design ideas

Edited by Bill Travis and Anne Watson Swager

µC eliminates offboard EEPROM with "virtual-EEPROM" routines

Wallace Ly, National Semiconductor Corp, Santa Clara, CA

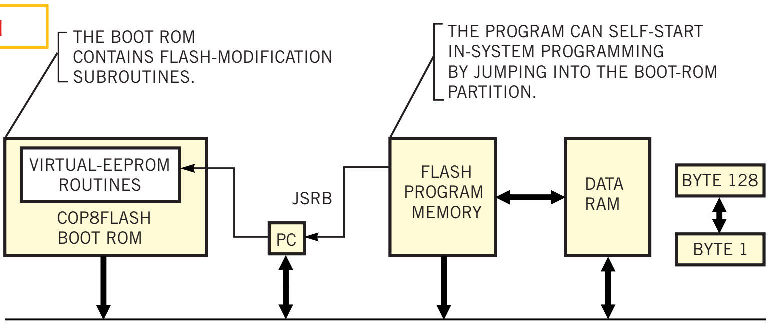
COP8FLASH µCs let you allocate some of their internal flash memory as a nonvolatile storage space. This line of flash µCs accomplishes this task by accessing some memory functions built into the boot ROM. In addition to some of these memory functions, the COP8flash µC can initiate and control in-system 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 E^2. 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 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. A system designer can set the WRITE_COUNTER to 0 to immediately copy

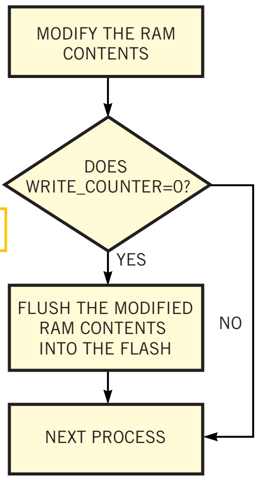
Figure 1



THE BOOT ROM CONTAINS FLASH-MODIFICATION SUBROUTINES.
THE PROGRAM CAN SELF-START IN-SYSTEM PROGRAMMING BY JUMPING INTO THE BOOT-ROM PARTITION.
VIRTUAL-EEPROM ROUTINES
COP8FLASH BOOT ROM
JSRB
PC
FLASH PROGRAM MEMORY
DATA RAM
BYTE 128
BYTE 1

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

Figure 2



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

the contents of the RAM into the flash µC. 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

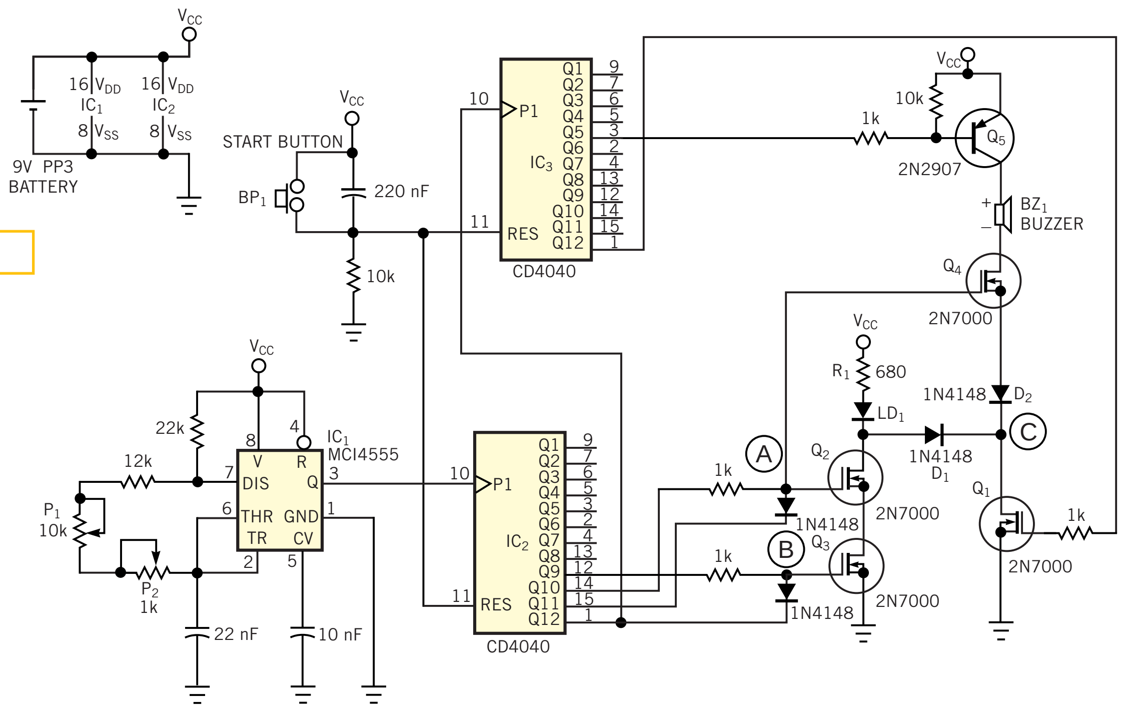
JM Terrade, Clermont-Ferrand, France

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



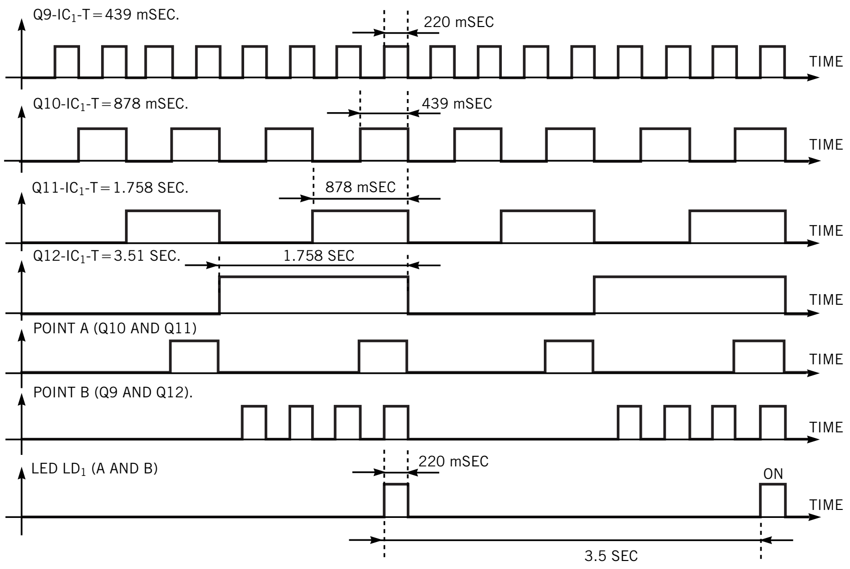
The buzzer in this circuit sounds at precise two-hour intervals.

press the start button, BP$_1$, a two-hour delay commences. During this delay, LED LD$_1$ flashes to indicate the delay is in progress and the battery voltage is satisfactory. After the two-hour delay elapses, the buzzer, BZ$_1$, emits short beeps for one minute If you don't press the start button again, LD$_1$ stays lit continuously to indicate that the delay has elapsed. Pressing the start button initiates a new two-hour delay cycle.

IC$_1$, 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. P$_1$ and P$_2$ permit gross and fine adjustments of the period. For the two-hour delay, P$_1$ yields a ±20-minute adjustment; P$_2$ produces a ±2-minute adjustment. For a precise adjustment, observe the Q7 output of IC$_2$ and trim to obtain a period of 110 msec. IC$_2$ is a 2^{12} divider whose Q12 output has a 3.51-second period. This output connects to IC$_3$'s clock input. IC$_3$ 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 IC$_3$ to another output or trimming P$_1$ and P$_2$ allows you to set different delays. **Table 1** shows the timing details for the various outputs of IC$_2$ and IC$_3$.

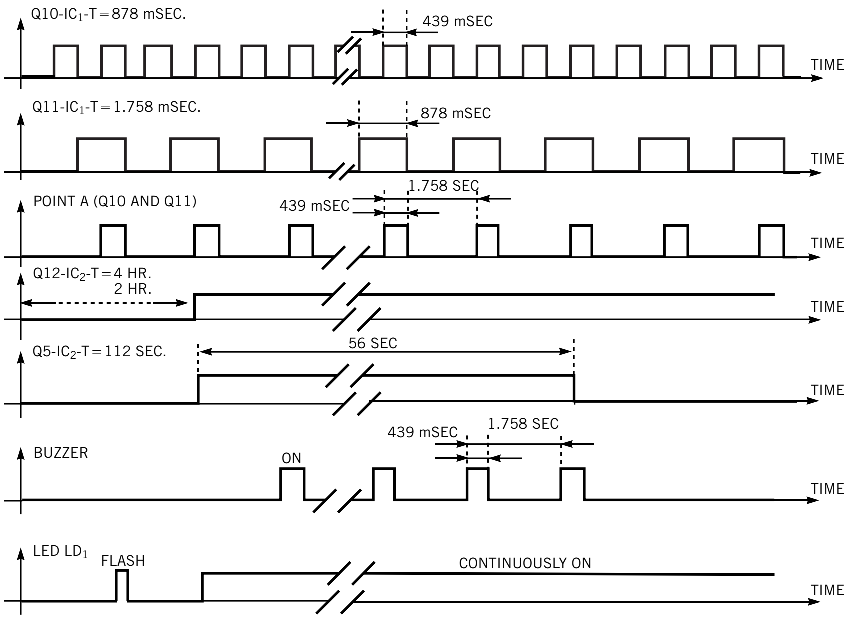
Figure 2 shows timing details for the first two hours after asserting start for the circuit in **Figure 1**. Q12 of IC$_3$ is at a low level, and transistor Q$_1$ is off. Thus, D$_1$and D$_2$ cannot conduct. To light LD$_1$, Q$_2$ and Q$_3$ must be on, which is the case when Q9 to Q12 of IC$_2$ are at a high level. This high-level condition occurs for 220 msec every 3.5 seconds. This time period might seem short, but R$_1$'s low value and the choice of a high-luminosity LED makes LD$_1$'s flashing clearly visible. **Figure 3** shows timing details after the two hour time-out delay. Q12 of IC$_3$ assumes a high level, and Q$_1$ grounds Point C. As a result, D$_1$ conducts, and LD$_1$ stays continuously on, indicating that the delay has terminated. D$_2$ conducts when Q$_4$ and Q$_5$ are on, and the buzzer sounds. These transistors turn on when Q5 of IC$_3$ is at a low level and Q10 and Q12 of IC$_2$ are at a high level. These conditions occur for 439 msec every 1.8 seconds, for a 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.

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.

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. IC_1 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.

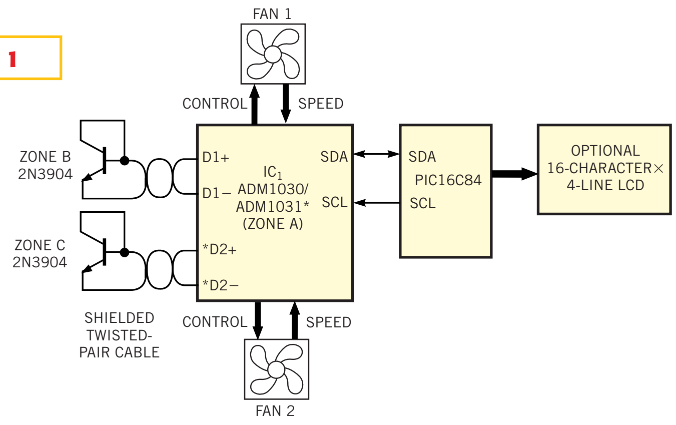


Figure 1

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

You program IC_1'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.

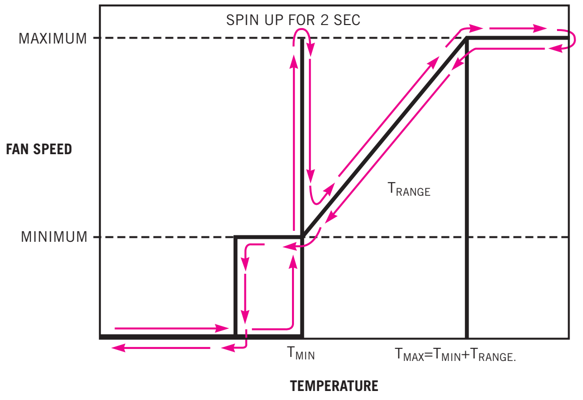


Figure 2

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

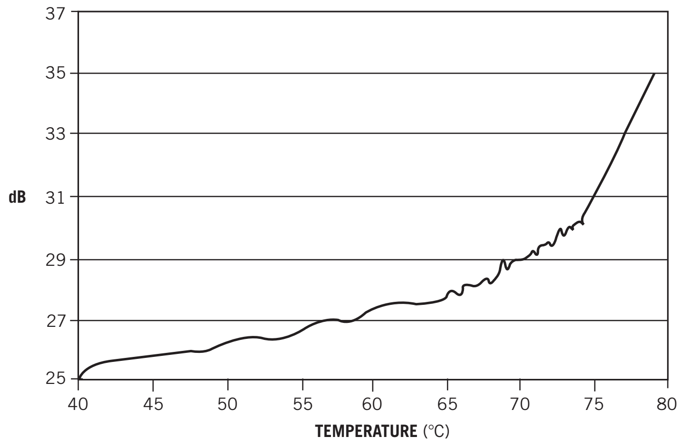
An example of programming IC_1 follows. Setting Configuration Register 1 (Reg 0x00) to 0x99 starts the device in automatic-fan-speed-control mode, with the $\overline{\text{FANFAULT}}$ function enabled. A setting of Remote Temp 1 T_{MIN}/T_{RANGE} (Reg 0x25)=0x63 sets T_{MIN} for fan 1 to 48°C and T_{RANGE} 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 flexibility 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 IC_1 and optionally drives an LCD. The μC bit-bangs pins 2 and 3 to provide serial clock and data for IC_1. The μC reads and displays all temperatures and fan speeds. The LEDs that connect to the $\overline{\text{THERM}}$ and $\overline{\text{FANFAULT}}$ outputs illuminate whenever an overtemperature condition occurs or a fan fails. The $\overline{\text{THERM}}$ output is a fail-safe output that goes low if a preprogrammed overtemperature $\overline{\text{THERM}}$ limit is exceeded. In the event of an overtemperature condition, both fans automatically run at full speed. If some external device pulls the $\overline{\text{THERM}}$ pin low, the fans also run at full speed. The $\overline{\text{FANFAULT}}$ 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.

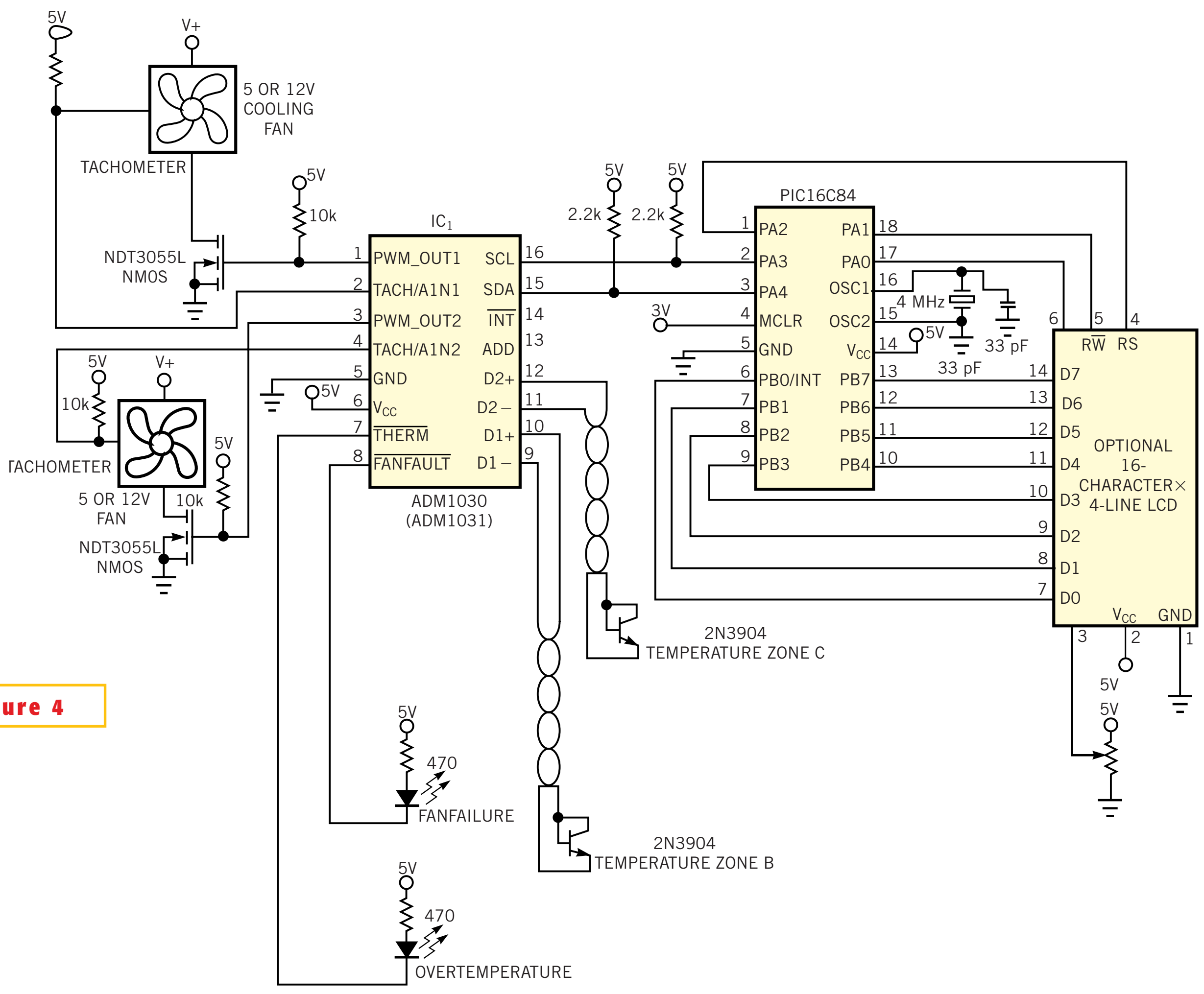
You can extend this idea for multiple fans with redundant cooling using two ICs to monitor six temperature zones (**Figure 5**). This scheme cross-connects the THERM and FANFAULT signals. The same pin drives fans A_1 and A_2, which will run at the same speed. Likewise, fans B_1 and B_2 connect in parallel 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 A_1 or A_2 fails, fans B_1 and B_2 automatically run at full speed. The $\overline{\text{FANFAULT}}$ output of IC_1 asserts low, pulling $\overline{\text{THERM}}$ of IC_2 low,

Figure 3



Automatic-fan-speed control reduces acoustic noise.

Figure 4

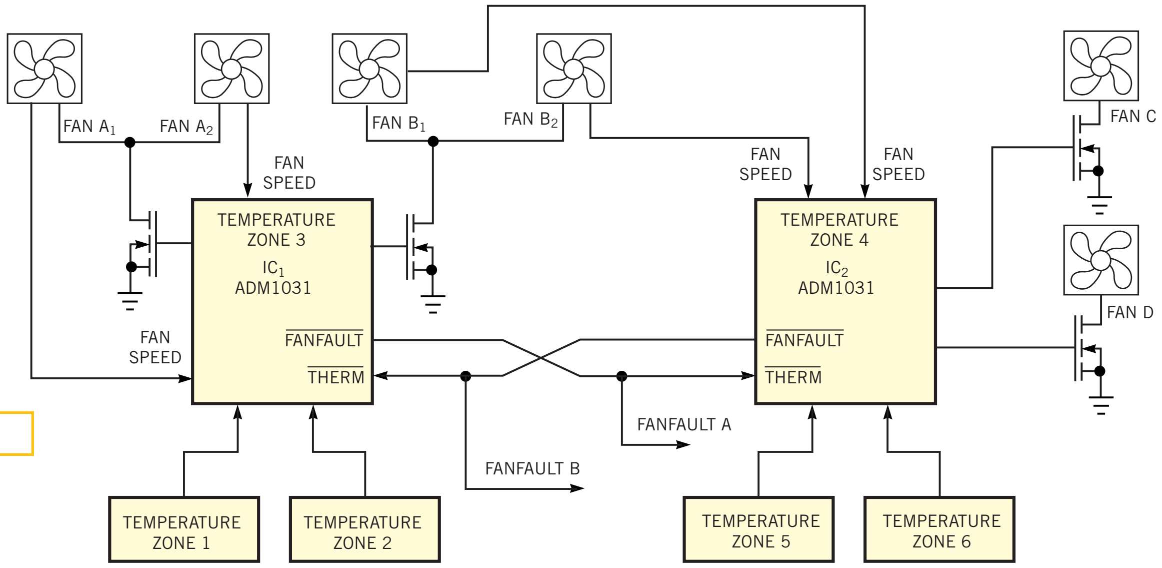


The μC bit-bangs pins 2 and 3 to provide serial clock and data for IC_1 and reads temperatures and fan speeds.

which causes redundant fans C and D to run at full speed. If either fan B_1 or B_2 fails, fans A_1 and A_2 automatically run at full speed. The $\overline{\text{FANFAULT}}$ output of IC_2 pulls $\overline{\text{THERM}}$ of IC_1 low, causing fans A_1 and A_2 to run at full speed. Fans C and D also automatically run at full speed. If the faulty fan is replaced, all fans return to their normal operating speeds. $\overline{\text{FANFAULT}}$ signals alert the system to fan failure.

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

Figure 5



Using two ICs provides for monitoring six temperature zones.

Circuit forms random-bit-sequence generator

Przemyslaw Krehlik and Lukasz Sliwczynski,
University of Mining and Metallurgy, Krakow, Poland

A RANDOM-BIT-SEQUENCE generator is basic equipment for prototyping and testing any data-transmission system. You use such a generator when measuring BER (bit-error rate) and pattern-dependent effects in a transmission system. Such effects can include baseline wander, pattern-dependent data jitter, and recovered-clock jitter. Most sequence generators yield a PRBS (pseudorandom bit sequence) from a shift register with appropriate feedback. Thus, the sequence has limited length, and the generator continuously repeats the same pattern. The generator in **Figure 1** overcomes these limitations by using random noise to form the outgoing data stream. The circuit uses the ECLinPS logic family (**Reference 1**). The MC10EL16 liner receiver converts the incoming noise to a digital signal. Next, a rising clock edge of the first MC10EL31 flip-flop samples this random signal. Ideally, this flip-flop should provide a random bit sequence.

Unfortunately, when the data at the flip-flop's input changes simultaneously with the clock's rising edge, the flip-flop may fall into a metastable state. Thus, the resulting output state is indeterminate, and a significantly extended propagation delay may result, producing a jitter in the generated bit sequence (**Reference 2**). The second MC10EL31 flip-flop eliminates the jitter problem. We tested the generator with clock frequencies to 1 GHz and observed no anomalies in the output eye pattern or frequency spectrum. Note that the ECLinPS devices are ultrafast ICs, so you need to exercise special care in your pc-board design. You should terminate the generator's inputs and outputs with 50Ω, keep all connections short, and decouple all ICs with local capacitors. You can use a circuit from **Reference 3** for a noise source. The voltage of the noise source should be 100 mV to 1V rms, and its frequency spectrum should be at least equal to the clock frequency.

References

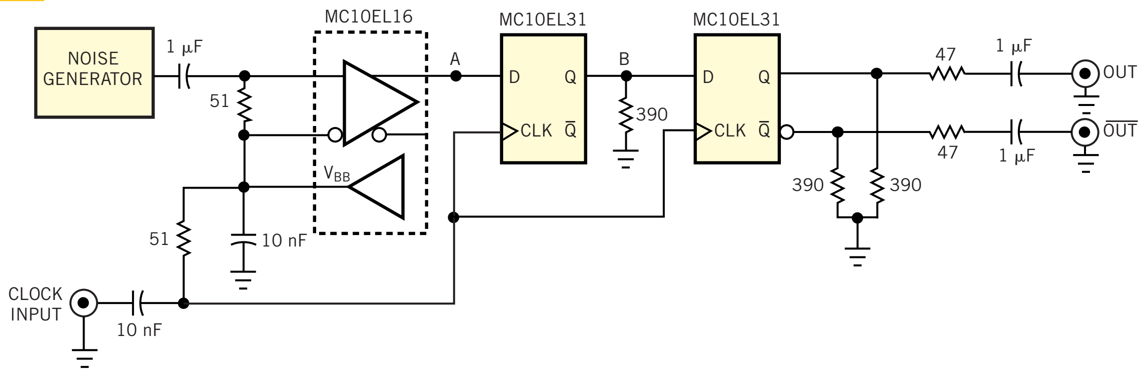
1. High Performance ECL Data, Motorola, 1995.

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

3. Sliwczynski, Lukasw, "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.