SlightTLC5957 Library Documentation

Release 1.0

Stefan Krüger

Apr 19, 2019

Contents

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.

CHAPTER 4

Building locally

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.

CHAPTER 5

Table of Contents

5.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/slight_tlc5957_simpletest.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# CircuitPython

"""Simple & Minimallistic example for the TLC5957 library."""

# import time

import board
# import busio
import bitbangio
import digitalio
import pulseio

import slight_tlc5957


spi_clock = digitalio.DigitalInOut(board.SCK)
spi_clock.direction = digitalio.Direction.OUTPUT
spi_mosi = digitalio.DigitalInOut(board.MOSI)
spi_mosi.direction = digitalio.Direction.OUTPUT
spi_miso = digitalio.DigitalInOut(board.MISO)
spi_miso.direction = digitalio.Direction.INPUT

# spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
spi = bitbangio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)

```

(continues on next page)

(continued from previous page)

```
# 6MHz for the grayscale clock
gsclk = pulseio.PWMOut(
    board.D9, duty_cycle=(2 ** 15), frequency=(6000 * 1000))

latch = digitalio.DigitalInOut(board.D7)
latch.direction = digitalio.Direction.OUTPUT

# define pixel array
num_leds = 16
pixels = slight_tlc5957.TLC5957(
    spi=spi,
    latch=latch,
    gsclk=gsclk,
    spi_clock=spi_clock,
    spi_mosi=spi_mosi,
    spi_miso=spi_miso,
    pixel_count=num_leds)


# set first pixel to orange
# using floating point values (0..1)
# pixels[0] = (1, 0.5, 0)
# set first pixel to sky blue
# using 16bit integer values (0..65535)
# pixels[1] = (0, 32000, 65535)

# write data to chips
pixels.show()

fade_value = 0
step = 500

print("loop..")

pixel_index = 3
buffer_index = (
    pixel_index * pixels.COLORS_PER_PIXEL * pixels.BUFFER_BYTES_PER_COLOR)

while True:
    # pixels[3] = (0, 100, fade_value)
    pixels.set_pixel_all_16bit_value(1, 1, 1)
    pixels.show()
    if (fade_value + step) > 65535 or (fade_value + step) < 0:
        step *= -1
    fade_value += step
    # 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

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# CircuitPython

"""Develop and Test TLC5957."""

__doc__ = """
Develop and Test TLC5957.

this script contains a bunch of tests and debug outputs.
its mainly the playground during the development of the library.
"""

import time

import board
# import busio
import bitbangio
import digitalio
import pulseio

import slight_tlc5957

#########################################
print(
    "\n" +
    (42 * '*') + "\n" +
    __doc__ + "\n" +
    (42 * '*') + "\n" +
    "\n"
)

#########################################
print(42 * '*')
print("initialise digitalio pins for SPI")
spi_clock = digitalio.DigitalInOut(board.SCK)
spi_clock.direction = digitalio.Direction.OUTPUT
spi_mosi = digitalio.DigitalInOut(board.MOSI)
spi_mosi.direction = digitalio.Direction.OUTPUT
spi_miso = digitalio.DigitalInOut(board.MISO)
spi_miso.direction = digitalio.Direction.INPUT

# print((42 * '*') + "\n" + "init busio.SPI")
# spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
print("init bitbangio.SPI")
spi = bitbangio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)

# maximum frequency is currently hardcoded to 6MHz
# https://github.com/adafruit/circuitpython/blob/master/ports/atmel-samd/common-hal/
↪pulseio/PWMOut.c#L119
gsclk_freqency = (6000 * 1000)  # 6MHz
gsclk = pulseio.PWMOut(
    board.D9, duty_cycle=(2 ** 15), frequency=gsclk_freqency)
print("gsclk.frequency: {:}MHz".format(gsclk.frequency / (1000*1000)))

```

(continues on next page)

(continued from previous page)

```
latch = digitalio.DigitalInOut(board.D7)
latch.direction = digitalio.Direction.OUTPUT

########################################
print(42 * '*')
print("define pixel array / init TLC5957")
num_leds = 16
pixels = slight_tlc5957.TLC5957(
    spi=spi,
    latch=latch,
    gsclk=gsclk,
    spi_clock=spi_clock,
    spi_mosi=spi_mosi,
    spi_miso=spi_miso,
    pixel_count=num_leds)

print("pixel_count", pixels.pixel_count)
print("chip_count", pixels.chip_count)
print("channel_count", pixels.channel_count)


########################################
print(42 * '*')
print("set colors")
for index in range(num_leds):
    pixels[index] = (1, 1, 1)
# write data to chips
pixels.show()
time.sleep(10)

########################################
print(42 * '*')
print("loop..")
value_high = 1000
value_low = 1
while True:
    pixel_active_index = 0
    for index in range(pixels.channel_count):
        if index == pixel_active_index:
            pixels.set_channel(index, value_high)
            pixels.set_channel((index - 1) % pixels.channel_count, value_low)
        pixel_active_index += 1
        # write data to chips
        pixels.show()
        # wait a second
        time.sleep(0.5)
    # set all to minimal
    for index in range(num_leds):
        pixels[index] = (value_low, value_low, value_low)
    # write data to chips
    pixels.show()
    time.sleep(2)
```

5.3 TLC5957 with FancyLED

Example how to combine TLC5957 with FancyLED.

Listing 3: examples/slight_tlc5957_fancyled.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# CircuitPython

"""TLC5957 & FancyLED."""

__doc__ = """
TLC5957 & FancyLED.

this is an example for combining the TLC5957 library with FancyLED.
Enjoy the colors :-)
"""

import board
# import busio
import bitbangio
import digitalio
import pulseio

import slight_tlc5957
import adafruit_fancyled.adafruit_fancyled as fancyled

########################################
print(
    "\n" +
    (42 * '*') + "\n" +
    __doc__ + "\n" +
    (42 * '*') + "\n" +
    "\n"
)

########################################
print(42 * '*')
print("initialise digitalio pins for SPI")
spi_clock = digitalio.DigitalInOut(board.SCK)
spi_clock.direction = digitalio.Direction.OUTPUT
spi_mosi = digitalio.DigitalInOut(board.MOSI)
spi_mosi.direction = digitalio.Direction.OUTPUT
spi_miso = digitalio.DigitalInOut(board.MISO)
spi_miso.direction = digitalio.Direction.INPUT

# print((42 * '*') + "\n" + "init busio.SPI")
# spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
print("init bitbangio.SPI")
spi = bitbangio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)

# maximum frequency is currently hardcoded to 6MHz
# https://github.com/adafruit/circuitpython/blob/master/ports/atmel-samd/common-hal/
↪pulseio/PWMOut.c#L119
gsclk_freqency = (6000 * 1000)  # 6MHz
gsclk = pulseio.PWMOut(
```

(continues on next page)

(continued from previous page)

```
    board.D9, duty_cycle=(2 ** 15), frequency=gsclk_freqency)
print("gsclk.frequency: {:}MHz".format(gsclk.frequency / (1000*1000)))

latch = digitalio.DigitalInOut(board.D7)
latch.direction = digitalio.Direction.OUTPUT

########################################
print(42 * '*')
print("define pixel array / init TLC5957")
num_leds = 32
pixels = slight_tlc5957.TLC5957(
    spi=spi,
    latch=latch,
    gsclk=gsclk,
    spi_clock=spi_clock,
    spi_mosi=spi_mosi,
    spi_miso=spi_miso,
    pixel_count=num_leds)

print("pixel_count", pixels.pixel_count)
print("chip_count", pixels.chip_count)
print("channel_count", pixels.channel_count)


########################################
# setup chip configuration
pixels.set_fc_CC_all(0x1FF, 0x1FF, 0x0FF)
pixels.set_fc_BC_all(0x4)
pixels.set_fc_ESPWM_all(enable=True)
pixels.print_buffer_fc()
pixels.update_fc()

########################################
# helper function


########################################
# Declare a 6-element RGB rainbow palette
palette = [
    fancyled.CRGB(1.0, 0.0, 0.0),  # Red
    fancyled.CRGB(0.5, 0.5, 0.0),  # Yellow
    fancyled.CRGB(0.0, 1.0, 0.0),  # Green
    fancyled.CRGB(0.0, 0.5, 0.5),  # Cyan
    fancyled.CRGB(0.0, 0.0, 1.0),  # Blue
    fancyled.CRGB(0.5, 0.0, 0.5),  # Magenta
]

# Positional offset into color palette to get it to 'spin'
offset = 0

########################################
# main loop
print(42 * '*')
print("rainbow loop")
while True:
    for i in range(num_leds):
        # Load each pixel's color from the palette using an offset, run it
```

(continues on next page)

(continued from previous page)

```
        # through the gamma function, pack RGB value and assign to pixel.
        # color = fancyled.palette_lookup(palette, offset + i / num_leds)
        brightness = 0.2
        color_offset = offset
        if i % 2 == 0:
            color_offset = offset + 0.5
            # brightness = 0.1
        if i >= num_leds/2:
            brightness = 0.1
        color = fancyled.palette_lookup(palette, color_offset)
        color = fancyled.gamma_adjust(color, brightness=brightness)
        pixels[i] = color
    pixels.show()

    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

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# CircuitPython

"""TLC5957 & FancyLED & 2D-Array mapping."""

__doc__ = """
TLC5957 & FancyLED & 2D-Array mapping.

this is an example for combining the TLC5957 library with FancyLED.
Enjoy the colors :-)
"""

import time

import board
# import busio
import bitbangio
import digitalio
import pulseio

import slight_tlc5957

########################################
print(
    "\n" +
    (42 * '*') + "\n" +
    __doc__ + "\n" +
    (42 * '*') + "\n" +
    "\n"
)

```

(continues on next page)

(continued from previous page)

```
##########################################
print(42 * '*')
print("initialise digitalio pins for SPI")
spi_clock = digitalio.DigitalInOut(board.SCK)
spi_clock.direction = digitalio.Direction.OUTPUT
spi_mosi = digitalio.DigitalInOut(board.MOSI)
spi_mosi.direction = digitalio.Direction.OUTPUT
spi_miso = digitalio.DigitalInOut(board.MISO)
spi_miso.direction = digitalio.Direction.INPUT

# print((42 * '*') + "\n" + "init busio.SPI")
# spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
print("init bitbangio.SPI")
spi = bitbangio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)

# maximum frequency is currently hardcoded to 6MHz
# https://github.com/adafruit/circuitpython/blob/master/ports/atmel-samd/common-hal/
↪pulseio/PWMOut.c#L119
gsclk_freqency = (6000 * 1000)  # 6MHz
gsclk = pulseio.PWMOut(
    board.D9, duty_cycle=(2 ** 15), frequency=gsclk_freqency)
print("gsclk.frequency: {:}MHz".format(gsclk.frequency / (1000*1000)))

latch = digitalio.DigitalInOut(board.D7)
latch.direction = digitalio.Direction.OUTPUT

##########################################
print(42 * '*')
print("define pixel array / init TLC5957")
rows = 4
cols = 4
num_leds = rows * cols
pixels = slight_tlc5957.TLC5957(
    spi=spi,
    latch=latch,
    gsclk=gsclk,
    spi_clock=spi_clock,
    spi_mosi=spi_mosi,
    spi_miso=spi_miso,
    pixel_count=num_leds)

print("pixel_count", pixels.pixel_count)
print("chip_count", pixels.chip_count)
print("channel_count", pixels.channel_count)

##########################################
# setup chip configuration

pixels.set_fc_CC_all(0x1FF, 0x1FF, 0x0FF)
pixels.set_fc_BC_all(0x4)
pixels.set_fc_ESPWM_all(enable=True)
pixels.print_buffer_fc()
pixels.update_fc()


##########################################
# 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)

```
time.sleep(2)

print("set color range")
for x in range(cols):
    # xN = x / cols
    xN = map_range_int(x, 0, cols, 1, 100)
    for y in range(rows):
        # yN = y / rows
        yN = map_range_int(y, 0, rows, 1, 100)
        # print(
        #     "x: {:>2} xN: {:>2} "
        #     "y: {:>2} yN: {:>2} "
        #     "pixel_index: {:>2}".format(
        #         x, xN,
        #         y, yN,
        #         get_pixel_index_from_row_col(x, y)
        #     )
        # )
        pixels[get_pixel_index_from_row_col(x, y)] = (xN, yN, 0)

pixels.show()
time.sleep(2)


########################################
# main loop
# Positional offset for blue part
offset = 0
print(42 * '*')
print("loop")
while True:
    offsetN = map_range_int(offset, 0.0, 1.0, 1, 200)
    for x in range(cols):
        xN = map_range_int(x, 0, cols, 1, 500)
        for y in range(rows):
            yN = map_range_int(y, 0, rows, 1, 500)
            pixels[get_pixel_index_from_row_col(x, y)] = (xN, yN, offsetN)
    pixels.show()
    offset += 0.01  # Bigger number = faster spin
    if offset > 1.0:
        offset = 0
```

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_pcbs/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 frequence is: - TLC5957: 33MHz
- **latch** (*DigitalInOut*) – The chip LAT (latch) pin object that implements the DigitalInOut 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 funciton hase some advanced error checking. it is much slower than the other provided 'bare' variants.. but therefor 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

Python Module Index

S

Index