



Edited by Brad Thompson

## High-side current-sensing switched-mode regulator provides constant-current LED drive

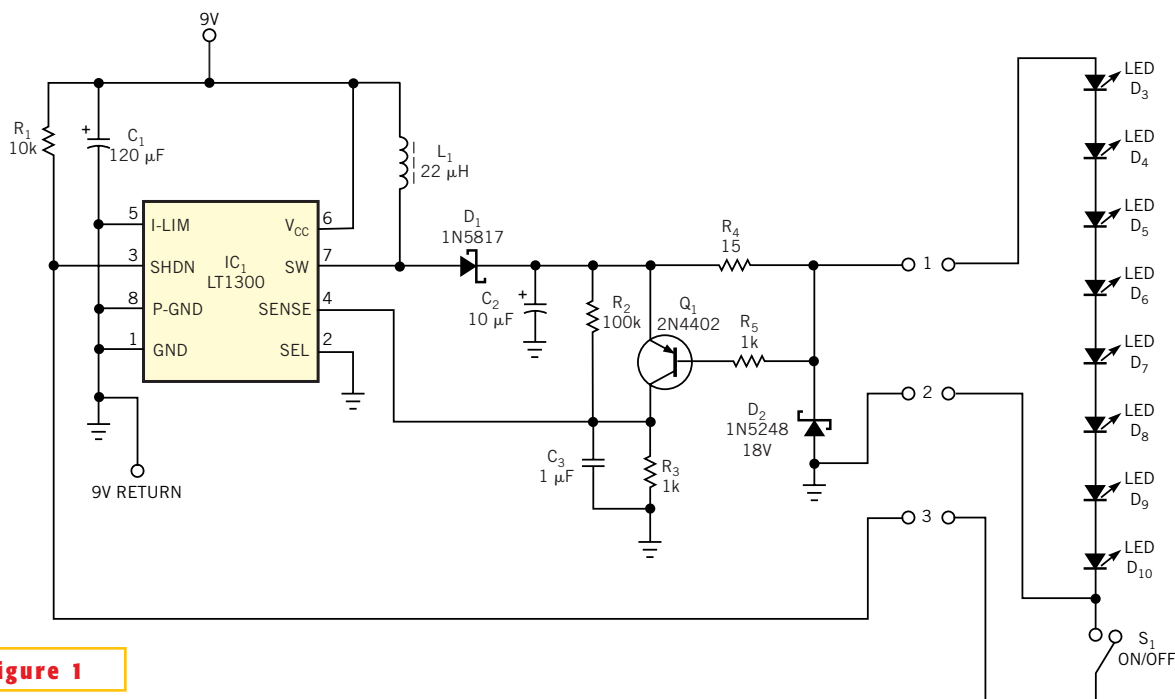
Bradley Albing, Philips Medical Systems Inc, Cleveland, OH

**M**ANY SUITABLE CIRCUITS exist for driving LEDs in constant-current mode and from low-voltage sources. For example, references 1, 2, and 3 show circuits that use switched-mode-regulator ICs and low-voltage sources to supply LED current. To produce a constant-current output using the circuit in Reference 2, you configure the regulator IC as a boost-mode switcher and use a resistor to sense the load current flowing in the LED string's low side, or negative-return leg. The sense resistor produces a proportional voltage that's applied to the LT1300's Sense input through a 2.5V ref-

erence diode. A voltage of 3.3V appearing at the LT1300's feedback-input terminal, Pin 4, indicates that the circuit's output is within regulation.

In applications that require a series string of LEDs to operate with its low side connected to ground, current sensing must take place in the string's high side. You can use either a rail-to-rail op amp and a handful of passive components or a dedicated current-sensing IC, such as Maxim's MAX4073T, to accomplish high-side sensing. However, adding a current-sensing IC increases circuit cost. To complicate matters, in this applica-

High-side current-sensing switched-mode regulator provides constant-current LED drive .....	95
Microcontroller's DAC provides code analysis .....	96
Two gates and a microprocessor form digital PLL .....	98
Simple sine synthesizer generates 19-kHz pilot tone for FM baseband signal .....	100
<b>Publish your Design Idea in EDN. See the What's Up section at <a href="http://www.edn.com">www.edn.com</a>.</b>	



**Figure 1**

A single switched-mode-regulator IC drives a series-connected string of LEDs in constant-current mode.

tion, only three conductors are available to connect a remotely mounted LED string,  $D_3$  through  $D_{10}$ , and on/off switch  $S_1$  to the regulator circuitry.

In this Design Idea, an LT1300,  $IC_1$ , boosts 9V to drive the LED string, which presents a total forward-voltage drop of approximately 12V (Figure 1). Resistor  $R_4$  serves as a current-sense resistor. At a cur-

rent of approximately 40 mA, transistor  $Q_1$  conducts and forces current through  $R_3$ , developing sufficient voltage drop to produce the requisite 3.3V at Sense Pin 4 of  $IC_1$ , bringing its output current into regulation. Zener diode  $D_2$  limits the regulator's output voltage in case the LED string or connector opens. Switch  $S_1$  turns on the circuit by grounding  $IC_1$ 's Pin 3. □

REFERENCES

1. LT1300 data sheet, Linear Technology Corp.
2. Application Note 59, Linear Technology Corp.
3. Caldwell, Steve, "1.5V battery powers white-LED driver," *EDN*, Sept 30, 2004, pg 96.

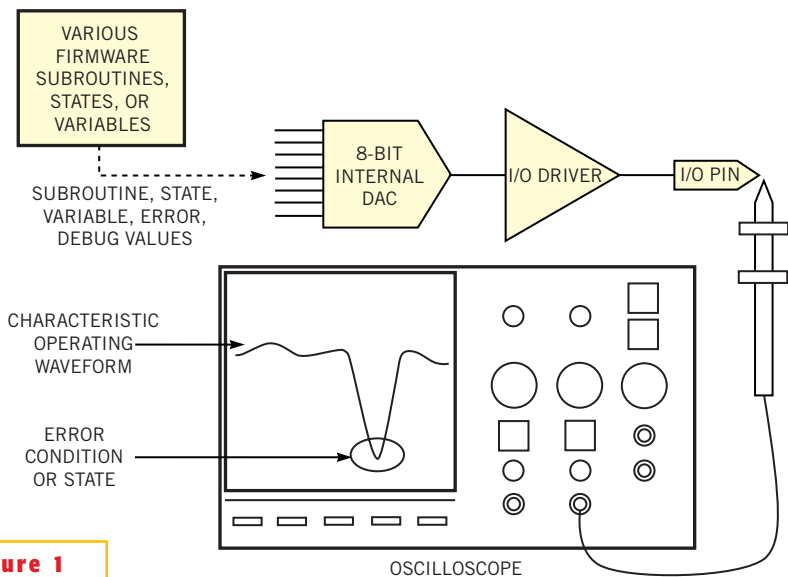
# Microcontroller's DAC provides code analysis

Dave Bordui, Cypress Semiconductor, Heathrow, FL

**F**INDING OUT WHERE your microcontroller's firmware spends most of its time can be a tedious task when you use a conventional in-circuit emulator and breakpoint techniques. Other such tasks include discovering why a state machine doesn't work as you intended and where your code goes during real-time operation or during an error condition. Classic debugging methods can also become cumbersome when you attempt to observe error states or debug a program-flow problem. Fortunately, a technique that takes advantage of a feature that many microcontrollers now include offers a simple debugging aid that allows designers to easily monitor these and other operations.

The technique uses the DAC in microcontrollers such as Cypress Microsystems's PSoC (programmable-system-on-chip) family (Figure 1). These devices provide a microcontroller core and an array of mixed-signal building blocks that includes true DACs that can deliver fixed dc levels. Other types of DACs that deliver pulse-width-modulated outputs are unsuitable for this application. To use this technique, you create firmware "state constants" that represent operating states of your design. If your code structure comprises a state machine or contains a large "switch/case" statement, then you have already defined these constants. Otherwise, you can easily add the constants as needed.

Once you define the constants, you enable a DAC and configure it to drive one



**Figure 1**

Put an unused internal DAC to work as a code analyzer and see where a microcontroller's routines go wrong.

of the microcontroller's unused analog output pins. Then, you write the state constant to the DAC whenever the firmware enters a particular location. If you use an 8-bit DAC, you can also monitor the value of any 8-bit variable. Next, you connect an oscilloscope's vertical input to the DAC's output pin and observe its output voltage, which the instrument displays as a characteristic waveform that represents the firmware's operation. If your oscilloscope includes measurement cursors, you can easily determine which

portion of the firmware is executing simply by measuring the DAC's dc output voltage at a given time.

You can define state constants to highlight certain conditions. For example, by reserving all state constants' values greater than 127 for error states, you can set the oscilloscope's horizontal sweep generator to trigger at a level that indicates an error. As a precaution, make sure that the microcontroller's DAC operates within its allowable update-rate range. □

# Two gates and a microprocessor form digital PLL

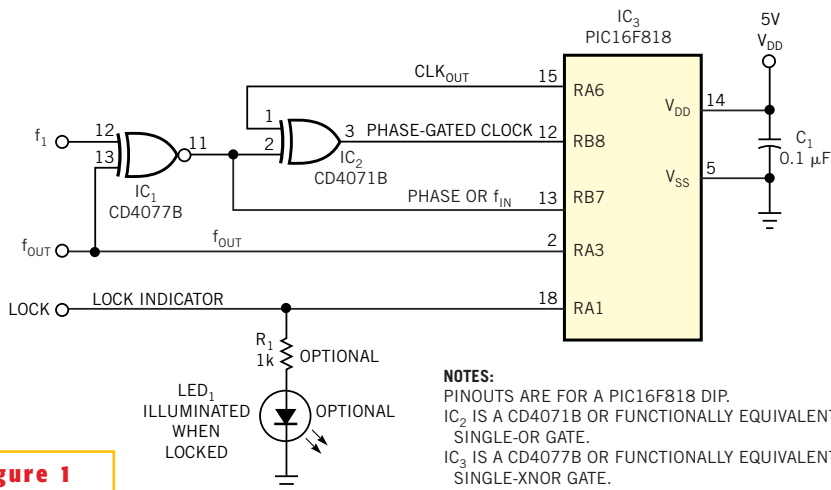
Kenneth Martin, TareTronics Inc, Corinth, MS

**Y**OU CAN USE Microchip's low-cost PIC16F818 microprocessor and a pair of gates to construct a digital PLL that can clean noisy digital signals over a range of 4 to 40 kHz. Featuring programmable lock range, phase differential, and loop gain, the digital-PLL engine and lock detector can extract clock and data information from noisy, short-range radio signals (**Figure 1**). When you construct it using a QFN-packaged microprocessor and discrete single-gate logic devices, the circuit occupies a pc-board area that's approximately as large as an aspirin.

**Figure 2** is analogous to a first-order analog PLL (**Reference 1**). With its associated period register, Timer 2 functions as a DCO (digitally controlled oscillator). When Timer 2's count matches the byte in its period register, the timer generates an interrupt. The microprocessor then computes a byte of information that writes to the period register to set the duration of the next half-period. In addition, the interrupt toggles an I/O port's output to produce square-wave signal-output frequency, which drives one input of the external XNOR (exclusive-nor) gate, IC<sub>1</sub>. The external input signal's input frequency drives the XNOR gate's remaining input to produce an output signal, which represents the phase difference, between the output and the input frequency. This XNOR-based phase detector provides good performance with noisy digital input signals.

The phase difference signal's duty cycle remains linear over a 0 to 180° range of two same-frequency signals. Applying the phase detector's output along with clock-signal clock frequency to OR gate IC<sub>2</sub> produces an output burst of 2-MHz clock pulses during each half-period interval of output frequency. The burst's length and the number of clock pulses it contains depend directly on the duty cycle or phase interval of the output frequency relative to the input frequency.

The circuit applies phase-difference pulses from IC<sub>2</sub> to the internal prescaler associated with IC<sub>3</sub>'s Timer 1, which divides them by a preset factor of one, two,



**Figure 1**

**NOTES:**  
 PINOUTS ARE FOR A PIC16F818 DIP.  
 IC<sub>2</sub> IS A CD4071B OR FUNCTIONALLY EQUIVALENT SINGLE-OR GATE.  
 IC<sub>3</sub> IS A CD4077B OR FUNCTIONALLY EQUIVALENT SINGLE-XNOR GATE.  
 CONNECT PIN 14 TO V<sub>DD</sub> AND PIN 7 TO GROUND ON IC<sub>1</sub> AND IC<sub>2</sub>.  
 GROUND ALL UNUSED GATE INPUTS.

**This microcomputer-based digital-PLL circuit locks to signals over a 4- to 40-kHz range and requires a minimal number of components.**

four, or eight. During each of the output frequency's half-periods, Timer 1 accumulates (integrates) the prescaled pulses.

The interrupt-service-routine software for Timer 2 closes the loop and determines the digital PLL's key parameters. This routine comprises 19 instructions that execute in about 10 μsec when IC<sub>3</sub>'s internal clock oscillator runs at 8 MHz. After each Timer 2 interrupt, the interrupt-service routine toggles the output frequency, checks for phase lock, and then divides the output of Timer 1 by two, making K equal to one, two, four, eight, or 16. The routine subtracts the resulting value from N<sub>0</sub> and writes the difference to Timer 2's period register, which sets the length of the output frequency's next half-period. Although some interrupt-service-routine operations slightly modify the result, this count is typically as follows:

$$\frac{f_{\text{CLK}}}{2f_{\text{OUT}}} = N_0 - \frac{|\Phi_{\text{COUT}}|}{K} \quad (1)$$

For phase lock, the output-frequency half-period must equal the input-frequency half-period. The computed variable half-period count adjusts the output's frequency and phase. If the input frequency is within lock range, the vari-

able count changes in the direction necessary to achieve and maintain phase lock between the input and the output frequency. As an example, assume that the input frequency is 10 kHz; the maximum clock-frequency cycle count for each output-frequency half-period, N<sub>0</sub>, is 110; the division factor for the clock-frequency count that represents the phase of the output frequency, K, is two; and the output frequency is phase-locked to the input frequency. The half-period of 10 kHz is 50 μsec, or 100 counts, when the clock frequency is 2 MHz. Substituting these values in **Equation 1** and solving for the variable phase count yields a value of 20, which corresponds to a phase of 0.1, or 36°. Thus, with these parameters, the digital PLL's output frequency locks to the input frequency with a phase difference of 36°.

If the input frequency decreases, its half-period lengthens, and the variable phase count becomes smaller. According to **Equation 1**, after the division factor divides the variable phase count and you subtract it from the maximum half-period clock-frequency count, the half-period of the output frequency increases, lowering the output frequency and driving it toward a new match with the input

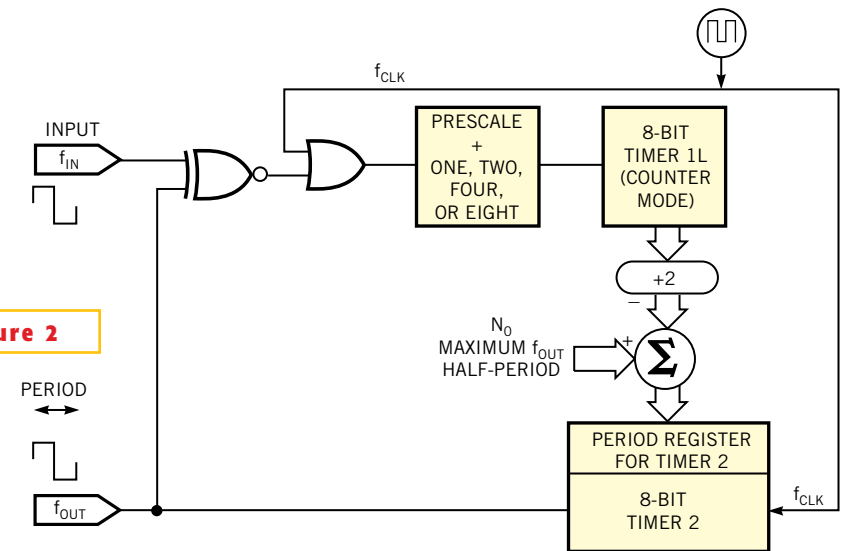
frequency. If the input frequency increases, the reverse occurs.

**Equation 2** defines the digital PLL's operation in phase-lock frequency, and design-selected system parameters:

$$f = \frac{f_{CLK}}{\left(1 + \frac{2\Phi}{K}\right) + 2.5} \text{ Hz}, \quad (2)$$

$\sim 34 \leq N_0 \leq 255$ ;  $K = \text{two, four, eight, or } 16$ ; and  $0 \leq \Phi \leq 0.5$ , where  $f$  is the frequency,  $f_{CLK}$  is the clock frequency,  $N_0$  is the maximum clock-frequency cycle count for each output-frequency half-period,  $\Phi$  is the phase of the output frequency relative to the input frequency, and  $K$  is the division factor for the clock-frequency cycle count that represents the output phase of the output frequency relative to the input frequency. Adding a constant value of 2.5 to the output frequency's period count compensates for interrupt-service-routine operations that slightly affect the timing. The calculated value of phase-lock frequency is accurate to within  $\pm 1.5\%$  over most of the PLL's usable range. Because the PLL comprises only digital circuits and software, operation with any combination of parameters is repeatable.

You can manipulate **Equation 2** to solve for any variable in terms of the remaining four. To calculate the upper and lower limits of the lock range, set  $\Phi$  at 0.5 and 0, respectively. To calculate the digital PLL's "center frequency," set  $\Phi$  at 0.25, which corresponds to a  $90^\circ$  phase angle. In the previous example, maximum frequency is 13,408 Hz, center frequency is 11,204 Hz, and minimum frequency is 8989 Hz. The lock range is 13,408 to 8989, or 4419 Hz. Increasing  $K$  to 16 yields a maximum frequency of 9544 Hz



**Figure 2** An XNOR-gate phase detector provides good performance with noisy signals, and a microprocessor handles signal processing.

a center frequency of 9266 Hz, a minimum frequency of 8989 Hz, and a lock range of 555 Hz.

Resolution of the DCO using Timer 2 establishes the time jitter of the output frequency relative to the input frequency. Depending on the integer count written to its period register, Timer 2 produces discrete frequencies for output frequency. When the input frequency falls between discrete output frequencies that two adjacent counts produce, the PLL switches between the counts to produce an averaged but jittery output-frequency signal at the same frequency as the input frequency. Using a relatively large value of  $N_0$  reduces jitter, whereas a smaller value increases jitter. To improve resolution and reduce jitter, you can increase the clock frequency to 5 MHz by configuring the microprocessor's on-chip oscillator to use an external 20-MHz crystal.

You can adapt the digital PLL's basic design to a variety of applications by modifying the software and extending the interrupt-service routine. For example, stopping updates to Timer 2's period register puts the PLL in "coast" mode. Other expansion possibilities include implementing more sophisticated lock-detection circuitry to determine whether the input frequency falls within a certain frequency range and making dynamic adjustments of the values of  $N_0$  and  $K$  for better performance. You can download **Listing 1**, which is the assembly-language source code, as well as the hex programming file for IC<sub>3</sub>, from the online version of this Design Idea at [www.edn.com](http://www.edn.com). □

#### REFERENCE

1. Gardner, Floyd M, *Phaselock Techniques, Second Edition*, Wiley-Interscience, 1979, ISBN 0-471-04294-3.

## Simple sine synthesizer generates 19-kHz pilot tone for FM baseband signal

Carlos Bernal and Diego Puyal, Departamento Ingeniería Electrónica y Comunicaciones, Universidad de Zaragoza, Zaragoza, Spain

**A** MULTIPLEX SIGNAL comprises baseband information transmitted on a stereo analog FM-broadcast system, plus one or more SCA (Subsidiary Com-

munications Authorization) channels (**Figure 1**). This Design Idea presents a low-cost method of generating the basic 19-kHz pilot tone. The 19-kHz pilot tone

comprises a baseband signal, and the L+R and L-R signals consist of DSBSC (double-sideband-suppressed-carrier) modulation centered at 38 kHz. For a re-

ceiver to correctly demodulate the signal, the transmitted pilot tone and L-R signal must synchronize at their respective zero crossings. In addition, any distortion in the pilot tone produces harmonics that can interfere with adjacent sections of the signal.

The low-distortion, 19-kHz pilot-tone generator comprises a resistive voltage divider,  $R_1$  through  $R_{11}$ , connected between the  $V_{CC}$  and  $-V_{CC}$  supply rails (Figure 2). The resistors' values are weighted to provide  $N=8$  approximate sampled values of a sine wave and are relatively low to present "stiff," low-impedance sources to eight-channel analog multiplexer  $IC_1$ . An up/down counter,  $IC_2$ , drives  $IC_1$  and takes advantage of a sine wave's inherent symmetry to enhance the resolution and reduce the distortion of the 19-kHz pilot sine wave.

In effect, analog multiplexer  $IC_1$  acts as

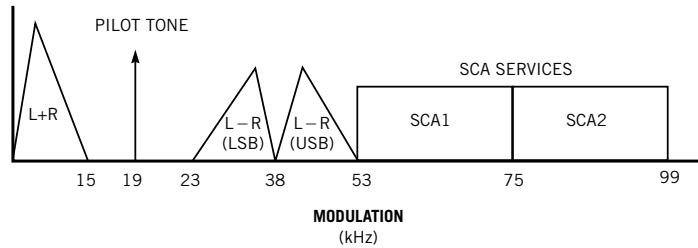
a zero-order hold circuit, producing an  $N$  times Nyquist oversampled sine wave of frequency  $f_{SIN}$ , plus several attenuated alias frequencies centered at:  $f_{ALLIAS} = m \times (2 \times N \times f_{SINE})$ , where  $m=1, 2, 3$ . For most applications, a simple passive RC filter at the multiplexer's output adequately removes the alias frequencies. Binary counter  $IC_3$  generates a 608-kHz clock signal plus a 19-kHz up/down control signal for counter  $IC_2$ , and sections

of hex inverter  $IC_1$  serve as a crystal oscillator and buffer.

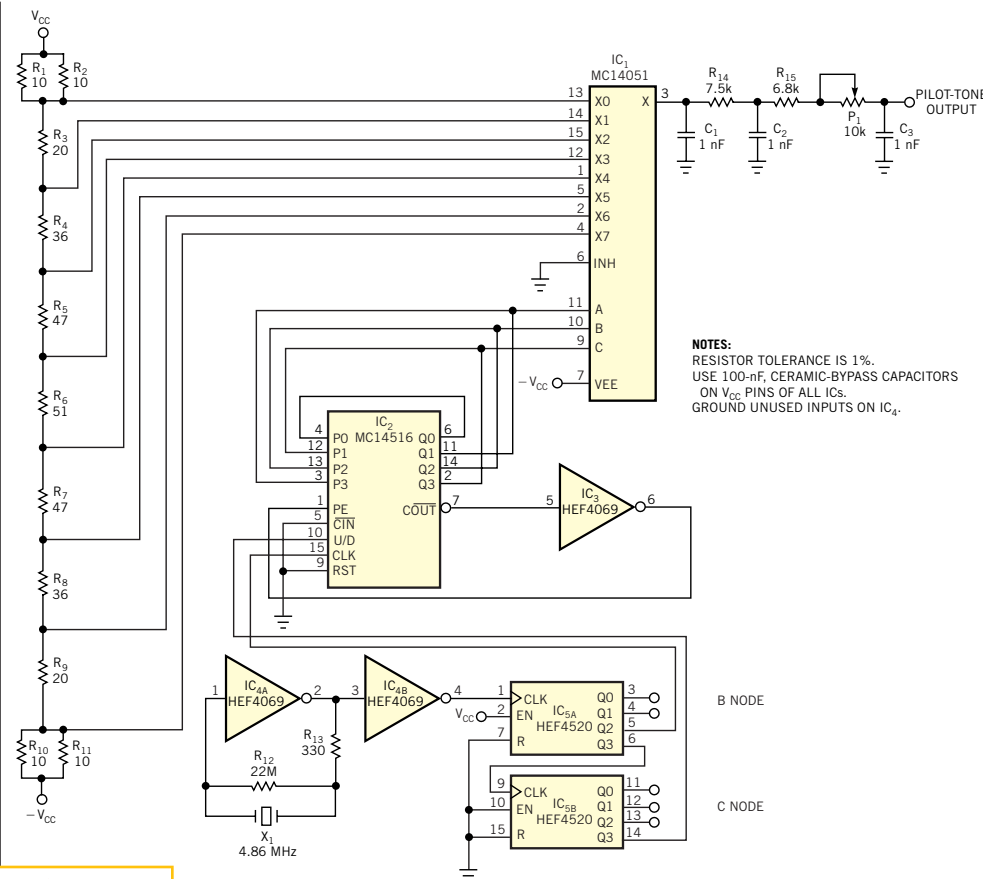
You can expand the basic circuit by duplicating the resistor network, multiplexer, and up/down counter. An external audio source drives the resistor network's upper and lower ends with L and R audio signals that have undergone lowpass-filtering to eliminate components with frequencies greater than 15 kHz. A 1.216-MHz signal clocks the second up/down counter and a 38-kHz up/down control signal derived from higher frequency taps on counter

$IC_2$ . The added circuitry generates the baseband L+R channel, and the L-R modulation in synchronism with the 19-kHz pilot tone because all clock pulses originate from a common counter. To produce the composite multiplexed signal, the outputs of both analog multiplexers sum in an external network.

Using the specified components, the circuit generates a 19-kHz pilot tone with harmonics 60 dB below the fundamental and synchronous with the maximums of the suppressed 38-kHz carrier. The same circuit structure produces L+R- and L-R-channel generation without changing components' values. Potentiometer  $P_1$  allows a  $90 \pm 10^\circ$  fine phase adjustment to correct distortion and to resynchronize at zero crossing. □



**Figure 1** A typical FM-broadcast signal contains a complex spectrum.



**NOTES:**  
RESISTOR TOLERANCE IS 1%.  
USE 100-nF, CERAMIC-BYPASS CAPACITORS  
ON  $V_{CC}$  PINS OF ALL ICs.  
GROUND UNUSED INPUTS ON  $IC_4$ .

**Figure 2**

The pilot-tone-generator circuit uses low-cost CMOS-logic circuits plus an analog multiplexer.