Technical Article

Choosing the Right Oscillator for Your Microcontroller

May 17, 2016 by Robert Keim

Internal or external? Quartz or ceramic? Crystal oscillator or silicon oscillator? So many clocking options . . . which one is right for your design?

Internal or external? Quartz or ceramic? Crystal oscillator or silicon oscillator? So many clocking options . . . which one is right for your design?

Supporting Information

- Microprocessors

Oscillating Options

Every microcontroller needs a clock source. The CPU, the memory bus, the peripherals—clock signals are everywhere inside a microcontroller. They govern the speed at which the processor executes instructions, the baud rate of serial-communication signals, the amount of time needed to perform an analog-to-digital conversion, and so much more.

All of this clocking action goes back to the source of the clock signal, namely the oscillator. Therefore, you need to make sure that your oscillator can support whatever performance you expect from your microcontroller. At the same time, though, some oscillator options are more complex or expensive than others, so your choice of oscillator should also reflect the importance of reducing cost and complexity whenever possible.

There are quite a few ways to generate a clock signal for a microcontroller. The datasheet for your particular device should provide a good deal of information about what types of oscillators you can use and how to implement them in a way that is compatible with the device's hardware. This article will focus on the advantages and disadvantages of various clock sources so that you can better choose among the oscillator options discussed in your microcontroller's datasheet.

So let's start with a list, followed by a discussion of each option:

- Internal
  - Usually (as far as I know, always) a resistor–capacitor circuit
  - Phase-locked loop for frequency multiplication
- External
  - CMOS clock
  - Crystal
  - Ceramic resonator
  - Resistor–capacitor
  - Capacitor only

Internal Oscillators: The KIS Option

I'm an advocate of the Keep It Simple principle; consequently, I have a special appreciation for internal oscillators, and I encourage you to avail yourself of the internal oscillator whenever possible. No external components are required: You can safely assume that the frequency is well chosen since the oscillator was designed by the same people who designed the rest of the microcontroller. Also, the salient performance specs—e.g., initial accuracy, duty cycle, temperature dependency—are (hopefully) right there in the datasheet.

The dominant disadvantage with internal oscillators is the lack of precision and frequency stability. The baseline frequency depends on the values of the passive components that make up the oscillator circuit, and the tolerances for the values of these passive components are not particularly tight. Furthermore, capacitance and resistance are influenced by ambient temperature, so internal RC oscillators experience temperature "drift"—i.e., changes in temperature lead to changes in frequency.

In my experience, many applications can tolerate the shortcomings of an internal oscillator, especially when the frequency has been calibrated at the factory. With older microcontrollers, the internal oscillator might have tolerance as bad as ±20%. However, a newer device can give you ±1.5% (or better), which is accurate enough for RS-232 communication and even (in conjunction with clock-recovery circuitry) for USB.

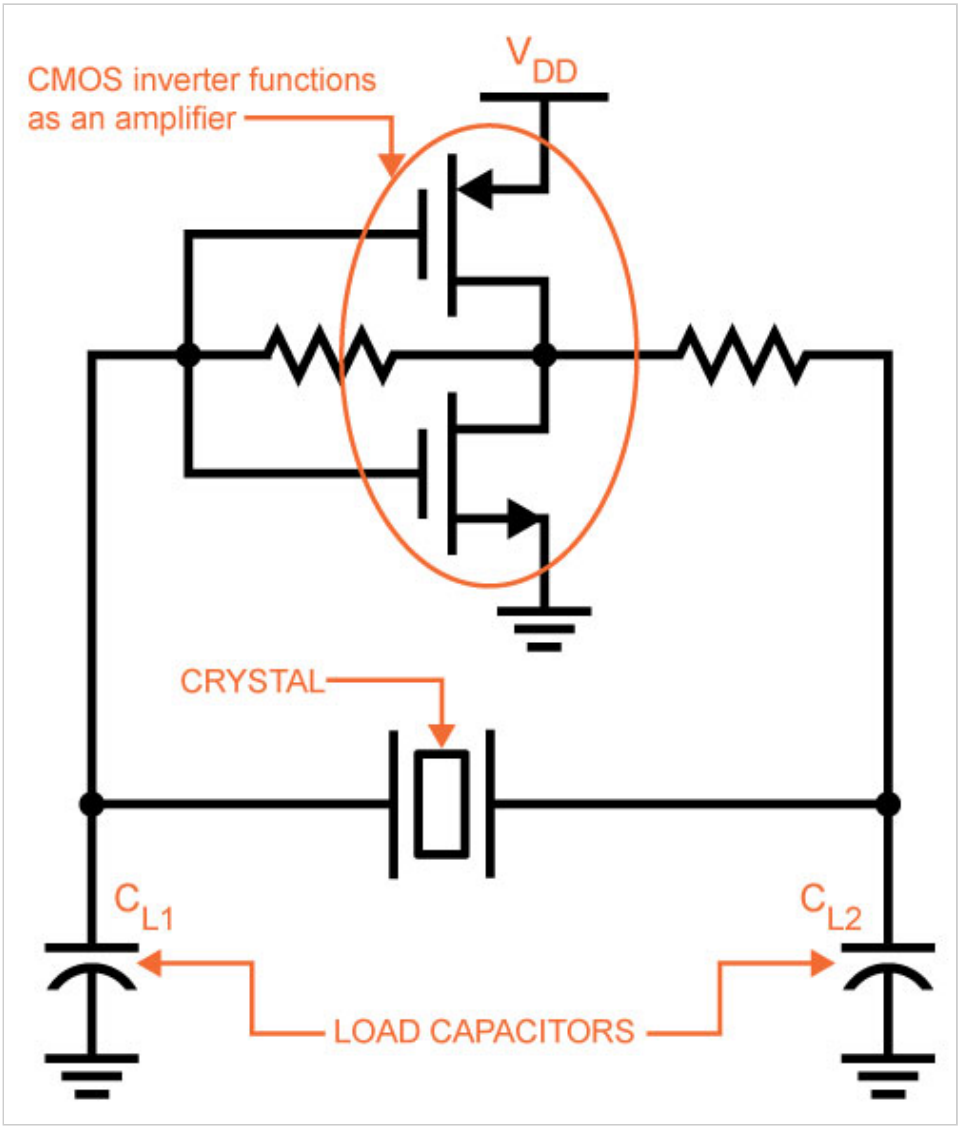
Another way to expand the capabilities of an internal oscillator is manual "trimming"—if your microcontroller includes a trimming/calibration register, you can adjust the frequency by modifying the value in this register. This is a perfectly practical technique for low-quantity designs: Simply measure the clock frequency with an oscilloscope or frequency counter and then trim the oscillator accordingly.

A variation on the internal-oscillator theme is the phase-locked loop (PLL). A PLL allows a low-quality, high-speed internal oscillator to benefit from the stability and precision of an external oscillator. In general, a PLL doesn't help you to avoid external components because it requires a reference clock that is usually derived from a crystal. An exception, though, is when you have a high-quality clock somewhere on the PCB but don't want to use it for the microcontroller because it's too slow—you could use a PLL to multiply this clock up to an acceptable frequency.

CMOS Clock

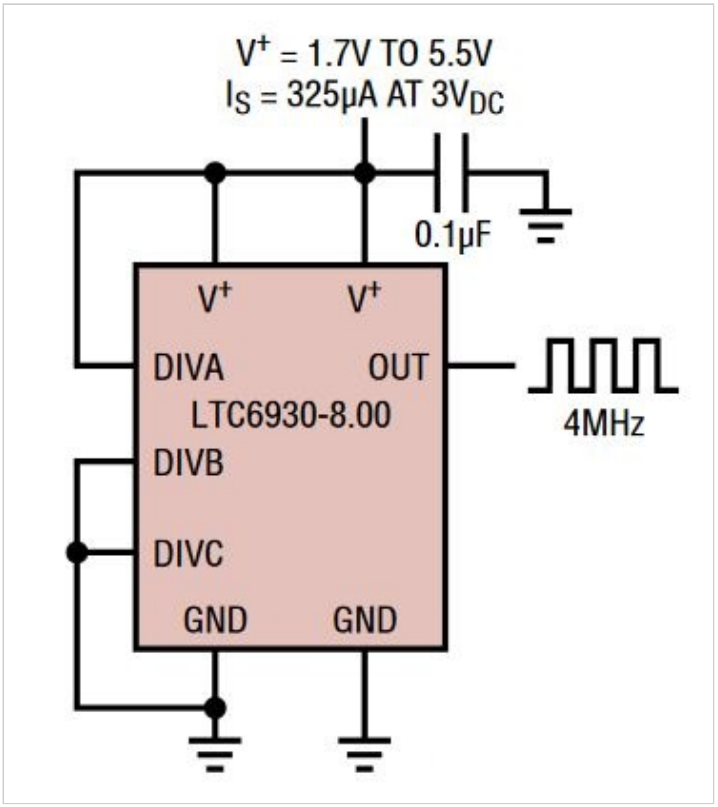
Another straightforward clocking option is the so-called "CMOS clock," which falls into the for-lack-of-a-better-term category. "CMOS clock" is a vague (though convenient) way of referring to any clock signal driven by some other component on the board. The CMOS clock is a great option if your design already includes a clock signal with 1) a workable frequency and 2) electrical characteristics that are compatible with the microcontroller's CMOS-clock-input circuitry. Often, though, this is not the case, so let's look at two options for generating a CMOS clock.

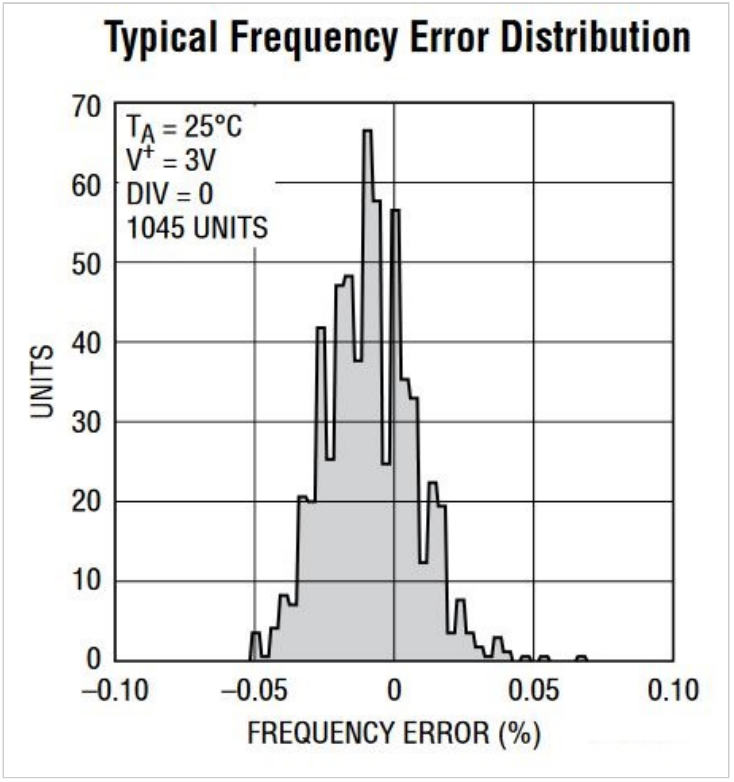
First is the "crystal oscillator." This is a good time to point out that a quartz crystal is not an oscillator; rather, a quartz crystal is the central component in a quartz-crystal oscillator circuit, which might look something like this:



Crystal oscillators are handy devices that consist of a quartz crystal and the additional circuitry needed to generate a standard digital clock signal. Thus, you get the stability and precision of a crystal without worrying about load capacitance and the careful PCB layout needed to ensure robust operation with a standalone crystal.

The second option is a “silicon oscillator”. This term refers to oscillator ICs that are not based on quartz crystals or ceramic resonators. These devices are versatile and easy to use, and they can be quite accurate. For example, the LTC6930 series from Linear Tech requires only one bypass capacitor, and the vast majority of the parts are within .05% of the nominal frequency:





Silicon oscillators are more reliable than crystals and ceramic resonators, especially in harsh environments subject to shock or vibration. But they're also more expensive.

Quartz and Ceramic

When you need seriously high precision and stability without the additional cost of a crystal-based oscillator IC, opt for the standalone-crystal approach. Parts with tolerance below 20 parts per million (i.e., 0.002%) are readily available. The oscillator circuit shown above is partially integrated into microcontrollers that support the standalone-crystal configuration; you will need to provide the correct load capacitors. The total load capacitance (C_{LTOTAL}) is specified in the crystal's datasheet, and the load capacitors are chosen as follows:

$$C_{LTOTAL} = \frac{C_{L1} \times C_{L2}}{C_{L1} + C_{L2}} + C_P$$

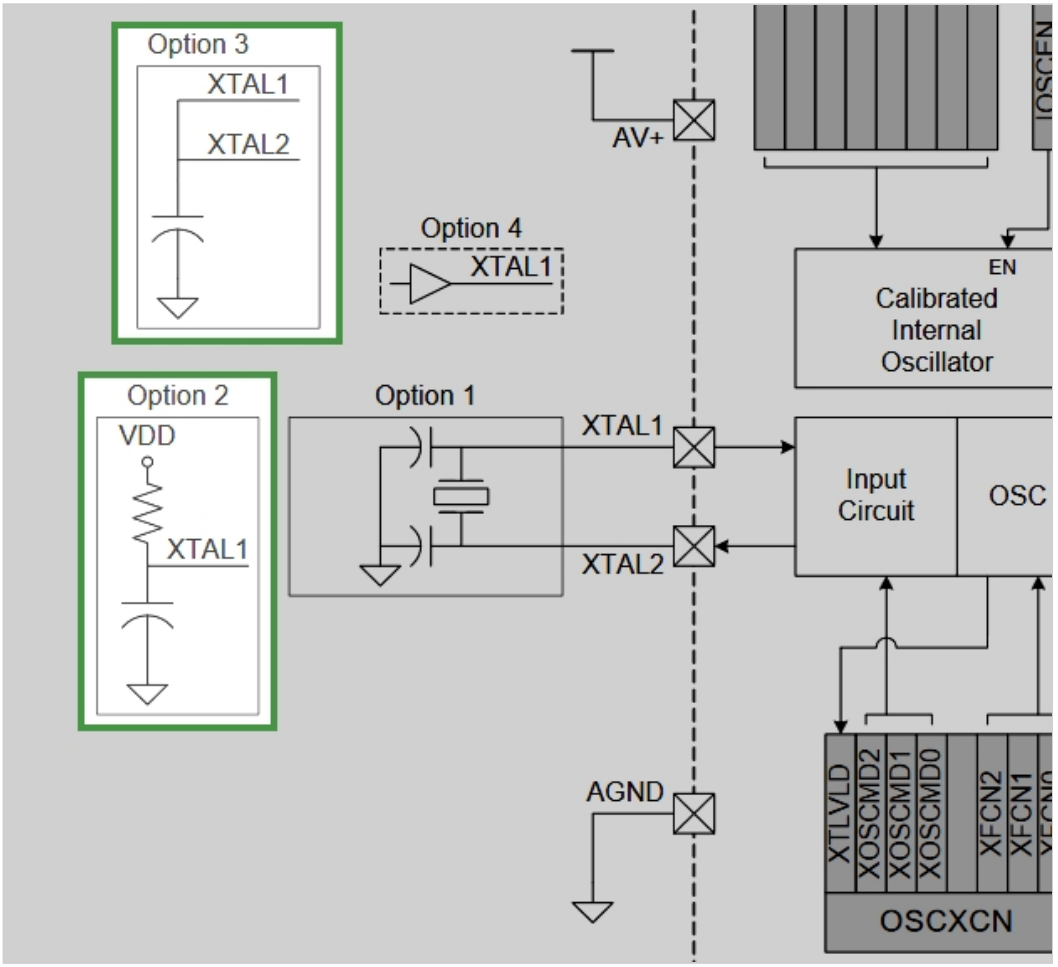
where C_P represents whatever parasitic capacitance is present. This calculation is pretty simple in practice: Choose a reasonable value for C_P (say, 5 pF), subtract this from C_{LTOTAL}, then multiply by two. So if the datasheet gives a load capacitance of 18 pF, we have

$$C_{L1} = C_{L2} = (18\ \text{pF} - 5\ \text{pF}) \times 2 = 26\ \text{pF}$$

Ceramic resonators are less accurate than crystals; common tolerances are 1000 to 5000 parts per million. They may save you a few cents if you don't need the accuracy of quartz, but in my mind, the main advantage is that you can get ceramic resonators with integrated load capacitors.

Last, and Least . . .

There aren't many situations that call for an external resistor–capacitor or capacitor-only oscillator. If for some reason you are opposed to the external-oscillator options discussed thus far, choose a microcontroller with an internal oscillator and use that. If, however, you are determined to dig out a passive or two from your box of spare parts, refer to the microcontroller's datasheet for guidance on how to connect and design the oscillator circuit. Here is an example of how to connect the components, taken from the datasheet for the C8051F12x–13x (PDF) microcontrollers from Silicon Labs:



And you can refer to page 190 (PDF) of this same datasheet for an example of information on choosing component values.

Conclusion

I hope you now know enough to make an informed, confident decision next time you need to choose an oscillator for your microcontroller. Here are my recommendations in a nutshell:

- Internal oscillator whenever possible
- Silicon oscillator if the accuracy is adequate and the cost is acceptable—otherwise, quartz crystal