

Edited by Bill Travis and Anne Watson Swager

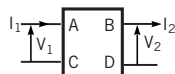
Cascade ABCD two-port networks

Bert Erickson, Thomson Consumer Electronics, Fayetteville, NY

TELEPHONE SUBSCRIBER LINES have become a topic of intense interest for organizations attempting to transmit Internet signals on telephone lines. Basic loop standards exist, and tables of twisted-pair primary constants extending to 20 MHz are in the literature. Asymmetric digital-subscriber lines (ADSLs) are now available for high-speed Internet service. The telephone industry has always used two-port networks in the form of ABCD matrices to cascade sections of telephone cable. These configurations allow you to join the sections by matrix multiplication. However, because the elements in the matrices are complex numbers and several sections make up a chain, you need to organize the process of matrix multiplication. A short computer program performs this task (**Listing 1**). You enter the number of matrices in the chain and then enter the real and imaginary parts for each A, B, C, and D element. The product then appears on the screen.

The two-port matrix equation with

the ABCD parameters has the following format:

$$\begin{bmatrix} V_1 \\ I_1 \end{bmatrix} = \begin{bmatrix} A & C \\ B & D \end{bmatrix} \begin{bmatrix} V_2 \\ I_2 \end{bmatrix}$$


where

$$A = \left. \frac{V_1}{V_2} \right|_{I_2=0} \quad B = \left. \frac{V_2}{I_2} \right|_{V_2=0} \quad C = \left. \frac{I_1}{V_2} \right|_{I_2=0} \quad D = \left. \frac{I_1}{I_2} \right|_{V_2=0}$$

A represents the open-circuit transfer function, B represents the short-circuit transfer impedance, C represents the open-circuit transfer admittance, and D represents the short-circuit current ratio. You can find the ABCD matrix for a ladder network composed of RLC elements by slicing the network into series and shunt matrices. You then multiply the

product of the first two by the third, and the progression continues with additional subscripts identifying the components. In the final product, the elements A, B, C, and D—with all their component symbols—may become quite cumbersome. To avoid these complicated expressions in the matrix for the ladder network, don't use them. Rather, enter the numerical values for the series and shunt components along with the necessary 0,0s and 1,0s. You can use this concept for any network comprising passive components that you can separate into isolated two-port sections. You can also use the method to predict the degradation a bridged tap produces in a telephone cable.

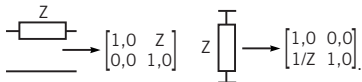
The progression of entering values for cascaded networks is normally from left to right, or from source to load with the

Cascade ABCD two-port networks.....	107
Simple tester checks	
Christmas-tree lights	108
Power meter uses low-cost multiplier.....	110
Add music to your next project	112
PWM circuit uses fuse to sense current.....	114
Bias supply accepts high inputs	116
Method provides simple error-rate generator	118

LISTING 1—CASCADING TWO-PORT NETWORKS

```
CLS
PRINT SPC(8); : INPUT "Enter number of 2X2 matrices, n => 2 "; n
DIM a(n - 1), b(n - 1), c(n - 1), d(n - 1)
DIM e(n - 1), f(n - 1), g(n - 1), h(n - 1), t(7)
FOR j = 0 TO n - 1
PRINT SPC(8); "For matrix number "; j + 1
PRINT SPC(16); : INPUT "Enter A and E "; a(j), e(j)
PRINT SPC(16); : INPUT "Enter B and F "; b(j), f(j)
PRINT SPC(16); : INPUT "Enter C and G "; c(j), g(j)
PRINT SPC(16); : INPUT "Enter D and H "; d(j), h(j)
NEXT j
GOSUB 1
END
1 FOR j = 1 TO n - 1
t(0) = a(j - 1) * a(j) - e(j - 1) * e(j) + b(j - 1) * c(j) - f(j - 1) * g(j)
t(1) = a(j - 1) * e(j) + e(j - 1) * a(j) + b(j - 1) * g(j) + f(j - 1) * c(j)
t(2) = a(j - 1) * b(j) - e(j - 1) * f(j) + b(j - 1) * d(j) - f(j - 1) * h(j)
t(3) = a(j - 1) * f(j) + e(j - 1) * b(j) + b(j - 1) * h(j) + f(j - 1) * d(j)
t(4) = c(j - 1) * a(j) - g(j - 1) * e(j) + d(j - 1) * c(j) - h(j - 1) * g(j)
t(5) = c(j - 1) * e(j) + g(j - 1) * a(j) + d(j - 1) * g(j) + h(j - 1) * c(j)
t(6) = c(j - 1) * b(j) - g(j - 1) * f(j) + d(j - 1) * d(j) - h(j - 1) * h(j)
t(7) = c(j - 1) * f(j) + g(j - 1) * b(j) + d(j - 1) * h(j) + h(j - 1) * d(j)
a(j) = t(0); b(j) = t(2); c(j) = t(4); d(j) = t(6)
e(j) = t(1); f(j) = t(3); g(j) = t(5); h(j) = t(7)
PRINT : PRINT SPC(16); t(0); t(1); t(2); t(3)
PRINT SPC(16); t(4); t(5); t(6); t(7)
NEXT j
RETURN
```

current pointing toward the load. If you reverse the progression, the direction of the current is usually reversed, and the location of elements A and D is reversed to conform to the reciprocity theorem. The matrices for a single series component and a single shunt component are as follows:

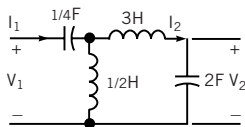


You can use these simple matrices to represent RLC components, transformers, and bridged taps in telephone cables. However, for a telephone cable, you must consider the reflected wave. Thus, you must express the A, B, C, D elements by the following hyperbolic functions:

$$\begin{aligned} A &= \cosh(\gamma d), \\ B &= Z_0 \sinh(\gamma d), \\ C &= (1/Z_0) \sinh(\gamma d), \text{ and} \\ D &= \cosh(\gamma d). \end{aligned}$$

where the propagation constant $\gamma = \alpha + j\beta$, d is the electrical length of the cable, and Z_0 is the characteristic impedance. In the United States, the nominal

value for Z_0 is 100V, which is also the specified value for the source and load impedance. Because γ is a complex number, A, B, C, and D are also complex numbers, or the polar equivalent thereof. These numbers are not difficult to calculate; however, the parameters depend on the application (references 1 and 2). The following example shows how to enter the data and read the results:



$$\begin{aligned} N_1 &= \begin{bmatrix} 1,0 & 0,-4 \\ 1/Z & 1,0 \end{bmatrix} & N_2 &= \begin{bmatrix} 1,0 & 0,0 \\ 0,-2 & 1,0 \end{bmatrix} \\ N_3 &= \begin{bmatrix} 1,0 & 0,3 \\ 0,0 & 1,0 \end{bmatrix} & N_4 &= \begin{bmatrix} 1,0 & 0,0 \\ 0,2 & 1,0 \end{bmatrix} \\ N_1 \cdot N_2 &= \begin{bmatrix} -7,0 & 0,-4 \\ 0,2 & 1,0 \end{bmatrix} & N_1 \cdot N_2 \cdot N_3 &= \begin{bmatrix} -7,0 & 0,-25 \\ 0,-2 & 7,0 \end{bmatrix} \\ N_1 \cdot N_2 \cdot N_3 \cdot N_4 &= \begin{bmatrix} 43,0 & 0,-25 \\ 0,12 & 7,0 \end{bmatrix} \end{aligned}$$

Within the solid bars, N_1 through N_4 show the entered data for the compo-

nents. The bottom line shows the matrix product as it appears on the screen. For this network, the matrix elements are $A=43+j0$, $B=0-j25$, $C=0+j12$, and $D=7+j0$. The open-circuit voltage ratio is $V_1/V_2=A=43$. The open-circuit transfer admittance $I1/V2=C=0+j12$. The input impedance is $Z_{IN}=A/C=0-j(43/12)$. Listing 1 uses easy-to-read and-compile QuickBasic. You can download this listing 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 #2455. (DI #2455)

REFERENCES

1. Rauschmayer, Dennis, *ADSL/VDSL Principles*, MacMillan Technical Publishing, 1999.
2. Chen, Walter, *DLS: Simulation Techniques and Standards Development of Digital Subscriber Line Systems*, MacMillan Technical Publishing, 1998.

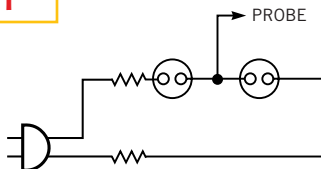
TO VOTE FOR THIS DESIGN,
CIRCLE NO. 426

Simple tester checks Christmas-tree lights

William Dias, Brown & Sharpe, North Kingstown, RI

WHY IS IT THAT YOU always test 48 bulbs before you find the bad one in a 50-light string? The simple circuit in Figure 1 allows you to divide and conquer, greatly reducing the time it takes to find the bad bulb. The circuit uses a pair of NE2 neon bulbs with current-limiting resistors. You can use a pair of Radio Shack 272-1100 bulb-resistor sets. It's convenient to house the tester in a clear piece of plastic tubing, with the probe tip emerging from one end and a light-duty power cord emerging from the other end. You place the bulbs in the tube such that one is close to the probe tip and the other is near the power cord, so it's easy to re-

Figure 1



A simple probe set cuts the time you spend troubleshooting a series-string light set.

member which bulb lit last. The probe tip connects to common point between the neon bulbs. It consists of thin spring wire with all but the last 1/4 in. insulated. You

use the bare tip to make contact with the crimp connectors in the base of the bulbs.

Series-string Christmas-tree lights come in two types. The first type is the continuous-series string (Figure 2a). In this configuration, one wire from the plug goes from bulb to bulb until it reaches the last bulb. A return wire bypasses all the bulbs and returns to the plug. The second type is the alternating-series string (Figure 2b). In this connection, one wire from the plug goes to the first bulb, and the other wire from the plug goes to the second bulb. The connections then alternate through the string. To troubleshoot a defective continuous series string:

- Plug in both the tester and the bulb set.
- Insert the tip of the tester's probe into the wire hole in the base of the first bulb. One of the neon bulbs should light; remember which one.
- Move halfway down the set and insert the probe again. If the same neon bulb lights, then the problem is in the second half of the set. If the other neon bulb lights, then the problem is in the first half of the set. Either way, you are testing 25 of the 50 bulbs without breaking into a sweat.
- If the original neon bulb lights, move halfway down the remaining

part of the set and try again. If the other neon bulb lights, you must move back halfway to the last bulb you tested and try again. This process should allow you to find a bad bulb in a set of 50 in only seven steps. You know you have the bad bulb when inserting the probe tip into one side of the bulb lights one neon bulb and placing the tip in the other side lights the other neon bulb.

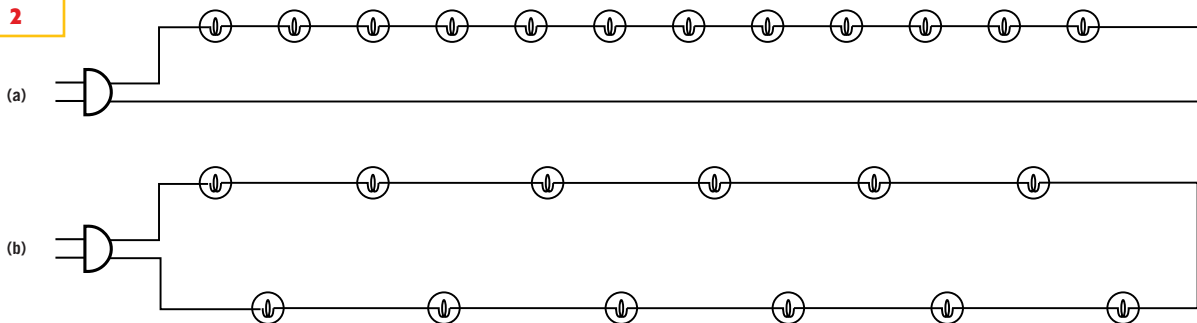
To troubleshoot a defective set with many bad bulbs, use the same process as above. At some point, you will reach the dead spot between two or more bad bulbs. When you reach this point, neither

neon lamp will light. Back up, just as if the other neon bulb had lit. You know you have a bad bulb if the probe lights when you plug it into one side and nothing lights when you plug it into the other side. Replace this bulb and start over.

To troubleshoot an alternating-series string, you must work in pairs. Test the first bulb, and one neon bulb lights. Test the second bulb, and the other neon bulb lights. Now move down the set an even number of lights and test the next pair of lights. When you pass the bad bulb, the same neon lamp lights for both series-string bulbs. (DI #2457).

TO VOTE FOR THIS DESIGN,
CIRCLE NO. 427

Figure 2



Series-string light sets come in two flavors: a continuous-series string (a) and an alternating-series string (b).

Power meter uses low-cost multiplier

Jeff Kotowski and Alan Johnston, National Semiconductor, Grass Valley, CA

A POWER-METER CIRCUIT typically requires either an analog or a digital multiplying circuit. These circuits can be complex, finicky, or expensive. A simpler way to achieve the multiplication first converts the current to a duty cycle proportional to the current. You can use this duty cycle to gate the input voltage; the gating effectively provides a multiplication function. A lowpass filter then provides an output voltage proportional

to power. The method sounds convoluted, but the implementation is simple. **Figure 1** shows the complete power-meter circuit. The National Semiconductor LM3812M-7.0 IC senses current and delivers a duty cycle proportional to the current. The current relates to the duty-cycle output as follows:

$$I = 22 \cdot (\text{DUTY_CYCLE} - 0.5).$$

The desired output is a voltage equal to

the input power divided by 10. The divide-by-10 operation keeps the voltage within a manageable range. The output is, then:

$$V_{\text{OUT}} = V_{\text{IN}} \cdot I / 10 = 2.2 \cdot \left[\text{DUTY_CYCLE} \cdot V_{\text{IN}} - \frac{1}{2} \cdot V_{\text{IN}} \right].$$

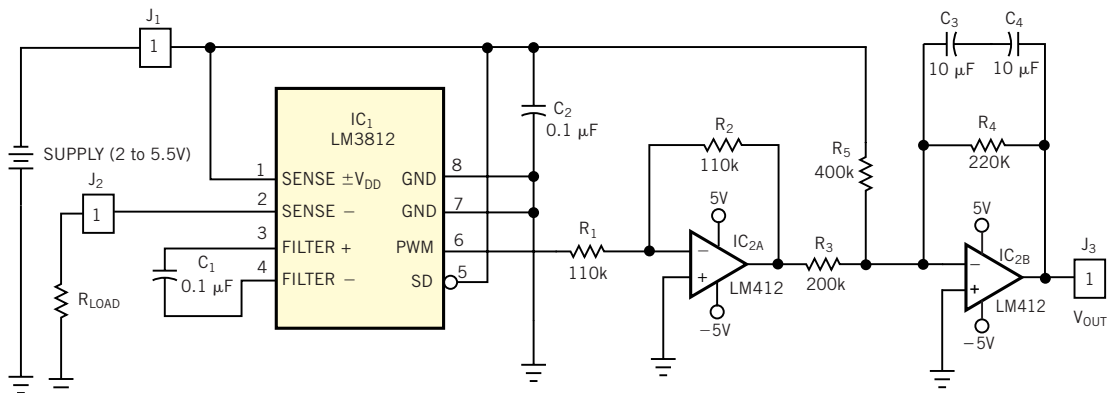
To subtract the $\frac{1}{2}V_{\text{IN}}$ term in the equation, the first op amp inverts the power signal. The second op amp adds in the

offset and again inverts the signal. Capacitors C_1 and C_2 provide filtering. These capacitors connect back-to-back to achieve nonpolar operation. Tests with input currents of -7 to $+7A$ and with a source voltage of 2 to 5.25V showed bet-

ter than 3% accuracy over the range of 0 to 25W. (DI # 2458)

TO VOTE FOR THIS DESIGN,
CIRCLE NO. 428

Figure 1



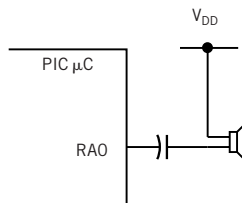
A low-cost duty-cycle IC forms the heart of a power-meter circuit.

Add music to your next project

Rodger Richey, Microchip Technology Inc, Chandler, AZ

ADDING MUSIC TO YOUR next design is as simple as using one I/O pin to drive the speaker and approximately 150 words of program memory to play the music. You can store the musical notes and durations internally in program memory or externally in a serial EEPROM. Frequently, implementing musical playback requires an external processor with memory or a specialized melody device. Both cases can bring increased cost and can limit the number of songs you can play back. The approach described here uses a Microchip Technology midrange PIC μC with a free 8-bit timer resource. Two implementations are possible. The one presented here uses internal program memory to store the musical notes. This method allows as many as 127 notes per song. The other technique is to use an ex-

Figure 1



You need only one I/O pin and a capacitor to generate music with a PIC μC .

ternal serial EEPROM to store the notes.

The first important point to consider is that the musical playback is completely interrupt-driven. Therefore, the main application can run in the foreground and can accept interrupts as necessary from the musical playback. Timer0 controls the playback; you can find it on any of the midrange PIC μC s—from the eight-pin PIC12C671 to the 68-pin

PIC16C924. This timer is a generic 8-bit timer with an 8-bit prescaler. You can download the assembly 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 #2556. The code is designed to run a PIC μC at 4 MHz. You can accommodate other frequencies by changing defines in the code. At 4 MHz, you use a 1-to-8 prescaler to provide the correct frequencies for the notes to play. Again, you can modify the prescale value to suit the desired operating frequency by changing the value in the Option register. One I/O pin provides the musical notes to the speaker. **Figure 1** shows the simple connections. You can modify the source code and hardware to drive the speaker differentially using two I/O pins.

Toggleing an I/O pin at twice the fre-

quency of the desired note generates each note. This technique creates software-based pulse-width modulation with a 50% duty cycle and a tone at the desired frequency. At each Timer0 interrupt, the routine reloads TMR0 with the number of ticks for the desired frequency. You use the following formula to calculate the number of ticks for each note:

$$\text{TICKS}(\text{Freqx}) = [256 - ((\text{CLOCK}/4/\text{PRESCALER}/\text{Freqx}/2) - 1)]$$

Using $\text{CLOCK}=4\text{MHz}$, $\text{PRESCALER}=8$, and $\text{Freqx}5523$, the calculated number of ticks for an A tone is 137. The number of ticks required is actually 118, but by subtracting this number from 256 in the formula, you can get the value to load into TMR0. A table designated Notes in the source code lists only seven notes, but it's easy to add more. You can determine the duration of each note by counting the number of Timer0 inter-

rupts. Because the timer interrupt period differs for every note, the number of Timer0 interrupts differs for every note. A table called Durations in the source code contains the note duration for the corresponding note in the Notes table. On each Timer0 interrupt, the Notes value loads into TMR0 and the note duration decrements. At the end of each note, a pause of $1/64$ occurs to emphasize the note you are playing. The routine creates the pause by not toggling the output pin during the pause interrupts. Again, each note requires a different number of interrupts to generate a pause of a set duration. To simplify the pause generation, the program subtracts the number of interrupts required for a pause from each note duration. The routine creates the pause by setting the timer such that the pause occurs from one timer interrupt. If the clock frequency is such that you cannot create the pause with one interrupt, you can alter the value `STOP_LENGTH` in the source code. The program loads

the next note to play after the pause interrupt. A 0 in the Notes and Durations tables in the source code denotes the end of the song. At this point, the program plays the song again.

Using internal memory to store the notes and durations has some limitations. Without some extra table-handling code, tables are limited to 255 elements and must start on a 256-byte boundary. Because the duration values are 16 bits long, the number of notes the PIC μC can play is 127. Depending on the size of program memory, you can place many songs in 256-byte blocks and play them individually. By using two more I/O pins (SCL and SDA), the PIC μC can interface to an external serial EEPROM and play any number of notes up to the maximum memory size divided by 3. A 24LC01B from Microchip Technology can hold 341 notes plus corresponding durations. (DI #2456)

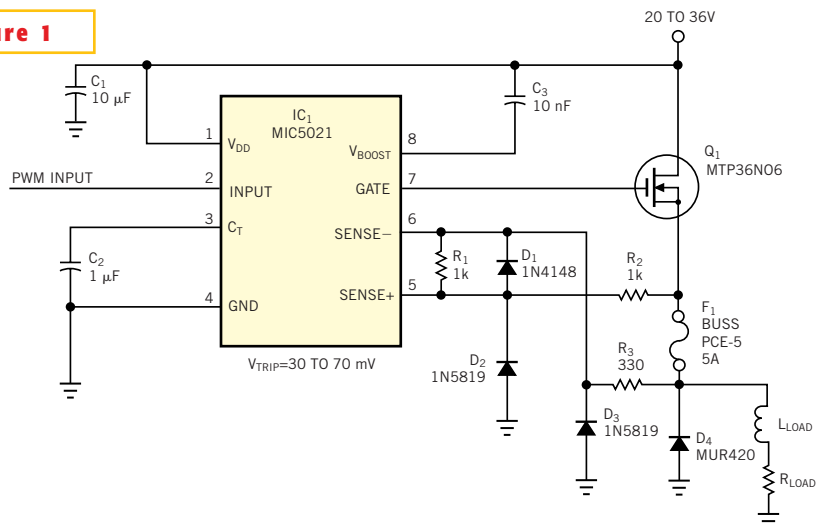
TO VOTE FOR THIS DESIGN,
CIRCLE NO. 429

PWM circuit uses fuse to sense current

Eugene Kaplounovski, Nautilus International, Burnaby, BC, Canada

HIGH EFFICIENCY AND, hence, low loss is a usual design goal for a PWM circuit. **Figure 1** shows one of several channels of low-loss PWM control for hydraulic valves used in heavy industrial machinery. These valves usually have dc resistance of approximately 12Ω and substantial inductive reactance. The operating voltage is 24V, and the PWM circuit usually operates at frequencies of 50 to 500 Hz to prevent valve sticking. To minimize on-resistance losses in the power MOSFET, the circuit uses a Motorola MTP36N06, even though that device's current-handling capacity greatly exceeds the load requirements. A Micrel MIC5021 high-side driver provides gate control; its high slew rate minimizes the MOSFET's switching losses. To implement the overload-protection feature in

Figure 1

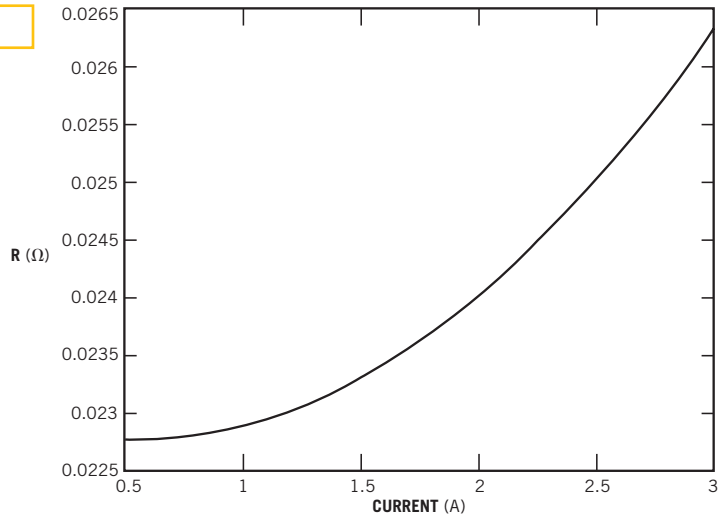


A fuse does double duty as a protection device and a current-sensing resistor.

the gate driver, a low-resistance current-sensing resistor is necessary. Safety requirements and common sense mandate protection against catastrophic component failure. Unfortunately, the resistance of the fuse contributes to circuit losses. The combined losses in the fuse and current-sensing resistor are comparable with those in the rest of the circuit.

You can sometimes replace the current-sensing resistor by the fuse, thereby eliminating one of the two sources of loss. The dc resistance of a Buss PCE-5 5A fuse is 20 to 30 m Ω , which is quite close to the value that the overload-protection circuit requires. The trip point of the MIC5021 overcurrent comparator is nominally 50 mV, but it may vary from 30 to 70 mV. Such wide tolerance makes setting a precise trip point with a precision resistor impossible. Another of the fuse's benefits is its positive temperature coefficient of resistance. The comparator's trip differential voltage also has a positive temperature coefficient. These two temperature coefficients track somewhat, offering some temperature compensation. **Figure 2** plots the fuse-resistance behavior. The curve is an approximated mean value of resistance versus current. Data for the graph uses 10 sam-

Figure 2



Self-heating causes a positive temperature coefficient of resistance in the fuse in Figure 1.

ples of the fuse and a second-order polynomial function using Matlab software. According to Micrel's data sheet, the trip point is a linear function of the ambient temperature.

R_2 and R_3 , along with D_2 and D_3 , provide protection against negative inductive-kick spikes at the comparator's inputs. Adding R_1 allows adjustment of the current trip point. D_1 does not conduct during normal operation but protects the

comparator from excessive differential voltage if the fuse fails. R_2 and R_3 limit the load current under this condition to a low and safe level. Disconnecting the fuse during experimentation leads to immediate shutdown, with only short (several-microsecond) pulses at the output. C_2 determines the time between the circuit's attempts to restart. (DI #2459)

TO VOTE FOR THIS DESIGN,
CIRCLE NO. 430

Bias supply accepts high inputs

Robert Sheehan, Linear Technology Corp, Milpitas, CA

WHEN YOU DESIGN dc/dc converters, it is often necessary to generate a bias supply to operate the control circuitry from the raw input voltage. Many methods are available for configuring the bias supply, each with its own benefits and shortcomings. Some methods use a "trickle-charge" start-up circuit, with a back-feed winding to provide power under normal operating conditions. With a low parts count and cycling short-circuit protection (if leakage inductance to the back-feed winding doesn't cause cycling to stop), this configuration is widely used in the power-supply industry. However, the configuration suf-

fers from a long turn-on time and a limit on the capacitive load into which the converter starts up. Another popular approach uses a transistor-based pass regulator, often in conjunction with an overwinding to keep power dissipation low in normal run conditions. Although this method can provide a fast start-up, it generally exhibits high power dissipation during an output short circuit. Attempts to gate this type of circuit can be messy and complicated. A buck regulator overcomes the disadvantages but can be complex and costly. The circuit in **Figure 1** uses IC₁, an LT1431 shunt regulator, low-cost transistors, and an off-the-shelf in-

ductor to form a high-voltage converter for use as a bias supply.

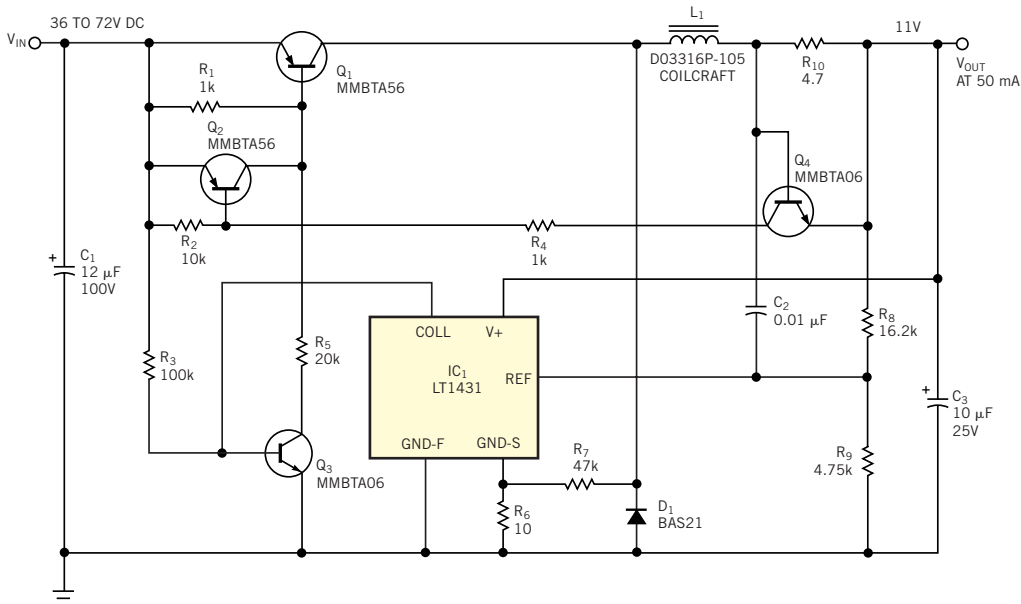
The circuit needs no bleeder resistors or tertiary winding. The selection of Q_1 and Q_3 limits the maximum input voltage to 80V in the circuit as shown. The circuit operates as a hysteretic regulator, also called a ripple regulator, relaxation oscillator, or bang-bang controller. Positive feedback comes from R_6 and R_7 ; negative feedback comes from C_2 , R_8 , and R_9 . With a 48V input, inductor current is discontinuous, and the switching frequency is 50 kHz. Efficiency is typically 74% with a 48V input and a 50-mA load, an acceptable figure for bias-sup-

ply power. Q_2 and Q_4 provide short-circuit protection; thus, the design is robust. Short-circuit current is typically

120 mA. You can repeatedly hot-plug the circuit with no adverse effects. (DI #2454)

TO VOTE FOR THIS DESIGN,
CIRCLE NO. 431

Figure 1



A shunt-regulator IC and a handful of low-cost components form an efficient, robust bias supply.

Method provides simple error-rate generator

JM Williams, Marconi Communications, Liverpool, UK

TESTING THE PERFORMANCE of a system being fed random errors can be tricky; it is not easy to add an accurate, low-level bit-error rate to a data stream. The technique shown in **Figure 1** (page 121) uses a PC, some simple software, and a few external components or an FPGA. The PC computes an n-bit threshold from the chosen bit-error rate, which then downloads to the FPGA. Using a magnitude comparator, the FPGA compares the threshold with the contents of the linear-feedback-shift register (LFSR), which clocks at the same rate as the data stream. If the LFSR's contents are lower in magnitude than the threshold, an error occurs. If they are greater than or equal to the threshold, no error occurs. Thus, if you need 50% errors, you set the threshold to 1000...0; if you need 25% errors, you set the threshold to 0100...0, and so on. The PC translates from rates ex-

pressed as, for example, 1 in 10^4 , to such a threshold. The n-bit threshold should be long enough to achieve the desired accuracy. The LFSR must be longer than the threshold to minimize the effects of the LFSR's not generating the all-zeros state. Making the LFSR twice the length of the threshold vector is satisfactory. It is also necessary to randomize the starting point of the LFSR. You can do this by downloading random digits from the PC at the same time the threshold downloads. You use the serial data port and a UART to control the FPGA and load data into it, by choosing one of the UART's output bits to set the Run/Load mode and two other output bits to drive the Data and Clock lines. You then write routines to translate loading information into sequences of asynchronous characters. Other methods are obviously available. Building the UART into the FPGA

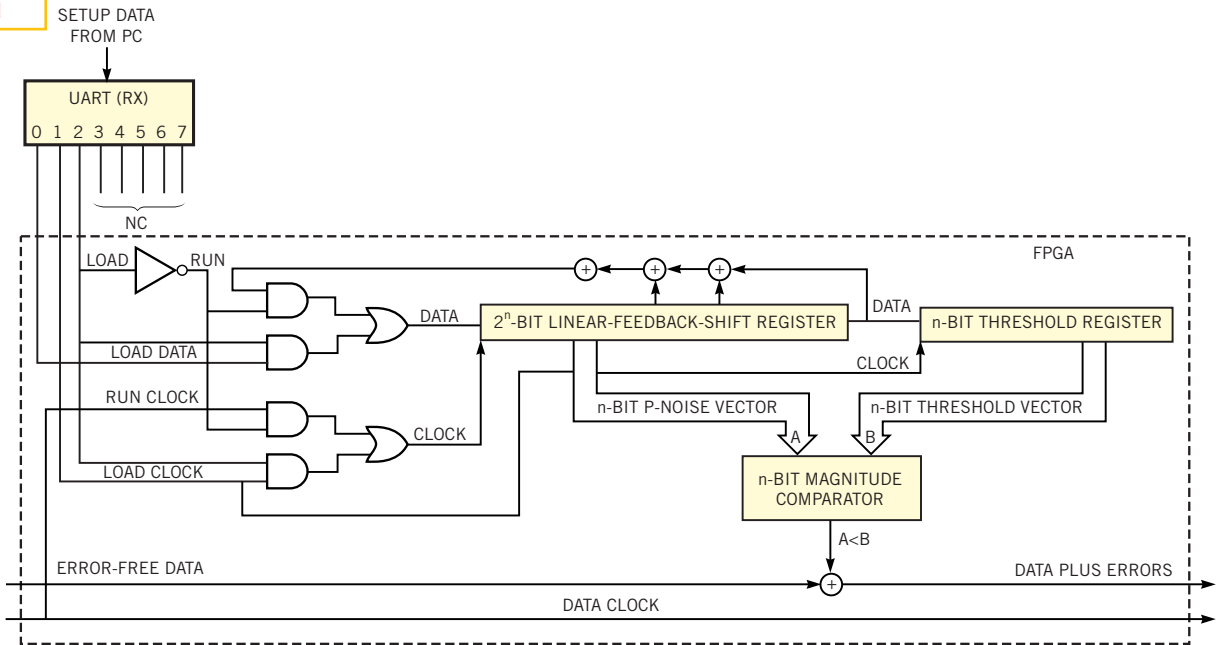
design is another clear improvement. If the threshold register is 32 bits long, the LFSR should be 64 bits long. You should choose the taps to give a sequence length of $2^{64}-1$. In other words, the choice should be in accordance with a primitive polynomial. **Reference 1** gives a useful list of taps that yield a maximal length sequence. One choice (chosen at random) requires three taps located at stages 2, 19, and 25 (**Figure 1**). The speed of operation depends on the capabilities of the FPGA. With modern FPGAs and a little care in the design, rates in the hundreds of megahertz are possible. (DI #2460)

REFERENCE

- Schneier, Bruce, *Applied Cryptography, Second Edition*, John Wiley, pg 408.

TO VOTE FOR THIS DESIGN,
CIRCLE NO. 432

Figure 1



Using a PC and an FPGA, you can generate any percentage of random errors in a data stream.