleep(5)
37 # Display blinking cursor
38 lcd.clear()
39 lcd.blink = True
40 lcd.message = "Blinky Cursor!"
41 # Wait 5s
42 time.sleep(5)
43 lcd.blink = False
44 lcd.clear()
45 # Create message to scroll
46 scroll_msg = '<-- Scroll'

```

(continues on next page)

(continued from previous page)

```

47 lcd.message = scroll_msg
48 # Scroll message to the left
49 for i in range(len(scroll_msg)):
50     time.sleep(0.5)
51     lcd.move_left()
52 lcd.clear()
53 lcd.message = "Going to sleep\nCya later!"
54 time.sleep(5)
55 # Turn backlight off
56 lcd.backlight = False
57 time.sleep(2)

```

Listing 4: examples/charlcd_spi_mono_simpletest.py

```

1  """Simple test for 16x2 character LCD connected to 74HC595 SPI LCD backpack."""
2  import time
3  import board
4  import busio
5  import digitalio
6  import adafruit_character_lcd.character_lcd_spi as character_lcd
7
8  # Modify this if you have a different sized character LCD
9  lcd_columns = 16
10 lcd_rows = 2
11
12 # Backpack connection configuration:
13 clk = board.SCK # Pin connected to backpack CLK.
14 data = board.MOSI # Pin connected to backpack DAT/data.
15 latch = board.D5 # Pin connected to backpack LAT/latch.
16
17 # Initialise SPI bus.
18 spi = busio.SPI(clk, MOSI=data)
19
20 # Initialise the LCD class
21 latch = digitalio.DigitalInOut(latch)
22 lcd = character_lcd.Character_LCD_SPI(spi, latch, lcd_columns, lcd_rows)
23
24 # Turn backlight on
25 lcd.backlight = True
26 # Print a two line message
27 lcd.message = "Hello\nCircuitPython"
28 # Wait 5s
29 time.sleep(5)
30 lcd.clear()
31 # Print two line message right to left
32 lcd.text_direction = lcd.RIGHT_TO_LEFT
33 lcd.message = "Hello\nCircuitPython"
34 # Wait 5s
35 time.sleep(5)
36 # Return text direction to left to right
37 lcd.text_direction = lcd.LEFT_TO_RIGHT
38 # Display cursor
39 lcd.clear()
40 lcd.cursor = True
41 lcd.message = "Cursor! "
42 # Wait 5s

```

(continues on next page)

(continued from previous page)

```

43 time.sleep(5)
44 # Display blinking cursor
45 lcd.clear()
46 lcd.blink = True
47 lcd.message = "Blinky Cursor!"
48 # Wait 5s
49 time.sleep(5)
50 lcd.blink = False
51 lcd.clear()
52 # Create message to scroll
53 scroll_msg = '<-- Scroll'
54 lcd.message = scroll_msg
55 # Scroll message to the left
56 for i in range(len(scroll_msg)):
57     time.sleep(0.5)
58     lcd.move_left()
59 lcd.clear()
60 lcd.message = "Going to sleep\nCya later!"
61 # Turn backlight off
62 lcd.backlight = False
63 time.sleep(2)

```

Listing 5: examples/charlcd_keypad_simpletest.py

```

1  """Simple test for keypad on I2C RGB character LCD Shield or Pi Plate kits"""
2  import time
3  import board
4  import busio
5  import adafruit_character_lcd.character_lcd_rgb_i2c as character_lcd
6
7  # Modify this if you have a different sized Character LCD
8  lcd_columns = 16
9  lcd_rows = 2
10
11 # Initialise I2C bus.
12 i2c = busio.I2C(board.SCL, board.SDA)
13
14 # Initialise the LCD class
15 lcd = character_lcd.Character_LCD_RGB_I2C(i2c, lcd_columns, lcd_rows)
16
17 lcd.clear()
18 lcd.color = [100, 0, 0]
19 while True:
20     if lcd.left_button:
21         print("Left!")
22         lcd.message = "Left!"
23
24     elif lcd.up_button:
25         print("Up!")
26         lcd.message = "Up!"
27
28     elif lcd.down_button:
29         print("Down!")
30         lcd.message = "Down!"
31
32     elif lcd.right_button:

```

(continues on next page)

(continued from previous page)

```

33     print("Right!")
34     lcd.message = "Right!"
35
36     elif lcd.select_button:
37         print("Select!")
38         lcd.message = "Select!"
39
40     else:
41         time.sleep(0.1)
42         lcd.clear()

```

7.2 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)

7.2.1 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

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()
```

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*

Parameters

- **column** – column location
- **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 (1, 1).

message

Display a string of text on the character LCD.

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
```

(continues on next page)

(continued from previous page)

```
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
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)
```

7.3 adafruit_character_lcd.character_lcd_i2c

Module for using I2C with I2C/SPI character LCD backpack

- Author(s): Kattni Rembor

7.3.1 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

class `adafruit_character_lcd.character_lcd_i2c.Character_LCD_I2C` (*i2c, columns, lines, back-light_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)
```

7.4 adafruit_character_lcd.character_lcd_spi

Module for using SPI with I2C/SPI character LCD backpack

- Author(s): Kattni Rembor

7.4.1 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

```
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)
```

7.5 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

7.5.1 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

class adafruit_character_lcd.character_lcd_rgb_i2c.**Character_LCD_RGB_I2C** (*i2c*,
columns,
lines)

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

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)
```

(continues on next page)

(continued from previous page)

```
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
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 8

Indices and tables

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

a

`adafruit_character_lcd.character_lcd`,
 [21](#)
`adafruit_character_lcd.character_lcd_i2c`,
 [26](#)
`adafruit_character_lcd.character_lcd_rgb_i2c`,
 [27](#)
`adafruit_character_lcd.character_lcd_spi`,
 [27](#)

A

adafruit_character_lcd.character_lcd (module), 21
adafruit_character_lcd.character_lcd_i2c (module), 26
adafruit_character_lcd.character_lcd_rgb_i2c (module), 27
adafruit_character_lcd.character_lcd_spi (module), 27

B

backlight (adafruit_character_lcd.character_lcd.Character_LCD_Mono attribute), 25
blink (adafruit_character_lcd.character_lcd.Character_LCD attribute), 21

C

Character_LCD (class in adafruit_character_lcd.character_lcd), 21
Character_LCD_I2C (class in adafruit_character_lcd.character_lcd_i2c), 26
Character_LCD_Mono (class in adafruit_character_lcd.character_lcd), 24
Character_LCD_RGB (class in adafruit_character_lcd.character_lcd), 25
Character_LCD_RGB_I2C (class in adafruit_character_lcd.character_lcd_rgb_i2c), 28
Character_LCD_SPI (class in adafruit_character_lcd.character_lcd_spi), 27
clear() (adafruit_character_lcd.character_lcd.Character_LCD method), 22
color (adafruit_character_lcd.character_lcd.Character_LCD attribute), 26
create_char() (adafruit_character_lcd.character_lcd.Character_LCD method), 22
cursor (adafruit_character_lcd.character_lcd.Character_LCD attribute), 22
cursor_position() (adafruit_character_lcd.character_lcd.Character_LCD method), 22

D

display (adafruit_character_lcd.character_lcd.Character_LCD attribute), 23
down_button (adafruit_character_lcd.character_lcd_rgb_i2c.Character_LCD attribute), 28

H

home() (adafruit_character_lcd.character_lcd.Character_LCD method), 23

L

left_button (adafruit_character_lcd.character_lcd_rgb_i2c.Character_LCD attribute), 28

M

message (adafruit_character_lcd.character_lcd.Character_LCD attribute), 23
move_left() (adafruit_character_lcd.character_lcd.Character_LCD method), 23
move_right() (adafruit_character_lcd.character_lcd.Character_LCD method), 24

R

right_button (adafruit_character_lcd.character_lcd_rgb_i2c.Character_LCD attribute), 29

S

select_button (adafruit_character_lcd.character_lcd_rgb_i2c.Character_LCD attribute), 29

T

text_direction (adafruit_character_lcd.character_lcd.Character_LCD attribute), 24

U

up_button (adafruit_character_lcd.character_lcd_rgb_i2c.Character_LCD attribute), 29