

# designideas

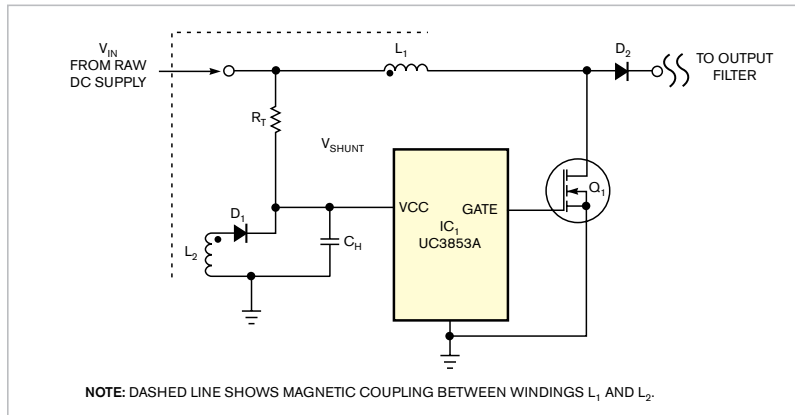
READERS SOLVE DESIGN PROBLEMS

## Shunt regulator speeds power supply's start-up

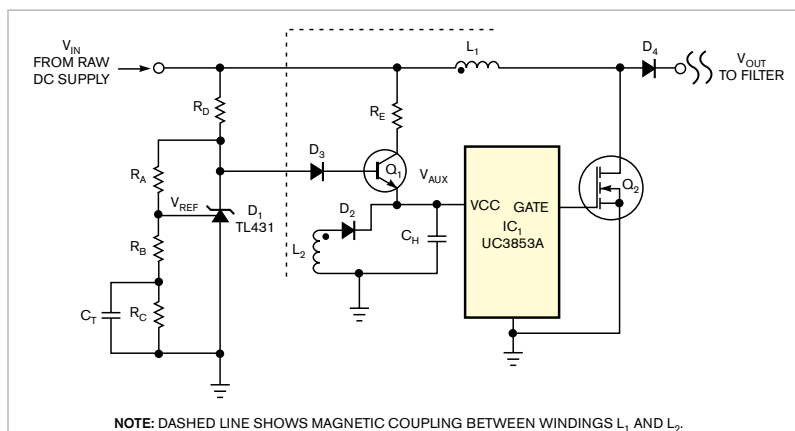
Michael O'Loughlin, Texas Instruments, Nashua, NH

**▶** In certain applications, design requirements may call upon a system's switched-mode power supply to more promptly deliver its output than would the garden-variety power supply. **Figure 1** shows such a supply's bootstrap, or start-up, circuit. In a

switched-mode power supply's PFC (power-factor-corrected) preregulator, the circuit's PWM (pulse-width modulator), IC<sub>1</sub>, draws its normal operating power from auxiliary winding L<sub>1</sub>, wound on boost inductor L<sub>2</sub>'s magnetic core and diode D<sub>1</sub>.



**Figure 1** In a conventional switched-mode power supply's bootstrap circuit, trickle-charge resistor R<sub>T</sub> and capacitor C<sub>H</sub> supply start-up power to the pulse-width modulator and controller, IC<sub>1</sub>.



**Figure 2** In this augmented bootstrap circuit, transistor Q<sub>1</sub> delivers a robust initial pulse of current to capacitor C<sub>H</sub>, ensuring faster start-up and power delivery.

### DIs Inside

**64** Build a USB-based GPIB controller

**66** Programs calculate 1% and ratio-resistor pairs

**68** Simplify worst-case PSpice simulations with customized measurement expressions

**70** Tiny twisted-pair transmission line solves test-fixture woes

**▶** What are your design problems and solutions? Publish them here and receive \$150! Send your Design Ideas to [edndesignideas@reedbusiness.com](mailto:edndesignideas@reedbusiness.com).

Resistor R<sub>T</sub> and capacitor C<sub>H</sub> form a trickle-charge circuit that supplies power for bootstrapping IC<sub>1</sub> into normal operation. In conventional designs, R<sub>T</sub> comprises a high resistance that delivers just enough current to overcome the standby current and supply a trickle charge to holdup capacitor C<sub>H</sub>, which stores enough energy to power the PWM circuit until the power converter begins operation. Under normal circumstances, the circuit's slow start-up response poses no problems.

When faster power-on response becomes important, you can reduce the bootstrap time by reconfiguring the start-up shunt regulator (**Figure 2**). Capacitor C<sub>T</sub>; shunt-regulator IC D<sub>1</sub>, a TL431; diode D<sub>3</sub>; transistor Q<sub>1</sub>; and resistors R<sub>A</sub> through R<sub>D</sub> form the bootstrap circuit. At power application, capacitor C<sub>T</sub> holds no charge, and the series-pass regulator that Q<sub>1</sub> and D<sub>1</sub> form determines the voltage at the PWM's power input, V<sub>AUX</sub>.

At turn-on, the V<sub>AUX</sub> voltage reaches its peak voltage, V<sub>AUX PEAK</sub>, which the ratio of resistors R<sub>A</sub> and R<sub>B</sub> determines.

Capacitor  $C_T$  and resistor  $R_C$  conserve energy by setting the bootstrap circuit's turn-off time and voltage. Resistor  $R_D$  supplies bias current to  $D_1$ , the TL431 shunt-regulator IC, and resistor  $R_E$  keeps transistor  $Q_1$  within its safe operating area by limiting its collector current.

To design the circuit, begin by selecting resistors  $R_A$  and  $R_B$  to establish the peak charging voltage, as the following equation shows:

$$\frac{V_{REF}}{R_B} = \frac{V_{AUX\_PEAK} + V_{D3} + V_{BE} - V_{REF}}{R_A + R_B},$$

where  $V_{REF}$  represents the TL431's internal reference voltage. Next, select resistor  $R_C$  to reduce the shunt-regulated voltage below the nominal  $V_{AUX}$  voltage,  $V_{VAUX\_NOMINAL}$ , which the auxiliary winding supplies:

$$RC_T = \frac{V_{REF} \times R_A + (V_{REF} - V_{AUX\_NOMINAL})R_B}{V_{AUX\_NOMINAL} - V_{REF} - IV}.$$

Choose capacitor  $C_T$ 's value to set the bootstrap time,  $T_{BOOT}$ , as follows:

$$C_T = \frac{2 \times T_{BOOT}}{R_C}.$$

As in **Figure 1**, diode  $D_2$  and auxiliary winding  $L_2$  provide normal operating power to IC<sub>1</sub>. **EDN**

## Build a USB-based GPIB controller

Boštjan Glazar, Marko Jankovec, and Marko Topic,  
Laboratory of Semiconductor Devices, Ljubljana, Slovenia

Contemporary research laboratories include a variety of instruments that connect using any of several interface methods to a PC for automating procedures and collecting data. Although different communication interfaces exist, the GPIB (general-purpose-interface bus) still enjoys wide popularity. The host PC must include a suitable GPIB controller—an internal interface card or an external device. Newer PC designs are phasing out traditional internal buses, such as PCI, ISA, and EISA, in favor of other standards, so using an external controller offers a more appropriate approach because external I/O ports, such as RS-232 and USB, tend to maintain backward compatibility.

This Design Idea covers the development of a GPIB controller, which turned out to be easier and cheaper than commercially available alternatives. The design uses easy-to-obtain components with a total component cost of approximately \$50. For comparison, a commercial controller costs at least 10 times more: \$500 to \$1000. The USB 2.0-compliant controller, an external device, draws its operating power from the bus and provides plug-and-play operation and high-speed data transfer. In addition, a USB-controller design extends its applications to notebooks and other computers that lack available I/O slots. The controller re-

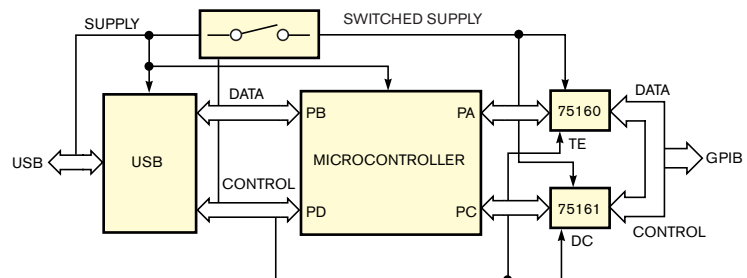
sides on a double-sided pc board and fits into a 123×30×70-mm enclosure (**Figure 1**). To simplify controller use, the design uses National Instruments' (www.ni.com) LabView graphical programming language to develop the appropriate driver.

The design uses the FT245BM USB-controller IC from Future Technology Devices International Ltd (www.ftdichip.com), which features an 8-bit parallel connection to the host microcontroller and a virtual-communications port to the PC-interface side. The circuit operates at a full speed of 12 Mbps. Targeting use in GPIB applications, the 75160 and 75161 ICs drive GPIB I/O lines. An Atmel (www.atmel.com) AVR AT90S8515 microcontroller provides firmware-resident sequence control and in-circuit-pro-

grammable flash memory that simplifies firmware design and upgrades. The USB also can supply 5V of power at as much as 500 mA, which eliminates the requirement for an external power supply. The controller also supports the required low-power mode to reduce consumption to less than 1 mA.

The designers used the Protel (www.altium.com) schematic-capture and pc-board-layout software to design the circuit. They used a milling machine to produce the prototype pc board and partially assembled the board with a manual SMD placer. You can also use a commercial prototype pc-board-fabrication service to prepare a double-sided pc board with plated through holes and manually assemble the circuit. **Figure 2** shows an internal view, and **Figure 3** shows the completely assembled controller, which is easy and fast to build.

The controller communicates with the host computer through a logical serial interface that enables use of the



**Figure 1** This USB-based GPIB controller requires only four ICs.



**Figure 2** A top view of the controller's pc board shows the USB connector (left) and the GPIB connector (right).

controller with any programming language that supports serial-port communications. The LabView driver is compatible with LabView's built-in GPIB driver, thus simplifying adapta-

tion of programs to the new interface. The driver is a collection of virtual instruments, which require only one more input—that is, a serial-port number—than a built-in GPIB driver.

Thanks to its open-source design, the controller provides a highly cost-effective approach to controlling GPIB instruments that's adaptable to many computational platforms. You can obtain the microcontroller's firmware; descriptions of the protocols; and all other necessary files, including a pc-board layout, at <http://lsd.fe.uni-lj.si/gpib/>. With that information, you can write a driver for whatever operating system or programming language you choose. In addition, the Web page includes



**Figure 3** The controller in its housing presents a profile that's not much larger than a GPIB cable connector.

firmware for the Atmel AVR microprocessor, a user's manual for the assembled interface, and additional notes on GPIB and LabView. To download a 1505-kbyte, zip-formatted archive containing the entire project, go to: <http://lsd.fe.unilj.si/gpib/complete.zip>. **EDN**

## Programs calculate 1% and ratio-resistor pairs

Carl Rutschow, Upland, CA

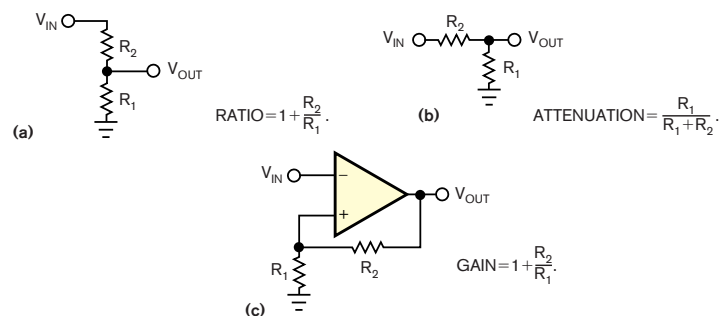
➔ If you perform analog-circuit design, you'll occasionally need to use a resistor with a nonstandard value to produce a particular gain, ratio, or attenuation factor. You can create resistors of unusual values by connecting two standard-value resistors of 1% tolerance in parallel. Because it is impossible to readily predict which resistor pairs will fall closest to the desired value, a computer program can help by calculating all combinations of standard 1% resistors to determine the best values for your application.

The Visual Basic, compiled, executable file Rratio2.exe checks all standard 1%-resistor values in a given range for a desired ratio, attenuation factor, or noninverting operational-amplifier gain (**Figure 1**). You can download the program from [www.edn.com/051216di1](http://www.edn.com/051216di1). You select the calculation mode via the program's window buttons. As an op-

tion, you can choose whether the program displays all possible values or only the values closest to the target value.

Using standard 1%-resistor values, a second program, RPar2a.exe, also available at [www.edn.com/051216di1](http://www.edn.com/051216di1), checks and displays all appropriate combinations that generate a desired parallel resistor's value. The program

generally calculates several parallel combination values that fall well within 0.1% of the desired value. By comparison, a single 1% resistor's nominal value may differ by as much as 1.45% from the desired value. Note that, for both programs, the calculated resistance values depend on the paired resistors' tolerances. **EDN**



**Figure 1** You can use the program Rratio2.exe to calculate ratio (a), attenuation (b), or gain (c).

# Simplify worst-case PSpice simulations with customized measurement expressions

Wayne Huang and Jeff Van Auken, Picor Corp, North Smithfield, RI

During IC design, worst-case simulations help designers account for variations in characteristics of PNP and NPN transistors and base and polysilicon resistors. These four classes of devices alone produce more than 16 combinations of simulation conditions. To accommodate temperature variations, each combination undergoes simulation at  $-40$ ,  $+27$  (room temperature), and  $+125^{\circ}\text{C}$ , producing at least 48 series of data to analyze when simulations are complete. To help an IC designer evaluate simulated waveforms' characteristics, PSpice provides a library of ready-to-use, predefined measurements, including bandwidth, gain/phase margins, and more. PSpice also allows a designer to use predefined YatX and XatNthY measurements to measure a waveform's y value at a given x value—usually, time—and to find an x value that corresponds to the nth instance of a given y value (Reference 1).

However, when a designer must measure the value of Waveform 1 when Waveform 2 crosses a certain y value, predefined measurements do not apply because, unlike many programming languages, PSpice allows no embedment. This Design Idea describes how to create a customized PSpice measurement expression that solves the problem. As Listing 1 shows, the measurement expression itself is straightforward. Line 1 finds the X\_value (x1) when Trace 1 crosses the y1\_value for the nth positive slope. Line 2, denoted by braces {} at the bottom of the listing, searches for the value of Trace 2 (y2) at x1. Similarly, Listing 2 shows how a designer can create a measurement ex-

pression to find a y2 value when Trace 1 crosses a y1\_value for the nth negative slope or when Trace 1 crosses a given percentage of its full y-axis range.

Figure 1 shows a simulation example in which the input and the output voltages represent a comparator's input and output, respectively. When the input voltage is greater than the positive threshold voltage, then the output voltage is high; when the input voltage is less than the negative threshold voltage, the output voltage is low. Using customized measurement expressions, a designer can easily find the rising and falling thresholds and the comparator's hysteresis voltage for all conditions immediately after the probe data becomes available. If any condition exists in which the threshold doesn't meet the design specification, the designer can then go directly to that condition and spend time on further analysis.

The simulation example describes an input-voltage monitor comprising a comparator that acts as a "power-good" block in a power-management IC. When the input voltage rises above a

13V enable threshold, the output voltage goes high and enables other circuit blocks. When the input voltage falls below a 10V disable threshold, the output voltage goes low and disables other circuits. The difference between the enable and the disable thresholds—that is, 3V—defines the hysteresis voltage. A worst-case simulation of the circuit must account for variations in characteristics of NPN and PNP transistors, base resistors, and polysilicon resistors in the circuit. Each device's characteristics can fall at either the low or the high end of the process specifications and thus produce 16 combinations.

The toolbar lists a few of the 16 possible combinations. For example, LLLL refers to the case in which characteristics of NPN and PNP transistors and base and polysilicon resistors all fall at their low values. In addition, one pass of the simulation uses nominal values; that is, the components' specifications fall in the centers of their nominal characteristics. For each combination, PSpice simulates the circuit's behavior at low, room, and high temperatures, respectively, producing 51 data traces for the block's input and output voltages for a total of 102 displayed traces. After PSpice assembles the data, the circuit's designer must extract the actual threshold voltages for each condition for comparison with the circuit's specifications.

Given the large number of displayed traces, using the display's cursor to measure each threshold consumes much of a designer's time. Using a customized PSpice measurement extracts the threshold voltages in a fraction of the time and presents the data in tabular form. The table immediately below the waveform plot contains simulation results for all 51 traces. Columns 1, 2, and 3 list results for nominal characteris-

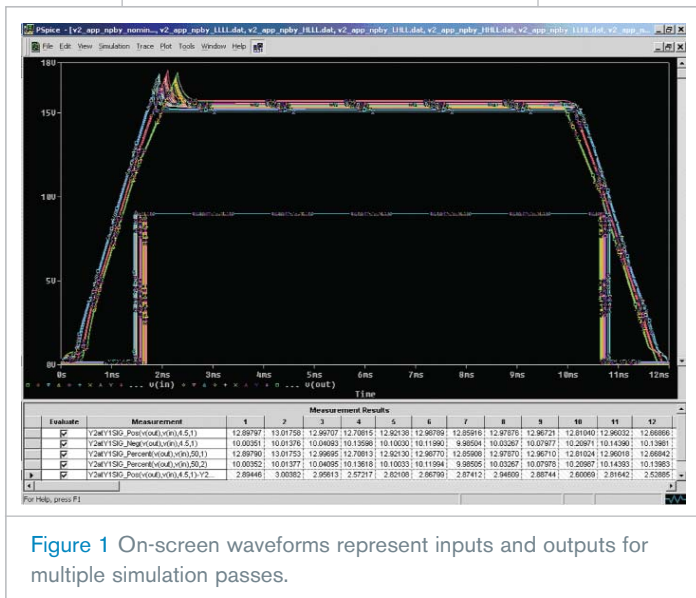


Figure 1 On-screen waveforms represent inputs and outputs for multiple simulation passes.

LISTING 1 MEASUREMENT EXPRESSION FOR x1

```

View Measurement
Y2atY1SIG_Pos(1,2,y1_value,n_occur)=y2
#Desc# Find the value of trace 2 at the X_value corresponding to the
#Desc# nth positive slope crossing the given Y1_value for trace 1
#Arg1# Name of trace 1
#Arg2# Name of trace 2
#Arg3# nth occurrence
#Arg4# Y1 Value
{
1| search forward for n_occur:level (y1_value, positive) !1;
2| search forward xval (x1) !2;
}
    
```

LISTING 2 MEASUREMENT EXPRESSION FOR y2

```

Y2atY1SIG_Neg(1,2,y1_value,n_occur)=y2
{
1| search forward for n_occur:level (y1_value, negative) !1;
2| search forward Xvalue (x1) !2;
}

Y2atY1SIG_Percent(1,2,y1_pct,n_occur)=y2
{
1| search forward for n_occur:level (y1_pct% ) !1;
2| search forward xval (x1) !2;
}
    
```

tics, and columns 4, 5, and 6 list results for low, room, and high temperatures when all devices' specifications reside at their lower extremes.

Row 1 of the table displays the measurement expression and results for the enable-voltage threshold. When the output voltage first crosses 4.5V (one-half the simulated cir-

cuit's 9V power-supply bus voltage) on the positive slope, the simulation records the value of the input voltage as the enable-threshold voltage, and row 2 measures the disable-threshold voltage. Rows 3 and 4 measure the enable- and disable-threshold voltages by another method: When the output voltage passes 50% of the full-scale

value for the first and second times, PSpice measures the value of the input voltage. Row 5 calculates the hysteresis voltage.**EDN**

REFERENCE

1 *PSpice User's Guide*, Cadence Design Systems Inc, June 2003, [www.cadence.com](http://www.cadence.com).

## Tiny twisted-pair transmission line solves test-fixture woes

Glen Chenier, Allen, TX

Engineers often construct test fixtures that include high-speed differential signals. Although miniature coaxial cable is widely available, there's no commercial off-the-shelf source for small-gauge twisted-pair cable that's suitable for differential signals. Although Category 5 Ethernet cable contains four twisted pairs, it's too large for crowded fixtures and for attachment to the Amp Z-Pack connectors some fixtures require. Many engineers are unaware that they can twist together two lengths of AWG #30 Kynar-insulated wire—garden-variety wire-wrap and prototype cut-and-jumper wire—to make a 102Ω differential-transmission line. If you use Kynar's dielectric constant and the insulation's thickness to compute its properties, the line's calculated differential impedance works out to 110Ω. In practice, differential TDR (time-domain-reflectometer) measurements

show that the line's actual impedance consistently measures 102Ω—only 2% away from the target impedance and thus close enough for most practical purposes.

To make your own twisted pair, start with a long AWG #30 Kynar-insulated wire and fold it in half. Enlist a co-worker to hold the cable's closed end by slipping the loop around a screwdriver's blade. If you're working alone, slip the loop around a doorknob. Tightly twist the two wires' free ends together and insert the twisted ends into the chuck of a Dremel ([www.dremel.com](http://www.dremel.com)) rotary tool. Tighten the chuck and hold the Dremel tool so that the wires are stretched tightly, are of the same length, and lie parallel with each other.

Apply a slight amount of tension to the wires and start the tool. As the wires twist together, the pair shortens and pulls the tool's operator toward the loop support. A variable-speed Dremel

tool works best when you operate it at its slowest setting. If you have only a fixed-speed Dremel tool, avoid over-twisting the wires by preparing a length of 10 to 20 ft of cable at a time. The extra length allows time to switch the tool off and avoid overtwisting the wires. Cut off and discard the cable's nonuniformly twisted end sections.

The amount of twist in the wires is not critical, but the wires should be firmly twisted together. Using approximately eight to 10 twists/in. works well. To count the twists, hold a portion of the cable against a ruler or measuring scale under a magnifier and count 16 to 20 "bumps," or half-twists, per inch. Using too many twists per inch uses excess wire and increases losses and propagation delay. For the lengths in a test fixture, losses are insignificant except at extremely high frequencies.

You can also use a variable-speed hand drill with a 1/4- or 3/8-in. chuck to twist the wires, but you need to fold the wires' free ends several times and wrap them in duct tape to ensure a snug fit in the drill's chuck. When using any power tool, wear safety glasses or other eye protection during the procedure.**EDN**