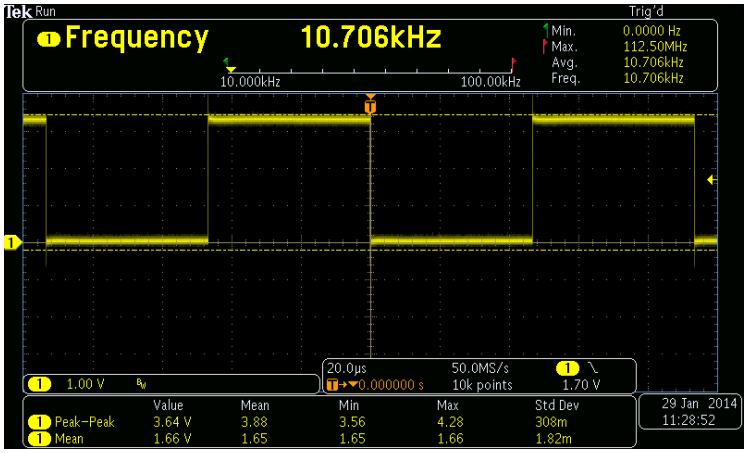


That's it! If you want to change the frequencies, adjust the example sketch and re-upload.

Library Reference

The library we have is simple and easy to use

You can create the **Adafruit_Si5351** object with:
Download file
Copy Code

```
1. Adafruit_SI5351 clockgen = Adafruit_SI5351();
```

I2C does not have pins, as they are fixed in hardware.

Begin!

To initialize the chip, call **clockgen.begin()** which will check that it can be found. Begin() returns true/false depending on these checks. We suggest you wrap begin() in a statement that will check if the chip was located:
Download file
Copy Code

```
1.   if (clockgen.begin() != ERROR_NONE)
2.   {
3.     /* There was a problem detecting the IC ... check your connections */
4.     Serial.print("Ooops, no Si5351 detected ... Check your wiring or I2C ADDR!");
5.     while(1);
6.   }
```

Set up the PLL

The chip uses two subsections to generate clock outputs. First it **multiplies** the 25MHz reference clock by some amount (setting up the PLL), then it **divides** that new clock by some other amount (setting up the clock divider)

By noodling with the multiplier and divider you can generate just about any clock frequency!

There are **two** PLL multipliers (A and B), so if you want to have three outputs, two outputs will have to share one PLL.

Set up the PLL with 'integer mode'

The cleanest way to run the PLL is to do a straight up integer multiplication:
Download file
Copy Code

```
1. clockgen.setupPLLInt(SI5351_PLL_A or SI5351_PLL_B, m);
```

This sets **PLL_A** or **PLL_B** to be **25MHz * m** and **m** (the integer multipler) can range from **15** to **90**!

Set up the PLL with 'fractional mode'

This mode allows a much more flexible PLL setting by using fractional multipliers for the PLL setup, however, the output may have a slight amount of jitter so if possible, try to use integer mode!
Download file
Copy Code

```
1. clockgen.setupPLLInt(SI5351_PLL_A or SI5351_PLL_B, m, n, d);
```

This sets **PLL_A** or **PLL_B** to be **25MHz * (m** + **n/d)**

- **m** (the integer multipler) can range from **15** to **90**
- **n** (the numerator) can range from **0 to 1,048,575**

- **d** (the denominator) can range from **1** to **1,048,575**

Set up the clock divider

Once you have the PLLs set up, you can now divide that high frequency down to get the number you want for the output

Each output has its own divider. You can use the cleaner Integer-only divider:
Download file
Copy Code

```
clockgen.setupMultisynthInt(output, SI5351_PLL_x, SI5351_MULTISYNTH_DIV_x);
```

- For the **output** use 0, 1 or 2
- For the PLL input, use either **SI5351_PLL_A** or **SI5351_PLL_B**
- For the divider, you can divide by **SI5351_MULTISYNTH_DIV_4**, **SI5351_MULTISYNTH_DIV_6**, or **SI5351_MULTISYNTH_DIV_8**

Again, integer output will give you the cleanest clock. If you need more flexibility, use the fractional generator/divider:
Download file
Copy Code

```
clockgen.setupMultisynth(output, SI5351_PLL_x, div, n, d);
```

- For the **output** use 0, 1 or 2
- For the PLL input, use either **SI5351_PLL_A** or **SI5351_PLL_B**
- The final frequency is equal to the **PLL / (div + n/d)**
- **div** can range from **4** to **900**
- **n** can range from **0 to 1,048,575**
- **d** can range from **1 to 1,048,575**

Additional R Divider

If you need to divide even more, to get to the < 100 KHz frequencies, there's an additional R divider, that divides the output once more by a fixed number:
Download file
Copy Code

```
clockgen.setupRdiv(output, SI5351_R_DIV_x);
```

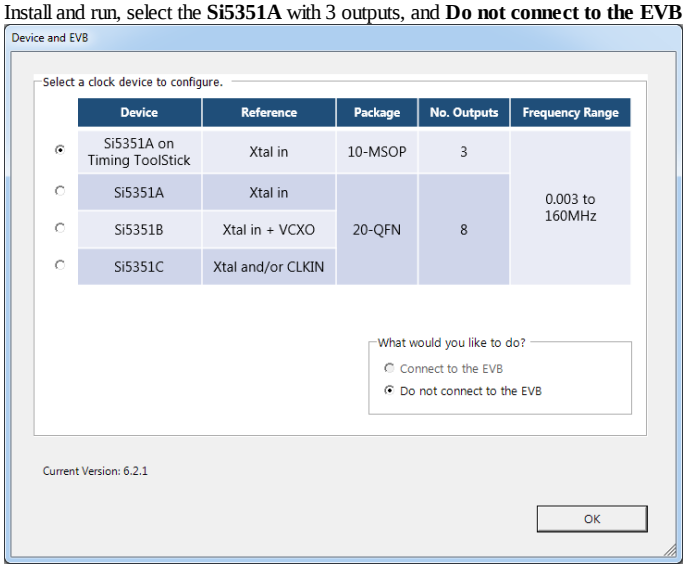
output is the clock output #
The R divider can be any of the following:

- SI5351_R_DIV_1
- SI5351_R_DIV_2
- SI5351_R_DIV_4
- SI5351_R_DIV_8
- SI5351_R_DIV_16
- SI5351_R_DIV_32
- SI5351_R_DIV_64
- SI5351_R_DIV_128

Software

As you can see, the annoying part here is figuring out the best choice for PLL multipler & divider! SiLabs has a desktop application called ClockBuilder that can do some calculation of the PLL divider/multiplier for you. It's windows only, but you only need to use it once for calculation.

Install and run, select the **Si5351A** with 3 outputs, and **Do not connect to the EVB**



Enable the output you want, and set the frequency as floating point or fraction