
Adafruit
CircuitPython *CharLCDLibrary* Documentation
Release 1.0

Brent Rubell

Jun 07, 2021

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 tests	13
6.1.1	Mono	13
6.1.2	RGB	14
6.1.3	I2C Mono	16
6.1.4	I2C RGB	17
6.1.5	SPI Mono	19
6.1.6	Keypad	20
6.1.7	Raspberry Pi Mono	21
6.1.8	Raspberry Pi RGB	22
6.2	Other tests	24
6.2.1	Custom Character	24
6.2.2	Nyan Cat	25
6.3	adafruit_character_lcd.character_lcd	27
6.3.1	Implementation Notes	27
6.4	adafruit_character_lcd.character_lcd_i2c	32
6.4.1	Implementation Notes	32
6.5	adafruit_character_lcd.character_lcd_rgb_i2c	33
6.5.1	Implementation Notes	33
6.6	adafruit_character_lcd.character_lcd_spi	35
6.6.1	Implementation Notes	35
7	Indices and tables	37
	Python Module Index	39
	Index	41

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 tests

Ensure your device works with these simple tests.

6.1.1 Mono

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
```

(continues on next page)

(continued from previous page)

```

24 lcd = characterlcd.Character_LCD_Mono(
25     lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7, lcd_columns, lcd_rows, lcd_
    ↳backlight
26 )
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)

```

6.1.2 RGB

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"""

```

(continues on next page)

(continued from previous page)

```
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
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"
```

(continues on next page)

(continued from previous page)

```

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
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()

```

6.1.3 I2C Mono

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 adafruit_character_lcd.character_lcd_i2c as character_lcd
8
9 # Modify this if you have a different sized Character LCD
10 lcd_columns = 16
11 lcd_rows = 2
12
13 # Initialise I2C bus.
14 i2c = board.I2C() # uses board.SCL and board.SDA
15
16 # Initialise the lcd class
17 lcd = character_lcd.Character_LCD_I2C(i2c, lcd_columns, lcd_rows)
18

```

(continues on next page)

(continued from previous page)

```

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

```

6.1.4 I2C RGB

Listing 4: examples/charlcd_i2c_rgb_simpletest.py

```

1 # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2 # SPDX-License-Identifier: MIT
3
4 """Simple test for I2C RGB character LCD shield kit"""
5 import time
6 import board
7 import adafruit_character_lcd.character_lcd_rgb_i2c as character_lcd
8
9 # Modify this if you have a different sized Character LCD

```

(continues on next page)

(continued from previous page)

```
10 lcd_columns = 16
11 lcd_rows = 2
12
13 # Initialise I2C bus.
14 i2c = board.I2C() # uses board.SCL and board.SDA
15
16 # Initialise the LCD class
17 lcd = character_lcd.Character_LCD_RGB_I2C(i2c, lcd_columns, lcd_rows)
18
19 lcd.clear()
20 # Set LCD color to red
21 lcd.color = [100, 0, 0]
22 time.sleep(1)
23 # Print two line message
24 lcd.message = "Hello\nCircuitPython"
25 # Wait 5s
26 time.sleep(5)
27 # Set LCD color to blue
28 lcd.color = [0, 100, 0]
29 time.sleep(1)
30 # Set LCD color to green
31 lcd.color = [0, 0, 100]
32 time.sleep(1)
33 # Set LCD color to purple
34 lcd.color = [50, 0, 50]
35 time.sleep(1)
36 lcd.clear()
37 # Print two line message right to left
38 lcd.text_direction = lcd.RIGHT_TO_LEFT
39 lcd.message = "Hello\nCircuitPython"
40 # Wait 5s
41 time.sleep(5)
42 # Return text direction to left to right
43 lcd.text_direction = lcd.LEFT_TO_RIGHT
44 # Display cursor
45 lcd.clear()
46 lcd.cursor = True
47 lcd.message = "Cursor! "
48 # Wait 5s
49 time.sleep(5)
50 # Display blinking cursor
51 lcd.clear()
52 lcd.blink = True
53 lcd.message = "Blinky Cursor!"
54 # Wait 5s
55 time.sleep(5)
56 lcd.blink = False
57 lcd.clear()
58 # Create message to scroll
59 scroll_msg = "<-- Scroll"
60 lcd.message = scroll_msg
61 # Scroll to the left
62 for i in range(len(scroll_msg)):
63     time.sleep(0.5)
64     lcd.move_left()
65 lcd.clear()
66 time.sleep(1)
```

(continues on next page)

(continued from previous page)

```

67 lcd.message = "Going to sleep\nCya later!"
68 time.sleep(5)
69 # Turn off LCD backlights and clear text
70 lcd.color = [0, 0, 0]
71 lcd.clear()

```

6.1.5 SPI Mono

Listing 5: 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)
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

```

(continues on next page)

(continued from previous page)

```

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)

```

6.1.6 Keypad

Listing 6: 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 adafruit_character_lcd.character_lcd_rgb_i2c as character_lcd
8
9 # Modify this if you have a different sized Character LCD
10 lcd_columns = 16
11 lcd_rows = 2
12
13 # Initialise I2C bus.
14 i2c = board.I2C() # uses board.SCL and board.SDA
15
16 # Initialise the LCD class
17 lcd = character_lcd.Character_LCD_RGB_I2C(i2c, lcd_columns, lcd_rows)
18
19 lcd.clear()
20 lcd.color = [100, 0, 0]
21 while True:
22     if lcd.left_button:
23         print("Left!")
24         lcd.message = "Left!"
25
26     elif lcd.up_button:
27         print("Up!")
28         lcd.message = "Up!"
29

```

(continues on next page)

(continued from previous page)

```

30     elif lcd.down_button:
31         print("Down!")
32         lcd.message = "Down!"
33
34     elif lcd.right_button:
35         print("Right!")
36         lcd.message = "Right!"
37
38     elif lcd.select_button:
39         print("Select!")
40         lcd.message = "Select!"
41
42     else:
43         time.sleep(0.1)
44         lcd.clear()

```

6.1.7 Raspberry Pi Mono

Listing 7: examples/charlcd_rpi_mono_simpletest.py

```

1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  """Simple test for monochromatic character LCD on Raspberry Pi"""
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 # Raspberry Pi Pin Config:
15 lcd_rs = digitalio.DigitalInOut(board.D26)
16 lcd_en = digitalio.DigitalInOut(board.D19)
17 lcd_d7 = digitalio.DigitalInOut(board.D27)
18 lcd_d6 = digitalio.DigitalInOut(board.D22)
19 lcd_d5 = digitalio.DigitalInOut(board.D24)
20 lcd_d4 = digitalio.DigitalInOut(board.D25)
21 lcd_backlight = digitalio.DigitalInOut(board.D4)
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_
↳backlight
26 )
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()

```

(continues on next page)

(continued from previous page)

```

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 # Turn backlight off
66 lcd.backlight = False
67 time.sleep(2)

```

6.1.8 Raspberry Pi RGB

Listing 8: examples/charlcd_rpi_rgb_simpletest.py

```

1 # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2 # SPDX-License-Identifier: MIT
3
4 """Simple test for RGB character LCD on Raspberry Pi"""
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
12 lcd_columns = 16
13 lcd_rows = 2
14
15 # Raspberry Pi Pin Config:
16 lcd_rs = digitalio.DigitalInOut(board.D26) # LCD pin 4
17 lcd_en = digitalio.DigitalInOut(board.D19) # LCD pin 6

```

(continues on next page)

(continued from previous page)

```
18 lcd_d7 = digitalio.DigitalInOut(board.D27) # LCD pin 14
19 lcd_d6 = digitalio.DigitalInOut(board.D22) # LCD pin 13
20 lcd_d5 = digitalio.DigitalInOut(board.D24) # LCD pin 12
21 lcd_d4 = digitalio.DigitalInOut(board.D25) # LCD pin 11
22 lcd_rw = digitalio.DigitalInOut(
23     board.D4
24 ) # LCD pin 5. Determines whether to read to or write from the display.
25 # Not necessary if only writing to the display. Used on shield.
26
27 red = pwmio.PWMOut(board.D21)
28 green = pwmio.PWMOut(board.D12)
29 blue = pwmio.PWMOut(board.D18)
30
31 # Initialize the LCD class
32 # The lcd_rw parameter is optional. You can omit the line below if you're only
33 # writing to the display.
34 lcd = characterlcd.Character_LCD_RGB(
35     lcd_rs,
36     lcd_en,
37     lcd_d4,
38     lcd_d5,
39     lcd_d6,
40     lcd_d7,
41     lcd_columns,
42     lcd_rows,
43     red,
44     green,
45     blue,
46     lcd_rw,
47 )
48
49 RED = [100, 0, 0]
50 GREEN = [0, 100, 0]
51 BLUE = [0, 0, 100]
52
53 while True:
54     lcd.clear()
55     # Set LCD color to red
56     lcd.color = [100, 0, 0]
57     time.sleep(1)
58
59     # Print two line message
60     lcd.message = "Hello\nCircuitPython"
61
62     # Wait 5s
63     time.sleep(5)
64
65     # Set LCD color to blue
66     lcd.color = [0, 100, 0]
67     time.sleep(1)
68     # Set LCD color to green
69     lcd.color = [0, 0, 100]
70     time.sleep(1)
71     # Set LCD color to purple
72     lcd.color = [50, 0, 50]
73     time.sleep(1)
74     lcd.clear()
```

(continues on next page)

(continued from previous page)

```

75
76     # Print two line message right to left
77     lcd.text_direction = lcd.RIGHT_TO_LEFT
78     lcd.message = "Hello\nCircuitPython"
79     # Wait 5s
80     time.sleep(5)
81
82     # Return text direction to left to right
83     lcd.text_direction = lcd.LEFT_TO_RIGHT
84
85     # Display cursor
86     lcd.clear()
87     lcd.cursor = True
88     lcd.message = "Cursor! "
89     # Wait 5s
90     time.sleep(5)
91
92     # Display blinking cursor
93     lcd.clear()
94     lcd.blink = True
95     lcd.message = "Blinky Cursor!"
96     # Wait 5s
97     time.sleep(5)
98     lcd.blink = False
99     lcd.clear()
100
101     # Create message to scroll
102     scroll_msg = "<-- Scroll"
103     lcd.message = scroll_msg
104     # Scroll to the left
105     for i in range(len(scroll_msg)):
106         time.sleep(0.5)
107         lcd.move_left()
108     lcd.clear()
109
110     # Turn off LCD backlights and clear text
111     lcd.color = [0, 0, 0]
112     lcd.clear()
113     time.sleep(1)

```

6.2 Other tests

6.2.1 Custom Character

Listing 9: examples/charlcd_customcharacter.py

```

1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  """Display a custom character"""
5  import board
6  import digitalio
7  import adafruit_character_lcd.character_lcd as characterlcd

```

(continues on next page)

(continued from previous page)

```

8
9 # Modify this if you have a different sized character LCD
10 lcd_columns = 16
11 lcd_rows = 2
12
13 # Metro M0/M4 Pin Config:
14 lcd_rs = digitalio.DigitalInOut(board.D7)
15 lcd_en = digitalio.DigitalInOut(board.D8)
16 lcd_d7 = digitalio.DigitalInOut(board.D12)
17 lcd_d6 = digitalio.DigitalInOut(board.D11)
18 lcd_d5 = digitalio.DigitalInOut(board.D10)
19 lcd_d4 = digitalio.DigitalInOut(board.D9)
20 lcd_backlight = digitalio.DigitalInOut(board.D13)
21
22 # Initialise the LCD class
23 lcd = characterlcd.Character_LCD_Mono(
24     lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7, lcd_columns, lcd_rows, lcd_
25     ↪backlight
26 )
27
28 checkmark = bytes([0x0, 0x0, 0x1, 0x3, 0x16, 0x1C, 0x8, 0x0])
29
30 # Store in LCD character memory 0
31 lcd.create_char(0, checkmark)
32
33 lcd.clear()
34 lcd.message = "\x00 Success \x00"

```

6.2.2 Nyan Cat

Listing 10: examples/charlcd_custom_character_nyan_cat.py

```

1 # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2 # SPDX-License-Identifier: MIT
3
4 """Use custom characters to display Nyan cat"""
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

```

(continues on next page)

(continued from previous page)

```
24 lcd = characterlcd.Character_LCD_Mono(  
25     lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7, lcd_columns, lcd_rows, lcd_  
↪backlight  
26 )  
27  
28 head = [31, 17, 27, 17, 17, 21, 17, 31]  
29  
30 top_body = [31, 0, 31, 0, 18, 8, 2, 8]  
31 top_left_corner_body = [31, 16, 16, 17, 22, 20, 20, 20]  
32 top_right_corner_body = [31, 1, 1, 17, 13, 5, 5, 5]  
33  
34 # these three chars will be the above three reversed with a few minor changes to  
35 # fit feet into the bottom  
36 bot_body = []  
37 bot_left_corner_body = []  
38 bot_right_corner_body = []  
39  
40 tail_neutral = [0, 0, 0, 0, 31, 31, 0, 0]  
41 tail_up = [0, 8, 12, 6, 3, 1, 0, 0]  
42  
43 for i in range(7, -1, -1):  
44     bot_body.append(top_body[i])  
45     bot_left_corner_body.append(top_left_corner_body[i])  
46     bot_right_corner_body.append(top_right_corner_body[i])  
47  
48 # adding feet and making space for them  
49  
50 bot_body[6] = 31  
51 bot_body[5] = 0  
52 bot_body[4] = 31  
53 bot_body[7] = 24  
54 bot_left_corner_body[7] = 0  
55 bot_left_corner_body[6] = 31  
56 bot_left_corner_body[7] = 28  
57 bot_right_corner_body[7] = 0  
58 bot_right_corner_body[6] = 31  
59  
60 # bottom body with feet forward  
61 bot_body2 = bot_body[:-1] + [3]  
62  
63  
64 rainbow = [0, 0, 6, 25, 11, 29, 27, 12]  
65 rainbow2 = [0, 0, 6, 31, 13, 5, 23, 12]  
66  
67 lcd.create_char(0, top_body)  
68 lcd.create_char(1, top_left_corner_body)  
69 lcd.create_char(2, rainbow)  
70 lcd.create_char(3, bot_left_corner_body)  
71 lcd.create_char(4, bot_body)  
72 lcd.create_char(5, bot_right_corner_body)  
73 lcd.create_char(6, head)  
74 lcd.create_char(7, tail_neutral)  
75  
76 lcd.clear()  
77  
78 lcd.move_right()  
79 lcd.message = (
```

(continues on next page)

(continued from previous page)

```

80     "\x02\x02\x02\x02\x01\x00\x00\x00\x06\n\x02\x02\x02\x07\x03\x04\x04\x04\x05"
81 )
82
83 lcd.backlight = True
84
85 while True:
86     lcd.create_char(4, bot_body2)
87     lcd.create_char(7, tail_up)
88     lcd.create_char(2, rainbow2)
89     lcd.move_right()
90     time.sleep(0.4)
91     lcd.create_char(4, bot_body)
92     lcd.create_char(7, tail_neutral)
93     lcd.create_char(2, rainbow)
94     lcd.move_left()
95     time.sleep(0.4)

```

6.3 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)

6.3.1 Implementation Notes

Hardware:

- [Adafruit Character LCDs](#)

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://circuitpython.org/downloads>
- Adafruit's Bus Device library: 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 adafruit_character_lcd.character_lcd_i2c as character_lcd

i2c = board.I2C() # uses board.SCL and 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 adafruit_character_lcd.character_lcd_i2c as character_lcd

i2c = board.I2C() # uses board.SCL and 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 adafruit_character_lcd.character_lcd_i2c as character_lcd

i2c = board.I2C() # uses board.SCL and board.SDA
lcd = character_lcd.Character_LCD_I2C(i2c, 16, 2)
```

(continues on next page)

(continued from previous page)

```
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 adafruit_character_lcd.character_lcd_i2c as character_lcd

i2c = board.I2C() # uses board.SCL and 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 adafruit_character_lcd.character_lcd_i2c as character_lcd

i2c = board.I2C() # uses board.SCL and 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 adafruit_character_lcd.character_lcd_i2c as character_lcd
```

(continues on next page)

(continued from previous page)

```
i2c = board.I2C() # uses board.SCL and 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 adafruit_character_lcd.character_lcd_i2c as character_lcd

i2c = board.I2C() # uses board.SCL and 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 adafruit_character_lcd.character_lcd_i2c as character_lcd

i2c = board.I2C() # uses board.SCL and 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 adafruit_character_lcd.character_lcd_i2c as character_lcd

i2c = board.I2C() # uses board.SCL and 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
- **red** (*PWMOut, DigitalInOut*) – Red RGB Anode
- **green** (*PWMOut, DigitalInOut*) – Green RGB Anode
- **blue** (*PWMOut, DigitalInOut*) – 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 adafruit_character_lcd.character_lcd_rgb_i2c as character_lcd

i2c = board.I2C() # uses board.SCL and board.SDA

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

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

6.4 adafruit_character_lcd.character_lcd_i2c

Module for using I2C with I2C/SPI character LCD backpack

- Author(s): Kattni Rembor

6.4.1 Implementation Notes

Hardware:

- I2C / SPI character LCD backpack (Product ID: 292)

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://circuitpython.org/downloads>
- Adafruit’s Bus Device library: https://github.com/adafruit/Adafruit_CircuitPython_BusDevice

class adafruit_character_lcd.character_lcd_i2c.**Character_LCD_I2C** (*i2c, columns, lines, address=None, back-light_inverted=False*)

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

To use, import and initialise as follows:

```
import board
from adafruit_character_lcd.character_lcd_i2c import Character_LCD_I2C

i2c = board.I2C() # uses board.SCL and board.SDA
lcd = Character_LCD_I2C(i2c, 16, 2)
```

6.5 adafruit_character_lcd.character_lcd_rgb_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 (Product ID: 714)
- RGB LCD Shield Kit w/ 16x2 Character Display - Positive Display (Product ID: 716)
- Adafruit RGB Negative 16x2 LCD+Keypad Kit for Raspberry Pi (Product ID: 1110)
- Adafruit RGB Positive 16x2 LCD+Keypad Kit for Raspberry Pi (Product ID: 1109)

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://circuitpython.org/downloads>
- Adafruit's Bus Device library: 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
from adafruit_character_lcd.character_lcd_rgb_i2c import Character_LCD_RGB_I2C

i2c = board.I2C() # uses board.SCL and 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
from adafruit_character_lcd.character_lcd_rgb_i2c import Character_LCD_RGB_I2C

i2c = board.I2C() # uses board.SCL and 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
from adafruit_character_lcd.character_lcd_rgb_i2c import Character_LCD_RGB_I2C

i2c = board.I2C() # uses board.SCL and 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
from adafruit_character_lcd.character_lcd_rgb_i2c import Character_LCD_RGB_I2C

i2c = board.I2C() # uses board.SCL and 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
from adafruit_character_lcd.character_lcd_rgb_i2c import Character_LCD_RGB_I2C

i2c = board.I2C() # uses board.SCL and 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
from adafruit_character_lcd.character_lcd_rgb_i2c import Character_LCD_RGB_I2C
```

(continues on next page)

(continued from previous page)

```
i2c = board.I2C() # uses board.SCL and board.SDA
lcd = Character_LCD_RGB_I2C(i2c, 16, 2)

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

6.6 adafruit_character_lcd.character_lcd_spi

Module for using SPI with I2C/SPI character LCD backpack

- Author(s): Kattni Rembor

6.6.1 Implementation Notes

Hardware:

- I2C / SPI character LCD backpack (Product ID: 4566)

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://circuitpython.org/downloads>
- Adafruit's Bus Device library: 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_Mono* and implements all of the same functions and functionality.

To use, import and initialise as follows:

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

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


CHAPTER 7

Indices and tables

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

a

adafruit_character_lcd.character_lcd,
27
adafruit_character_lcd.character_lcd_i2c,
32
adafruit_character_lcd.character_lcd_rgb_i2c,
33
adafruit_character_lcd.character_lcd_spi,
35

A

adafruit_character_lcd.character_lcd
 (module), 27

adafruit_character_lcd.character_lcd_i2c
 (module), 32

adafruit_character_lcd.character_lcd_rgb_i2c
 (module), 33

adafruit_character_lcd.character_lcd_spi
 (module), 35

B

backlight (adafruit_character_lcd.character_lcd.Character_LCD_Mono
 attribute), 31

blink (adafruit_character_lcd.character_lcd.Character_LCD
 attribute), 27

C

Character_LCD (class in
 adafruit_character_lcd.character_lcd), 27

Character_LCD_I2C (class in
 adafruit_character_lcd.character_lcd_i2c),
 32

Character_LCD_Mono (class in
 adafruit_character_lcd.character_lcd), 30

Character_LCD_RGB (class in
 adafruit_character_lcd.character_lcd), 31

Character_LCD_RGB_I2C (class in
 adafruit_character_lcd.character_lcd_rgb_i2c),
 33

Character_LCD_SPI (class in
 adafruit_character_lcd.character_lcd_spi),
 35

clear () (adafruit_character_lcd.character_lcd.Character_LCD
 method), 28

color (adafruit_character_lcd.character_lcd.Character_LCD_RGB
 attribute), 32

column_align (adafruit_character_lcd.character_lcd.Character_LCD
 attribute), 28

create_char () (adafruit_character_lcd.character_lcd.Character_LCD
 method), 28

cursor (adafruit_character_lcd.character_lcd.Character_LCD
 attribute), 28

cursor_position ()
 (adafruit_character_lcd.character_lcd.Character_LCD
 method), 29

D

display (adafruit_character_lcd.character_lcd.Character_LCD
 attribute), 29

down_button (adafruit_character_lcd.character_lcd_rgb_i2c.Character_LCD_RGB_I2C
 attribute), 33

H

home () (adafruit_character_lcd.character_lcd.Character_LCD
 method), 29

L

left_button (adafruit_character_lcd.character_lcd_rgb_i2c.Character_LCD_RGB_I2C
 attribute), 34

M

message (adafruit_character_lcd.character_lcd.Character_LCD
 attribute), 29

move_left () (adafruit_character_lcd.character_lcd.Character_LCD
 method), 29

move_right () (adafruit_character_lcd.character_lcd.Character_LCD
 method), 30

R

right_button (adafruit_character_lcd.character_lcd_rgb_i2c.Character_LCD_RGB_I2C
 attribute), 34

S

select_button (adafruit_character_lcd.character_lcd_rgb_i2c.Character_LCD_RGB_I2C
 attribute), 34

T

`text_direction` (*adafruit_character_lcd.character_lcd.Character_LCD*
attribute), 30

U

`up_button` (*adafruit_character_lcd.character_lcd_rgb_i2c.Character_LCD_RGB_I2C*
attribute), 34