SI5351 driver User Guide

This code controls the frequency of the two independent channels CLK0 and CLK1 of an SI5351 chip over the range 8kHz to 120MHz. The chip is popular with Radio Ham constructors, as it replaces drift-prone LC oscillators and bandswitching in many applications.

N.B. The SI5351 **must be powered from a 3.3V power source**, such as provided by the ESP32, to avoid damage to the GPIO pins used for I2C control.

The user commands are as follows:-

Wire.start (sdapin sclpin –) Set up the I2C interface and define the GPIO pins used for communication with the SI5351. Called only once in an application before any of the following commands.

The following words are located in the **SI5351** vocabulary:-

Rfinit (--) Initialise the SI5351 ready for use. Turns output pins CLK0 and CLK1 off (no output pulses) Must be called before any other commands listed below.
CLK0 (--) Select the output from pin CLK0 as the currently selected channel - all commands will go to that channel until the other channel is selected
CLK1 (--) Select the output from pin CLK1 as the currently selected channel - all commands will go to that channel until the other channel is selected
Rftune (frequency --) Set the currently selected channel to 'frequency' Hz. This has been tested over the frequency range 8000 to 120000000 Hz
Rfon (--) Turn the currently selected channel on, approx 3.3V high pulses will appear at the output
Rfoff (--) Turn the currently selected channel off (0V output)

N.B. A value will need changing, unique to each SI5351, for best frequency accuracy.
fref needs changing to suit the clock frequency driving your SI5351. The 25MHz crystal oscillator is not trimmed exactly on frequency. This must be done whilst monitoring either CLK0 or CLK1 output with a frequency counter or calibrated receiver. e.g. This code would set fref to 25.008325MHz and set the SI5351 output to 7MHz for measurement, repeat until actually on frequency:-

25008325 to fref
7000000 RFtune

There are some optional test words at the bottom of the source code to check the data being generated and read the SI5351 settings. During development, the application note AN619 from SI was useful, as was this website. Referring to the SI5351 diagram on the latter website, my code holds three of the frequency setting parameters constant - e=0, f=1 and c = 1048575, to simplify the maths. The value of c is the largest permitted and this allows b to vary over the widest range

and thus give the smallest tuning resolution. The OMD division ratio is maintained as an even integer which is said to minimise clock jitter. The code only sends changed bytes, when the demanded frequency changes. All these measures are to improve tuning speed, when controlled by a rotary encoder, so as to sound as smooth as possible - radio enthusiasts spend a lot of time hunting for weak signals, so any notchy tuning artifact would be unwelcome. I've swept the output in frequency using test word **RFsweep** and listened to it with a receiver and it sounds like an analogue oscillator, which is great. Only if used in a radio application, for best purity of the signal generated, I've read it's better to use just one channel per SI5351 as the cross talk between channels increases clock jitter slightly. For other applications, it's usually not an issue.

There are debug statements in words **VFOC!** and **INCVFOC!**. When these are uncommented, you can see the data being sent to the SI5351 on the terminal. Notice after a **CLK0** or **CLK1** command, the next **RFtune** command sends a lot of bytes. Subsequent **RFtune** commands to nearby frequencies only send the few bytes that have changed, so that tuning takes as little time as possible.

Bob Edwards, radio ham G4BBY in SW U.K., adapted this code from his driver for the Parallax P2 processor in September 2025. He used ESP32forth version 7.0.7.21. Hope you find it useful!