
AdafruitTLC59711 Library Documentation

Release 1.0

Tony DiCola

Sep 19, 2021

Contents

1	Dependencies	3
2	Installing from PyPI	5
3	Usage Example	7
4	Contributing	9
5	Documentation	11
6	Table of Contents	13
6.1	Simple test	13
6.2	FancyLED	14
6.3	Animations Example	15
6.4	Single Chip Autoshow	24
6.5	Brightness Correction Data	25
6.6	Fastset test	26
6.7	adafruit_tlc59711	28
6.7.1	Implementation Notes	28
7	Indices and tables	33
	Python Module Index	35
	Index	37

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](#).

6.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/tlc59711_simpletest.py

```
1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  # simple demo of the TLC59711 16-bit 12 channel LED PWM driver.
5  # Shows the minimal usage - how to set pixel values in a few ways.
6  # Author: Tony DiCola
7
8  import board
9  import busio
10
11 import adafruit_tlc59711
12
13 print("tlc59711_simpletest.py")
14
15 # Define SPI bus connected to chip.
16 # You only need the clock and MOSI (output) line to use this chip.
17 spi = busio.SPI(board.SCK, MOSI=board.MOSI)
18 pixels = adafruit_tlc59711.TLC59711(spi, pixel_count=16)
19
20 # examples how to set the pixels:
21 # range:
22 # 0 - 65535
23 # or
24 # 0.0 - 1.0
25 # every pixel needs a color -
26 # give it just a list or tuple with 3 integer values: R G B
27
```

(continues on next page)

(continued from previous page)

```

28 # set all pixels to a very low level
29 pixels.set_pixel_all((10, 10, 10))
30
31 # every chip has 4 Pixels (=RGB-LEDs = 12 Channel)
32 pixels[0] = (100, 100, 100)
33 pixels[1] = (0, 0, 100)
34 pixels[2] = (0.01, 0.0, 0.01)
35 pixels[3] = (0.1, 0.01, 0.0)
36 # if you are ready to show your values you have to call
37 pixels.show()
38
39 # there are a bunch of other ways to set pixel.
40 # 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

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  # CircuitPython
4
5  # SPDX-FileCopyrightText: 2021 s-light
6  # SPDX-License-Identifier: MIT
7  # Author Stefan Krüger (s-light)
8
9  """TLC59711 & FancyLED."""
10
11  __doc__ = """
12  TLC59711 & FancyLED.
13
14  this is an example for combining the TLC5957 library with FancyLED.
15  Enjoy the colors :-)
16  """
17
18  import board
19
20  import busio
21
22  import adafruit_fancyled.adafruit_fancyled as fancyled
23  import adafruit_tlc59711
24
25  #####
26  print("\n" + (42 * "*") + "\n" + __doc__ + "\n" + (42 * "*") + "\n" + "\n")
27
28  #####
29  # print(42 * "*")
30  # print("initialise digitalio pins for SPI")
31  # spi_clock = digitalio.DigitalInOut(board.SCK)
32  # spi_clock.direction = digitalio.Direction.OUTPUT
33  # spi_mosi = digitalio.DigitalInOut(board.MOSI)
34  # spi_mosi.direction = digitalio.Direction.OUTPUT

```

(continues on next page)

(continued from previous page)

```

35 # spi_miso = digitalio.DigitalInOut(board.MISO)
36 # spi_miso.direction = digitalio.Direction.INPUT
37
38 # print((42 * '*') + "\n" + "init busio.SPI")
39 spi = busio.SPI(board.SCK, MOSI=board.MOSI)
40
41 #####
42 print(42 * "*")
43 print("init TLC5957")
44 NUM_LEDS = 16
45 pixels = adafruit_tlc59711.TLC59711(
46     spi=spi,
47     pixel_count=NUM_LEDS,
48 )
49
50 print("pixel_count", pixels.pixel_count)
51 print("chip_count", pixels.chip_count)
52 print("channel_count", pixels.channel_count)
53
54
55 #####
56 # main loop
57 print(42 * "*")
58 print("rainbow loop")
59 hue_offset = 0
60 while True:
61     brightness = 0.8
62     color = fancyled.CHSV(hue_offset, 1.0, 1.0)
63     color = fancyled.gamma_adjust(color, brightness=brightness)
64     pixels.set_pixel_all(color)
65     pixels.show()
66
67     # Bigger number = faster spin
68     hue_offset += 0.000005
69     if hue_offset >= 1:
70         hue_offset = 0
71         print("hue_offset reset")

```

6.3 Animations Example

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

Listing 3: examples/tlc59711_dev.py

```

1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 # CircuitPython
4
5 # SPDX-FileCopyrightText: 2021 s-light
6 # SPDX-License-Identifier: MIT
7 # Author Stefan Krüger (s-light)
8
9 """TLC5971 / TLC59711 Multi Development."""
10

```

(continues on next page)

(continued from previous page)

```

11  __doc__ = """
12  TLC59711 development helper.
13
14  this sketch contains a bunch of timing tests and other development things..
15  Enjoy the colors :-)
16  """
17
18  import time
19
20  import board
21  import busio
22
23  import adafruit_tlc59711
24
25
26  #####
27  PIXEL_COUNT = 16 * 1
28
29  spi = busio.SPI(board.SCK, MOSI=board.MOSI)
30  pixels = adafruit_tlc59711.TLC59711(spi, pixel_count=PIXEL_COUNT)
31
32
33  #####
34  # test function
35
36  VALUE_HIGH = 1000
37
38
39  def channelcheck_update_pixel(offset):
40      """Channel check pixel."""
41      # print("offset", offset)
42
43      # pixels[offset] = (VALUE_HIGH, 0, 0)
44      pixels.set_pixel_16bit_value(offset, VALUE_HIGH, 0, 0)
45      # clear last pixel
46      last = offset - 1
47      if last < 0:
48          last = PIXEL_COUNT - 1
49      # pixels[last] = (0, 0, 1)
50      pixels.set_pixel_16bit_value(last, 0, 0, 1)
51      # pixels[offset] = (0xAAAA, 0xBBBB, 0xCCCC)
52      pixels.show()
53
54      offset += 1
55      if offset >= PIXEL_COUNT:
56          time.sleep(0.2)
57          offset = 0
58          print("clear")
59          pixels.set_pixel_all((0, 1, 0))
60          pixels.show()
61      return offset
62
63
64  def channelcheck_update(offset):
65      """Channel check."""
66      # print("offset", offset)
67

```

(continues on next page)

(continued from previous page)

```

68     pixels.set_channel(offset, VALUE_HIGH)
69     # clear last set channel
70     last = offset - 1
71     if last < 0:
72         last = pixels.channel_count - 1
73     pixels.set_channel(last, 0)
74     pixels.show()
75
76     offset += 1
77     if offset >= pixels.channel_count:
78         offset = 0
79         print("offset overflow. start from 0")
80     return offset
81
82
83 #####
84
85
86 def timeit_call(message, test_function, loop_count=1000):
87     """Measure timing."""
88     duration = 0
89     start_time = time.monotonic()
90     for _index in range(loop_count):
91         start_time = time.monotonic()
92         test_function()
93         end_time = time.monotonic()
94         duration += end_time - start_time
95     # print(
96     #     "duration:\n"
97     #     "\t{s} for {} loops\n"
98     #     "\t{:.2f}ms per call"
99     #     """.format(
100     #         duration,
101     #         loop_count,
102     #         (duration/loop_count)*1000
103     #     )
104     # )
105     # print(
106     #     "\t{:.2f}ms per call"
107     #     """.format((duration / loop_count) * 1000)
108     # )
109     # "{:>8.2f}ms".format(3.56)
110     print(
111         "{call_duration:>10.4f}ms\t{message}"
112         """.format(
113             call_duration=(duration / loop_count) * 1000,
114             message=message,
115         )
116     )
117
118
119 def timeit_pixels_show():
120     """Measure timing."""
121     print("*** pixels show:")
122     loop_count = 1000
123
124     def _test():

```

(continues on next page)

(continued from previous page)

```

125     pixels.show()
126
127     timeit_call("'pixels.show()'", _test, loop_count)
128
129
130 def timeit_pixels_set_single():
131     """Measure timing pixels set."""
132     print("*** pixels set single:")
133     loop_count = 1000
134
135     def _test():
136         pixels[3] = (500, 40500, 1000)
137
138     timeit_call("'pixels[3] = (500, 40500, 1000)'", _test, loop_count)
139
140     def _test():
141         pixels[3] = (0.1, 0.5, 0.9)
142
143     timeit_call("'pixels[3] = (0.1, 0.5, 0.9)'", _test, loop_count)
144
145     def _test():
146         pixels.set_pixel(3, (500, 40500, 1000))
147
148     timeit_call("'pixels.set_pixel(3, (500, 40500, 1000))'", _test, loop_count)
149
150     def _test():
151         pixels.set_pixel(3, (0.1, 0.5, 0.9))
152
153     timeit_call("'pixels.set_pixel(3, (0.1, 0.5, 0.9))'", _test, loop_count)
154
155
156 def timeit_pixels_set_loop():
157     """Measure timing pixels set."""
158     print("*** pixels set loop:")
159     loop_count = 10
160
161     def _test():
162         for i in range(PIXEL_COUNT):
163             pixels[i] = (500, 40500, 1000)
164
165     timeit_call(
166         "'pixels[0..{}] = (500, 40500, 1000)'.format(PIXEL_COUNT)", _test, loop_count
167     )
168
169     def _test():
170         for i in range(PIXEL_COUNT):
171             pixels[i] = (0.1, 0.5, 0.9)
172
173     timeit_call(
174         "'pixels[0..{}] = (0.1, 0.5, 0.9)'.format(PIXEL_COUNT)", _test, loop_count
175     )
176
177     def _test():
178         for i in range(PIXEL_COUNT):
179             pixels.set_pixel(i, (500, 40500, 1000))
180
181     timeit_call(

```

(continues on next page)

(continued from previous page)

```

182     'pixels.set_pixel(0..{}, (500, 40500, 1000))'.format(PIXEL_COUNT),
183     _test,
184     loop_count,
185 )
186
187 def _test():
188     for i in range(PIXEL_COUNT):
189         pixels.set_pixel(i, (0.1, 0.5, 0.9))
190
191 timeit_call(
192     'pixels.set_pixel(0..{}, (0.1, 0.5, 0.9))'.format(PIXEL_COUNT),
193     _test,
194     loop_count,
195 )
196
197
198 def timeit_pixels_set_all():
199     """Measure timing pixels set."""
200     print("*** pixels set all:")
201     loop_count = 10
202
203     def _test():
204         pixels.set_pixel_all((500, 40500, 1000))
205
206     timeit_call("'pixels.set_pixel_all((500, 40500, 1000))'", _test, loop_count)
207
208     def _test():
209         pixels.set_pixel_all((0.1, 0.5, 0.9))
210
211     timeit_call("'pixels.set_pixel_all((0.1, 0.5, 0.9))'", _test, loop_count)
212
213     def _test():
214         pixels.set_pixel_all_16bit_value(500, 40500, 1000)
215
216     timeit_call(
217         'pixels.set_pixel_all_16bit_value(500, 40500, 1000)', _test, loop_count
218     )
219
220     def _test():
221         pixels.set_all_black()
222
223     timeit_call("'pixels.set_all_black()'", _test, loop_count)
224
225
226 def timeit_pixels_set_16bit():
227     """Measure timing pixels set."""
228     print("*** pixels set 16bit:")
229     loop_count = 1000
230
231     def _test():
232         pixels.set_pixel_16bit_value(3, 500, 40500, 1000)
233
234     timeit_call(
235         'pixels.set_pixel_16bit_value(3, 500, 40500, 1000)', _test, loop_count
236     )
237
238     def _test():

```

(continues on next page)

(continued from previous page)

```

239     pixels.set_pixel_16bit_color(3, (500, 40500, 1000))
240
241     timeit_call(
242         "'pixels.set_pixel_16bit_color(3, (500, 40500, 1000))'", _test, loop_count
243     )
244
245     def _test():
246         for i in range(PIXEL_COUNT):
247             pixels.set_pixel_16bit_value(i, 500, 40500, 1000)
248
249     timeit_call(
250         "'pixels.set_pixel_16bit_value(0..{}), 500, 40500, 1000)'"
251         ".format(PIXEL_COUNT),
252         _test,
253         10,
254     )
255
256     def _test():
257         for i in range(PIXEL_COUNT):
258             pixels.set_pixel_16bit_color(i, (500, 40500, 1000))
259
260     timeit_call(
261         "'pixels.set_pixel_16bit_color(0..{}), (500, 40500, 1000))'"
262         ".format(PIXEL_COUNT),
263         _test,
264         10,
265     )
266
267
268     def timeit_pixels_set_float():
269         """Measure timing pixels set."""
270         print("*** pixels set float:")
271         loop_count = 1000
272
273         def _test():
274             pixels.set_pixel_float_value(3, 0.1, 0.5, 0.9)
275
276         timeit_call("'pixels.set_pixel_float_value(3, 0.1, 0.5, 0.9)'", _test, loop_count)
277
278         def _test():
279             pixels.set_pixel_float_color(3, (0.1, 0.5, 0.9))
280
281         timeit_call("'pixels.set_pixel_float_color(3, (0.1, 0.5, 0.9))'", _test, loop_
282         ↪count)
283
284         def _test():
285             for i in range(PIXEL_COUNT):
286                 pixels.set_pixel_float_value(i, 0.1, 0.5, 0.9)
287
288         timeit_call(
289             "'pixels.set_pixel_float_value(0..{}), 0.1, 0.5, 0.9)'" ".format(PIXEL_COUNT),
290             _test,
291             10,
292         )
293
294         def _test():
295             for i in range(PIXEL_COUNT):

```

(continues on next page)

(continued from previous page)

```

295         pixels.set_pixel_float_color(i, (0.1, 0.5, 0.9))
296
297     timeit_call(
298         "'pixels.set_pixel_float_color(0..{}), (0.1, 0.5, 0.9))' ".format(PIXEL_
299 ←COUNT),
300         _test,
301         10,
302     )
303
304     def _test():
305         for i in range(PIXEL_COUNT):
306             pixels.set_pixel_16bit_value(
307                 i, int(0.1 * 65535), int(0.5 * 65535), int(0.9 * 65535)
308             )
309
310     timeit_call(
311         "'pixels.set_pixel_16bit_value(0..{}), f2i 0.1, f2i 0.5, f2i 0.9)' ".
312         .format(PIXEL_COUNT),
313         _test,
314         10,
315     )
316
317 def timeit_channel_set():
318     """Measure timing channel set."""
319     print("*** channel set:")
320     loop_count = 1000
321
322     def _test():
323         pixels.set_channel(0, 10000)
324
325     timeit_call("'set_channel(0, 10000)'", _test, loop_count)
326
327     def _test():
328         pixels.set_channel(0, 10000)
329         pixels.set_channel(1, 10000)
330         pixels.set_channel(2, 10000)
331
332     timeit_call("'set_channel(0..2, 10000)'", _test, loop_count)
333
334     channel_count = PIXEL_COUNT * 3
335
336     def _test():
337         for i in range(channel_count):
338             pixels.set_channel(i, 500)
339
340     timeit_call("'set_channel(for 0..{}, 10000)' ".format(channel_count), _test, 10)
341
342
343 def timeit_channel_set_internal():
344     """Measure timing channel set internal."""
345     print("*** channel set internal:")
346     # loop_count = 1000
347     #
348     # def _test():
349     #     pixels._set_channel_16bit_value(0, 10000)
350     # timeit_call(

```

(continues on next page)

(continued from previous page)

```

351 #     "'_set_channel_16bit_value(0, 10000) '",
352 #     _test,
353 #     loop_count
354 # )
355 #
356 # def _test():
357 #     pixels._set_channel_16bit_value(0, 10000)
358 #     pixels._set_channel_16bit_value(1, 10000)
359 #     pixels._set_channel_16bit_value(2, 10000)
360 #     timeit_call(
361 #         "'_set_channel_16bit_value(0..2, 10000) '",
362 #         _test,
363 #         loop_count
364 #     )
365 #
366 # def _test():
367 #     for i in range(PIXEL_COUNT * 3):
368 #         pixels._set_channel_16bit_value(i, 500)
369 #     timeit_call(
370 #         "'_set_channel_16bit_value(for 0..{}), 10000) '",
371 #         "{}.format(PIXEL_COUNT * 3),
372 #         _test,
373 #         10
374 #     )
375 print("    must be uncommented in code to work..")
376
377
378 def timeit_pixels_get():
379     """Measure timing pixels get."""
380     print("*** pixels get:")
381
382     pixels.set_pixel_all((1, 11, 111))
383
384     def _test():
385         print("[", end="")
386         for i in range(PIXEL_COUNT):
387             print("{}:{}", ".format(i, pixels[i]), end="")
388         print("]")
389
390     timeit_call("'print('{}:{}'.format(i, pixels[i]), end='')'", _test, 1)
391
392
393 def time_measurement():
394     """Measure timing."""
395     print("meassure timing:")
396     timeit_pixels_show()
397     timeit_pixels_set_single()
398     timeit_pixels_set_loop()
399     timeit_pixels_set_all()
400     timeit_pixels_set_16bit()
401     timeit_pixels_set_float()
402     timeit_channel_set()
403     timeit_channel_set_internal()
404     timeit_pixels_get()
405     pixels.set_pixel_all((0, 1, 1))
406
407

```

(continues on next page)

(continued from previous page)

```

408 #####
409
410
411 def test_bcdata():
412     """Test BC-Data setting."""
413     print("test BC-Data setting:")
414     print("set pixel all to 100, 100, 100")
415     pixels.set_pixel_all((100, 100, 100))
416     pixels.show()
417     time.sleep(2)
418     print(
419         "bcr: {:>3}\n"
420         "bcg: {:>3}\n"
421         "bcb: {:>3}\n"
422         "".format(
423             pixels.bcr,
424             pixels.bcg,
425             pixels.bcb,
426         )
427     )
428     # calculate bc values
429     Ioclmx = adafruit_tlc59711.TLC59711.calculate_Ioclmx(Riref=2.7)
430     print("Ioclmx = {}".format(Ioclmx))
431     Riref = adafruit_tlc59711.TLC59711.calculate_Riref(Ioclmx=Ioclmx)
432     print("Riref = {}".format(Riref))
433     BCValues = adafruit_tlc59711.TLC59711.calculate_BCData(
434         Ioclmx=Ioclmx,
435         IoutR=18,
436         IoutG=11,
437         IoutB=13,
438     )
439     # (127, 77, 91)
440     print("BCValues = {}".format(BCValues))
441
442     print("set bcX")
443     pixels.bcr = BCValues[0]
444     pixels.bcg = BCValues[1]
445     pixels.bcb = BCValues[2]
446     pixels.update_BCData()
447     pixels.show()
448     print(
449         "bcr: {:>3}\n"
450         "bcg: {:>3}\n"
451         "bcb: {:>3}\n"
452         "".format(
453             pixels.bcr,
454             pixels.bcg,
455             pixels.bcb,
456         )
457     )
458     time.sleep(2)
459
460 #####
461
462
463
464 def test_main():

```

(continues on next page)

(continued from previous page)

```

465     """Test Main."""
466     print(42 * "*", end="")
467     print(__doc__, end="")
468     print(42 * "*")
469     # print()
470     # time.sleep(0.5)
471     # print(42 * '*')
472
473     pixels.set_pixel_all_16bit_value(1, 10, 100)
474     pixels.show()
475     time.sleep(0.5)
476
477     test_bcddata()
478     time.sleep(0.5)
479     print(42 * "*")
480
481     time_measurement()
482     time.sleep(0.5)
483     print(42 * "*")
484
485     offset = 0
486
487     print("loop:")
488     while True:
489         offset = channelcheck_update(offset)
490         time.sleep(0.5)
491         print(offset)
492
493
494     #####
495     # main loop
496
497     if __name__ == "__main__":
498
499         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

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  # CircuitPython
4
5  # SPDX-FileCopyrightText: 2021 s-light
6  # SPDX-License-Identifier: MIT
7  # Author Stefan Krüger (s-light)
8
9  """TLC5971 / TLC59711 Example."""
10
11  __doc__ = """
12  tlc59711_singlechip_autoshow.py - TLC59711AutoShow minimal usage example.

```

(continues on next page)

(continued from previous page)

```

13
14 simple demo of the TLC59711 16-bit 12 channel LED PWM driver.
15 Shows the minimal usage - how to set pixel values.
16 the TLC59711AutoShow class automatically writes the pixel values on each change.
17 this makes it very slow on lots of pixel changes but is convenient for only a handful_
  ↳ of pixels..
18
19 Author: Tony DiCola, Stefan Krueger
20
21 Enjoy the colors :-)
22 """
23
24 import board
25 import busio
26
27 import adafruit_tlc59711
28
29 print(__doc__)
30
31 # Define SPI bus connected to chip.
32 # You only need the clock and MOSI (output) line to use this chip.
33 spi = busio.SPI(board.SCK, MOSI=board.MOSI)
34 pixels = adafruit_tlc59711.TLC59711AutoShow(spi)
35
36 # Ways to set the values:
37 # just a list or tuple with 3 integer values: R G B
38 # each 0 - 65535 or 0.0 - 1.0
39 # every chip has 4 RGB-LEDs (=12 Channel)
40 pixels[0] = (100, 100, 10111)
41 pixels[1] = (0, 0, 100)
42 pixels[2] = (0.01, 0.0, 0.01)
43 pixels[3] = (0.1, 0.01, 0.0)
44
45 # You can also explicitly control each R0, G0, B0, R1, B1, etc. channel of the first_
  ↳ ic
46 # by getting and setting its 16-bit value directly with properties.
47 # For example set channel 2 to 1/4 green (i.e. G2):
48 pixels.g2 = 65535 // 4
49
50 # there are a bunch of other advanced ways to set pixel.
51 # have a look at the other examples.

```

6.5 Brightness Correction Data

Test brightness correction data (BC)

Listing 5: examples/tlc59711_simpletest.py

```

1 # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2 # SPDX-License-Identifier: MIT
3
4 # simple demo of the TLC59711 16-bit 12 channel LED PWM driver.
5 # Shows the minimal usage - how to set pixel values in a few ways.
6 # Author: Tony DiCola

```

(continues on next page)

(continued from previous page)

```

7
8 import board
9 import busio
10
11 import adafruit_tlc59711
12
13 print("tlc59711_simpletest.py")
14
15 # Define SPI bus connected to chip.
16 # You only need the clock and MOSI (output) line to use this chip.
17 spi = busio.SPI(board.SCK, MOSI=board.MOSI)
18 pixels = adafruit_tlc59711.TLC59711(spi, pixel_count=16)
19
20 # examples how to set the pixels:
21 # range:
22 # 0 - 65535
23 # or
24 # 0.0 - 1.0
25 # every pixel needs a color -
26 # give it just a list or tuple with 3 integer values: R G B
27
28 # set all pixels to a very low level
29 pixels.set_pixel_all((10, 10, 10))
30
31 # every chip has 4 Pixels (=RGB-LEDs = 12 Channel)
32 pixels[0] = (100, 100, 100)
33 pixels[1] = (0, 0, 100)
34 pixels[2] = (0.01, 0.0, 0.01)
35 pixels[3] = (0.1, 0.01, 0.0)
36 # if you are ready to show your values you have to call
37 pixels.show()
38
39 # there are a bunch of other ways to set pixel.
40 # 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

```

1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 # CircuitPython
4
5 # SPDX-FileCopyrightText: 2021 s-light
6 # SPDX-License-Identifier: MIT
7 # Author Stefan Krüger (s-light)
8
9 """TLC5971 / TLC59711 Example."""
10
11 __doc__ = """
12 tlc59711_fastset.py - TLC59711 fast set example.
13

```

(continues on next page)

(continued from previous page)

```

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

```

```

17
18 Enjoy the colors :-)
19 """
20
21
22 import time
23
24 import board
25 import busio
26
27 import adafruit_tlc59711
28
29
30 #####
31 PIXEL_COUNT = 16
32
33 spi = busio.SPI(board.SCK, MOSI=board.MOSI)
34 pixels = adafruit_tlc59711.TLC59711(spi, pixel_count=PIXEL_COUNT)
35
36
37 #####
38 # test function
39
40
41 def channelcheck_update_pixel(offset):
42     """Channel check pixel."""
43     # print("offset", offset)
44
45     pixels.set_pixel_16bit_value(offset, 1000, 100, 0)
46     # clear last pixel
47     last = offset - 1
48     if last < 0:
49         last = PIXEL_COUNT - 1
50     pixels.set_pixel_16bit_value(last, 0, 0, 1)
51     pixels.show()
52
53     offset += 1
54     if offset >= PIXEL_COUNT:
55         time.sleep(0.2)
56         offset = 0
57         print("clear")
58         pixels.set_pixel_all_16bit_value(0, 1, 0)
59         pixels.show()
60     return offset
61
62
63 def test_main():
64     """Test Main."""
65     print(42 * "*", end="")
66     print(__doc__, end="")
67     print(42 * "*")
68
69     bcvalues = adafruit_tlc59711.TLC59711.calculate_BCData(
70         Io1max=18,

```

(continues on next page)

(continued from previous page)

```

71         IoutR=18,
72         IoutG=11,
73         IoutB=13,
74     )
75     print("bcvalues = {}".format(bcvalues))
76     pixels.bcr = bcvalues[0]
77     pixels.bcg = bcvalues[1]
78     pixels.bcb = bcvalues[2]
79     pixels.update_BCData()
80     pixels.show()
81
82     offset = 0
83
84     print("loop:")
85     while True:
86         offset = channelcheck_update_pixel(offset)
87         time.sleep(0.2)
88
89 #####
90 # main loop
91
92 if __name__ == "__main__":
93     test_main()
94
95

```

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 frequency is: - TLC59711: 10MHz - TLC5971: 20MHz
- **pixel_count** (*bool*) – Number of RGB-LEDs (=Pixels) that are connected. (default=4)

static calculate_BCData (*, *IocI*max=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>

$I_{out} = I_{ocI}max * (BCX / 127)$ $BCX = I_{out} / I_{ocI}max * 127$

Parameters

- **IocI**max (*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_IocImax (*, *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>

$Riref = (Viref / I_{ocI}max) * 41$ $I_{ocI}max = (41 / Riref) * Viref$

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

Return tuple *IocI*max (mA)

static calculate_Riref (*, *IocI*max=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>

$Riref = (Viref / I_{ocI}max) * 41$

Parameters IocImax (*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 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. it's 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 frequency 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 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_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`

a

adafruit_tlc59711, [28](#)

A

adafruit_tlc59711 (*module*), 28

C

calculate_BCData() (*adafruit_tlc59711.TLC59711 static method*), 29

calculate_Ioc1max() (*adafruit_tlc59711.TLC59711 static method*), 29

calculate_Riref() (*adafruit_tlc59711.TLC59711 static method*), 29

chip_set_BCData() (*adafruit_tlc59711.TLC59711 method*), 29

S

set_all_black() (*adafruit_tlc59711.TLC59711 method*), 29

set_all_black() (*adafruit_tlc59711.TLC59711AutoShow method*), 31

set_channel() (*adafruit_tlc59711.TLC59711 method*), 29

set_channel() (*adafruit_tlc59711.TLC59711AutoShow method*), 31

set_chipheader_bits_in_buffer() (*adafruit_tlc59711.TLC59711 method*), 30

set_pixel() (*adafruit_tlc59711.TLC59711 method*), 30

set_pixel() (*adafruit_tlc59711.TLC59711AutoShow method*), 32

set_pixel_16bit_color() (*adafruit_tlc59711.TLC59711 method*), 30

set_pixel_16bit_value() (*adafruit_tlc59711.TLC59711 method*), 30

set_pixel_all() (*adafruit_tlc59711.TLC59711 method*), 30

set_pixel_all() (*adafruit_tlc59711.TLC59711AutoShow method*), 32

set_pixel_all_16bit_value() (*adafruit_tlc59711.TLC59711 method*), 30

set_pixel_float_color() (*adafruit_tlc59711.TLC59711 method*), 30

set_pixel_float_value() (*adafruit_tlc59711.TLC59711 method*), 31

show() (*adafruit_tlc59711.TLC59711 method*), 31

T

TLC59711 (*class in adafruit_tlc59711*), 28

TLC59711AutoShow (*class in adafruit_tlc59711*), 31

U

update_BCData() (*adafruit_tlc59711.TLC59711 method*), 31

update_fc() (*adafruit_tlc59711.TLC59711 method*), 31