2.8-GHz prescaler keeps cost down

Neil Eaton, Emsys Engineering, Peterborough, ON, Canada

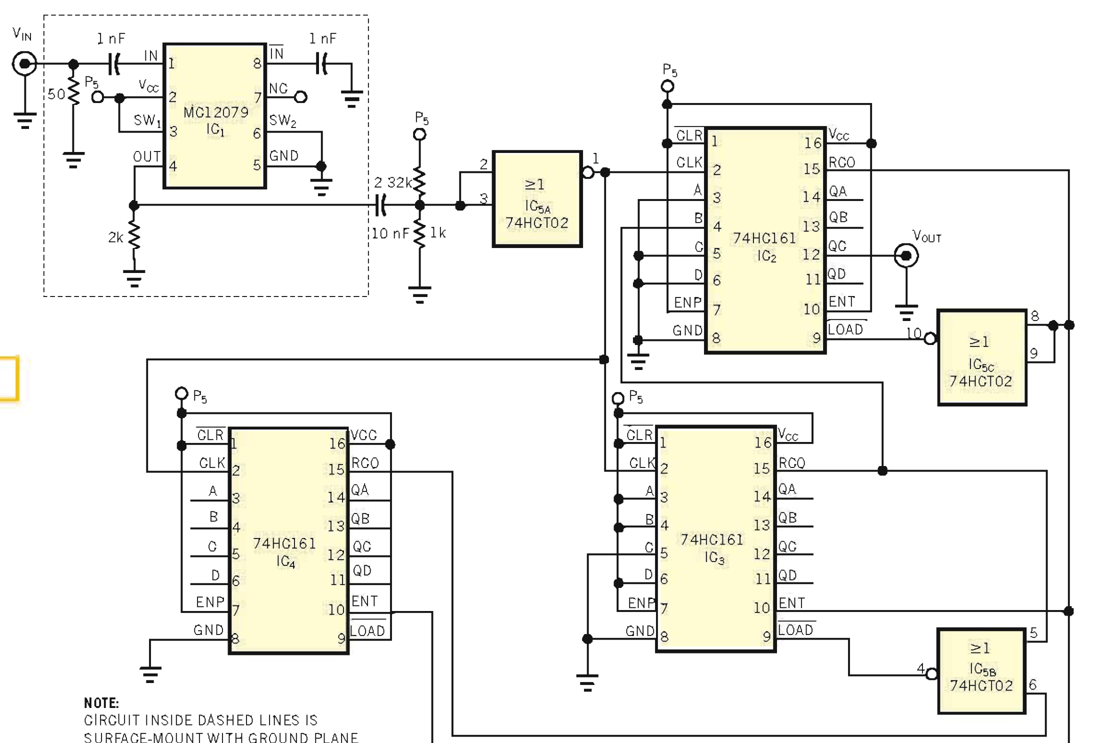
The prescaler in **Figure 1** inexpensively extends the range of a frequency counter by dividing the input signal's frequency by a factor of 1000. The guaranteed input-frequency range of the input prescaler, IC$_1$, is 250 MHz to 2.8 GHz, although typical values are 100 MHz to 3.5 GHz. The prototype operates at frequencies well below 100 MHz, but its fastest generator goes only to 1.7 GHz, so you cannot confirm the upper range. The input-voltage range is 400 to 1000 mV p-p from 250 to 500 MHz and 100 to 1000 mV p-p for higher than 500 MHz. IC$_1$ serves as a divide-by-128 prescaler, whose output is a 1.6V p-p square wave. The RC network level-shifts the output of IC$_1$ to ensure that the top of the square wave is above the 2V input threshold of IC$_{5A}$. The output of IC$_{5A}$ is a 5V, CMOS-compatible square wave with a frequency of $^1/_{128}$ of the input frequency. Most frequency counters can handle these frequencies, but the submultiple is inconvenient for an operator. A further division by a factor of 7.8125 (1000/128) produces a scaling factor of 1000.

Fortunately, the frequency counter averages its input over many cycles, so the output of the prescaler need not be exactly $^1/_{1000}$ of the input frequency for every input pulse. The 0.8125 figure is 13 divided by 16. The average frequency ratio is therefore 7.8125 if you divide 13 output pulses of 16 by eight and the remaining three by seven. For best results, the divide-by-seven pulses should be as evenly spaced as possible. The result is a repeating sequence 16 output pulses long with the following pattern:

Pulse number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Divide by	8	8	8	8	7	8	8	8	8	7	8	8	8	8	7	8

IC$_2$ divides the prescaler's output by seven or eight, depending on the state of the B input. IC$_3$ and IC$_5$ both connect to the output of IC$_2$, so they count output pulses, not prescaler pulses. IC$_5$ divides the output by five to generate the divide-by-seven periods. IC$_4$ divides the output by 16 to reset the cycle upon completion. Without IC$_4$, the cycle would continue to divide by seven at every fifth output pulse for a ratio of 7.8. The construction of the circuitry inside the dashed lines in **Figure 1** is critical. The MC12079 is available only as a surface-mount device. In the design, it and its associated passive components is mounted on a Surfboard (Capital Advanced Technologies Model 9081, available from Digikey). It is then fastened, component side up, to a bare copper-clad board to create a ground plane. All connections to ground from the high-

Figure 1



Extend your frequency-measurement capability with this inexpensive divide-by-1000 prescaler.

speed circuitry go directly to the ground plane using short lengths of copper braid. (Desoldering wick is ideal.) The input uses a BNC chassis-mount connector with its shell soldered directly to the ground plane, and the center pin connected to the Surfboard with the shortest possible wire. The rest of the circuit is noncritical. To avoid clutter, Figure 1 shows no bypass capacitors, but you should place them near every IC. (DI #2564)

To Vote For This Design, Enter No. 332

Add harmony to your system

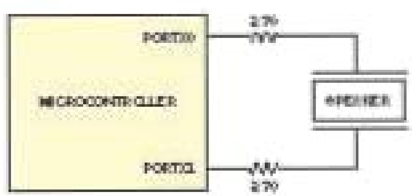
Eugene O'Bryan, Food and Drug Administration, Rockville, MD

Have you ever wished you could distinguish one device's operating state from another's by the sounds they make or that error states would sound harsh while normal operations would sound harmonious? By combining the NCO technique from a previous Design Idea (Reference 1) with digital mixing, you can obtain musical chords or intervals with a minimal amount of hardware and software. Any µC system can thus produce a variety of sounds. In Figure 1, a piezoelectric speaker, Radio Shack Model 273-091, and two 270Ω resistors transduce a pulse stream from a µC. Differential drive to this transducer increases the volume by doubling the effective voltage. The capacitance of the piezoelectric speaker reacts with the 270Ω resistors to integrate and smooth the pulse stream. The software (Listing 1), a tight loop, comprises a square-wave generator and an NCO's (numerically controlled oscillator's) summing part. After it sets up some registers, the sound-generating loop establishes an output level for each of two or more square waves.

The pseudocode example in Listing 1 demonstrates a two-note generator in which the output levels of two square waves are established in registers r_vol1 and r_vol2. The frequency of each square wave is a function of the values set for variables first_note and second_note and by the cycle rate of the loop. Note that a half-cycle of a square wave concludes when the corresponding counter register (r_cnt1 or r_cnt2) reaches zero. The summing part of the loop uses the NCO technique to generate an output level for the digital mixture of the square waves generated in the first part of the loop. The mixing of two square waves with frequencies of 880 Hz, the C above middle C, and 988 Hz, the second D above middle C, in the equal-tempered scale produces the oscilloscope waveform in Figure 2. An AVR AT90S Series µC runs through the sound-generating loop in 16 clock cycles. Thus, to produce an 988-Hz square wave with this type of µC running at 8 MHz, the first_note or second_note value is set at 253, which is equal to 8 MHz/16/988/2.

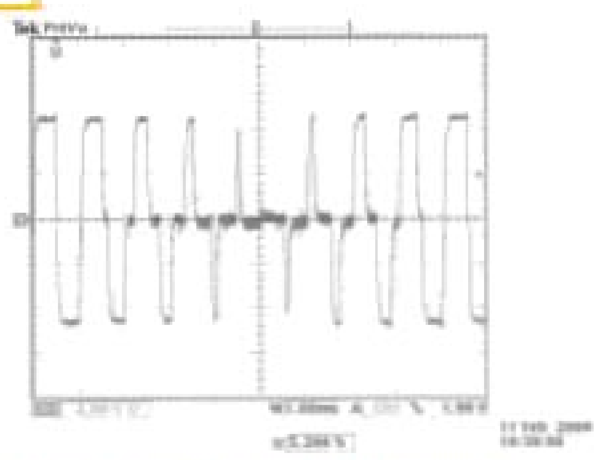
A note of caution is in order in calculating this value in an assembly program. Be careful of truncation issues; the musical intervals sound wrong if the frequencies are off. Also, when you use another type of µC, you must adjust the loop software to compensate for differences between the µC's instruction timing and the assumed instruction timing of the pseudocode. All instructions should complete in one clock cycle except for jump or branch executions, which take two clock cycles. You can create chords by expanding the software loop to include a third square-wave generator and modifying the mixer to add in the third note. A by-product of this sound-generation scheme is that it lets you control volume by changing the volume variable in Listing 1. The loudest possible volume occurs when the volume variable equals 40hex for a two-note generator. Setting the volume variable lower than 40hex reduces the voltage output from each square wave. With this scheme, you can produce 10 distinct volume levels, using hex values 40, 6A, 60, 54,

Figure 1



Produce harmonious sounds in your system, using this simple scheme.

Figure 2



Mixing square waves of 880 and 988 Hz produces this waveform.