# AdafruitSI5351 Library Documentation

*Release 1.0*

**Tony DiCola**

**Jul 28, 2019**

# Contents

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

# Usage Example

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

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 adafruit-circuitpython-si5351 --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/si5351_simpletest.py

```
# Simple demo of the SI5351 clock generator.
# This is like the Arduino library example:
#   https://github.com/adafruit/Adafruit_Si5351_Library/blob/master/examples/si5351/
↪si5351.ino
# Which will configure the chip with:
#  - PLL A at 900mhz
#  - PLL B at 616.66667mhz
#  - Clock 0 at 112.5mhz, using PLL A as a source divided by 8
#  - Clock 1 at 13.553115mhz, using PLL B as a source divided by 45.5
#  - Clock 2 at 10.76khz, using PLL B as a source divided by 900 and further
#    divided with an R divider of 64.
import board
import busio

import adafruit_si5351


# Initialize I2C bus.
i2c = busio.I2C(board.SCL, board.SDA)

# Initialize SI5351.
si5351 = adafruit_si5351.SI5351(i2c)
# Alternatively you can specify the I2C address if it has been changed:
#si5351 = adafruit_si5351.SI5351(i2c, address=0x61)

# Now configue the PLLs and clock outputs.
# The PLLs can be configured with a multiplier and division of the on-board
```

(continues on next page)

(continued from previous page)

```
# 25mhz reference crystal.  For example configure PLL A to 900mhz by multiplying
# by 36.  This uses an integer multiplier which is more accurate over time
# but allows less of a range of frequencies compared to a fractional
# multiplier shown next.
si5351.pll_a.configure_integer(36)  # Multiply 25mhz by 36
print('PLL A frequency: {0}mhz'.format(si5351.pll_a.frequency/1000000))

# And next configure PLL B to 616.6667mhz by multiplying 25mhz by 24.667 using
# the fractional multiplier configuration.  Notice you specify the integer
# multiplier and then a numerator and denominator as separate values, i.e.
# numerator 2 and denominator 3 means 2/3 or 0.667.  This fractional
# configuration is susceptible to some jitter over time but can set a larger
# range of frequencies.
si5351.pll_b.configure_fractional(24, 2, 3)  # Multiply 25mhz by 24.667 (24 2/3)
print('PLL B frequency: {0}mhz'.format(si5351.pll_b.frequency/1000000))

# Now configure the clock outputs.  Each is driven by a PLL frequency as input
# and then further divides that down to a specific frequency.
# Configure clock 0 output to be driven by PLL A divided by 8, so an output
# of 112.5mhz (900mhz / 8).  Again this uses the most precise integer division
# but can't set as wide a range of values.
si5351.clock_0.configure_integer(si5351.pll_a, 8)
print('Clock 0: {0}mhz'.format(si5351.clock_0.frequency/1000000))

# Next configure clock 1 to be driven by PLL B divided by 45.5 to get
# 13.5531mhz (616.6667mhz / 45.5).  This uses fractional division and again
# notice the numerator and denominator are explicitly specified.  This is less
# precise but allows a large range of frequencies.
si5351.clock_1.configure_fractional(si5351.pll_b, 45, 1, 2) # Divide by 45.5 (45 1/2)
print('Clock 1: {0}mhz'.format(si5351.clock_1.frequency/1000000))

# Finally configure clock 2 to be driven by PLL B divided once by 900 to get
# down to 685.15 khz and then further divided by a special R divider that
# divides 685.15 khz by 64 to get a final output of 10.706khz.
si5351.clock_2.configure_integer(si5351.pll_b, 900)
# Set the R divider, this can be a value of:
#  - R_DIV_1: divider of 1
#  - R_DIV_2: divider of 2
#  - R_DIV_4: divider of 4
#  - R_DIV_8: divider of 8
#  - R_DIV_16: divider of 16
#  - R_DIV_32: divider of 32
#  - R_DIV_64: divider of 64
#  - R_DIV_128: divider of 128
si5351.clock_2.r_divider = adafruit_si5351.R_DIV_64
print('Clock 2: {0}khz'.format(si5351.clock_2.frequency/1000))

# After configuring PLLs and clocks, enable the outputs.
si5351.outputs_enabled = True
# You can disable them by setting false.
```

## 5.2 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 6

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

## a

# Index

## A



## O



## S