
AdafruitSI5351 Library Documentation

Release 1.0

Tony DiCola

May 31, 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	adafruit_si5351	15
7	Indices and tables	17
	Python Module Index	19
	Index	21

SI5351 clock generator module.

CHAPTER 1

Dependencies

This driver depends on:

- [Adafruit CircuitPython](#)
- [Bus Device](#)

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

CHAPTER 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-si5351
```

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

```
sudo pip3 install adafruit-circuitpython-si5351
```

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-si5351
```


CHAPTER 3

Usage Example

See `examples/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/si5351_simpletest.py

```
1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  # Simple demo of the SI5351 clock generator.
5  # This is like the Arduino library example:
6  #   https://github.com/adafruit/Adafruit_SI5351_Library/blob/master/examples/si5351/
   ↪ si5351.ino
7  # Which will configure the chip with:
8  #   - PLL A at 900mhz
9  #   - PLL B at 616.66667mhz
10 #   - Clock 0 at 112.5mhz, using PLL A as a source divided by 8
11 #   - Clock 1 at 13.553115mhz, using PLL B as a source divided by 45.5
12 #   - Clock 2 at 10.76khz, using PLL B as a source divided by 900 and further
13 #     divided with an R divider of 64.
14 import board
15 import busio
16
17 import adafruit_si5351
18
19
20 # Initialize I2C bus.
21 i2c = busio.I2C(board.SCL, board.SDA)
22
23 # Initialize SI5351.
24 si5351 = adafruit_si5351.SI5351(i2c)
25 # Alternatively you can specify the I2C address if it has been changed:
26 # si5351 = adafruit_si5351.SI5351(i2c, address=0x61)
```

(continues on next page)

(continued from previous page)

```

27
28 # Now configure the PLLs and clock outputs.
29 # The PLLs can be configured with a multiplier and division of the on-board
30 # 25mhz reference crystal. For example configure PLL A to 900mhz by multiplying
31 # by 36. This uses an integer multiplier which is more accurate over time
32 # but allows less of a range of frequencies compared to a fractional
33 # multiplier shown next.
34 si5351.pll_a.configure_integer(36) # Multiply 25mhz by 36
35 print("PLL A frequency: {0}mhz".format(si5351.pll_a.frequency / 1000000))
36
37 # And next configure PLL B to 616.6667mhz by multiplying 25mhz by 24.667 using
38 # the fractional multiplier configuration. Notice you specify the integer
39 # multiplier and then a numerator and denominator as separate values, i.e.
40 # numerator 2 and denominator 3 means 2/3 or 0.667. This fractional
41 # configuration is susceptible to some jitter over time but can set a larger
42 # range of frequencies.
43 si5351.pll_b.configure_fractional(24, 2, 3) # Multiply 25mhz by 24.667 (24 2/3)
44 print("PLL B frequency: {0}mhz".format(si5351.pll_b.frequency / 1000000))
45
46 # Now configure the clock outputs. Each is driven by a PLL frequency as input
47 # and then further divides that down to a specific frequency.
48 # Configure clock 0 output to be driven by PLL A divided by 8, so an output
49 # of 112.5mhz (900mhz / 8). Again this uses the most precise integer division
50 # but can't set as wide a range of values.
51 si5351.clock_0.configure_integer(si5351.pll_a, 8)
52 print("Clock 0: {0}mhz".format(si5351.clock_0.frequency / 1000000))
53
54 # Next configure clock 1 to be driven by PLL B divided by 45.5 to get
55 # 13.5531mhz (616.6667mhz / 45.5). This uses fractional division and again
56 # notice the numerator and denominator are explicitly specified. This is less
57 # precise but allows a large range of frequencies.
58 si5351.clock_1.configure_fractional(si5351.pll_b, 45, 1, 2) # Divide by 45.5 (45 1/2)
59 print("Clock 1: {0}mhz".format(si5351.clock_1.frequency / 1000000))
60
61 # Finally configure clock 2 to be driven by PLL B divided once by 900 to get
62 # down to 685.15 khz and then further divided by a special R divider that
63 # divides 685.15 khz by 64 to get a final output of 10.706khz.
64 si5351.clock_2.configure_integer(si5351.pll_b, 900)
65 # Set the R divider, this can be a value of:
66 # - R_DIV_1: divider of 1
67 # - R_DIV_2: divider of 2
68 # - R_DIV_4: divider of 4
69 # - R_DIV_8: divider of 8
70 # - R_DIV_16: divider of 16
71 # - R_DIV_32: divider of 32
72 # - R_DIV_64: divider of 64
73 # - R_DIV_128: divider of 128
74 si5351.clock_2.r_divider = adafruit_si5351.R_DIV_64
75 print("Clock 2: {0}khz".format(si5351.clock_2.frequency / 1000))
76
77 # After configuring PLLs and clocks, enable the outputs.
78 si5351.outputs_enabled = True
79 # You can disable them by setting false.

```

6.2 adafruit_si5351

CircuitPython module to control the SI5351 clock generator. See `examples/simpletest.py` for a demo of the usage. This is based on the Arduino library at: https://github.com/adafruit/Adafruit_Si5351_Library/

- Author(s): Tony DiCola

```
class adafruit_si5351.SI5351(i2c, *, address=96)
```

SI5351 clock generator. Initialize this class by specifying:

- `i2c`: The I2C bus connected to the chip.

Optionally specify:

- `address`: The I2C address of the device if it differs from the default.

`outputs_enabled`

Get and set the enabled state of all clock outputs as a boolean. If true then all clock outputs are enabled, and if false then they are all disabled.

CHAPTER 7

Indices and tables

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

a

adafruit_si5351, [14](#)

A

`adafruit_si5351` (*module*), [14](#)

O

`outputs_enabled` (*adafruit_si5351.SI5351* attribute), [15](#)

S

`SI5351` (*class in adafruit_si5351*), [15](#)