AdafruitTLC59711 Library Documentation

Release 1.0

Tony DiCola

Apr 29, 2021

Contents

CircuitPython module for the TLC59711 16-bit 12 channel LED PWM driver.

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

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-tlc59711
```

To install system-wide (this may be required in some cases):

```
sudo pip3 install adafruit-circuitpython-tlc59711
```

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-tlc59711
```

CHAPTER 3

Usage Example

See examples/tlc59711_simpletest.py for a demo of the usage.

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/tlc59711_simpletest.py

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

# simple demo of the TLC59711 16-bit 12 channel LED PWM driver.
# Shows the minimal usage - how to set pixel values in a few ways.
# Author: Tony DiCola

import board
import busio

import adafruit_tlc59711

print("tlc59711_simpletest.py")

# Define SPI bus connected to chip.
# You only need the clock and MOSI (output) line to use this chip.
spi = busio.SPI(board.SCK, MOSI=board.MOSI)
pixels = adafruit_tlc59711.TLC59711(spi, pixel_count=16)

# examples how to set the pixels:
# range:
# 0 - 65535
# or
# 0.0 - 1.0
# every pixel needs a color -
# give it just a list or tuple with 3 integer values: R G B

```

(continues on next page)

(continued from previous page)

```
# set all pixels to a very low level
pixels.set_pixel_all((10, 10, 10))

# every chip has 4 Pixels (=RGB-LEDs = 12 Channel)
pixels[0] = (100, 100, 100)
pixels[1] = (0, 0, 100)
pixels[2] = (0.01, 0.0, 0.01)
pixels[3] = (0.1, 0.01, 0.0)
# if you are ready to show your values you have to call
pixels.show()

# there are a bunch of other ways to set pixel.
# have a look at the other examples.
```

6.2 FancyLED

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

Listing 2: examples/tlc59711_fancyled.py

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

# SPDX-FileCopyrightText: 2021 s-light
# SPDX-License-Identifier: MIT
# Author Stefan Krüger (s-light)

"""TLC59711 & FancyLED."""

__doc__ = """
TLC59711 & FancyLED.

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

import board

import busio

import adafruit_fancyled.adafruit_fancyled as fancyled
import adafruit_tlc59711

########################################
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
```

(continues on next page)

(continued from previous page)

```
# 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)

########################################
print(42 * "*")
print("init TLC5957")
NUM_LEDS = 16
pixels = adafruit_tlc59711.TLC59711(
    spi=spi,
    pixel_count=NUM_LEDS,
)

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


########################################
# main loop
print(42 * "*")
print("rainbow loop")
hue_offset = 0
while True:
    brightness = 0.8
    color = fancyled.CHSV(hue_offset, 1.0, 1.0)
    color = fancyled.gamma_adjust(color, brightness=brightness)
    pixels.set_pixel_all(color)
    pixels.show()

    # Bigger number = faster spin
    hue_offset += 0.000005
    if hue_offset >= 1:
        hue_offset = 0
        print("heu_offset reset")
```

6.3 Animations Example

You should have a plethora of functions to animate your lights.

Listing 3: examples/tlc59711_dev.py

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

# SPDX-FileCopyrightText: 2021 s-light
# SPDX-License-Identifier: MIT
# Author Stefan Krüger (s-light)

"""TLC5971 / TLC59711 Multi Development."""

```

(continues on next page)

(continued from previous page)

```
__doc__ = """
TLC59711 development helper.

this sketch contains a bunch of timing tests and other development things..
Enjoy the colors :-)
"""

import time

import board
import busio

import adafruit_tlc59711


########################################
PIXEL_COUNT = 16 * 1

spi = busio.SPI(board.SCK, MOSI=board.MOSI)
pixels = adafruit_tlc59711.TLC59711(spi, pixel_count=PIXEL_COUNT)


########################################
# test function

VALUE_HIGH = 1000


def channelcheck_update_pixel(offset):
    """Channel check pixel."""
    # print("offset", offset)

    # pixels[offset] = (VALUE_HIGH, 0, 0)
    pixels.set_pixel_16bit_value(offset, VALUE_HIGH, 0, 0)
    # clear last pixel
    last = offset - 1
    if last < 0:
        last = PIXEL_COUNT - 1
    # pixels[last] = (0, 0, 1)
    pixels.set_pixel_16bit_value(last, 0, 0, 1)
    # pixels[offset] = (0xAAAA, 0xBBBB, 0xCCCC)
    pixels.show()

    offset += 1
    if offset >= PIXEL_COUNT:
        time.sleep(0.2)
        offset = 0
        print("clear")
        pixels.set_pixel_all((0, 1, 0))
        pixels.show()
    return offset


def channelcheck_update(offset):
    """Channel check."""
    # print("offset", offset)

```

(continues on next page)

(continued from previous page)

```
    pixels.set_channel(offset, VALUE_HIGH)
    # clear last set channel
    last = offset - 1
    if last < 0:
        last = pixels.channel_count - 1
    pixels.set_channel(last, 0)
    pixels.show()

    offset += 1
    if offset >= pixels.channel_count:
        offset = 0
        print("offset overflow. start from 0")
    return offset


########################################


def timeit_call(message, test_function, loop_count=1000):
    """Measure timing."""
    duration = 0
    start_time = time.monotonic()
    for _index in range(loop_count):
        start_time = time.monotonic()
        test_function()
        end_time = time.monotonic()
        duration += end_time - start_time
    # print(
    #     "duration:\n"
    #     "\t{}s for {} loops\n"
    #     "\t{:.2f}ms per call"
    #     "".format(
    #         duration,
    #         loop_count,
    #         (duration/loop_count)*1000
    #     )
    # )
    # print(
    #     "\t{:.2f}ms per call"
    #     "".format((duration / loop_count) * 1000)
    # )
    # "{:>8.2f}ms".format(3.56)
    print(
        "{call_duration:>10.4f}ms\t{message}"
        "".format(
            call_duration=(duration / loop_count) * 1000,
            message=message,
        )
    )


def timeit_pixels_show():
    """Measure timing."""
    print("*** pixels show:")
    loop_count = 1000

    def _test():
```

(continues on next page)

(continued from previous page)

```
        pixels.show()

    timeit_call("'pixels.show()'", _test, loop_count)


def timeit_pixels_set_single():
    """Measure timing pixels set."""
    print("*** pixels set single:")
    loop_count = 1000

    def _test():
        pixels[3] = (500, 40500, 1000)

    timeit_call("'pixels[3] = (500, 40500, 1000)'", _test, loop_count)

    def _test():
        pixels[3] = (0.1, 0.5, 0.9)

    timeit_call("'pixels[3] = (0.1, 0.5, 0.9)'", _test, loop_count)

    def _test():
        pixels.set_pixel(3, (500, 40500, 1000))

    timeit_call("'pixels.set_pixel(3, (500, 40500, 1000))'", _test, loop_count)

    def _test():
        pixels.set_pixel(3, (0.1, 0.5, 0.9))

    timeit_call("'pixels.set_pixel(3, (0.1, 0.5, 0.9))'", _test, loop_count)


def timeit_pixels_set_loop():
    """Measure timing pixels set."""
    print("*** pixels set loop:")
    loop_count = 10

    def _test():
        for i in range(PIXEL_COUNT):
            pixels[i] = (500, 40500, 1000)

    timeit_call(
        "'pixels[0..{}] = (500, 40500, 1000)'".format(PIXEL_COUNT), _test, loop_count
    )

    def _test():
        for i in range(PIXEL_COUNT):
            pixels[i] = (0.1, 0.5, 0.9)

    timeit_call(
        "'pixels[0..{}] = (0.1, 0.5, 0.9)'".format(PIXEL_COUNT), _test, loop_count
    )

    def _test():
        for i in range(PIXEL_COUNT):
            pixels.set_pixel(i, (500, 40500, 1000))

    timeit_call(
```

(continues on next page)

(continued from previous page)

```
        "'pixels.set_pixel(0..{}, (500, 40500, 1000))'".format(PIXEL_COUNT),
        _test,
        loop_count,
    )

    def _test():
        for i in range(PIXEL_COUNT):
            pixels.set_pixel(i, (0.1, 0.5, 0.9))

    timeit_call(
        "'pixels.set_pixel(0..{}, (0.1, 0.5, 0.9))'".format(PIXEL_COUNT),
        _test,
        loop_count,
    )


def timeit_pixels_set_all():
    """Measure timing pixels set."""
    print("*** pixels set all:")
    loop_count = 10

    def _test():
        pixels.set_pixel_all((500, 40500, 1000))

    timeit_call("'pixels.set_pixel_all((500, 40500, 1000))'", _test, loop_count)

    def _test():
        pixels.set_pixel_all((0.1, 0.5, 0.9))

    timeit_call("'pixels.set_pixel_all((0.1, 0.5, 0.9))'", _test, loop_count)

    def _test():
        pixels.set_pixel_all_16bit_value(500, 40500, 1000)

    timeit_call(
        "'pixels.set_pixel_all_16bit_value(500, 40500, 1000)'", _test, loop_count
    )

    def _test():
        pixels.set_all_black()

    timeit_call("'pixels.set_all_black()'", _test, loop_count)


def timeit_pixels_set_16bit():
    """Measure timing pixels set."""
    print("*** pixels set 16bit:")
    loop_count = 1000

    def _test():
        pixels.set_pixel_16bit_value(3, 500, 40500, 1000)

    timeit_call(
        "'pixels.set_pixel_16bit_value(3, 500, 40500, 1000)'", _test, loop_count
    )

    def _test():
```

(continues on next page)

(continued from previous page)

```
        pixels.set_pixel_16bit_color(3, (500, 40500, 1000))

    timeit_call(
        "'pixels.set_pixel_16bit_color(3, (500, 40500, 1000))'", _test, loop_count
    )

    def _test():
        for i in range(PIXEL_COUNT):
            pixels.set_pixel_16bit_value(i, 500, 40500, 1000)

    timeit_call(
        "'pixels.set_pixel_16bit_value(0..{}, 500, 40500, 1000)'"
        "".format(PIXEL_COUNT),
        _test,
        10,
    )

    def _test():
        for i in range(PIXEL_COUNT):
            pixels.set_pixel_16bit_color(i, (500, 40500, 1000))

    timeit_call(
        "'pixels.set_pixel_16bit_color(0..{}, (500, 40500, 1000))'"
        "".format(PIXEL_COUNT),
        _test,
        10,
    )


def timeit_pixels_set_float():
    """Measure timing pixels set."""
    print("*** pixels set float:")
    loop_count = 1000

    def _test():
        pixels.set_pixel_float_value(3, 0.1, 0.5, 0.9)

    timeit_call("'pixels.set_pixel_float_value(3, 0.1, 0.5, 0.9)'", _test, loop_count)

    def _test():
        pixels.set_pixel_float_color(3, (0.1, 0.5, 0.9))

    timeit_call("'pixels.set_pixel_float_color(3, (0.1, 0.5, 0.9))'", _test, loop_
↪count)

    def _test():
        for i in range(PIXEL_COUNT):
            pixels.set_pixel_float_value(i, 0.1, 0.5, 0.9)

    timeit_call(
        "'pixels.set_pixel_float_value(0..{}, 0.1, 0.5, 0.9)'" "".format(PIXEL_COUNT),
        _test,
        10,
    )

    def _test():
        for i in range(PIXEL_COUNT):
```

(continues on next page)

(continued from previous page)

```
            pixels.set_pixel_float_color(i, (0.1, 0.5, 0.9))

    timeit_call(
        "'pixels.set_pixel_float_color(0..{}, (0.1, 0.5, 0.9))'" "".format(PIXEL_
↪COUNT),
        _test,
        10,
    )

    def _test():
        for i in range(PIXEL_COUNT):
            pixels.set_pixel_16bit_value(
                i, int(0.1 * 65535), int(0.5 * 65535), int(0.9 * 65535)
            )

    timeit_call(
        "'pixels.set_pixel_16bit_value(0..{}, f2i 0.1, f2i 0.5, f2i 0.9)'"
        "".format(PIXEL_COUNT),
        _test,
        10,
    )


def timeit_channel_set():
    """Measure timing channel set."""
    print("*** channel set:")
    loop_count = 1000

    def _test():
        pixels.set_channel(0, 10000)

    timeit_call("'set_channel(0, 10000)'", _test, loop_count)

    def _test():
        pixels.set_channel(0, 10000)
        pixels.set_channel(1, 10000)
        pixels.set_channel(2, 10000)

    timeit_call("'set_channel(0..2, 10000)'", _test, loop_count)

    channel_count = PIXEL_COUNT * 3

    def _test():
        for i in range(channel_count):
            pixels.set_channel(i, 500)

    timeit_call("'set_channel(for 0..{}, 10000)'" "".format(channel_count), _test, 10)


def timeit_channel_set_internal():
    """Measure timing channel set internal."""
    print("*** channel set internal:")
    # loop_count = 1000
    #
    # def _test():
    #     pixels._set_channel_16bit_value(0, 10000)
    # timeit_call(
```

(continues on next page)

(continued from previous page)

```
    #     "'_set_channel_16bit_value(0, 10000)'",
    #     _test,
    #     loop_count
    # )
    #
    # def _test():
    #     pixels._set_channel_16bit_value(0, 10000)
    #     pixels._set_channel_16bit_value(1, 10000)
    #     pixels._set_channel_16bit_value(2, 10000)
    # timeit_call(
    #     "'_set_channel_16bit_value(0..2, 10000)'",
    #     _test,
    #     loop_count
    # )
    #
    # def _test():
    #     for i in range(PIXEL_COUNT * 3):
    #         pixels._set_channel_16bit_value(i, 500)
    # timeit_call(
    #     "'_set_channel_16bit_value(for 0..{}, 10000)'"
    #     "".format(PIXEL_COUNT * 3),
    #     _test,
    #     10
    # )
    print("    must be uncommented in code to work..")


def timeit_pixels_get():
    """Measure timing pixels get."""
    print("*** pixels get:")

    pixels.set_pixel_all((1, 11, 111))

    def _test():
        print("[", end="")
        for i in range(PIXEL_COUNT):
            print("{}:{}, ".format(i, pixels[i]), end="")
        print("]")

    timeit_call("'print('{}:{}, '.format(i, pixels[i]), end='')'", _test, 1)


def time_measurement():
    """Measure timing."""
    print("meassure timing:")
    timeit_pixels_show()
    timeit_pixels_set_single()
    timeit_pixels_set_loop()
    timeit_pixels_set_all()
    timeit_pixels_set_16bit()
    timeit_pixels_set_float()
    timeit_channel_set()
    timeit_channel_set_internal()
    timeit_pixels_get()
    pixels.set_pixel_all((0, 1, 1))


```

(continues on next page)

(continued from previous page)

```
########################################


def test_bcdata():
    """Test BC-Data setting."""
    print("test BC-Data setting:")
    print("set pixel all to 100, 100, 100")
    pixels.set_pixel_all((100, 100, 100))
    pixels.show()
    time.sleep(2)
    print(
        "bcr: {:>3}\n"
        "bcg: {:>3}\n"
        "bcb: {:>3}\n"
        "".format(
            pixels.bcr,
            pixels.bcg,
            pixels.bcb,
        )
    )
    # calculate bc values
    Ioclmax = adafruit_tlc59711.TLC59711.calculate_Ioclmax(Riref=2.7)
    print("Ioclmax = {}".format(Ioclmax))
    Riref = adafruit_tlc59711.TLC59711.calculate_Riref(Ioclmax=Ioclmax)
    print("Riref = {}".format(Riref))
    BCValues = adafruit_tlc59711.TLC59711.calculate_BCData(
        Ioclmax=Ioclmax,
        IoutR=18,
        IoutG=11,
        IoutB=13,
    )
    # (127, 77, 91)
    print("BCValues = {}".format(BCValues))

    print("set bcX")
    pixels.bcr = BCValues[0]
    pixels.bcg = BCValues[1]
    pixels.bcb = BCValues[2]
    pixels.update_BCData()
    pixels.show()
    print(
        "bcr: {:>3}\n"
        "bcg: {:>3}\n"
        "bcb: {:>3}\n"
        "".format(
            pixels.bcr,
            pixels.bcg,
            pixels.bcb,
        )
    )
    time.sleep(2)


########################################


def test_main():
```

(continues on next page)

(continued from previous page)

```
    """Test Main."""
    print(42 * "*", end="")
    print(__doc__, end="")
    print(42 * "*")
    # print()
    # time.sleep(0.5)
    # print(42 * '*')

    pixels.set_pixel_all_16bit_value(1, 10, 100)
    pixels.show()
    time.sleep(0.5)

    test_bcdata()
    time.sleep(0.5)
    print(42 * "*")

    time_measurement()
    time.sleep(0.5)
    print(42 * "*")

    offset = 0

    print("loop:")
    while True:
        offset = channelcheck_update(offset)
        time.sleep(0.5)
        print(offset)


########################################
# main loop

if __name__ == "__main__":

    test_main()
```

6.4 Single Chip Autoshow

This makes it very slow on lots of pixel changes but is convenient for only a handful of pixels.

Listing 4: examples/tlc59711_singlechip_autoshow.py

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

# SPDX-FileCopyrightText: 2021 s-light
# SPDX-License-Identifier: MIT
# Author Stefan Krüger (s-light)

"""TLC5971 / TLC59711 Example."""

__doc__ = """
tlc59711_singlechip_autoshow.py - TLC59711AutoShow minimal usage example.
```

(continues on next page)

(continued from previous page)

```

simple demo of the TLC59711 16-bit 12 channel LED PWM driver.
Shows the minimal usage - how to set pixel values.
the TLC59711AutoShow class automatically writes the pixel values on each change.
this makes it very slow on lots of pixel changs but is convenient for only a handfull of pixels..

Author: Tony DiCola, Stefan Krueger

Enjoy the colors :-)
"""

import board
import busio

import adafruit_tlc59711

print(__doc__)

# Define SPI bus connected to chip.
# You only need the clock and MOSI (output) line to use this chip.
spi = busio.SPI(board.SCK, MOSI=board.MOSI)
pixels = adafruit_tlc59711.TLC59711AutoShow(spi)

# Ways to set the values:
# just a list or tuple with 3 integer values: R G B
# each 0 - 65535 or 0.0 - 1.0
# every chip has 4 RGB-LEDs (=12 Channel)
pixels[0] = (100, 100, 10111)
pixels[1] = (0, 0, 100)
pixels[2] = (0.01, 0.0, 0.01)
pixels[3] = (0.1, 0.01, 0.0)

# You can also explicitly control each R0, G0, B0, R1, B1, etc. channel of the first ic
# by getting and setting its 16-bit value directly with properties.
# For example set channel 2 to 1/4 green (i.e. G2):
pixels.g2 = 65535 // 4

# there are a bunch of other advanced ways to set pixel.
# have a look at the other examples.
```

6.5 Brightness Correction Data

Test brightness correction data (BC)

Listing 5: examples/tlc59711_simpletest.py

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

# simple demo of the TLC59711 16-bit 12 channel LED PWM driver.
# Shows the minimal usage - how to set pixel values in a few ways.
# Author: Tony DiCola
```

(continues on next page)

(continued from previous page)

```

import board
import busio

import adafruit_tlc59711

print("tlc59711_simpletest.py")

# Define SPI bus connected to chip.
# You only need the clock and MOSI (output) line to use this chip.
spi = busio.SPI(board.SCK, MOSI=board.MOSI)
pixels = adafruit_tlc59711.TLC59711(spi, pixel_count=16)

# examples how to set the pixels:
# range:
# 0 - 65535
# or
# 0.0 - 1.0
# every pixel needs a color -
# give it just a list or tuple with 3 integer values: R G B

# set all pixels to a very low level
pixels.set_pixel_all((10, 10, 10))

# every chip has 4 Pixels (=RGB-LEDs = 12 Channel)
pixels[0] = (100, 100, 100)
pixels[1] = (0, 0, 100)
pixels[2] = (0.01, 0.0, 0.01)
pixels[3] = (0.1, 0.01, 0.0)
# if you are ready to show your values you have to call
pixels.show()

# there are a bunch of other ways to set pixel.
# have a look at the other examples.
```

6.6 Fastset test

Showcases the usage of set_pixel_16bit_value for fastest setting of values.

Listing 6: examples/tlc59711_fastset.py

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

# SPDX-FileCopyrightText: 2021 s-light
# SPDX-License-Identifier: MIT
# Author Stefan Krüger (s-light)

"""TLC5971 / TLC59711 Example."""

__doc__ = """
tlc59711_fastset.py - TLC59711 fast set example.

```

(continues on next page)

(continued from previous page)

```
showcases the usage of set_pixel_16bit_value for fastest setting of values.
for speed comparision of all the available set calls
look at the tlc59711_dev.py file.

Enjoy the colors :-)
"""


import time

import board
import busio

import adafruit_tlc59711


########################################
PIXEL_COUNT = 16

spi = busio.SPI(board.SCK, MOSI=board.MOSI)
pixels = adafruit_tlc59711.TLC59711(spi, pixel_count=PIXEL_COUNT)


########################################
# test function


def channelcheck_update_pixel(offset):
    """Channel check pixel."""
    # print("offset", offset)

    pixels.set_pixel_16bit_value(offset, 1000, 100, 0)
    # clear last pixel
    last = offset - 1
    if last < 0:
        last = PIXEL_COUNT - 1
    pixels.set_pixel_16bit_value(last, 0, 0, 1)
    pixels.show()

    offset += 1
    if offset >= PIXEL_COUNT:
        time.sleep(0.2)
        offset = 0
        print("clear")
        pixels.set_pixel_all_16bit_value(0, 1, 0)
        pixels.show()
    return offset


def test_main():
    """Test Main."""
    print(42 * "*", end="")
    print(__doc__, end="")
    print(42 * "*")

    bcvalues = adafruit_tlc59711.TLC59711.calculate_BCData(
        Ioclmax=18,
```

(continues on next page)

(continued from previous page)

```
        IoutR=18,
        IoutG=11,
        IoutB=13,
    )
    print("bcvalues = {}".format(bcvalues))
    pixels.bcr = bcvalues[0]
    pixels.bcg = bcvalues[1]
    pixels.bcb = bcvalues[2]
    pixels.update_BCData()
    pixels.show()

    offset = 0

    print("loop:")
    while True:
        offset = channelcheck_update_pixel(offset)
        time.sleep(0.2)


########################################
# main loop

if __name__ == "__main__":

    test_main()
```

6.7 adafruit_tlc59711

CircuitPython module for the TLC59711 or TLC5971 16-bit 12 channel LED PWM driver. See examples/tlc59711_simpletest.py for a demo of the usage.

- Author(s): Tony DiCola, Stefan Kruger

6.7.1 Implementation Notes

Hardware:

- Adafruit 12-Channel 16-bit PWM LED Driver - SPI Interface - TLC59711 (Product ID: 1455) or TLC5971

Software and Dependencies:

- **The API is mostly compatible to the DotStar / NeoPixel Libraries** and is therefore also compatible with FancyLED.
- Adafruit CircuitPython firmware for the supported boards: https://circuitpython.org/downloads

class adafruit_tlc59711.**TLC59711**(*spi*, *, *pixel_count=4*)

TLC5971 & TLC59711 16-bit 12 channel LED PWM driver.

The TLC59711 & TLC5971 chip is designed to drive 4 RGB LEDs with 16-bit PWM per Color. This Library can control 1..many chips. The class has an interface compatible with the FancyLED library - and the API 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/outout must be set, the MISO/input is unused. Maximal data clock frequence is: - TLC59711: 10MHz - TLC5971: 20MHz
- **pixel_count** (*bool*) – Number of RGB-LEDs (=Pixels) that are connected. (default=4)

static calculate_BCData(**, Ioclmax=18, IoutR=17, IoutG=15, IoutB=9*)
Calculate Global Brightness Control Values.

see: 8.5.1 Global Brightness Control (BC) Function (Sink Current Control) http://www.ti.com/lit/ds/symlink/tlc5971.pdf#page=19&zoom=200,0,697

Iout = Ioclmax * (BCX / 127) BCX = Iout / Ioclmax * 127

Parameters

- **Ioclmax** (*float*) – max output current set by Riref (mA) (default=20)
- **IoutR** (*float*) – max output current for red color group (mA) (default=9)
- **IoutG** (*float*) – max output current for green color (mA) (default=15)
- **IoutB** (*float*) – max output current for blue color (mA) (default=17)

Return tuple (bcr, bcg, bcb)

static calculate_Ioclmax(**, Riref=2.48*)
Calculate Maximum Constant Sink Current Value.

see: 8.4.1 Maximum Constant Sink Current Setting http://www.ti.com/lit/ds/symlink/tlc5971.pdf#page=18&zoom=160,0,524

Riref = (Viref / Ioclmax) * 41 Ioclmax = (41 / Riref) * Viref

Parameters **Riref** (*float*) – resistor value (kΩ) (default=20)

Return tuple Ioclmax (mA)

static calculate_Riref(**, Ioclmax=20*)
Calculate Maximum Constant Sink Current Value.

see: 8.4.1 Maximum Constant Sink Current Setting http://www.ti.com/lit/ds/symlink/tlc5971.pdf#page=19&zoom=200,0,697

Riref = (Viref / Ioclmax) * 41

Parameters **Ioclmax** (*float*) – target max output current (mA) (default=20)

Return tuple Riref (kΩ)

chip_set_BCData(*chip_index*, *bcr=127*, *bcg=127*, *bcb=127*)
Set BC-Data.

Parameters

- **chip_index** (*int*) – Index of Chip to set.
- **bcr** (*int*) – 7-bit value from 0-127 (default=127)
- **bcg** (*int*) – 7-bit value from 0-127 (default=127)
- **bcb** (*int*) – 7-bit value from 0-127 (default=127)

set_all_black()
Set all pixels to black.

set_channel(*channel_index*, *value*)
Set the value for the provided channel.

Parameters

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

set_chipheader_bits_in_buffer(*, *chip_index=0*, *part_bit_offset=0*, *field=None*, *value=0*)
Set chip header 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 the current LED PWM state to the chip.

update_BCData()
Update BC-Data for all Chips in Buffer.

need to be called after you changed on of the BC-Data Parameters. (bcr, bcg, bcb)

update_fc()
Update Function Control Bits for all Chips in Buffer.

need to be called after you changed on of the Function Control Bit Parameters. (outtmg, extgck, tmgrst, dsprpt, blank)

class adafruit_tlc59711.**TLC59711AutoShow**(*spi*, *pixel_count=4*)
TLC59711 16-bit 12 channel LED PWM driver with Auto-Show.

This chip is designed to drive 4 RGB LEDs with 16-bit PWM per Color. The class has an interface compatible with the FancyLED library. and with this is similar to the NeoPixel and DotStar Interfaces.

this TLC59711AutoShow is a subclass of TLC59711 that adds automatically sending changed data to the chips. this creates very slows responses on big changes. It is mainly usefull if you only have a very small number of pixels.

Parameters

- **spi** (*SPI*) – An instance of the SPI bus connected to the chip. The clock and MOSI/outout must be set, the MISO/input is unused. Maximal data clock frequence is: - TLC59711: 10MHz - TLC5971: 20MHz
- **pixel_count** (*bool*) – Number of RGB-LEDs (=Pixels) that are connected.

set_all_black()
Set all pixels to black.

set_channel(*channel_index*, *value*)
Set the value for the provided channel.

Parameters

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

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_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

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index