Determining baud rates for 8051 UARTs and other UART issues

AN448

Author: Greg Goodhue

The purpose of this note is to expand upon and clarify some aspects of determining baud rates and crystal frequencies for using a standard 8051 or 80C51 UART for ordinary RS-232 type serial communication. The standard baud rate equation is simplified here and is restated to allow solving for other variables such as the crystal frequency and timer reload values.

The following discussion assumes that the reader has some knowledge of the 8051/80C51 UART and timers. This should be considered a supplement to the information presented in the Philips 80C51 Family Microcontroller Data Book sections on Timer/Counters and the Standard Serial Interface.

Since this discussion assumes the use of a standard UART for RS-232 serial communications, the UART will be used in modes 1 or 3 (variable baud rates) and timer 1 will be used in mode 2 (8-bit auto-reload mode) as the baud rate generator. All of the equations shown here give an option for two clock divisors depending on whether the SMOD bit is used on a CMOS microcontroller. For an NMOS device, always use the default value (SMOD is not = 1).

The basic equation for a timer reload value can be stated as:

$$TH1 = 256 - \frac{\text{Crystal Frequency/384 (or 192 if SMOD = 1)}}{\text{Baud Rate}}$$

Example:

To obtain a timer reload value for a 9600 baud serial data rate with an 11.0592 MHz crystal:

$$256 - \frac{11{,}059{,}200/384}{9600} = 256 - 3 = 253, \text{ or FD hexadecimal}$$

The equation can also be solved to derive the baud rate or the crystal frequency from the other information as follows:

$$\text{Baud Rate} = \frac{\text{Crystal Frequency/384 (or 192 if SMOD = 1)}}{256 - TH1}$$

$$\text{Minimum crystal frequency for a given baud rate} = \text{Baud Rate} \times 384 \text{ (or 192 if SMOD = 1)}$$

Thus, the minimum crystal frequency that may be used for 19.2k baud communication on a CMOS part with SMOD = 1 would be 19200×192, which gives 3.6864 MHz. When using this equation, the timer reload value TH1 for the maximum baud rate is always 255 (256 – 1) or FF hexadecimal.

Of course, any even multiple of the frequency obtained in this manner will also support the same baud rate with a different timer reload value. For instance, four times 3.6864 MHz is 14.7456 MHz. At that crystal frequency, 19.2k baud is attained with a timer reload value that gives one fourth of the timer overflow rate: 252 (256 – 4) or FC hexadecimal.

CRYSTAL FREQUENCIES USED FOR STANDARD BAUD RATES

The following chart shows possible crystal frequencies for use with the 80C51 UART at standard baud rates. The chart assumes use of the UART in modes 1 or 3 (variable baud rates) and timer mode 2 (8-bit auto-reload mode). The chart also assumes a minimum requirement of at least 9600 baud (including the use of SMOD for baud rate doubling). More crystal frequencies are available if a lower maximum baud rate is required.

The minimum timer count column indicates how many timer counts are required at the stated crystal frequency in order to obtain the maximum baud rate shown. The last column shows the timer reload value that is used to obtain the minimum timer count. This is simply 256 minus the minimum timer count.

Timer reload values for other baud rates at the same crystal frequency are determined by multiplying the minimum timer count by two successively and calculating a new reload value as previously mentioned. For instance, for 4800 baud at 1.8432 MHz, the timer count would be 2 (twice what it is for 9600 baud), giving a timer reload value of 254 (256 – 2) or FE hexadecimal.

Maximum Standard Crystal (MHz)	Maximum Baud Rate	Timer Count	Timer Reload Value (in hex)
1.8432	9600	1	FF
3.6864	19200	1	FF
5.5296	9600	3	FD
7.3728	38400	1	FF
9.2160	9600	5	FB
11.0592	19200	3	FD
12.9024	9600	7	F9
14.7456	76800 (2×3840)	1	FF
16.5888	9600	9	F7
18.4320	19200	5	FB
20.2752	9600	11	F5
22.1184	38400	3	FD
23.9616	9600	13	F3
25.8048	19200	7	F9
27.6840	9600	15	F1
29.4912	153600 (4×38400)	1	FF
31.3344	9600	17	EF
33.1776	19200	9	F7
35.0208	9600	19	ED
36.8640	38400	5	FB

THE EFFECT OF USING OFF-FREQUENCY CRYSTALS

Occasionally, one may wish to use an off-frequency crystal in a design, but still want to make use of the UART for debug purposes.

Since most terminals (or other RS-232 devices) will communicate with another device that has a baud rate that is off by several percent, this can often be done successfully. WARNING: running the UART off-frequency is NOT recommended if part of the application's normal operation involves communication with other RS-232 devices.

There is no exact limit on how much frequency error is tolerable, since this depends on the devices communicating, the baud rates, precise frequencies used by both devices, etc. However, a rule of thumb may be used that the communication is likely to work if the frequency is off by less than 5%. This somewhat arbitrary number was arrived at as follows: for a ten-bit serial code (one start, 8 data bits, one stop), a 10% data rate error will put the receiver off by about plus or minus one bit time at the end of one data frame. A one bit-time error seems rather excessive if one wants fairly reliable communications. So, consider using half of that value (5%) as a rule of thumb.

The consequence of all this is that one may often find a more standard "off-the-shelf" crystal frequency to use in an application, if the UART is being used for debugging, factory testing, etc. As an example, consider the well-known "color burst" crystal. At 3.579545 MHz, this crystal is only about 3% slower than the 3.6864 MHz crystal that may be desired for baud rate generation. As such, this lower cost crystal could be used in place of the less standard one in some cases. Another obvious replacement is to use the standard 14.31818 MHz crystal in place of the not-so-standard 14.7456 MHz crystal that appears in the table. This replacement also yields a less than 3% error and may be handy because it gives a fast instruction execution rate for the 80C51, whereas 3.58 MHz may be too slow for many applications.

It should also be remembered that RS-232 communications are most robust if characters are not transmitted back-to-back. This can become more important when the UART is deliberately used out of spec as described here. When data is sent at full speed, there is no chance for the receiver to re-synchronize to the transmitted frames if it once gets out of synch. However, when there is a short pause between characters (about 2 to 3 bit times or longer), the receiver will generally be able to correctly locate start bits without framing errors. In the worst case, a pause of one byte-time or longer in a transmission should ALWAYS re-synchronize any receiver no matter how out of synch it has become.

A LITTLE KNOWN PHENOMENON

In the UART setup code for most applications, the actual timer count register (TL1) is not initialized. In many applications, this DOES have an affect on the way the UART behaves on the first character sent, although the chances of this being noticed are slight. This can be seen by trying to observe the first character sent from the UART on a logic analyzer that is being triggered by the end of microcontroller reset. The first character will begin so far down the time line that it will not be seen at any resolution on the logic analyzer that would show any of the individual bits.

This effect occurs because TL1 has to time-out once before the first character is transmitted. If TL1 is not initialized in the program, it will have a reset value of 0. This could give the first timeout a duration of up to 255 normal bit times, depending on the reload value for TH1 (which again depends on the baud rate and crystal frequency).

Again, in most applications, this would never be an issue. In fact, it may often be an advantage to have a delay before the first serial character is sent after power-up. But if the first serial character should start sooner, TL1 may be initialized to some value other than zero. For no delay, the same value used in TH1 should be used.