

Edited by Bill Travis and Anne Watson Swager

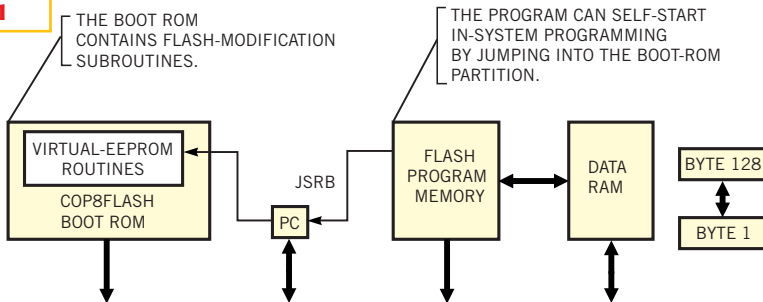
μC eliminates offboard EEPROM with “virtual-EEPROM” routines

Wallace Ly, National Semiconductor Corp, Santa Clara, CA

COP8FLASH μCs let you allocate some of their internal flash memory as a nonvolatile storage space. This line of flash μCs accomplishes this task by accessing some memory functions built into the boot ROM. In addition to some of these memory functions, the COP8flash μC can initiate and control in-system programming under software. You can use “virtual EEPROM,” although not truly EEPROM, to mimic the behavior of the storage space of an offboard EEPROM. Note that the flash inside the COP8flash μC is not just renamed E². This fact has significant implications because the flash μC is rated for 100,000 erase/write cycles and 100-year data retention.

Figure 1 depicts how you can allocate a 128-byte virtual EEPROM. Although users may not write over the same byte twice, they may modify RAM contents. Additionally, you can “shadow” the flash μC with RAM. If users want to permanently save the contents of the virtual

Figure 1



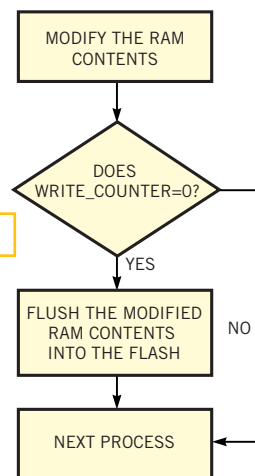
This example of a 128-byte virtual EEPROM shows how you can mimic the behavior of the storage space of an offboard EEPROM.

EEPROM, then they may execute a “flush” to dump the contents of the modified RAM into the flash μC.

A system designer may bring up the issue that the “dumping” of the RAM into the flash μC may take too long. However, the opposite is true if you take the following into consideration: A typical write to the flash μC takes a few microseconds, and a page erase takes 8 msec, independently of the μC system clock. So, the dumping process may take less than 10 msec. However, a designer is hard-pressed to find an offboard serial EEPROM that has less than 100 msec of access time.

Figure 2 shows how to visualize the sample virtual-EEPROM code. As you shadow the flash μC, you write bytes to the RAM, and a WRITE_COUNTER decrements. When the WRITE_COUNTER equals 0, the routine flushes the contents of the RAM. The flash μC then saves the contents of the RAM.

Figure 2



When the WRITE_COUNTER equals 0, the routine flushes the contents of the RAM.

A system designer can set the WRITE_COUNTER to 0 to immediately copy

μC eliminates offboard EEPROM with “virtual-EEPROM” routines	123
Medical timer warns when pills are due	124
Temperature monitor and fan controller reduce fan noise.....	128
Circuit forms random bit-sequencer generator	132

the contents of the RAM into the flash μ C. You can download the Virtual EEPROM C code from EDN's Web site, www.ednmag.com. Click on "Search

Databases" and then enter the Software Center to download the file for Design Idea #2688.

Is this the best Design Idea in this issue? Vote at www.ednmag.com/ednmag/vote.asp.

Medical timer warns when pills are due

JM Terrade, Clermont-Ferrand, France

SOME PEOPLE NEED to take medication at precise, regular intervals. When you're in a hospital, a medical staff is present to ensure that you take your medication on time. But when you're taking medication at home, you must frequently look at the clock—a clear annoyance. When my wife was pregnant, she needed to take medication every two hours from 8 am to 10 pm. To help her time the two-hour intervals, I built the battery-powered circuit in **Figure 1**. The circuit derives its power from a 9V PP3 battery. All the ICs are CMOS-based; the low power consumption of the circuit allows one-week autonomy. When you

TABLE 1—TIMING DETAILS FOR CIRCUIT IN FIGURE 1

	IC ₂		IC ₃	
	Positive edge	Period	Positive edge	Period
Q1	0.858 msec	1.17 msec	3.51 sec	7.03 sec
Q2	1.717 msec	3.43 msec	7.03 sec	14.06 sec
Q3	3.433 msec	6.866 msec	14.06 sec	28.12 sec
Q4	6.866 msec	13.73 msec	28.12 sec	56.25 sec
Q5	13.73 msec	27.5 msec	56.25 sec	112.5 sec
Q6	27.46 msec	55 msec	112.5 sec	225 sec
Q7	54.93 msec	110 msec	225 sec	7.5 minutes
Q8	109.86 msec	220 msec	7.5 minutes	15 minutes
Q9	219.7 msec	439 msec	15 minutes	30 minutes
Q10	439 msec	878 msec	30 minutes	One hour
Q11	879 msec	1.758 sec	1 hour	Two hour
Q12	1.758 sec	3.51 sec	2 hour	Four hour

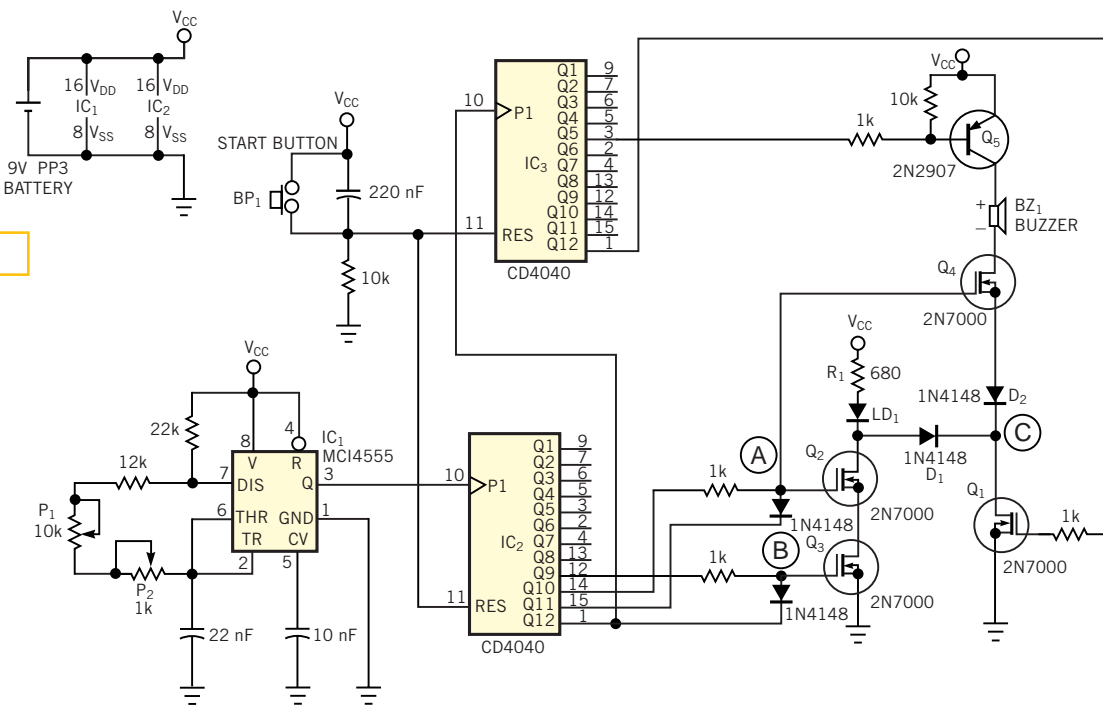


Figure 1

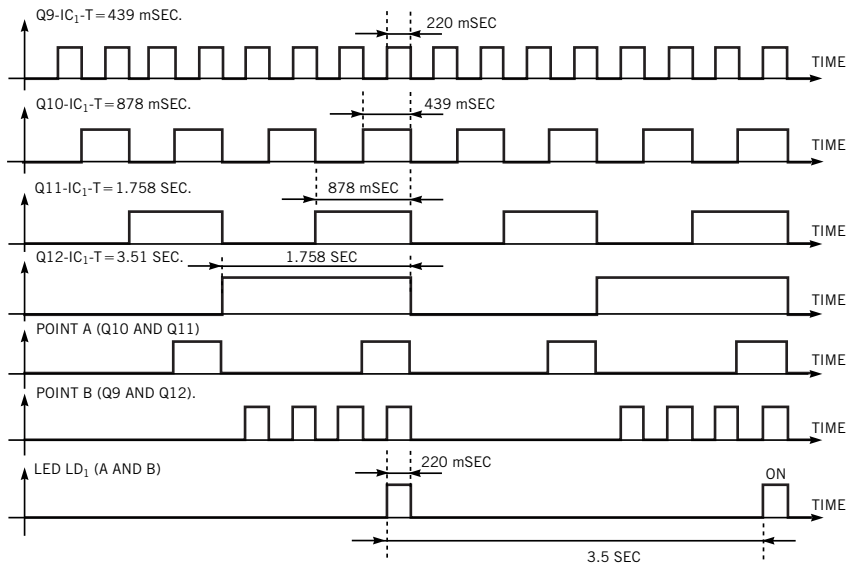
The buzzer in this circuit sounds at precise two-hour intervals.

press the start button, BP₁, a two-hour delay commences. During this delay, LED LD₁ flashes to indicate the delay is in progress and the battery voltage is satisfactory. After the two-hour delay elapses, the buzzer, BZ₁, emits short beeps for one minute. If you don't press the start button again, LD₁ stays lit continuously to indicate that the delay has elapsed. Pressing the start button initiates a new two-hour delay cycle.

IC₁, a 555 timer, operates as an astable multivibrator and produces a rectangular waveform at its output. The period should be equal to 858.3 μsec. P₁ and P₂ permit gross and fine adjustments of the period. For the two-hour delay, P₁ yields a ±20-minute adjustment; P₂ produces a ±2-minute adjustment. For a precise adjustment, observe the Q7 output of IC₂ and trim to obtain a period of 110 msec. IC₂ is a 2¹² divider whose Q12 output has a 3.51-second period. This output connects to IC₃'s clock input. IC₃ then yields a 14,400-second period, or four hours. The circuit detects the positive edge of Q12, which occurs after two hours. Changing the connection to IC₃ to another output or trimming P₁ and P₂ allows you to set different delays. **Table 1** shows the timing details for the various outputs of IC₂ and IC₃.

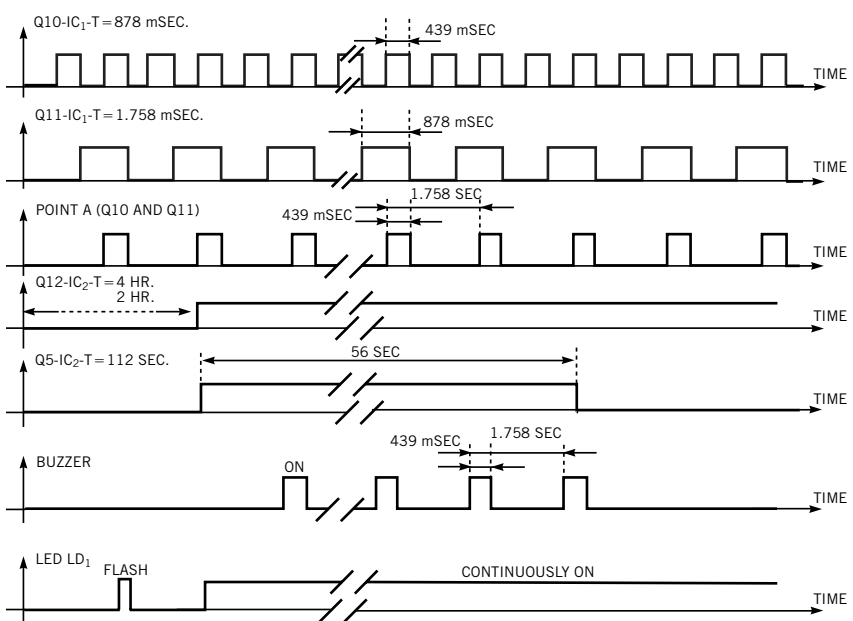
Figure 2 shows timing details for the first two hours after asserting start for the circuit in **Figure 1**. Q12 of IC₃ is at a low level, and transistor Q₁ is off. Thus, D₁ and D₂ cannot conduct. To light LD₁, Q₂ and Q₃ must be on, which is the case when Q9 to Q12 of IC₂ are at a high level. This high-level condition occurs for 220 msec every 3.5 seconds. This time period might seem short, but R₁'s low value and the choice of a high-luminosity LED makes LD₁'s flashing clearly visible. **Figure 3** shows timing details after the two-hour time-out delay. Q12 of IC₃ assumes a high level, and Q₁ grounds Point C. As a result, D₁ conducts, and LD₁ stays continuously on, indicating that the delay has terminated. D₂ conducts when Q₄ and Q₅ are on, and the buzzer sounds. These transistors turn on when Q5 of IC₃ is at a low level and Q10 and Q12 of IC₂ are at a high level. These conditions occur for 439 msec every 1.8 seconds, for a

Figure 2



These waveforms show what elapses in the circuit of Figure 1 during the two-hour time-out.

Figure 3



The buzzer sounds, and the LED stays on after the two-hour time-out.

total duration of 56 seconds. This time period might seem short, but it is long enough to create an audible beep. The start button restarts the cycle.

Is this the best Design Idea in this issue? Vote at www.ednmag.com/ednmag/vote.asp.

Temperature monitor and fan controller reduce fan noise

David Hanrahan, Analog Devices Inc, Limerick, Ireland

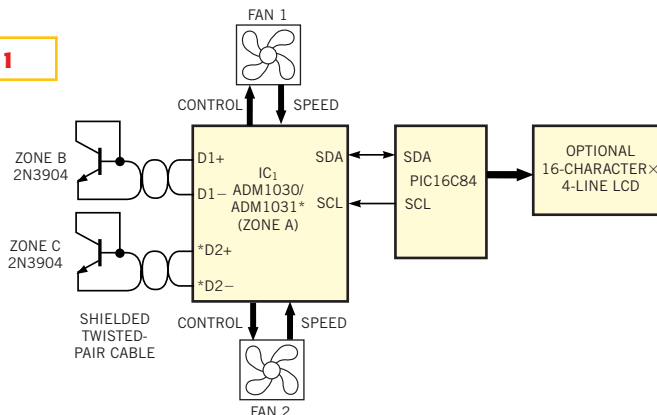
THE SCHEME IN **Figure 1** reduces system acoustic noise by running system fans at their optimum speeds for a given temperature. IC₁ combines ±1°C-accurate temperature measurement of three temperatures with automatic fan-speed control of two channels. A two-wire serial interface allows you to oversee critical temperature and fan-speed data. NPN transistors, such as the 2N3904, can measure temperatures in remote locations. The microcontroller interface allows you to connect an LCD to display all monitored parameters.

You program IC₁'s automatic-fan-speed-control function using T_{MIN} and T_{RANGE} (**Figure 2**). T_{MIN} is the temperature at which the fan automatically turns on and runs at minimum speed. T_{RANGE} is the temperature-to-fan speed-control slope, with options of 5, 10, 20, 40, and 80°C. Choosing one of the T_{RANGE} options allows you to define how the fan reacts to temperature variation.

An example of programming IC₁ follows. Setting Configuration Register 1 (Reg 0x00) to 0x99 starts the device in automatic-fan-speed-control mode, with the FANFAULT function enabled. A setting of Remote Temp 1 T_{MIN}/T_{RANGE} (Reg 0x25)=0x63 sets T_{MIN} for fan 1 to 48°C and T_{RANGE} to 40°C. The Fan reaches full speed at 88°C. A setting of Remote Temp 2 T_{MIN}/T_{RANGE} (Reg 0x26) = 0x62 results in a T_{MIN} for Fan 1 of 48°C and a T_{RANGE} of 20°C. The fan reaches full speed at 68°C.

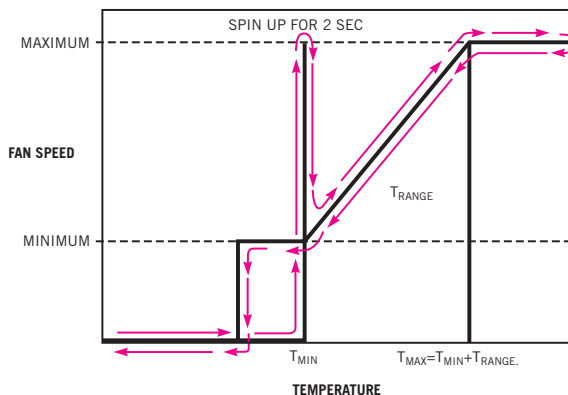
The fan runs at low speed until the system requires a higher level of cooling. The fan speed responds automatically to temperature variation, reducing acoustic noise (**Figure 3**). You can program additional features, such as fan spin-up time and minimum fan speed. The automatic-fan-speed-control mode allows flexi-

Figure 1



A scheme with multiple temperature-measurement and fan-control channels can run system fans at the optimum speeds for a given temperature.

Figure 2



The values of T_{MIN} and T_{RANGE} set the parameters of the automatic-fan-speed control loop.

bility over which temperature channel controls each fan. You can also decide that the maximum speed calculated for all temperature channels controls the fans.

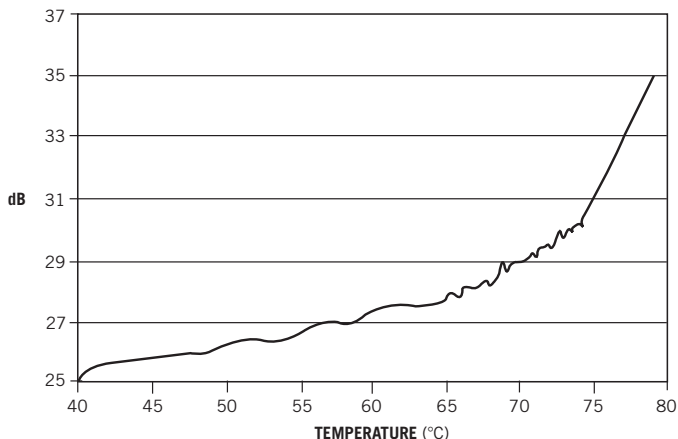
Figure 4 shows the circuit diagram for the temperature-monitor/fan-control circuit. The PIC16C84 writes the configuration settings to IC₁ and optionally drives an LCD. The μC bit-bangs pins 2 and 3 to provide serial clock and data for IC₁. The μC reads and displays all temperatures and fan speeds. The LEDs that connect to the THERM and FANFAULT

outputs illuminate whenever an over-temperature condition occurs or a fan fails. The THERM output is a fail-safe output that goes low if a preprogrammed overtemperature THERM limit is exceeded. In the event of an overtemperature condition, both fans automatically run at full speed. If some external device pulls the THERM pin low, the fans also run at full speed. The FANFAULT pin signals catastrophic fan failure. If one fan fails, the second fan automatically spins at full speed to compensate for the loss

of airflow. Should the failing fan recover or be replaced, both fans automatically return to their normal operating speeds.

You can extend this idea for multiple fans with redundant cooling using two ICs to monitor six temperature zones (Figure 5). This scheme cross-connects the THERM and FANFAULT signals. The same pin drives fans A₁ and A₂, which will run at the same speed. Likewise, fans B₁ and B₂ connect in parallel to a common FET. Fans C and D are redundant coolers that the system uses only if it gets excessively hot or a fan fails. If either fan A₁ or A₂ fails, fans B₁ and B₂ automatically run at full speed. The FANFAULT output of IC₁ asserts low, pulling THERM of IC₂ low,

Figure 3



Automatic-fan-speed control reduces acoustic noise.

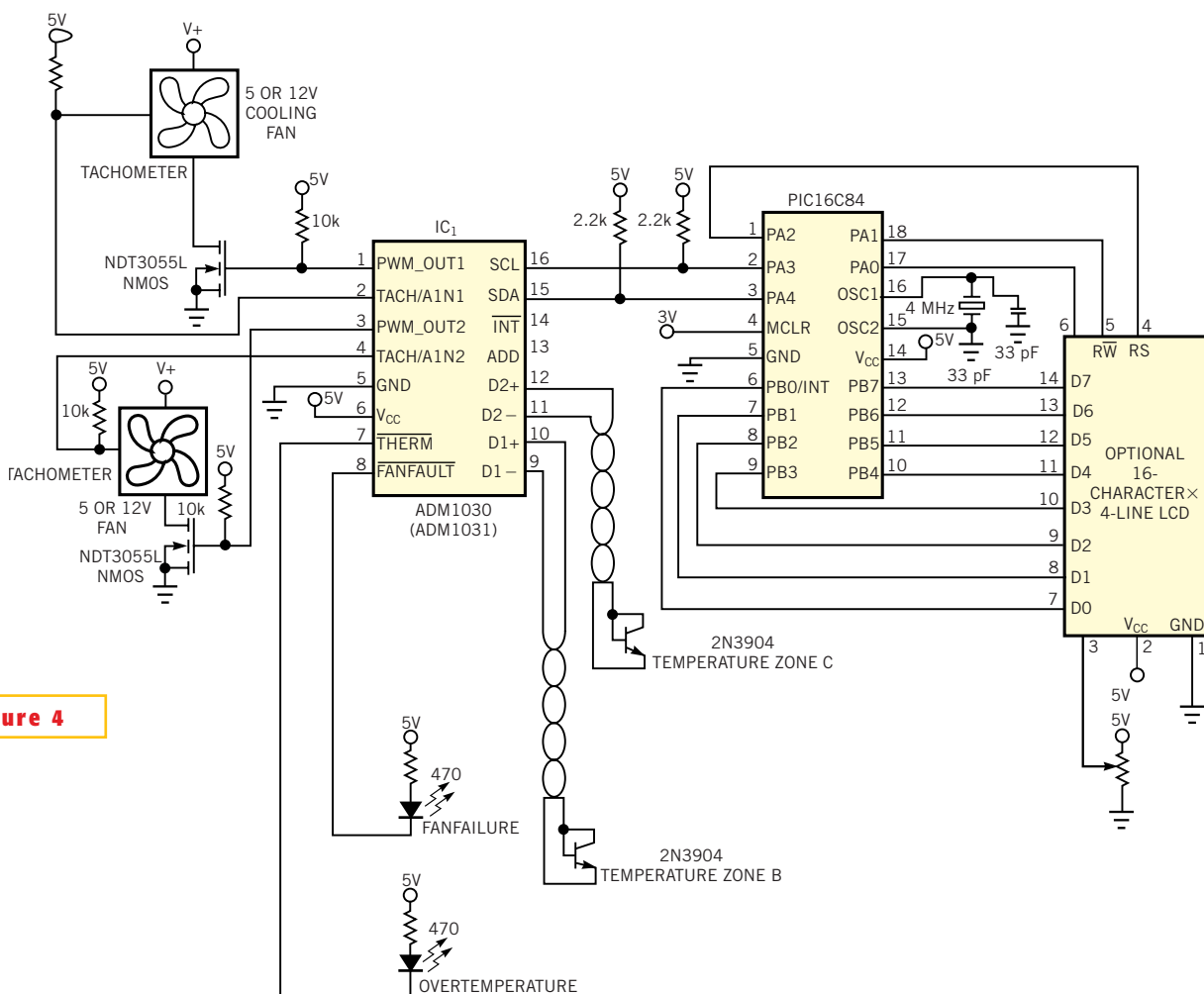


Figure 4

The μ C bit-bangs pins 2 and 3 to provide serial clock and data for IC₁ and reads temperatures and fan speeds.

its frequency spectrum should be at least equal to the clock frequency.

REFERENCES

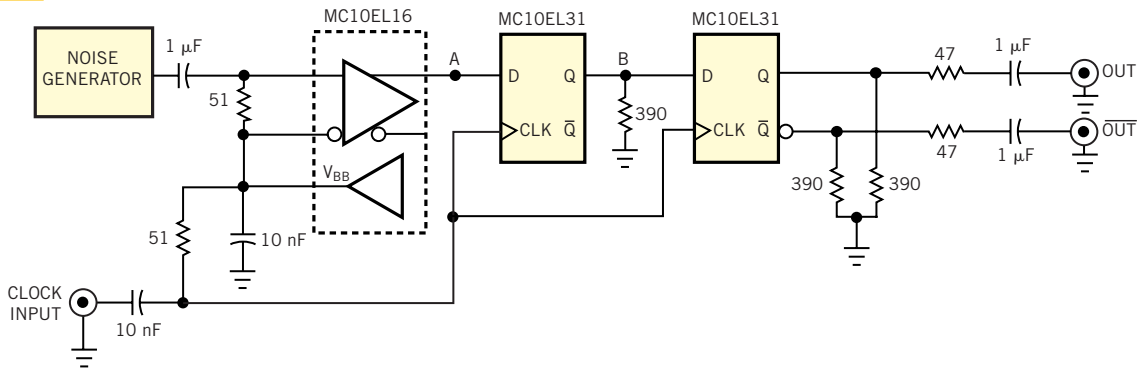
1. High Performance ECL Data, Motorola, 1995.

2. Metastability and the ECLinPS Family, Motorola Application Note AN1504.

3. Sliwczynski, Lukasz, "Zener diode and MMICs produce true broadband noise," *EDN*, Oct 14, 1999, pg 158.

Is this the best Design Idea in this issue? Vote at www.ednmag.com/ednmag/vote.asp.

Figure 1



This generator produces truly random bit sequences at frequencies to 1 GHz.