Adafruit CIRCUITPYTHON$_{CHARLCD}$ Library Documenta

Release 1.0

Brent Rubell

Nov 17, 2020

Contents

1 Installing from PyPI **3**

2 Dependencies **5**

3 Usage Example **7**

4 Contributing **9**

5 Documentation **11**

6 Table of Contents **13**
- 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 . . . 24
  - 6.3.1 Implementation Notes . . . 24
- 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

7 Indices and tables **29**

Python Module Index **31**

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
- 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.

CHAPTER 6

Table of Contents

6.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/charlcd_mono_simpletest.py

```
"""Simple test for monochromatic character LCD"""
import time
import board
import digitalio
import adafruit_character_lcd.character_lcd as characterlcd

# Modify this if you have a different sized character LCD
lcd_columns = 16
lcd_rows = 2

# Metro M0/M4 Pin Config:
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)

# Initialise the LCD class
lcd = characterlcd.Character_LCD_Mono(
    lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7, lcd_columns, lcd_rows, lcd_
↪backlight
)

# Turn backlight on
lcd.backlight = True
```

(continues on next page)

(continued from previous page)

```
# Print a two line message
lcd.message = "Hello\nCircuitPython"
# Wait 5s
time.sleep(5)
lcd.clear()
# Print two line message right to left
lcd.text_direction = lcd.RIGHT_TO_LEFT
lcd.message = "Hello\nCircuitPython"
# Wait 5s
time.sleep(5)
# Return text direction to left to right
lcd.text_direction = lcd.LEFT_TO_RIGHT
# Display cursor
lcd.clear()
lcd.cursor = True
lcd.message = "Cursor! "
# Wait 5s
time.sleep(5)
# Display blinking cursor
lcd.clear()
lcd.blink = True
lcd.message = "Blinky Cursor!"
# Wait 5s
time.sleep(5)
lcd.blink = False
lcd.clear()
# Create message to scroll
scroll_msg = "<-- Scroll"
lcd.message = scroll_msg
# Scroll message to the left
for i in range(len(scroll_msg)):
    time.sleep(0.5)
    lcd.move_left()
lcd.clear()
lcd.message = "Going to sleep\nCya later!"
time.sleep(3)
# Turn backlight off
lcd.backlight = False
time.sleep(2)
```

Listing 2: examples/charlcd_rgb_simpletest.py

```
"""Simple test for RGB character LCD"""
import time
import board
import digitalio
import pulseio
import adafruit_character_lcd.character_lcd as characterlcd

# Modify this if you have a different sized character LCD
lcd_columns = 16
lcd_rows = 2

# Metro M0/M4 Pin Config:
lcd_rs = digitalio.DigitalInOut(board.D7)
lcd_en = digitalio.DigitalInOut(board.D8)
```

(continues on next page)

(continued from previous page)

```
lcd_d7 = digitalio.DigitalInOut(board.D12)
lcd_d6 = digitalio.DigitalInOut(board.D11)
lcd_d5 = digitalio.DigitalInOut(board.D10)
lcd_d4 = digitalio.DigitalInOut(board.D9)
red = pulseio.PWMOut(board.D3)
green = pulseio.PWMOut(board.D5)
blue = pulseio.PWMOut(board.D6)

# Initialise the LCD class
lcd = characterlcd.Character_LCD_RGB(
    lcd_rs,
    lcd_en,
    lcd_d4,
    lcd_d5,
    lcd_d6,
    lcd_d7,
    lcd_columns,
    lcd_rows,
    red,
    green,
    blue,
)

lcd.clear()
# Set LCD color to red
lcd.color = [100, 0, 0]
time.sleep(1)
# Print two line message
lcd.message = "Hello\nCircuitPython"
# Wait 5s
time.sleep(5)
# Set LCD color to blue
lcd.color = [0, 100, 0]
time.sleep(1)
# Set LCD color to green
lcd.color = [0, 0, 100]
time.sleep(1)
# Set LCD color to purple
lcd.color = [50, 0, 50]
time.sleep(1)
lcd.clear()
# Print two line message right to left
lcd.text_direction = lcd.RIGHT_TO_LEFT
lcd.message = "Hello\nCircuitPython"
# Wait 5s
time.sleep(5)
# Return text direction to left to right
lcd.text_direction = lcd.LEFT_TO_RIGHT
# Display cursor
lcd.clear()
lcd.cursor = True
lcd.message = "Cursor! "
# Wait 5s
time.sleep(5)
# Display blinking cursor
lcd.clear()
lcd.blink = True
```

(continues on next page)

(continued from previous page)

```
lcd.message = "Blinky Cursor!"
# Wait 5s
time.sleep(5)
lcd.blink = False
lcd.clear()
# Create message to scroll
scroll_msg = "<-- Scroll"
lcd.message = scroll_msg
# Scroll to the left
for i in range(len(scroll_msg)):
    time.sleep(0.5)
    lcd.move_left()
lcd.clear()
time.sleep(1)
lcd.message = "Going to sleep\nCya later!"
time.sleep(5)
# Turn off LCD backlights and clear text
lcd.color = [0, 0, 0]
lcd.clear()
```

Listing 3: examples/charlcd_i2c_mono_simpletest.py

```
"""Simple test for 16x2 character lcd connected to an MCP23008 I2C LCD backpack."""
import time
import board
import busio
import adafruit_character_lcd.character_lcd_i2c as character_lcd

# Modify this if you have a different sized Character LCD
lcd_columns = 16
lcd_rows = 2

# Initialise I2C bus.
i2c = busio.I2C(board.SCL, board.SDA)

# Initialise the lcd class
lcd = character_lcd.Character_LCD_I2C(i2c, lcd_columns, lcd_rows)

# Turn backlight on
lcd.backlight = True
# Print a two line message
lcd.message = "Hello\nCircuitPython"
# Wait 5s
time.sleep(5)
lcd.clear()
# Print two line message right to left
lcd.text_direction = lcd.RIGHT_TO_LEFT
lcd.message = "Hello\nCircuitPython"
# Wait 5s
time.sleep(5)
# Return text direction to left to right
lcd.text_direction = lcd.LEFT_TO_RIGHT
# Display cursor
lcd.clear()
lcd.cursor = True
lcd.message = "Cursor! "
```

(continues on next page)

(continued from previous page)

```
# Wait 5s
time.sleep(5)
# Display blinking cursor
lcd.clear()
lcd.blink = True
lcd.message = "Blinky Cursor!"
# Wait 5s
time.sleep(5)
lcd.blink = False
lcd.clear()
# Create message to scroll
scroll_msg = "<-- Scroll"
lcd.message = scroll_msg
# Scroll message to the left
for i in range(len(scroll_msg)):
    time.sleep(0.5)
    lcd.move_left()
lcd.clear()
lcd.message = "Going to sleep\nCya later!"
time.sleep(5)
# Turn backlight off
lcd.backlight = False
time.sleep(2)
```

Listing 4: examples/charlcd_spi_mono_simpletest.py

```
"""Simple test for 16x2 character LCD connected to 74HC595 SPI LCD backpack."""
import time
import board
import busio
import digitalio
import adafruit_character_lcd.character_lcd_spi as character_lcd

# Modify this if you have a different sized character LCD
lcd_columns = 16
lcd_rows = 2

# Backpack connection configuration:
clk = board.SCK  # Pin connected to backpack CLK.
data = board.MOSI  # Pin connected to backpack DAT/data.
latch = board.D5  # Pin connected to backpack LAT/latch.

# Initialise SPI bus.
spi = busio.SPI(clk, MOSI=data)

# Initialise the LCD class
latch = digitalio.DigitalInOut(latch)
lcd = character_lcd.Character_LCD_SPI(spi, latch, lcd_columns, lcd_rows)

# Turn backlight on
lcd.backlight = True
# Print a two line message
lcd.message = "Hello\nCircuitPython"
# Wait 5s
time.sleep(5)
lcd.clear()
```

(continues on next page)

(continued from previous page)

```
# Print two line message right to left
lcd.text_direction = lcd.RIGHT_TO_LEFT
lcd.message = "Hello\nCircuitPython"
# Wait 5s
time.sleep(5)
# Return text direction to left to right
lcd.text_direction = lcd.LEFT_TO_RIGHT
# Display cursor
lcd.clear()
lcd.cursor = True
lcd.message = "Cursor! "
# Wait 5s
time.sleep(5)
# Display blinking cursor
lcd.clear()
lcd.blink = True
lcd.message = "Blinky Cursor!"
# Wait 5s
time.sleep(5)
lcd.blink = False
lcd.clear()
# Create message to scroll
scroll_msg = "<-- Scroll"
lcd.message = scroll_msg
# Scroll message to the left
for i in range(len(scroll_msg)):
    time.sleep(0.5)
    lcd.move_left()
lcd.clear()
lcd.message = "Going to sleep\nCya later!"
# Turn backlight off
lcd.backlight = False
time.sleep(2)
```

Listing 5: examples/charlcd_keypad_simpletest.py

```
"""Simple test for keypad on I2C RGB character LCD Shield or Pi Plate kits"""
import time
import board
import busio
import adafruit_character_lcd.character_lcd_rgb_i2c as character_lcd

# Modify this if you have a different sized Character LCD
lcd_columns = 16
lcd_rows = 2

# Initialise I2C bus.
i2c = busio.I2C(board.SCL, board.SDA)

# Initialise the LCD class
lcd = character_lcd.Character_LCD_RGB_I2C(i2c, lcd_columns, lcd_rows)

lcd.clear()
lcd.color = [100, 0, 0]
while True:
    if lcd.left_button:
```

(continues on next page)

(continued from previous page)

```
        print("Left!")
        lcd.message = "Left!"

    elif lcd.up_button:
        print("Up!")
        lcd.message = "Up!"

    elif lcd.down_button:
        print("Down!")
        lcd.message = "Down!"

    elif lcd.right_button:
        print("Right!")
        lcd.message = "Right!"

    elif lcd.select_button:
        print("Select!")
        lcd.message = "Select!"

    else:
        time.sleep(0.1)
        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

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 (1, 1).

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 size 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*, *backlight_pin=None*, *backlight_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)
```

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

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

class adafruit_character_lcd.character_lcd_spi.**Character_LCD_SPI**(*spi*, *latch*, *columns*, *lines*, *backlight_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)
```

(continues on next page)

(continued from previous page)

```
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

class adafruit_character_lcd.character_lcd_rgb_i2c.**Character_LCD_RGB_I2C**(*i2c*, *columns*, *lines*, *address=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

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)
```

(continues on next page)

(continued from previous page)

```
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
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

Python Module Index

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

Index

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*), 19
Character_LCD_I2C (*class in adafruit_character_lcd.character_lcd_i2c*), 25
Character_LCD_Mono (*class in adafruit_character_lcd.character_lcd*), 22
Character_LCD_RGB (*class in adafruit_character_lcd.character_lcd*), 23
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*), 20
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 attribute*), 26

H

home() (*adafruit_character_lcd.character_lcd.Character_LCD method*), 21

L

left_button (*adafruit_character_lcd.character_lcd_rgb_i2c.Character_LCD_RGB attribute*), 27

M

message (*adafruit_character_lcd.character_lcd.Character_LCD attribute*), 21
move_left() (*adafruit_character_lcd.character_lcd.Character_LCD method*), 21
move_right() (*adafruit_character_lcd.character_lcd.Character_LCD method*), 22

R

right_button (*adafruit_character_lcd.character_lcd_rgb_i2c.Character_LCD_RGB attribute*), 27

S

select_button (*adafruit_character_lcd.character_lcd_rgb_i2c.Character_LCD_RGB 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*), 27