EDITED BY BRAD THOMPSON
AND FRAN GRANVILLE

designideas

READERS SOLVE DESIGN PROBLEMS

Temperature-to-period circuit provides linearization of thermistor response

S Kaliyugavaradan, Anna University,
Madras Institute of Technology, Chennai, India

Designers often use thermistors rather than other temperature sensors because thermistors offer high sensitivity, compactness, low cost, and small time constants. But most thermistors' resistance-versus-temperature characteristics are highly nonlinear and need correction for applications that require a linear response. Using a thermistor as a sensor, the simple circuit in **Figure 1** provides a time period varying linearly with temperature with a nonlinearity error of less than 0.1K over a range as high as 30K. You can use a frequency counter to convert the period into a digital output. An approximation derived from Bosson's Law for thermistor resistance, R_T, as a function of temperature, θ, comprises $R_T=AB^{-\theta}$ (see **sidebar** "Exploring Bosson's Law and its equation" on the Web version of this article at www.edn.com/051110di1). This relationship closely represents an actual thermistor's behavior over a narrow temperature range.

You can connect a parallel resistance, R_P, of appropriate value across the thermistor and obtain an effective resistance that tracks fairly close to $AB^{-\theta}\simeq$ 30K. In **Figure 1**, the network connected between terminals A and B provides an effective resistance of $R_{AB}\simeq AB^{-\theta}$. JFET Q_1 and resistance R_S form a current regulator that supplies a constant current sink, I_S, between terminals D and E.

Through buffer-amplifier IC_1, the voltage across R_4 excites the RC circuit comprising R_1 and C_1 in series, producing an exponentially decaying voltage across R_1 when R_2 is greater than R_{AB}. At the instant when the decaying voltage across R_1 falls below the voltage across thermistor R_T, the output of comparator IC_2 changes its state. The circuit oscillates, producing the voltage waveforms in **Figure 2** at IC_2's output. The period of oscillation, T, is $T=2R_1C_1\ln(R_2/R_{AB})\simeq 2R_1C_1[\ln(R_2/A)+\theta\ln B]$. This equation indicates that T varies linearly with thermistor temperature θ.

You can easily vary the conversion sensitivity, $\Delta T/\Delta\theta$, by varying resistor R_1's value. The current source comprising Q_1 and R_1 renders the output period, T, largely insensitive to variations in supply voltage and output load. You can vary the period, T, without affecting conversion sensitivity by

DIs Inside

82 Two wires control SPI high-speed ADC

84 Volume-unit meter spans 60-dB dynamic range

90 Pacer clock for microcontrollers saves subroutine calls

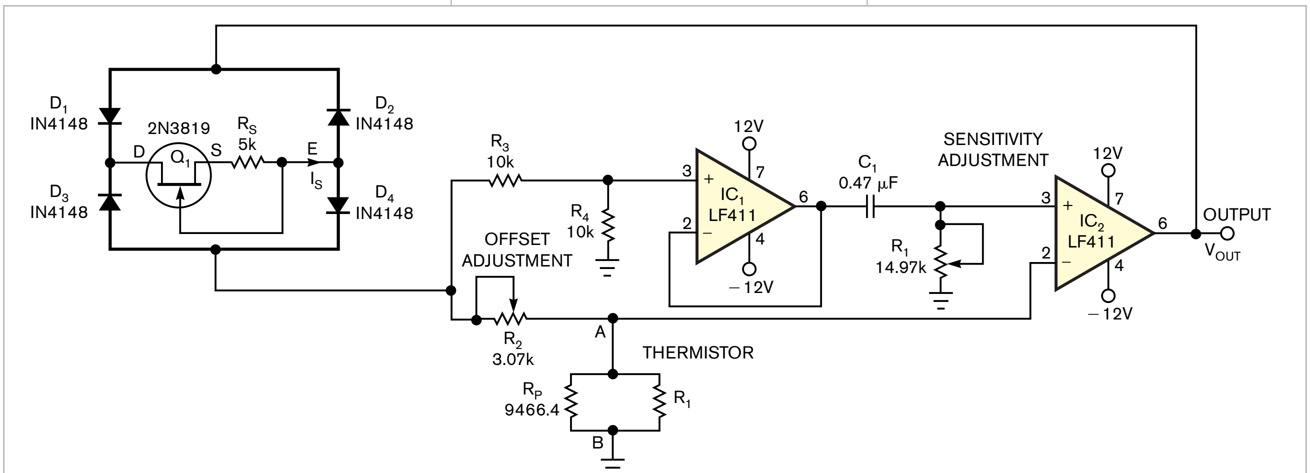

Figure 1 This simple circuit linearizes a thermistor's response and produces an output period that's proportional to temperature.

varying R_2. For a given temperature range, θ_L to θ_H, and conversion sensitivity, S_C, you can design the circuit as follows: Let θ_C represent the center temperature of the range. Measure the thermistor's resistance at temperatures θ_L, θ_C, and θ_H. Using the three resistance values R_L, R_C, and R_H, determine R_P, for which R_{AB} at θ_C represents the geometric mean of R_{AB} at θ_L and θ_H. For this value of R_P, you get R_{AB} exactly equal to $AB^{-\theta}$ at the three temperatures, θ_L, θ_C, and θ_H.

At other temperatures in the range, R_{AB} deviates from $AB^{-\theta}$, causing a nonlinearity error that is appreciably less than 0.1K for most thermistors when the temperature range is 30K or less. You can easily compute R_P using: $R_P=R_C[R_C(R_L+R_H)-2R_LR_H]/(R_LR_H-R_C^2)$. Because temperature-to-period-conversion sensitivity, S_C, is $2R_1C_1\ln b$, you can choose R_1 and C_1 such that $R_1C_1=S_C[\theta_H-\theta_C]/\ln(R_{AB}$ at θ_L/R_{AB} at $\theta_H)$ to obtain the required value of S_C. To get a specific output period, T_L, for the low temperature, θ_L, R_2 should equal $(R_{AB}$ at $\theta_L)e^Y$, in which Y represents $(T_L/2R_1C_1)$. In practice, use a lower value for R_2 because the nonzero response delay of IC_2 causes an increase in the output period.

Next, set potentiometers R_1 and R_2 close to their calculated values. After you adjust R_1 for the correct S_C, adjust R_2 until T equals T_L for temperature θ_L. The two voltage-divider resistances, R_3 and R_4, should be equal in value and of close tolerances. As a practical example, use a standard thermistor, such as a Yellow Springs Instruments 46004, to convert a temperature span of 20 to 50°C into periods of 5 to 20 msec. This thermistor exhibits resistances for R_L, R_C, and R_H of 2814, 1471, and 811.3Ω, respectively, at the low, midpoint, and high temperatures. Other parameters for the design include $S_C=0.5$ msec/K, $\theta_L=20$°C, $\theta_H=50$°C, $\theta_C=35$°C, and $T_L=5$ msec.

Because only a fraction of current I_S is through the thermistor, I_S should be low to avoid self-heating effects. This design uses an I_S of approximately 0.48 mA, which introduces a self-heating error of less than 0.03K for a thermistor's dissipation constant of 10 mW/K. **Figure 1** illustrates the values of the components in the example. All resistors are of 1% tolerance and 0.25W rating; use a polycarbonate-dielectric capacitor for C_1.

Simulating various temperatures from 20 to 50°C by replacing the thermistor with standard, 2814 to 811.3Ω, 0.01%-tolerance resistors produces T values of 5 to 20 msec with a maximum deviation from correct readings of less than 32 μsec, which corresponds to a maximum temperature error of less than 0.07K. Using an actual thermistor produces a maximum error of less than 0.1K for a thermistor dissipation constant of 10 mW/K or less. EDN

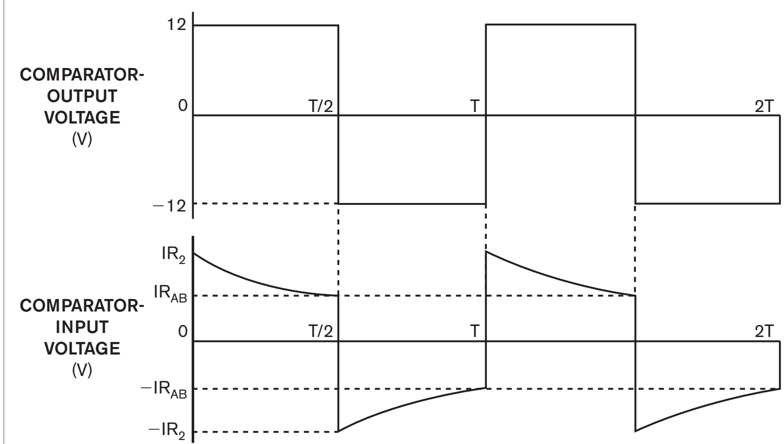

Figure 2 Waveforms show input to comparator IC_2 (lower trace) and its output (upper trace). In the lower trace, IR_2 represents the voltage across R_2.

Two wires control SPI high-speed ADC

Dan Meeks, Texas Instruments Inc, Austin, TX

Most current microprocessors, DSPs, and field-programmable gate arrays integrate hardware and software resources that support either or both of two common interface standards—SPI (serial-peripheral interface) and I²C (inter-IC)/SMBus. Both two-wire-interface standards suffer from a few crucial disadvantages. For example, I²C's throughput rates are 100 kbps, 400 kbps, or 3.4 Mbps in standard-, fast-, and high-speed modes, respectively, and can thus restrain a fast peripheral data converter's sample rate. Excluding framing and overhead bits, a 100k-sample/sec, 12-bit ADC must transfer at least 1.2 Mbps over the interface, a rate that only I²C's high-speed mode supports. Many processors and controllers currently offer no I²C high-speed mode and thus would be unable to support a fast data converter.

One of I²C's major benefits reduces the number of host-to-target interconnections. Using only two wires plus ground, the host controller can address the target device and exchange data, whereas SPI requires three wires—data, clock, and chip-selection—plus ground. Multiple SPI-target devices can share data and clock lines, but each device requires its own dedicated chip-selection line.

Given the perpetual demand for higher sample rates and resolution, I^2C's limited speed may restrict its use in some applications and instead force designers to select SPI. However, SPI requires an additional I/O pin on the host controller. In situations in which extra pins are unavailable but the application requires a fast SPI-bus converter, you can apply the technique in **Figure 1**.

For example, Texas Instruments' ADS7816 comprises a 200k-sample/sec, 12-bit-sampling ADC that requires a bit rate of 3M samples/sec to sample continuously at a 200k-sample/sec rate (**Reference 1**). Selecting the ADS7816's active-low $\overline{CS}$ (chip-select) pin initiates a conversion cycle. After toggling and holding $\overline{CS}$ low during the data transfer, $\overline{CS}$ returns high after transferring the data completes the process.

When the clock line initially goes low, it also asserts $\overline{CS}$ to a low state. The time constant of the peak detector comprising D_1, R_1, and C_1 ensures that $\overline{CS}$ does not go high until the clock line remains high for more than one clock cycle (**Figure 2**). Although the clock line toggles and retrieves data from IC_2, $\overline{CS}$ remains asserted low, and, upon completion of retrieval, the clock line goes high, and $\overline{CS}$ follows, readying the circuit for another conversion cycle.

Because C_1 must discharge at the end of a conversion cycle, the controller should delay the start of the next conversion cycle until C_1 fully discharges. Careful choice of R_1 and C_1 minimizes the delay to a minimum of three clock cycles. In addition, the voltage across C_1 must not fall below inverter IC_{1B}'s input threshold before the next clock pulse arrives to refresh the capacitor's voltage. Inverter IC_{1A}'s output voltage and current capabilities affect C_1's recharge time, and R_1 and IC_{1B}'s input impedance affect the discharge time. To ensure a robust design, include components' tolerances and temperature coefficients along with variations of logic-input and -output thresholds. EDN

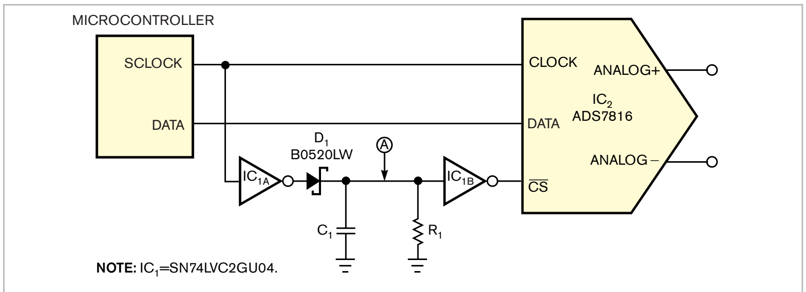

Figure 1 Two inverters and a few components can substitute for an SPI ADC's chip-select line.

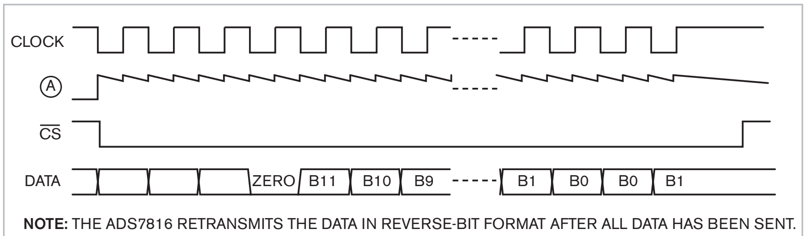

Figure 2 An SPI-clock waveform (top trace) evokes data (bottom trace), and peak detection of the clock (the waveform at Point A in Figure 1) yields a signal (next-to-bottom trace) that mimics the chip-selection line's behavior.

REFERENCE

1. ADS7816 data sheet, http://focus.ti.com/docs/prod/folders/print/ads7816.html, Texas Instruments.

Volume-unit meter spans 60-dB dynamic range

Jon Munson, Linear Technology Corp, Sunnyvale, CA

An audio volume-unit meter displays peak-related audio amplitudes to aid in accurately setting recording levels or for displaying an amplifier's operating conditions. A simple diode and capacitor network provides a classic volume-unit meter's peak-weighted response, but the circuit typically limits response to about 23 dB of displayable dynamic range, and the meter suffers from errors that its pointer's inertia and mechanical "ballistics" introduce. Contemporary displays eliminate the inertia problem by using arrays of lighted elements to form bar graphs, but any shortcomings in response and accuracy characteristics now shift to the signal-processing domain. You can use DSP techniques and applied mathematics to replicate a meter's functions in firmware, but this approach gets relatively expensive if the device doesn't already include DSP functions to spare.

An inexpensive analog meter's weakness remains its peak-hold element, a capacitor that must charge quickly to accommodate large signals and accurately for small signals—two mutually exclusive goals. In addition, the nonideal characteristics of the diodes for

(continued on pg 88)

full-wave rectification and peak-hold functions also limit an analog volume-unit meter's dynamic range. Preserving 20 dB of display dynamics and monitoring signal levels that can vary over a 40-dB range, which is typical in consumer electronics, call for a circuit with a dynamic range on the order of 60 dB.

In most instances, traditional circuits fail to simultaneously provide the intended accuracy and slew rate, particularly at low signal levels over a wide dynamic range. The circuit in **Figure 1** offers a simple configuration that delivers high accuracy over a dynamic range that exceeds 60 dB and provides the rapid-attack/slow-decay characteristics that a high-quality display requires.

The heart of the circuit is a Linear Technology LT1011 comparator, IC_2, which monitors the difference between the incoming signal's amplitude and the peak-detected output. It also delivers charging current to a 4.7-μF hold capacitor, C_6, whenever the state of its charge is too low. Unfortunately, the input-to-output delay inherent in comparators and nonlinear amplifiers determines the minimum output-pulse width. If the hold capacitor charges quickly to track large input bursts, the minimum charge step must greatly exceed the level of small signals and thus limits the dynamic range.

Inductor L_1 solves the capacitor-response problem by providing an adaptively variable source of charging current. Adding a 10-mH inductor limits the maximum current rate when the comparator generates narrow pulses, thus reducing the minimum charging amplitude step to a smaller level of 1 mV or less. For wider charging pulses, the current automatically ramps up to higher levels to provide the desired high slewing rate. The minimum charge step is essentially proportional to the signal-step size, ensuring a constant relative accuracy of better than 1 dB over a 60-dB signal range. A signal level of −59 dB corresponds to a 13-mV input, and a meter-scale factor of 0 dB of 2V peak corresponds to the input level necessary for a typical gain-of-20 audio power amplifier to deliver 100W rms into an 8Ω load, or approximately 40V peak output.

The circuit also includes two operational-amplifier stages based on Linear Technology's high-accuracy LT1469 dual op amp. The first stage, IC_{1A}, provides gain of a factor of six in this example, so that a 2V input peak provides a 12V output. The second op-amp stage, IC_{1B}, forms a precision inverting half-wave rectifier. The outputs from IC_{1A} and IC_{1B} and the positive-peak-detected voltage across C_6 combine at IC_2's input to provide a zero-crossing threshold to the comparator. When its input falls below 0V, IC_2's output switches on Q_1 and delivers charge to C_6 until the voltage across C_6 reaches or slightly exceeds the amplified audio voltage. The feedback network comprising R_8 and C_4 provides an optimal volume-unit-metering discharge. EDN

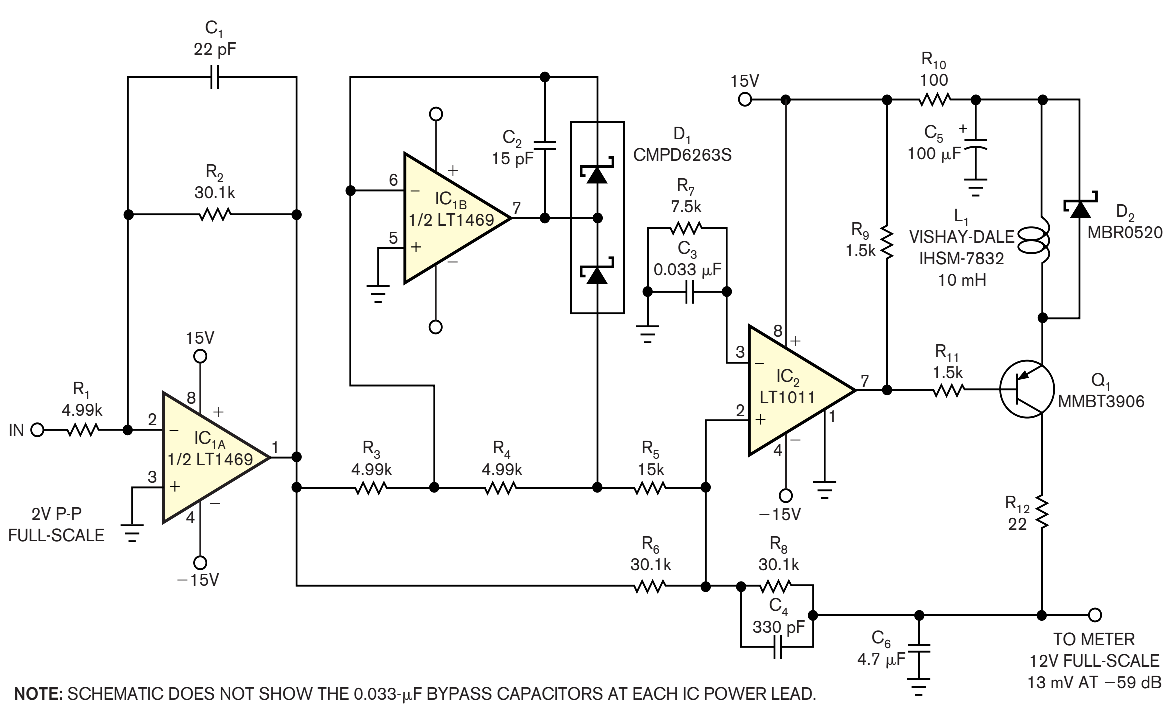

Figure 1 This inductor-compensated voltage-unit meter displays a 60-dB dynamic range.

Pacer clock saves subroutine calls

Enver Torlakovic, Willmot, New South Wales, Australia

This Design Idea outlines an easy-to-implement time-delay routine that requires no subroutine calls and thus avoids possible stack-overflow problems (**Listing 1**). This method also saves RAM space by requiring in most cases only two variables: the PACER_CLOCK as a free-running counter and another variable introduced at a particular instance (for example, TIME_VAR). The routine dedicates the microcontroller's Timer 0 to generate an interrupt-on-overflow instruction every 10 msec or at any other desired interval. You assign the Timer 0 interrupt a low priority in the initialization code and then enable the Timer 0 any convenient time. After assignment, do not alter the interval because many services likely depend on the pacer-clock routine. Note that the routine can achieve delays of as much as 255 times the Timer 0 overflow period.

Listing 1 is written for Microchip's PIC18F242 flash-memory controller, but porting the routine to another microcontroller should pose few problems. When copying the code to paste it into routines, note that you must change the labels—in this example, "wait_loop100"—at each application of the code between the rows of asterisks in the **listing**. EDN

MORE AT EDN.COM

You can download the listing from the Web version of this Design Idea at www.edn.com/051110di2.

LISTING 1—TIME-DELAY ROUTINE

```
int_test_timer0

        btfss       INTCON,     TMR0IF          ; test if Timer0 IRQ was active
        goto        int_test_INT1                               ;if not, check
next INT source
        bcf         INTCON,     TMR0IE                    ;disable Timer0 int.
        bcf         INTCON,     TMR0IF          ; clear Timer1 H/W flag

        ;Jobs to do here:     INC the PACER_CLOCK variable

        incf        PACER_CLOCK             ;This variable is a free running
counter

        ;RELOAD TIMER0 now: (Timer0 set for 10 milliseconds pacer clock)
        ;Note: Rollover occurs after the Timer reaches 0xFFFF
        ;Required number of ticks is calculated as: 0xFFFF-(TMR0H:TMR0L)

        movlw       0x0D                            ; High byte
        movwf       TMR0H                   ; Reload Timer0 high
        movlw       0x61                            ; LOW byte
        movwf       TMR0L                   ; Reload Timer0 low.

        ;Note: for an internal clock period of 0.161002 microseconds, count 62111 clocks
        ;to make up  the 10 millisecond interval.

        bcf         INTCON,     TMR0IF          ; clear Timer1 H/W flag
        bsf         INTCON,     TMR0IE                    ;Re-enable the Timer0
interrupt
        bsf         T0CON,          TMR0ON              ; turn ON Timer0
module

int_test_INT1
        ;code for INT1 starts from here…
        .
        .
retfie          ;return from interrupt

____________________________________________________________
Somewhere in the code:

                    ;******************
        movf                PACER_CLOCK, w          ;get the current Pacer_Clock
state
        addlw       D'10'                                           ; to
count up to 10  Timer0 overflow periods
        movwf       TIME_VAR                              ;load the time variable
with the contents of …

                ;…PACER_CLOCK plus 10
wait_loop100
        ;do some jobs here…
        ;….
        ;…
        movf                PACER_CLOCK, w
        xorwf               TIME_VAR, w                             ;check
if PACER_CLOCK was incremented 10 times
        tstfsz              WREG
        ;if zero, files are equal = Timed Out !
        goto                wait_loop100
        ;otherwise wait longer

                ;******************
```