
SlightTLC5957 Library Documentation

Release 1.0

Stefan Krüger

Apr 19, 2019

Contents

1	Dependencies	3
2	Usage Example	5
3	Contributing	7
4	Building locally	9
4.1	Zip release files	9
4.2	Sphinx documentation	9
5	Table of Contents	11
5.1	Simple test	11
5.2	Simple HW test	12
5.3	TLC5957 with FancyLED	15
5.4	TLC5957 with custom 2D-Array mapping	17
5.5	API	20
5.5.1	s-light CircuitPython TLC5957 library.	20
5.5.1.1	Implementation Notes	20
6	Indices and tables	25
	Python Module Index	27

CircuitPython library for [TI TLC5957](#) 48-channel 16bit LED-Driver

Setting of LED-Values / API is similar to NeoPixel and Dotstar APIs and compatible with [fancyled](#).

CHAPTER 1

Dependencies

This driver depends on:

- [Adafruit CircuitPython](#)

Please ensure all dependencies are available on the CircuitPython filesystem. This is easily achieved by downloading the [Adafruit library and driver bundle](#).

CHAPTER 2

Usage Example

have a look at the [examples](#) subfolder

CHAPTER 3

Contributing

Contributions are welcome! Please read our [Code of Conduct](#) before contributing to help this project stay welcoming.

4.1 Zip release files

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 slight-circuitpython-tlc5957 --library_
↳ location .
```

4.2 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.

5.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/slight_tlc5957_simpletest.py

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  # CircuitPython
4
5  """Simple & Minimallistic example for the TLC5957 library."""
6
7  # import time
8
9  import board
10 # import busio
11 import bitbangio
12 import digitalio
13 import pulseio
14
15 import slight_tlc5957
16
17
18 spi_clock = digitalio.DigitalInOut(board.SCK)
19 spi_clock.direction = digitalio.Direction.OUTPUT
20 spi_mosi = digitalio.DigitalInOut(board.MOSI)
21 spi_mosi.direction = digitalio.Direction.OUTPUT
22 spi_miso = digitalio.DigitalInOut(board.MISO)
23 spi_miso.direction = digitalio.Direction.INPUT
24
25 # spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
26 spi = bitbangio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
27
```

(continues on next page)

(continued from previous page)

```

28 # 6MHz for the grayscale clock
29 gsclock = pulseio.PWMOut(
30     board.D9, duty_cycle=(2 ** 15), frequency=(6000 * 1000))
31
32 latch = digitalio.DigitalInOut(board.D7)
33 latch.direction = digitalio.Direction.OUTPUT
34
35 # define pixel array
36 num_leds = 16
37 pixels = slight_tlc5957.TLC5957(
38     spi=spi,
39     latch=latch,
40     gsclock=gsclock,
41     spi_clock=spi_clock,
42     spi_mosi=spi_mosi,
43     spi_miso=spi_miso,
44     pixel_count=num_leds)
45
46
47 # set first pixel to orange
48 # using floating point values (0..1)
49 # pixels[0] = (1, 0.5, 0)
50 # set first pixel to sky blue
51 # using 16bit integer values (0..65535)
52 # pixels[1] = (0, 32000, 65535)
53
54 # write data to chips
55 pixels.show()
56
57 fade_value = 0
58 step = 500
59
60 print("loop..")
61
62 pixel_index = 3
63 buffer_index = (
64     pixel_index * pixels.COLORS_PER_PIXEL * pixels.BUFFER_BYTES_PER_COLOR)
65
66 while True:
67     # pixels[3] = (0, 100, fade_value)
68     pixels.set_pixel_all_16bit_value(1, 1, 1)
69     pixels.show()
70     if (fade_value + step) > 65535 or (fade_value + step) < 0:
71         step *= -1
72     fade_value += step
73     # time.sleep(0.3)

```

5.2 Simple HW test

This example lights up every LED after each other. Its great for production testing your boards.

Listing 2: examples/slight_tlc5957_pixel_checker.py

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  # CircuitPython
4
5  """Develop and Test TLC5957."""
6
7  __doc__ = """
8  Develop and Test TLC5957.
9
10 this script contains a bunch of tests and debug outputs.
11 its mainly the playground during the development of the library.
12 """
13
14 import time
15
16 import board
17 # import busio
18 import bitbangio
19 import digitalio
20 import pulseio
21
22 import slight_tlc5957
23
24 #####
25 print(
26     "\n" +
27     (42 * '*') + "\n" +
28     __doc__ + "\n" +
29     (42 * '*') + "\n" +
30     "\n"
31 )
32
33 #####
34 print(42 * '*')
35 print("initialise digitalio pins for SPI")
36 spi_clock = digitalio.DigitalInOut(board.SCK)
37 spi_clock.direction = digitalio.Direction.OUTPUT
38 spi_mosi = digitalio.DigitalInOut(board.MOSI)
39 spi_mosi.direction = digitalio.Direction.OUTPUT
40 spi_miso = digitalio.DigitalInOut(board.MISO)
41 spi_miso.direction = digitalio.Direction.INPUT
42
43 # print((42 * '*') + "\n" + "init busio.SPI")
44 # spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
45 print("init bitbangio.SPI")
46 spi = bitbangio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
47
48 # maximum frequency is currently hardcoded to 6MHz
49 # https://github.com/adafruit/circuitpython/blob/master/ports/atmel-samd/common-hal/
50 #   ↳ pulseio/PWMOut.c#L119
51 gsclock_frequency = (6000 * 1000) # 6MHz
52 gsclock = pulseio.PWMOut(
53     board.D9, duty_cycle=(2 ** 15), frequency=gsclock_frequency)
54 print("gsclock.frequency: {:.1f}MHz".format(gsclock.frequency / (1000*1000)))

```

(continues on next page)

(continued from previous page)

```

55 latch = digitalio.DigitalInOut(board.D7)
56 latch.direction = digitalio.Direction.OUTPUT
57
58 #####
59 print(42 * '*')
60 print("define pixel array / init TLC5957")
61 num_leds = 16
62 pixels = slight_tlc5957.TLC5957(
63     spi=spi,
64     latch=latch,
65     gsclock=gsclock,
66     spi_clock=spi_clock,
67     spi_mosi=spi_mosi,
68     spi_miso=spi_miso,
69     pixel_count=num_leds)
70
71 print("pixel_count", pixels.pixel_count)
72 print("chip_count", pixels.chip_count)
73 print("channel_count", pixels.channel_count)
74
75 #####
76
77 print(42 * '*')
78 print("set colors")
79 for index in range(num_leds):
80     pixels[index] = (1, 1, 1)
81 # write data to chips
82 pixels.show()
83 time.sleep(10)
84
85 #####
86
87 print(42 * '*')
88 print("loop..")
89 value_high = 1000
90 value_low = 1
91 while True:
92     pixel_active_index = 0
93     for index in range(pixels.channel_count):
94         if index == pixel_active_index:
95             pixels.set_channel(index, value_high)
96             pixels.set_channel((index - 1) % pixels.channel_count, value_low)
97             pixel_active_index += 1
98             # write data to chips
99             pixels.show()
100             # wait a second
101             time.sleep(0.5)
102         # set all to minimal
103         for index in range(num_leds):
104             pixels[index] = (value_low, value_low, value_low)
105         # write data to chips
106         pixels.show()
107         time.sleep(2)

```

5.3 TLC5957 with FancyLED

Example how to combine TLC5957 with FancyLED.

Listing 3: examples/slight_tlc5957_fancyled.py

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  # CircuitPython
4
5  """TLC5957 & FancyLED."""
6
7  __doc__ = """
8  TLC5957 & FancyLED.
9
10 this is an example for combining the TLC5957 library with FancyLED.
11 Enjoy the colors :-)
12 """
13
14 import board
15 # import busio
16 import bitbangio
17 import digitalio
18 import pulseio
19
20 import slight_tlc5957
21 import adafruit_fancyled.adafruit_fancyled as fancyled
22
23 #####
24 print(
25     "\n" +
26     (42 * '*') + "\n" +
27     __doc__ + "\n" +
28     (42 * '*') + "\n" +
29     "\n"
30 )
31
32 #####
33 print(42 * '*')
34 print("initialise digitalio pins for SPI")
35 spi_clock = digitalio.DigitalInOut(board.SCK)
36 spi_clock.direction = digitalio.Direction.OUTPUT
37 spi_mosi = digitalio.DigitalInOut(board.MOSI)
38 spi_mosi.direction = digitalio.Direction.OUTPUT
39 spi_miso = digitalio.DigitalInOut(board.MISO)
40 spi_miso.direction = digitalio.Direction.INPUT
41
42 # print((42 * '*') + "\n" + "init busio.SPI")
43 # spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
44 print("init bitbangio.SPI")
45 spi = bitbangio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
46
47 # maximum frequency is currently hardcoded to 6MHz
48 # https://github.com/adafruit/circuitpython/blob/master/ports/atmel-samd/common-hal/
49 ↳ pulseio/PWMOut.c#L119
50 gsclk_frequency = (6000 * 1000) # 6MHz
51 gsclk = pulseio.PWMOut(

```

(continues on next page)

(continued from previous page)

```

51     board.D9, duty_cycle=(2 ** 15), frequency=gsclk_frequency)
52 print("gsclk.frequency: {:.3f}MHz".format(gsclk.frequency / (1000*1000)))
53
54 latch = digitalio.DigitalInOut(board.D7)
55 latch.direction = digitalio.Direction.OUTPUT
56
57 #####
58 print(42 * '*')
59 print("define pixel array / init TLC5957")
60 num_leds = 32
61 pixels = slight_tlc5957.TLC5957(
62     spi=spi,
63     latch=latch,
64     gsclk=gsclk,
65     spi_clock=spi_clock,
66     spi_mosi=spi_mosi,
67     spi_miso=spi_miso,
68     pixel_count=num_leds)
69
70 print("pixel_count", pixels.pixel_count)
71 print("chip_count", pixels.chip_count)
72 print("channel_count", pixels.channel_count)
73
74
75 #####
76 # setup chip configuration
77 pixels.set_fc_CC_all(0x1FF, 0x1FF, 0x0FF)
78 pixels.set_fc_BC_all(0x4)
79 pixels.set_fc_ESPWM_all(enable=True)
80 pixels.print_buffer_fc()
81 pixels.update_fc()
82
83 #####
84 # helper function
85
86
87 #####
88 # Declare a 6-element RGB rainbow palette
89 palette = [
90     fancyled.CRGB(1.0, 0.0, 0.0), # Red
91     fancyled.CRGB(0.5, 0.5, 0.0), # Yellow
92     fancyled.CRGB(0.0, 1.0, 0.0), # Green
93     fancyled.CRGB(0.0, 0.5, 0.5), # Cyan
94     fancyled.CRGB(0.0, 0.0, 1.0), # Blue
95     fancyled.CRGB(0.5, 0.0, 0.5), # Magenta
96 ]
97
98 # Positional offset into color palette to get it to 'spin'
99 offset = 0
100
101 #####
102 # main loop
103 print(42 * '*')
104 print("rainbow loop")
105 while True:
106     for i in range(num_leds):
107         # Load each pixel's color from the palette using an offset, run it

```

(continues on next page)

(continued from previous page)

```

108     # through the gamma function, pack RGB value and assign to pixel.
109     # color = fancyled.palette_lookup(palette, offset + i / num_leds)
110     brightness = 0.2
111     color_offset = offset
112     if i % 2 == 0:
113         color_offset = offset + 0.5
114         # brightness = 0.1
115     if i >= num_leds/2:
116         brightness = 0.1
117     color = fancyled.palette_lookup(palette, color_offset)
118     color = fancyled.gamma_adjust(color, brightness=brightness)
119     pixels[i] = color
120     pixels.show()
121
122     offset += 0.005 # Bigger number = faster spin

```

5.4 TLC5957 with custom 2D-Array mapping

Example how to create a pixel-mapping with TLC5957.

Listing 4: examples/slight_tlc5957_2d_array.py

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  # CircuitPython
4
5  """TLC5957 & FancyLED & 2D-Array mapping."""
6
7  __doc__ = """
8  TLC5957 & FancyLED & 2D-Array mapping.
9
10 this is an example for combining the TLC5957 library with FancyLED.
11 Enjoy the colors :-)
12 """
13
14 import time
15
16 import board
17 # import busio
18 import bitbangio
19 import digitalio
20 import pulseio
21
22 import slight_tlc5957
23
24 #####
25 print(
26     "\n" +
27     (42 * '*') + "\n" +
28     __doc__ + "\n" +
29     (42 * '*') + "\n" +
30     "\n"
31 )
32

```

(continues on next page)

(continued from previous page)

```

33 #####
34 print(42 * '*')
35 print("initialise digitalio pins for SPI")
36 spi_clock = digitalio.DigitalInOut(board.SCK)
37 spi_clock.direction = digitalio.Direction.OUTPUT
38 spi_mosi = digitalio.DigitalInOut(board.MOSI)
39 spi_mosi.direction = digitalio.Direction.OUTPUT
40 spi_miso = digitalio.DigitalInOut(board.MISO)
41 spi_miso.direction = digitalio.Direction.INPUT
42
43 # print((42 * '*') + "\n" + "init busio.SPI")
44 # spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
45 print("init bitbangio.SPI")
46 spi = bitbangio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
47
48 # maximum frequency is currently hardcoded to 6MHz
49 # https://github.com/adafruit/circuitpython/blob/master/ports/atmel-samd/common-hal/
↳ pulseio/PWMOut.c#L119
50 gsclock_frequency = (6000 * 1000) # 6MHz
51 gsclock = pulseio.PWMOut(
52     board.D9, duty_cycle=(2 ** 15), frequency=gsclock_frequency)
53 print("gsclock.frequency: {:.3}MHz".format(gsclock.frequency / (1000*1000)))
54
55 latch = digitalio.DigitalInOut(board.D7)
56 latch.direction = digitalio.Direction.OUTPUT
57
58 #####
59 print(42 * '*')
60 print("define pixel array / init TLC5957")
61 rows = 4
62 cols = 4
63 num_leds = rows * cols
64 pixels = slight_tlc5957.TLC5957(
65     spi=spi,
66     latch=latch,
67     gsclock=gsclock,
68     spi_clock=spi_clock,
69     spi_mosi=spi_mosi,
70     spi_miso=spi_miso,
71     pixel_count=num_leds)
72
73 print("pixel_count", pixels.pixel_count)
74 print("chip_count", pixels.chip_count)
75 print("channel_count", pixels.channel_count)
76
77 #####
78 # setup chip configuration
79
80 pixels.set_fc_CC_all(0x1FF, 0x1FF, 0x0FF)
81 pixels.set_fc_BC_all(0x4)
82 pixels.set_fc_ESPWM_all(enable=True)
83 pixels.print_buffer_fc()
84 pixels.update_fc()
85
86
87 #####
88 # helper function

```

(continues on next page)

(continued from previous page)

```

89
90 def map_range(value, in_min, in_max, out_min, out_max):
91     """Map Value from one range to another."""
92     return (value - in_min) * (out_max - out_min) / (in_max - in_min) + out_min
93
94
95 def map_range_int(value, in_min, in_max, out_min, out_max):
96     """Map Value from one range to another."""
97     return int(
98         (value - in_min) * (out_max - out_min)
99         //
100         (in_max - in_min) + out_min
101     )
102
103
104 #####
105 # mapping function
106
107 pixel_map = [
108     # pylint: disable=bad-whitespace
109     [15, 14, 13, 12],
110     [11, 10, 9, 8],
111     [7, 6, 5, 4],
112     [3, 2, 1, 0],
113 ]
114
115
116 def get_pixel_index_from_row_col(row, col):
117     """Get pixel_index from row and column index."""
118     pixel_index = pixel_map[row][col]
119     return pixel_index
120
121
122 #####
123 # print(42 * '*')
124 # print("set colors")
125 # for index in range(num_leds):
126 #     pixels[index] = (1, 1, 1)
127 # # write data to chips
128 # pixels.show()
129 # time.sleep(2)
130 # print("set colors2")
131 # for index in range(num_leds):
132 #     pixels[index] = (0, 100, 1000)
133 # # write data to chips
134 # pixels.show()
135 # time.sleep(2)
136
137 #####
138 print(42 * '*')
139 print("set colors")
140 # set first pixel to orange
141 pixels[get_pixel_index_from_row_col(0, 0)] = (1.0, 0.5, 0.0)
142 pixels[get_pixel_index_from_row_col(0, 3)] = (0.1, 0.0, 1.0)
143 pixels[get_pixel_index_from_row_col(3, 0)] = (0.1, 0.5, 0.0)
144 pixels[get_pixel_index_from_row_col(3, 3)] = (0.0, 0.5, 1.0)
145 pixels.show()

```

(continues on next page)

(continued from previous page)

```

146 time.sleep(2)
147
148 print("set color range")
149 for x in range(cols):
150     # xN = x / cols
151     xN = map_range_int(x, 0, cols, 1, 100)
152     for y in range(rows):
153         # yN = y / rows
154         yN = map_range_int(y, 0, rows, 1, 100)
155         # print(
156             "x: {:>2} xN: {:>2} "
157             "y: {:>2} yN: {:>2} "
158             "pixel_index: {:>2}".format(
159                 x, xN,
160                 y, yN,
161                 get_pixel_index_from_row_col(x, y)
162             )
163         # )
164         pixels[get_pixel_index_from_row_col(x, y)] = (xN, yN, 0)
165
166 pixels.show()
167 time.sleep(2)
168
169 #####
170 # main loop
171 # Positional offset for blue part
172 offset = 0
173 print(42 * '*')
174 print("loop")
175 while True:
176     offsetN = map_range_int(offset, 0.0, 1.0, 1, 200)
177     for x in range(cols):
178         xN = map_range_int(x, 0, cols, 1, 500)
179         for y in range(rows):
180             yN = map_range_int(y, 0, rows, 1, 500)
181             pixels[get_pixel_index_from_row_col(x, y)] = (xN, yN, offsetN)
182         pixels.show()
183         offset += 0.01 # Bigger number = faster spin
184     if offset > 1.0:
185         offset = 0
186

```

5.5 API

5.5.1 s-light CircuitPython TLC5957 library.

CircuitPython library for TI TLC5957 48-channel 16bit LED-Driver

- Author(s): Stefan Krüger

5.5.1.1 Implementation Notes

Hardware:

- example PCB with TLC5957 and 4x4 SMD RGB LEDs https://github.com/s-light/magic_amulet_pcb/tree/master/LEDBoard_4x4_HD

Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://github.com/adafruit/circuitpython/releases>

class `slight_tlc5957.TLC5957` (*spi, spi_clock, spi_mosi, spi_miso, latch, gsclk, pixel_count=16*)
TLC5957 16-bit 48 channel LED PWM driver.

This chip is designed to drive 16 RGB LEDs with 16-bit PWM per Color. The class has an interface compatible with FancyLED. and with this is similar to the NeoPixel and DotStar Interfaces.

Parameters

- **spi** (*SPI*) – An instance of the SPI bus connected to the chip. The clock and MOSI must be set the MISO (input) is currently unused. Maximal data clock frequency is: - TLC5957: 33MHz
- **latch** (*DigitalInOut*) – The chip LAT (latch) pin object that implements the Digital-InOut API.
- **gsclk** (*PWMOut*) – The chip Grayscale Clock pin object that implements the PWMOut API.
- **pixel_count** (*bool*) – Number of RGB-LEDs (=Pixels) are connected.

get_fc_bits_in_buffer (*, *chip_index=0, part_bit_offset=0, field={'default': 0, 'length': 0, 'mask': 0, 'offset': 0}*)

Get function control bits in buffer.

print_buffer_fc ()

Print internal function_command buffer content.

print_buffer_fc_raw ()

Print internal function_command buffer content as raw binary.

set_all_black ()

Set all pixels to black.

static set_bit (*value, index, value_new*)

Set bit - return new value.

Set the index:th bit of value to 1 if value_new is truthy, else to 0, and return the new value.
<https://stackoverflow.com/a/12174051/574981>

static set_bit_with_mask (*value, mask, value_new*)

Set bit with help of mask.

set_channel (*channel_index, value*)

Set the value for the provided channel.

Parameters

- **channel_index** (*int*) – 0..channel_count
- **value** (*int*) – 0..65535

set_fc_BC (*chip_index=0, BC=4*)

Set brightness control.

set_fc_BC_all (*BC=4*)

Set brightness control for all chips.

set_fc_CC (*chip_index=0, CCR=256, CCG=256, CCB=256*)

Set color control for R, G, B.

set_fc_CC_all (*CCR=256, CCG=256, CCB=256*)

Set color control for R, G, B for all chips.

set_fc_ESPWM (*chip_index=0, enable=False*)

Set ESPWM.

set_fc_ESPWM_all (*enable=False*)

Set ESPWM for all chips.

set_fc_bits_in_buffer (*, *chip_index=0, part_bit_offset=0, field={'default': 0, 'length': 0, 'mask': 0, 'offset': 0}, value=0*)

Set function control bits in buffer.

set_pixel (*pixel_index, value*)

Set the R, G, B values for the pixel.

this function has some advanced error checking. it is much slower than the other provided 'bare' variants.. but therefore gives clues to what is going wrong.. ;-)

Parameters

- **pixel_index** (*int*) – 0..(pixel_count)
- **value** (*tuple*) – 3-tuple of R, G, B; each int 0..65535 or float 0..1

set_pixel_16bit_color (*pixel_index, color*)

Set color for pixel.

This is a Fast UNPROTECTED function: no error / range checking is done. its a little bit slower as *set_pixel_16bit_value*

Parameters

- **pixel_index** (*int*) – 0..(pixel_count)
- **color** (*int*) – 3-tuple of R, G, B; 0..65535

set_pixel_16bit_value (*pixel_index, value_r, value_g, value_b*)

Set the value for pixel.

This is a Fast UNPROTECTED function: no error / range checking is done.

Parameters

- **pixel_index** (*int*) – 0..(pixel_count)
- **value_r** (*int*) – 0..65535
- **value_g** (*int*) – 0..65535
- **value_b** (*int*) – 0..65535

set_pixel_all (*color*)

Set the R, G, B values for all pixels.

:param tuple 3-tuple of R, G, B; each int 0..65535 or float 0..1

set_pixel_all_16bit_value (*value_r, value_g, value_b*)

Set the R, G, B values for all pixels.

fast. without error checking.

Parameters

- **value_r** (*int*) – 0..65535
- **value_g** (*int*) – 0..65535

- **value_b** (*int*) – 0..65535

set_pixel_float_color (*pixel_index*, *color*)

Set color for pixel.

This is a Fast UNPROTECTED function: no error / range checking is done. its a little bit slower as *set_pixel_16bit_value*

Parameters

- **pixel_index** (*int*) – 0..(pixel_count)
- **color** (*tuple/float*) – 3-tuple of R, G, B; 0..1

set_pixel_float_value (*pixel_index*, *value_r*, *value_g*, *value_b*)

Set the value for pixel.

This is a Fast UNPROTECTED function: no error / range checking is done.

Parameters

- **pixel_index** (*int*) – 0..(pixel_count)
- **value_r** (*int*) – 0..1
- **value_g** (*int*) – 0..1
- **value_b** (*int*) – 0..1

show ()

Write out Grayscale Values to chips.

update_fc ()

Write out Function_Command Values to chips.

CHAPTER 6

Indices and tables

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

S

`slight_tlc5957`, [20](#)

G

`get_fc_bits_in_buffer()`
(*slight_tlc5957.TLC5957* method), 21

P

`print_buffer_fc()` (*slight_tlc5957.TLC5957*
method), 21
`print_buffer_fc_raw()` (*slight_tlc5957.TLC5957*
method), 21

S

`set_all_black()` (*slight_tlc5957.TLC5957*
method), 21
`set_bit()` (*slight_tlc5957.TLC5957* static method), 21
`set_bit_with_mask()` (*slight_tlc5957.TLC5957*
static method), 21
`set_channel()` (*slight_tlc5957.TLC5957* method),
21
`set_fc_BC()` (*slight_tlc5957.TLC5957* method), 21
`set_fc_BC_all()` (*slight_tlc5957.TLC5957*
method), 21
`set_fc_bits_in_buffer()`
(*slight_tlc5957.TLC5957* method), 22
`set_fc_CC()` (*slight_tlc5957.TLC5957* method), 21
`set_fc_CC_all()` (*slight_tlc5957.TLC5957*
method), 22
`set_fc_ESPWM()` (*slight_tlc5957.TLC5957* method),
22
`set_fc_ESPWM_all()` (*slight_tlc5957.TLC5957*
method), 22
`set_pixel()` (*slight_tlc5957.TLC5957* method), 22
`set_pixel_16bit_color()`
(*slight_tlc5957.TLC5957* method), 22
`set_pixel_16bit_value()`
(*slight_tlc5957.TLC5957* method), 22
`set_pixel_all()` (*slight_tlc5957.TLC5957*
method), 22
`set_pixel_all_16bit_value()`
(*slight_tlc5957.TLC5957* method), 22

`set_pixel_float_color()`
(*slight_tlc5957.TLC5957* method), 23
`set_pixel_float_value()`
(*slight_tlc5957.TLC5957* method), 23
`show()` (*slight_tlc5957.TLC5957* method), 23
`slight_tlc5957` (module), 20

T

`TLC5957` (class in *slight_tlc5957*), 21

U

`update_fc()` (*slight_tlc5957.TLC5957* method), 23