

Overview

This document is intended to address common questions about the Silicon Laboratories programmable oscillator Si570 XO and Si571 VCXO products. The term Si57x stands for both the Si570 and Si571.

FAQ

1. What is the tuning resolution of the Si57x?

< 1 ppb

2. What kind of memory is used in the Si57x?

The Si57x uses both registers and NVM (Non Volatile Memory). The NVM is write once, read many technology. The Si57x does not contain any flash memory.

3. Can I change the startup frequency or I2C address?

No. The startup or powerup frequency and I2C address are programmed in to the NVM at the factory. These items must be specified during the part number request process.

4. Why do we need to calculate f_{XTAL} when reconfiguring the output clock for large changes in output frequency?

Each Si57x device has a unique crystal. The crystal frequency is nominally 114.285MHz but may be offset from this value by +/-2000ppm. We calculate f_{XTAL} in order to determine the exact crystal frequency and then use this information to adjust the DSPLL feedback divider ratio and output divider ratios. (Crystal manufacturers cannot tune the crystals and achieve the same initial accuracy as is possible with the DSPLL approach.)

5. What happens if we attempt to program a new frequency outside the specified speed grade of the device?

The output is squelched.

6. Are the I2C inputs on a 1.8V or 2.5V Si57x 3.3V tolerant?

No. Digital I/O signals should be limited to a maximum voltage within the nominal VDD range supplying the Si57x, i.e. 1.8V, 2.5V, or 3.3V. While the device may operate otherwise, e.g. 3.3V I/O in to a 2.5V device, it is not specified for this, and reliability may be degraded.

7. When is a voltage translator needed when interfacing the Si57x?

The I2C pull up resistors should be tied to the same VDD as the Si57x. The master needs to be able to read and write on the I2C bus using these same pull ups. If that is not the case, then a voltage translator is called for.

For example, the Silicon Labs C8051F320 MCU uses open emitter outputs that can support direct connection of its I2C lines to an Si57x with pull ups to 2.5V or 3.3V. However, the MCU itself runs on nom 3.3V and has difficulty understanding the difference between a 1.8V logic HIGH and LOW. In this case, one can employ a voltage translator such as the NXP GTL2002 which is a 2-bit bidirectional low voltage translator. The MCU side of the translator is pulled up to 3.3V and the DUT side is pulled up to 1.8V.

8. How exactly should the Si57x device be addressed?

The I2C address is a 7 bit address. Our experience with MCUs has been to place the 7 bits in the upper bits of an 8 bit word with the LSB reserved for the data direction bit. In other words, to send 0x55 as the I2C address in a write command, the bit order would look like

S	Slave Address	Data Dir Bit	A
	1010101	0	
AA			

9. What are the digital threshold levels for SCL and SDA?

The I2C interface VIH and VIL specs are the same as that specified for Output Enable (OE) in the Si57x datasheet. The Si57x digital I/O VIH and VIL are specified as $0.75 \times VDD$ min and 0.5 V max respectively.

10. Are there any specific differences that make the Si57x programming interface I2C compatible as opposed to I2C conformant?

- One specific difference regards digital thresholds. The Si57x is designed to operate using several nominal supplies, i.e. $VDD = 1.8\text{V}$, 2.5V , or 3.3V . As noted above, Si57x VIH and VIL are specified as $0.75 \times VDD$ min and 0.5 V max respectively. However, the I2C spec states that VDD-related input levels VIH and VIL should be $0.7 \times VDD$ min and $0.3 \times VDD$ max respectively. So for example, if $VDD=3.3\text{V}$, then Si57x $VIL = 0.5\text{V}$ which is less than the I2C spec $0.3 \times 3.3\text{V} = 0.99\text{V}$.
- A second specific difference regards SCL. SCL is bidirectional per the I2C standard. However, it is only an input to the Si57x. Therefore, unlike other I2C slaves, the Si57x does not have the ability to hold SCL down until ready.

11. What speeds of operation can the Si57x support?

- 100 kbps - Standard Mode
- 400 kbps - Fast Mode

12. What determines whether the DCO needs to be “frozen” when programming a new frequency?

The DCO must be “frozen” as part of the reconfiguration sequence for large frequency steps. Small frequency steps are defined as frequency changes $\leq \pm 3500$ ppm from the anchor frequency. Large frequency steps are defined as frequency changes $> \pm 3500$ ppm from the anchor frequency.

13. What is the “anchor” frequency? The anchor frequency is the default start-up frequency or the new frequency obtained after following the large frequency step programming procedure.

14. Why does the device reset when attempting a large frequency step without properly freezing the DCO?

The DCO runs out of tuning range and can't reach the new frequency. The Si57x is designed so that when this happens, it resets and reloads the NVM. The output is squelched until the DCO is tuned to, and centered on, the programmed start-up frequency.

15. How does freezing the DCO help?

Freezing the DCO ensures that no one 8-bit register write will cause an intermediate RFREQ value to be out of the ± 3500 ppm range (from the anchor frequency). Freezing is necessary when changing the upper 8 bits of RFREQ; otherwise the NVM will be recalled asynchronously.

16. What happens to the output clock when making frequency changes?

The output clock runs continuously during small frequency steps as long as the total frequency change from the anchor frequency is $\leq \pm 3500$ ppm. The output clock will be squelched during a large frequency step for 10ms or less.

17. Will the Si57x respond if every transaction is terminated by a stop before another start is sent?

Yes.

18. Should a delay be assumed for RST_REG just as with power up?

Yes. Asserting internal reset RST_REG by writing 0x80 to register 135 resets all internal logic. The reset disrupts the I2C communication link. It is as if the power has been recycled and the slave device (the Si57x) is temporarily absent. Under these conditions, it would be appropriate to implement a soft reset startup delay at least as long as the specified power up delay of ~10msec. Another approach would be to have the master code loop and repeatedly attempt to communicate until the slave is back up.

19. What is a good software delay to use after a RECALL or RST_REG?

We usually put in a delay of 15ms or more after a RST_REG or RECALL to insure that the I2C is ready to work on the device.

20. What is the difference between RECALL and RST_REG?

Asserting a RECALL <Reg. 135 bit 0> maintains I2C communication while asserting a RST_REG <Reg 135 bit 7> interrupts I2C communication.

21. If one chooses to assert a RECALL, what is the general sequence of operations?

Here is a sequence that we have used in the past:

RECALL -> Freeze DCO -> <Write New Config> -> Unfreeze DCO + NewFreq

22. Is there a significant advantage to asserting a RECALL as opposed to a RST_REG?

No. There is a small time advantage to doing a RECALL over RST_REG. However, RST_REG is generally regarded as the safest thing to do to return a register memory + NVM chip back to a known state. There is not a significant or practical advantage to picking one or the other approach in the case of the Si57x.

23. What are some items to check or troubleshooting suggestions to consider if there are general I2C communication problems?

- **Can you confirm I2C communication by reading the contents of registers 07-12?** You should be able to verify that they are appropriate for the programmed startup frequency. If not, there may be a hardware or timing problem. Here are some follow-up questions and suggestions.
- **Are there pullups on the I2C bus?**
- **Is the programmed I2C address correct and is the device being addressed correctly?**
- **Is the NewFreq assertion arriving within 10ms of Unfreeze?** See the programming constraints table in the datasheet.
- **Are read commands terminated properly?** Make sure that the master is sending a Not Acknowledge and a Stop after the last read data byte to terminate the read command.
- **Obtain an Si57x eval board.** By using the Si57x-EVB eval board h/w and s/w and comparing scope traces you should be able to debug any problems by comparison.

24. **What if there are set up problems working with the Si57x-EVB?** Here is the relevant section from the software help that describes how to troubleshoot setup issues.

Communication Issues

It is a known issue that the I2C communication between the MCU and the Si57x hangs when there is an error.

The EVB must be reset after the error is addressed in order for the EVB software to work with the EVB. Press the RESET button on the motherboard to enable the reset.

Possible communication errors are:

- SDA or SCL are not connected correctly
- The jumpers are not configured correctly per the EVB datasheet including VDD, SDA, SCL, and J11 (level shifter)
- The daughterboard is not seated correctly
- The wrong VDD is selected on the motherboard
- The wrong I2C address is used in the host software, so that it does not match the one in the device

Software-Hardware Synchronization

Sometimes the software and/or driver needs to be reset as well. Power cycle the EVB by disconnecting and connecting the USB plug. Also click on Options > Find EVB in the Si57x Register Programmer or toggle the Connect/Disconnect controls in the Si57x Programmer. This will re-enable the software connection with the EVB.

No Output Frequency

Check these things if there is no output from the device:

- Check the OE (J9) state of the EVB versus the device's configuration
- Check the operating frequency range of the device versus how it was last programmed. It could be out of range. Note the speed grade of the device, which can also restrict the output frequency range.

25. What version USBXpress driver should be used when working with the Si57x-EVB programming software?

The Si57x-EVB programming software was written using the USBXpress drivers available at the time, i.e. version 1.x. Unfortunately, succeeding versions of these USBXpress drivers are incompatible. Therefore, the Si57x-EVB software should not use the latest USBXpress drivers.

As of this writing the latest USBXpress driver is listed as Rev. 3.1.1 on the Silicon Labs website and the revision will be expected to change in the future. However, these drivers are used for other Silicon Labs devices.

The required USBXpress driver version is available with the EVB distribution CD. The install procedure for this EVB is also different from some other Silicon Labs EVBs in that you have to connect the EVB to the PC and point the Hardware Install Wizard to the driver location on the CD. The OS should do the rest. The details of this install are with the EVB kit documentation. The driver is also available online with the Si57xEVBSoftware.zip file on both the Si570 and Si571 web pages.

To verify the correct version, right-click on the USBXpress entry in the Device Manager. There you can see in the Driver tab the date which should be 1/13/2004.

26. What if I have already installed an incompatible version USBXpress driver?

If you have already installed drivers for the EVB, you may have to uninstall or do an update for this EVB on your system. This can be done via the Device Manager for the particular USBXpress entry in the listing. Right-click on it and select Properties to get these options.

Different versions of the USBXpress drivers can coexist satisfactorily on the same computer. It is not necessary to uninstall and reinstall the versions globally on the computer. It's just a matter of associating the right driver files with the EVBs, when the EVBs are connected to the PC.