Edited by Bill Travis and Anne Watson Swager

Keep the heat down in power op amps

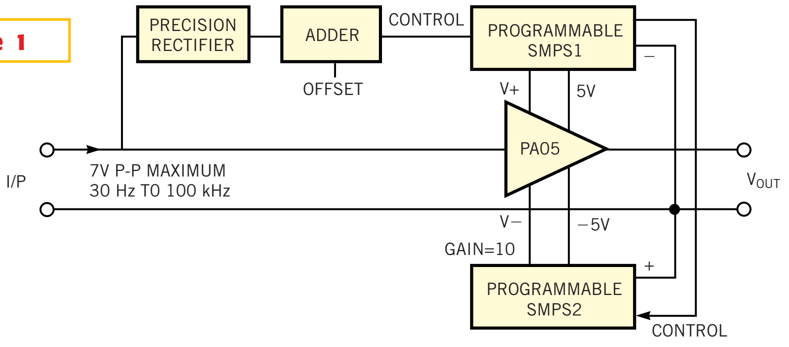
Vijay Damle, Digitronics, Pune, India

When you include a power op amp, such as PA05 from Apex Microtechnology, in your design, it is desirable to minimize the supply-to-output differential to a minimum to reduce power dissipation and to fully exploit the amplifier's output range. Our goal was to design a power amplifier to yield 70V p-p output at 10A with a fixed gain of 10 and a frequency of 30 Hz to 100 kHz. To obtain ±35V swing entailed dc supplies of approximately ±38V and two ±5V supplementary supplies. To derive the full 10A at lower voltage, you must reduce the supply voltage in proportion to the output voltage to decrease dissipation. In this case, the gain is fixed at 10. So, you can control the dc voltage proportional to the input voltage (**Figure 1**). SMPS1 and SMPS2 are identical voltage-programmed supplies (except for the ±5V supplementary supplies). The precision rectifier generates dc output proportional to the ac-input amplitude. To obtain approximately ±6V when no input signal is present, the circuit adds offset voltage to the signal.

As the input increases, the SMPS output increases from 6 to 38V for a 7V p-p ac input. To control the negative-side SMPS, you must transfer information from SMPS1 to SMPS2. The circuit in **Figure 2**, which generates current proportional to the input voltage, effects the transfer. Thus, SMPS2 generates an equal-value but opposite-polarity voltage to that of SMPS1. If you need higher output current, you may need to increase the voltage headroom, depending on the power amplifier you choose. Otherwise, you may experience output clipping. Note that for fast-changing input signals, the output may clip for a short time until the power-supply voltage rises. This phenomenon depends on the precision rectifier and the power supplies' response time.

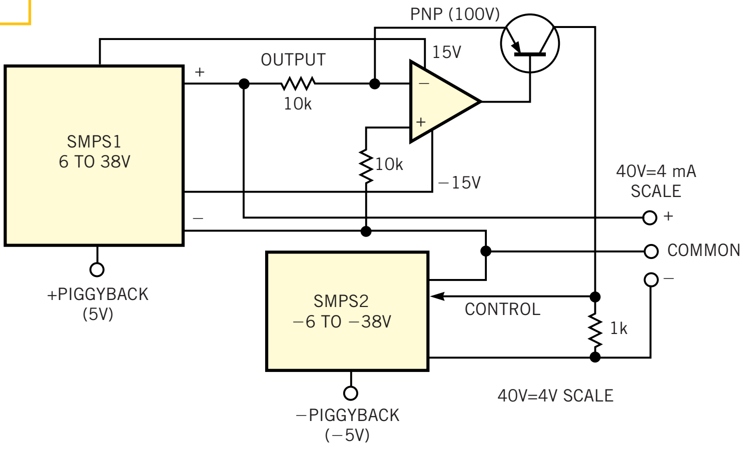
Is this the best Design Idea in this issue? Vote at www.ednmag.com/ednmag/vote.asp.

Figure 1



Power supplies track the input signal in bootstrap fashion, thus reducing power dissipation.

Figure 2



The switch-mode power supplies track each other with opposite-polarity outputs.

Buck converter works efficiently from phone line

Wayne Rewinkel, National Semiconductor, Santa Clara, CA

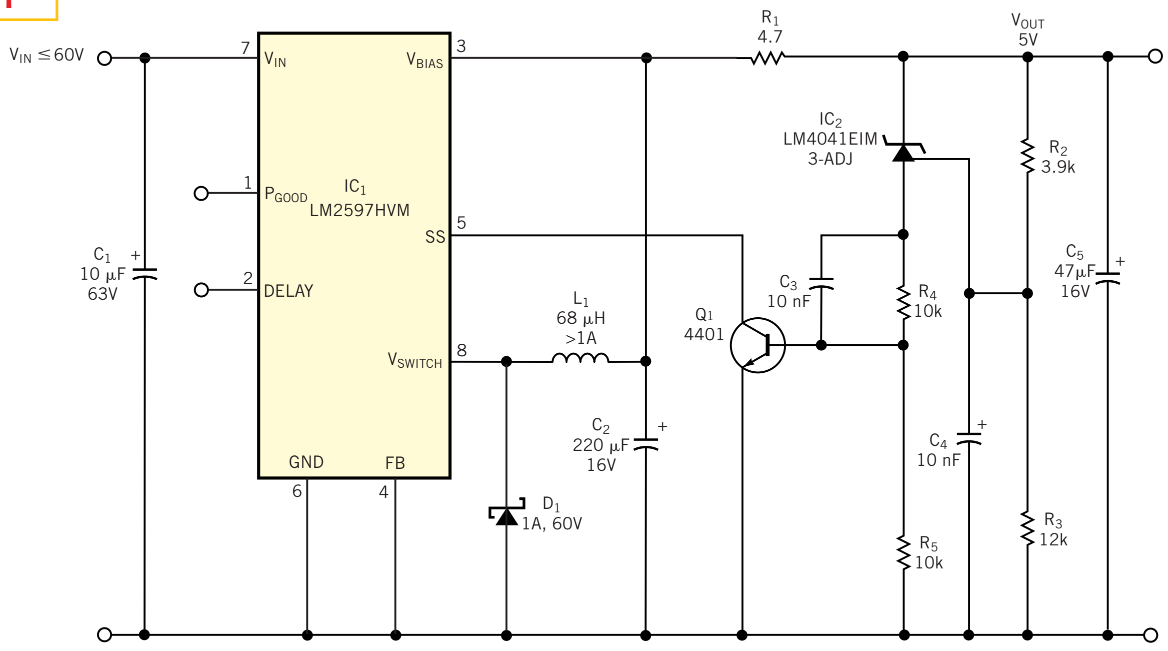
A SWITCHING CONVERTER provides an inexpensive way to generate 5V, 18 mA (48V, 5 mA maximum) directly from a standard phone line (**Figure 1**). The high input voltage and low available current require a unique design approach to achieve high efficiency. The circuit uses the LM2597HVM, which has a 60V rating and power-saving features. Its V_{BIAS} pin permits bootstrapping bias power whenever the output is higher than 4.4V. This feature reduces the bias current by at least a factor of four to a typical current of 1 mA. Additionally, the IC has a shutdown/soft-start pin that, when pulled low, shuts the regulator off (with 10-μA maximum quiescent current). When you release this pin, the IC starts switching with an increasing duty cycle. These features combine with an external comparator and voltage reference to provide the basis for a low-power switching regulator.

The circuit uses the LM4041 adjustable-voltage reference with Q_1 to form a small, low-power comparator. Its 1.23V threshold controls the regulator's output voltage according to the following expression: $V_{OUT}=1.23V\cdot(1+R_3/R_2)$. If the divided output voltage is below the threshold, IC_1 turns on, causing V_{OUT} to ramp up until the voltage crosses the threshold. Then Q_1 pulls Pin 5 low, forcing IC_1 into shutdown. IC_1 stays in this state until the output voltage again decays to the comparator's threshold and the cycle repeats. In this always-stable, discontinuous-switching mode, the inductor current is many times the load current, so its stored energy at turn-off always forces a slight overshoot that has an effect similar to hysteresis. This overshoot is the ripple voltage (ignoring capacitor ESR); you can estimate it from the energy transferred to C_2+C_5 during a switching cycle. The following equation gives ripple: $V_{RIPPLE}=\sqrt{((V_{OUT}\cdot V_{OUT})+L_1/(C_2+C_5))}$.

R_1 and C_5 remove the large ESR-induced spike from C_2 caused by the 1A peak charging current. This lowpass filter is small and inexpensive, and it allows the use of capacitors with almost any ESR ratings. C_3 and C_4 speed the comparator action and thus reduce output ripple. R_5 ensures that the 4041 receives sufficient bias current, and R_4 sets the maximum bias current. L_1 should have a value lower than 100 μH and must have a saturation rating exceeding 1A. Coilcraft's DO3316683 inductor fills the bill nicely. Smaller inductors degrade efficiency, because the 2597 goes into its pulse-current-limited protection mode. Larger values also degrade efficiency by using additional switching cycles. The circuit in **Figure 1** provides satisfactory results with input voltages of 10 to 60V. For an input of 48V, 5 mA, the available output current measures 34 mA. The output regulation is less than 2-mV output variation for inputs of 10 to 60V and load currents of 0 to 100 mA.

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

Figure 1



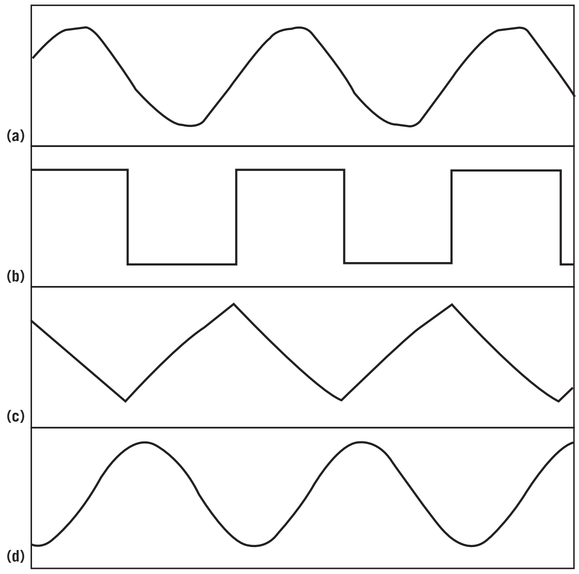
This inexpensive switching regulator derives its power directly from the phone line.

Sine reference is synchronous with ac line

P Seshanna, PhD, Assumption University, Bangkok, Thailand

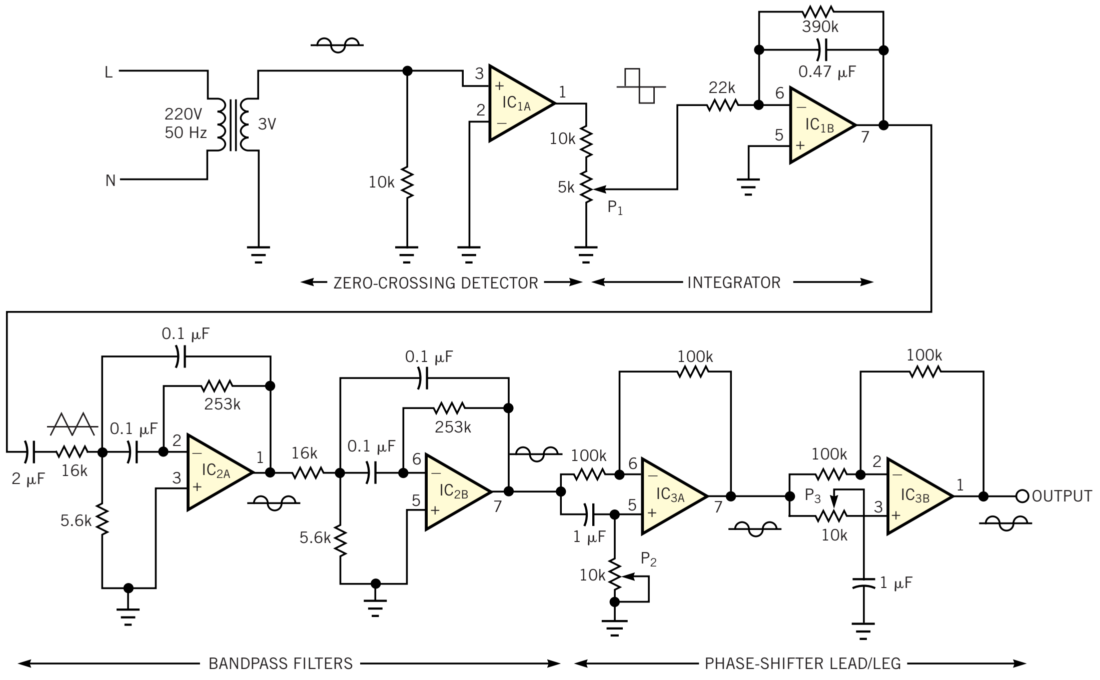
Many applications require a sinusoidal reference voltage synchronized to the ac line voltage. You cannot derive such a reference voltage directly from the ac line because the waveform of the ac line is distorted because of nonlinear loads connected to the line and because the amplitude of the line signal varies. Hence, you cannot use a simple step-down transformer to derive the reference signal. The circuit in **Figure 1** develops a line-synchronized reference signal whose phase and amplitude you can adjust using potentiometers P_1, P_2, and P_3. The circuit steps down and converts the line voltage to a square wave using the IC_{1A} zero-crossing detector. The integrator block, IC_{1B}, then integrates the square wave to produce a triangular wave. The triangular wave passes through two narrow bandpass filters comprising IC_{2A} and

Figure 2



Distorted line voltage (a) produces a square wave (b). An integrator creates a triangular wave (c), and bandpass filters produce a pure sine wave (d).

Figure 1



An op-amp circuit uses only resistors and capacitors to generate a line-synchronized sine wave.

IC_{2B}. The center frequency of the bandpass filters matches the 50-Hz line frequency. The sine-wave output from the filters passes through two phase shifters to set the phase at either lead or lag. The allpass-filter configuration comprising IC_{3A} and IC_{3B} sets the phase of the reference signal. You can adjust the amplitude of the output using potentiometer P_1, and you can adjust the phase with P_2, P_3, or both. The circuit uses three low-power LM358 dual op amps. **Figure 2** shows the measured waveforms at different points in the circuit. The reference sine wave has total harmonic distortion of 0.7%. Note that the circuit requires no heavy, bulky inductors.

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

Cascade bandpass filters for higher Q

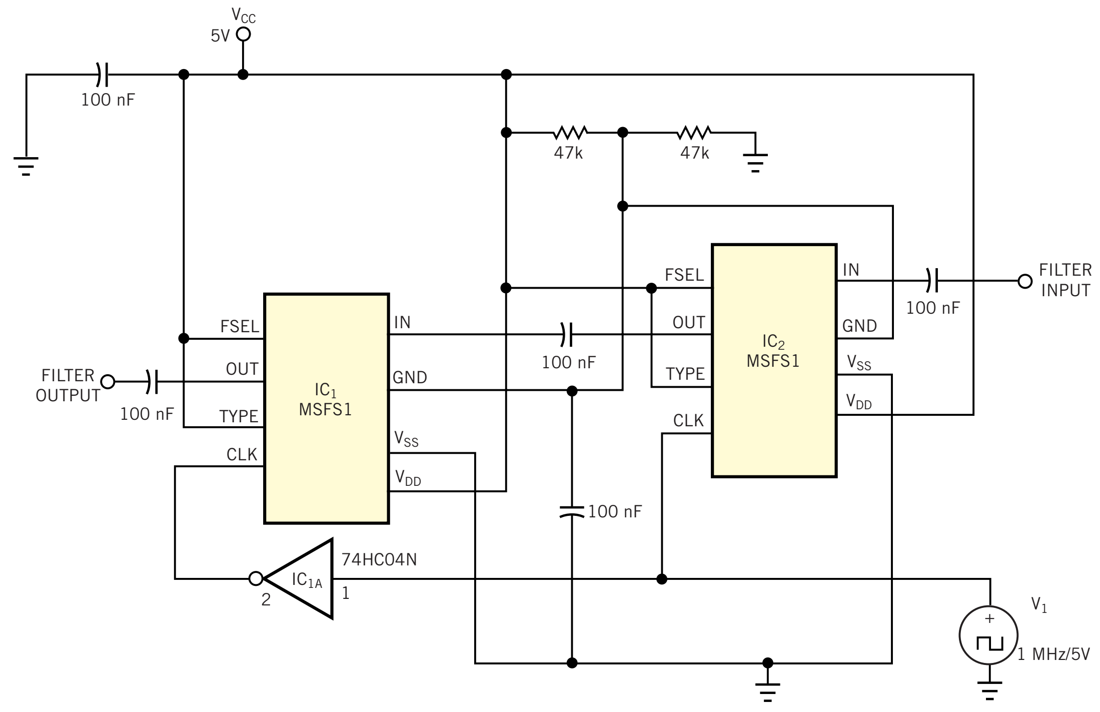
Trong Huynh and John Ambrose, Mixed Signal Integration Corp, Santa Clara, CA

Switched-capacitor filters that are preset for a given bandwidth sometimes do not deliver the bandwidth or Q an application requires. By inverting the clock between two switched-capacitor bandpass filters, such as the MSFS1 from Mixed Signal Integration Corp, you can configure a high-Q filter (**Figure 1**). The MSFS1 is a selectable, seventh-order, lowpass/bandpass, switched-capacitor filter. Using the FSEL pin, you can select either a lowpass or a bandpass response. With the TYPE pin, you can select a Butterworth, a Bessel, or an elliptic response if FSEL is low. When FSEL is high, you can select full-, third-, or sixth-octave bandpass response. The circuit in **Figure 1** shows both ICs set for sixth-octave bandpass response.

The clock-to-corner ratio of the MSFS1 is 50-to-1. With switched-capacitor filters, changing the clock frequency moves the center or corner frequency of the response by the same amount. For example, if the input clock to the MSFS1 is 1 MHz, the corner frequency is 20 kHz. Cutting the clock frequency to 500 kHz results in a corner frequency of 10 kHz. By inverting the clock between the two filter ICs, you obtain a 10th-octave filter. If you simply cascade the filters, without the clock inversion, the signal has a delay to the second filter equal to the clock period. The result is an increase in passband ripple and no change in the Q of the filter.

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

Figure 1



By inverting the clock to one switched-capacitor filter, you obtain a cascaded filter with enhanced Q.

C program calculates checksums

Ken Levine, Airshow Pacific Systems, Kirkland, WA

To ensure data integrity, it's wise to frequently calculate file checksums. The C program in **Listing 1** calculates the checksum of a file using a 16-bit CRC (cyclic redundancy check). The program assumes an 8-bit byte size. The routine reads the file as binary and processes one byte at a time. The CRC formula the program uses is X16+X12+X5+1, with a starting value of hexadecimal FFFF. The program displays the number of bytes processed. The file calccrc.c starts by including the header files needed. Next, the routine defines and initializes the constants. (The program does not use the C++ keyword const; therefore, a C compiler can compile the program.) Inside main(), the program defines variables and issues the initial starting value for the CRC. The routine performs a check to verify that at least two arguments passed to main(). If only one argument passes, the program terminates with a message that you need to supply a file name.

The program reads and processes one byte at a time until it reaches the end of the file. Each time the program reads a byte, the byte counter increments. At the end of the file, the program displays the CRC of the file and the number of bytes read. The routine calculates the 16-bit CRC, byte by byte, using the calcCRC16 function. This function passes the byte to be processed and the current value of the CRC and returns the new CRC value. The program calculates the CRC for each bit in the byte. The variable temp is assigned the current CRC value right-shifted 15 times, XORed with the current byte value right-shifted seven times. This operation XORs the MSB of the CRC with the MSB of the byte, so temp will have the value zero or one. Note that this operation does not change the values of the CRC or the byte. The CRC then left-shifts one place. If temp is 0, nothing happens. If temp is 1, the program XORs the CRC with hexadecimal 1021 (the X12+X5+1

LISTING 1—C PROGRAM FOR CALCULATING CRC

```
/* file calccrc.c
 * calculates the 16 bit CRC of a file
 * in Borland C++, an int is 16 bit
 *
 * file should be read in binary as this will read every character,
 * when a file is read in binary mode, the carriage return (CR) and
 * line feed (LF) are both read; in text mode, only the LF is read
 * should a file have CR in part of the data, text mode may not read
 * it
 */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <io.h>
#include <fcntl.h>

unsigned int INIT = 0xFFFF;  /* initial value of CRC */
unsigned int CRC16 = 0x1021; /* bits 12, 5 and 0 */
int SHIFT_CRC = 15;          /* how far to right shift crc */
int SHIFT_BYTE = 7;          /* how far to right shift byte */
int BYTE_SIZE = 8;           /* number of bits in a byte */


unsigned int calcCRC16(unsigned int, unsigned char);

int main(int argc, char *argv[]){

      unsigned int crc = INIT;
      long byteCounter = 0;
      FILE *Data;
      char ch;

if(argc  < 2){
            printf("\nNeed to supply file name for CRC calculation.");
            printf("\nProgram terminated.");
            exit(1);
      }

/* see if file can be opened*/
      if(NULL == (Data = fopen(argv[1], "rb"))){
            printf("\nUnable to open file %s, program terminated.", argv[1]);
            exit(1);
      }

      while(!feof(Data)){
            /* The end of file character is read and processed
             * by these statements.
            */
            ch = fgetc(Data);
            crc = calcCRC16(crc, ch);
            byteCounter++;
      }

      fclose(Data);
      /* Subtract one from byteCounter to compensate for the
       * end of file character being read. This character is

             * not counted when the operating system shows file
             * size.
            */
         byteCounter--;

            printf("\ncrc of %s is %x",argv[1], crc);
            printf("\n(read %ld bytes)", byteCounter);

            return 0;
      }


      unsigned int calcCRC16(unsigned int crc, unsigned char byte){

      /* Algorithim XOR's bit 15 (the MSB) of the current CRC with the current
       * MSB of byte.  The current CRC and byte are then left shifted by one.
       * This value is then XOR'ed with bits 12, 5 and 0 of the current CRC.
       * This is done until all bits in byte have been processed.
       */
```

(continued on pg 152)

term). Next, the byte left-shifts one place, so the program can process the next bit. This process repeats until the routine processes all eight bits of the byte.

If you wish to use a different formula to calculate the CRC, you need only change the variable CRC16, assuming that the formula still starts with the X16 term). If you wish to calculate a 32-bit CRC, the variables INIT and CRC16 cannot be of type int. You can set the variables SHIFT_CRC, SHIFT_BYTE, and BYTE_SIZE to other values to accommodate various byte and CRC sizes. This program is compiled using Borland C++ 3.0, Borland C++4.5, and Microsoft Visual C++ 1.0. You can download **Listing 1** 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 #2674.

LISTING 1—C PROGRAM FOR CALCULATING CRC ***(continued)***

```
        unsigned int temp;
        int index;

        for(index = 0; index < BYTE_SIZE; index++){
                temp = (crc >> SHIFT_CRC) ^ (byte >> SHIFT_BYTE);  /*temp is now MSB
of CRC X or'd

         with MSB of byte */
                crc <<= 1;  /* left shift one space */

                if(temp){  /* if temp is 1, then XOR bits 12, 5 and 0 with 1
                                * if temp is 0, no need to XOR because XOR
with 0
                                * does not change value */
                        crc ^= CRC16;
                }

                byte <<= 1; /* left shift one to get to next bit */
        }

        return crc;
}
```

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

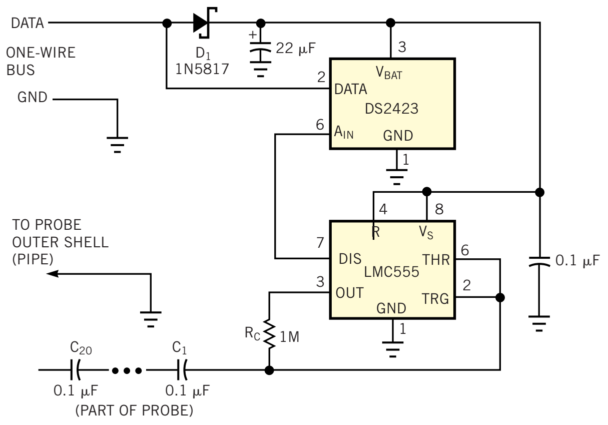
One-wire bus powers water-level sensor

Dale Litwhiler, Lockheed Martin, Newtown, PA

You can use the simple sensor circuit in **Figure 1** to remotely monitor the level of liquid water in a vessel such as a swimming pool. The LMC555 sensor oscillator provides an output-signal frequency that is a function of the water level. This signal drives a DS2423 pulse counter. A host PC or μC reads the output of the pulse counter via the Dallas Semiconductor one-wire bus (**Reference 1**). The circuit uses approximately 150 μA of current, allowing the circuit to steal its power from the bus via the Schottky diode, D_1. Because the circuit is sensing water that is part of the electrical circuit, you should use an ac-coupled signal to avoid polarization of the water and plating of the electrodes. One approach is to have the water in a circuit branch that is in series with some capacitance. In this sensor circuit, the water is in the branch containing the timing capacitance of a CMOS 555 timer, configured as a free-running oscillator with a 50% duty cycle. The sensor provides a capacitance that varies with water level.

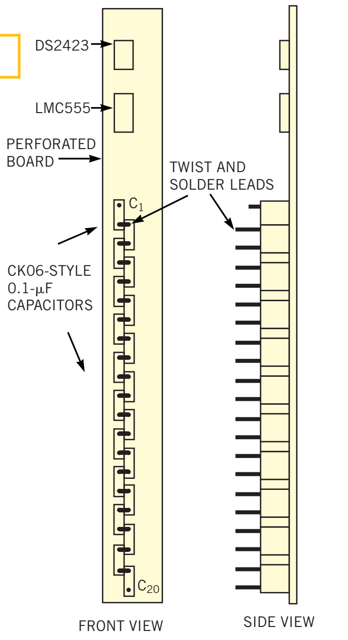
Figure 2 shows one method of fabri-

Figure 1



A simple 555-type timer and a counter form the heart of a water-level sensor.

Figure 2



A string of series-connected capacitors provides an indication of water level.

cating the sensor. You epoxy-bond a series string of N radial-leaded ceramic capacitors of equal value C underneath a pc board. Twist the leads of adjacent capacitors, solder them together, and trim them to serve as electrodes to contact the water. The outer shell of the sensor, a piece of 0.75-in. copper pipe, forms another electrode. If you place this assembly vertically in a vessel, the capacitance between the terminals of the sensor (the uppermost lead of C_1 and the outer shell) increases in steps as the water rises and covers more of the capacitors. The water effectively short-circuits the capacitors to the outer-shell electrode. Because the capacitors are in series, the total capacitance changes according to: $C_{TOTAL}=C/(N-n)$, where n is the number of capacitors with both leads covered by water. When you insert this expression in the equation for the 555 oscillation frequency, you obtain $f_{OSC}=(N-n)/1.4R_CC$. Note that the frequency changes linearly with water level.

This application uses 20 0.1-μF, CK06-style capacitors. The lead spacing of these capacitors is 0.2 in. These dimensions provide a measurement range of 4 in. with a resolution of 0.2 in. This design uses 1 MΩ for the oscillator timing resistor, R_C, because the timing resistance must be much larger than the impedance of the water to minimize timing error. (Measurements of several municipal and residential well-water samples revealed impedance values of approximately 5 kΩ in the frequency of 5 Hz to 1 kHz with 0.5-in. probe spacing.) Another reason for 1-MΩ oscillator timing is that the DS2423 counter has a maximum input frequency of approximately 2 kHz. Also, you must minimize the power stolen from the bus. Finally, you must maintain maximum isolation of the water from the bus in the event of a catastrophic bus fault.

With these values of capacitance and resistance, the sensor's output-frequency range is approximately 7 to 142 Hz in steps of approximately 7 Hz. In practice, you might read the counter at intervals of several seconds to several minutes to obtain an averaging effect. This sensor has found application for two summers in a residential swimming pool. Users noticed no change in performance from corrosion or plating of the electrodes.

REFERENCE

1. Awtrey, Dan, "Transmitting data and power over a one-wire bus," *Sensors*, February 1997.

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

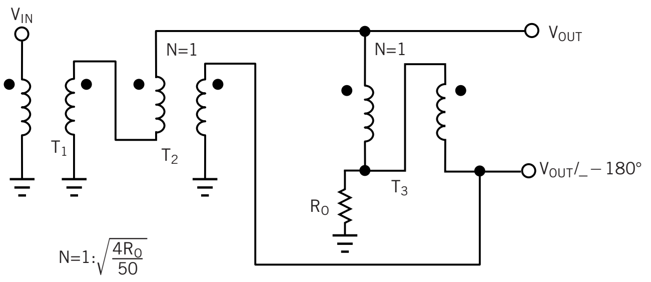
Ideal transformers aid in balanced-line analysis

Alan Victor, IBM Microelectronics, Research Triangle Park, NC

TRANSMISSION-LINE transformers combined with appropriate resistor values are useful in hybrid applications over limited bandwidths. One such device, the 180° hybrid combiner, is useful in CAD analysis for verifying the performance of balanced and differential circuits. The completed transformer provides matched signal levels, 180° phase-shifted, and all ports at an impedance Z_0. You don't need transmission-line transformers in the construction of this hybrid for analysis. Instead, the circuit uses an ideal 1-to-1 transformer. Combined with the appropriate termination resistor and one additional transformer, an ideal differential-excitation source is available. Extensive bandwidth, balanced and equal impedance at all three ports, and infinite isolation provide an excellent circuit block to assist in analyzing differential and balanced circuits. The circuit block is borrowed from power-amplifier hybrids where it is useful for signal splitting and combining (**Figure 1**).

After you add a third transformer, T_1, to the two transformers you use, and with R_0 set at 25Ω, a 50Ω, three-port unit becomes available. If you replace the termination resistor, R_0, with a transmission line and a variable resistor, then a 180° phase-shifted variable impedance becomes available at the excitation ports. You then have a tool for studying balanced-line performance sensitivity to source-impedance variations. You can study high-speed, RF, and microwave using differential or balanced topologies using the arrangement in **Figure 1**. You can also turn the circuit around end-to-end and then use it for combining the output of the balanced circuit. Tests show that the circuit yields a perfectly flat frequency response from 1 to greater than 22 GHz and phase shifts of exactly 0° and 180° (depending on the port).

Figure 1



A simple transformer circuit aids in the analysis of differential and balanced circuits.

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