Application Note 160

July 2016

Single-Channel Power Supply Monitor with Remote Temperature Sense

Nathan Enger

INTRODUCTION

Many applications with a single power regulator can benefit from the monitoring and control features of a power supply manager, but most power supply manager ICs have more than one channel. In an application that only has one power supply, there will be an unused set of DAC and ADC pins. Instead of letting the unused channel go to waste, we can use these pins, and a bit of microcontroller code, to sense remote temperature. The LTC®2970 is a 2-channel power supply monitor and controller. Each channel has two 14-bit ADCs to measure voltage and current, and one 8-bit DAC to servo the power supply voltage. It can drive an attached bipolar junction transistor to make a delta-V_{BE} measurement, and the microcontroller can use the measured voltages to calculate temperature. The cost of the added components is as little as 3 or 4 pennies.

The Circuit

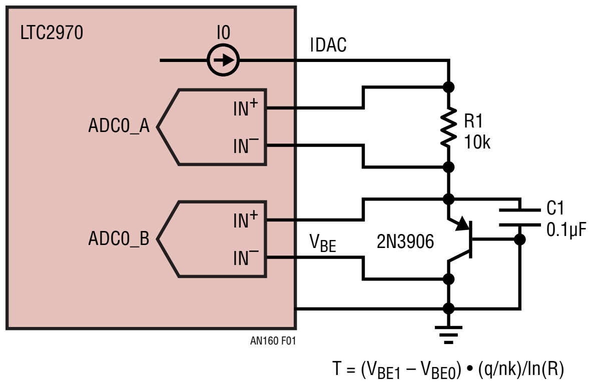

Figure 1. LTC2970 External Temperature Sense Circuit

To make a remote temperature sensor with the LTC2970, run two different currents (I_{LOW} and I_{HIGH}) through the transistor and measure V_{BE} at both currents (V_{BE_LOW} and V_{BE_HIGH}). The difference between the two V_{BE} measurements (ΔV_{BE}) is a direct function of temperature. Use Equation 1 to calculate temperature in the BJT.

$$T = \Delta V_{BE} \bullet \frac{q}{n \bullet k} \bullet \frac{1}{\ln\left[\frac{I_{HIGH}}{I_{LOW}}\right]} \qquad (1)$$

Shown in Figure 1, the LTC2970 has everything that we need in one of its two channels. The IDAC output can drive a programmable current between 0µA and 255µA. One of the ADC inputs can directly measure V_{BE} across the BJT. The other ADC input can measure a useful proxy for current by sampling the voltage across a resistor. This arrangement avoids several traditional problems with semiconductor temperature sensors. The first is knowing how much of the measured V_{BE} voltage is due to voltage drop in the wiring between the current source and the transistor. Here we measure V_{BE} directly at the transistor terminal, not at the DAC output pin, making resistance in the line less important (though transistor internal resistance will affect the measurement slightly). The second difficulty is knowing the precise ratio of I_{LOW} and I_{HIGH} currents. Because we measure voltage across a resistor, instead of relying on the IDAC to be accurate, we have very good knowledge of the ratio of currents. We don't need to know if the DAC is supplying precise ratios of currents, and we also don't need to know what the resistor value actually is. The voltage across it will be a pure function of current, and the ratio of two voltages will be equal to the ratio of the two currents.

Programming

The LTC2970 is a programmable device, containing registers to control the IDAC current, and to report the ADC readings, but the external temperature calculation must be done in software. A ready platform for such software is the Linduino™, an Arduino clone with isolated power, produced by Linear Technology®. Linduino communicates with the LTC2970 over the I^{2}C/SMBus, and with just a few lines of C code, performs the necessary temperature calculations. Any similar system that can talk to the LTC2970 over the I^{2}C/SMBus bus can run the C code and calculate temperature by this method.

Figure 3 shows a flow diagram for the very simple algorithm to measure temperature. Blue color indicates I^{2}C bus transactions, red indicates a delay, and green a calculation. Not represented are any steps to filter the temperature results. Filtering is separate, and we will treat it as an independent topic later in this document.

Figure 2 shows an excerpt of the Linduino code. The lines are color coded to indicate function, corresponding to the flow diagram in Figure 3. The essence of the algorithm is simply forcing two different currents and measuring the resulting voltages, then solving the equation for temperature.

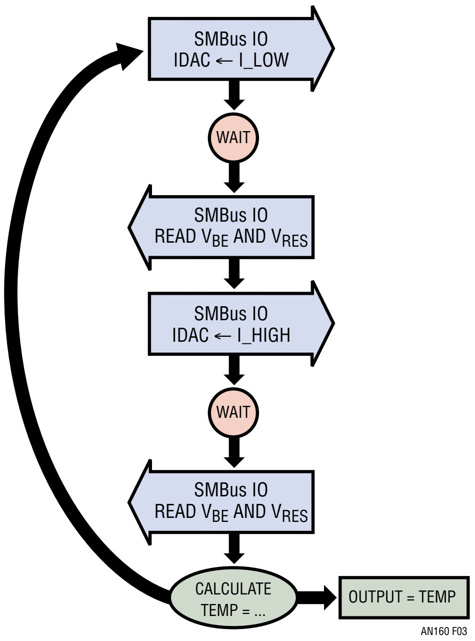

Figure 3. LTC2970 External Temperature Sense Flow Diagram

```
// BJT properties
const float bjt_n = 1.016;
const float q_by_k = 11.60452e3;
float q_by_n_k = (q_by_k / bjt_n);

//! Measure voltage and current at low current
smbus->writeWord(local_i2c_address, LTC2970_CH1_A_IDAC, dac_current_low);
delay(meas_delay);
reg_i_low =
 (MASK_ADC & smbus->readWord(local_i2c_address, LTC2970_CH1_B_ADC));
reg_vbe_low =
 (MASK_ADC & smbus->readWord(local_i2c_address, LTC2970_CH1_A_ADC));
voltage_low = ((float)(reg_vbe_low & MASK_ADC))*500e-6;
current_low = ((float)(reg_i_low & MASK_ADC))*500e-6;

//! Measure voltage and current at high current
smbus->writeWord(local_i2c_address, LTC2970_CH1_A_IDAC, dac_current_high);
delay(meas_delay);
reg_i_high =
 (MASK_ADC & smbus->readWord(local_i2c_address, LTC2970_CH1_B_ADC));
reg_vbe_high =
 (MASK_ADC & smbus->readWord(local_i2c_address, LTC2970_CH1_A_ADC));
voltage_high = ((float)(reg_vbe_high & MASK_ADC))*500e-6;
current_high = ((float)(reg_i_high & MASK_ADC))*500e-6;

//! Set DAC current to zero between measurements
smbus->writeWord(local_i2c_address, LTC2970_CH1_A_IDAC, MASK_IDAC);
delay(meas_delay);

//! Calculations
vbe_delta = voltage_high - voltage_low;
current_ratio = current_low / current_high;
ln_current_ratio = log(current_ratio); // natural log

temperature =
(vbe_delta * (-1 / ln_current_ratio) * q_by_n_k) - ABS_ZERO;
```

Figure 2. Linduino Temperature Calculation Code

The LTC2970 uses one ADC with a multiplexer front end to make measurements at each of seven inputs. The inputs are sampled in a round-robin fashion, and the time between subsequent samples of one input depends on how many inputs are selected for sampling in the ADC_MON register. When the round-robin ADC is programmed to sample every input, each input is sampled once every 240ms. The LTC2970 provides a single bit "new" indicator in each ADC register to indicate that the data has not been read yet. Each time the ADC stores a new conversion result, the "new" bit is set, and each time the register is read, the bit is cleared.

TEST HARDWARE

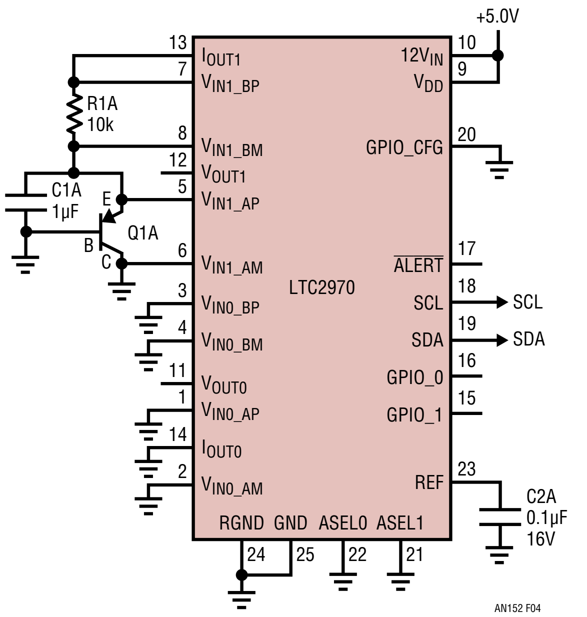

Figure 4. LTC2970 Temperature Test Schematic — One of Eight on the Board

The test hardware is very simple. In principle it is just a BJT and a resistor connected directly to the LTC2970. In practice there are additional considerations. Noise is the biggest concern. Since the circuit measures temperature remotely, there are interconnects, which can pick up capacitively-coupled noise and corrupt the measurements. Adding noise filters helps, but it also slows down the analog settling times, and makes the measurements take longer. The best approach is to run short sense lines differentially, with adequate isolation and shielding from noisy lines on the board. This reduces the required filter time constants and the settling time of the circuit[1]. Because the measurement requires two different operating points and sub-millivolt levels, there is a trade-off between noise filtering and measurement time.

For sensor evaluation and calibration, an oil bath provides a precisely controlled and stable temperature environment. See Figure 6. The measurement board contains eight LTC2970s and associated sensors, and is small enough to fit into the oil chamber. The only wires going to the board are +5V, GND, SDA, and SCL (power and communications). These are all supplied by the Linduino that controls the algorithm. Figure 4 shows a schematic of one of eight test devices on the board. Figure 5 shows what the board looks like before going into the oil bath.

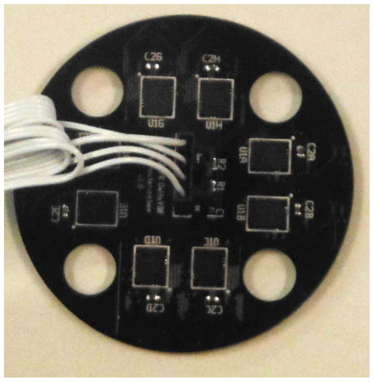

Figure 5. 8-Device LTC2970 Oil Bath Temperature Test Board

Figure 6. Temperature Test Oil Bath

Note 1. See Linear Technology Application Note 137: *Accurate Temperature Sensing with an External P-N Junction.*

The temperature conversion routine must program each new current, then wait for two things to happen: the voltages must settle and the ADC round-robin loop must refresh all ADC readings. We can accommodate the first only by waiting after writing to the IDAC register. This wait time should be no less than 90ms (9 time constants under the worst case). We can accommodate the second requirement by reading the "new" bit in the ADC result register to ensure that the result is fresh. We use a simple "READ_NEW" function to loop on a register read until the "new" bit becomes set, indicating a new ADC value. The code as written waits 350ms between programming current settings, and each temperature reading requires two current settings, for a total sample time of 700ms.

TEST RESULTS

Temperature Step Size

We must test several properties of this system. First is temperature step size; we want to verify that the temperature steps are as calculated in our theoretical framework (presented later) in Equations 10 through 14. Second, verify absolute accuracy: measure the circuit over a full range of temperatures.

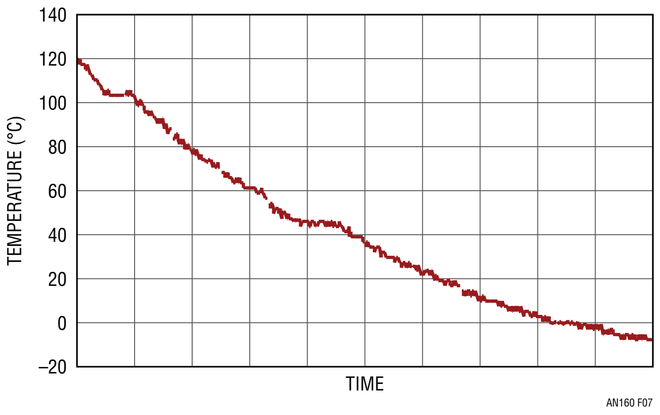

Figure 7. Temperatures Calculated During a Temperature Sweep from +120°C to –10°C

To verify the calculated predictions of temperature step size, we sweep temperature over a wide range so that the ADC is constantly measuring new values for I_{HIGH}, I_{LOW}, V_{BE_HIGH} and V_{BE_LOW}. We expect to see temperature steps of the calculated sizes, in various combinations of V_{BE}, I_{LOW}, and I_{HIGH}. In Figure 7 the temperature is swept from +120°C to –10°C as quickly as the equipment will allow, while repeated measurements are taken. The plot shows measurements taken by one LTC2970. Just visible at this scale are the discrete measurement steps.

Figure 8 shows the temperature step sizes during the temperature sweep from +120°C to –10°C. Note that the steps are not continuously distributed, but are clustered around discrete sizes predicted by the sensitivity calculations: ±2.3°C and ±0.25°C, caused by ADC LSB steps in V_{BE} and I_{LOW} samples. A few temperature steps in Figure 8 include two ADC LSB steps, so have magnitude near 4.5°C. We address these calculated predictions later in the Theory section of this document.

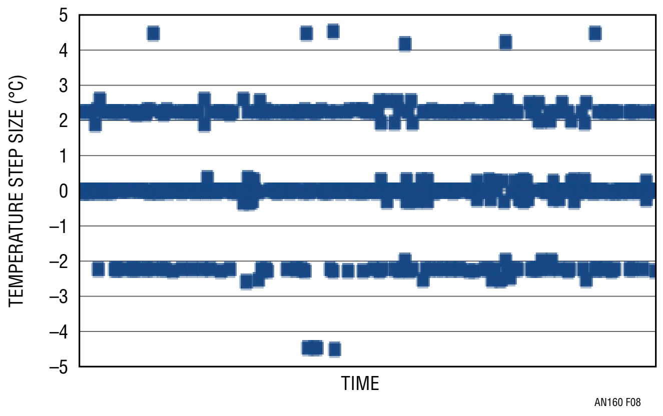

Figure 8. Temperature Step Size (°C) During a Temperature Sweep from +120°C to –10°C

Temperature Accuracy

To gauge measurement absolute accuracy we use the oil bath with a long-term stable temperature. Each of the eight LTC2970s on the board are measured separately. We collect data for each part with two averaging steps. The first includes 40 samples of the sensor with dithering of the IDAC current by ±1LSB to scramble noise sources. Taking the mean average over 40 samples smooths out the bumps. The second averaging uses 50 of the dithered-averaged measurements, and computes their average. The graphs below show mean and range of the temperature error (difference between measured and actual temperature) at temperatures from –15°C to 125°C. The red trace is the error between the mean average of all measurements and the actual oil temperature. The blue and green are the standard deviations of measurement errors at each temperature (how far any one temperature measurement deviated from actual).

The mean average of all samples (red trace) is within 1.2°C of the actual temperature, indicating that the analog errors in the system (transistor n, ADC errors, leakage, etc.) are small compared to the ADC measurement noise. The blue and green traces bound the temperature within 2.3°C of the actual temperature (±1LSB).

These measurements show that, even in an uncalibrated circuit, the system is limited primarily by ADC accuracy, and all other non-idealities are secondary for the LTC2970 in this configuration.

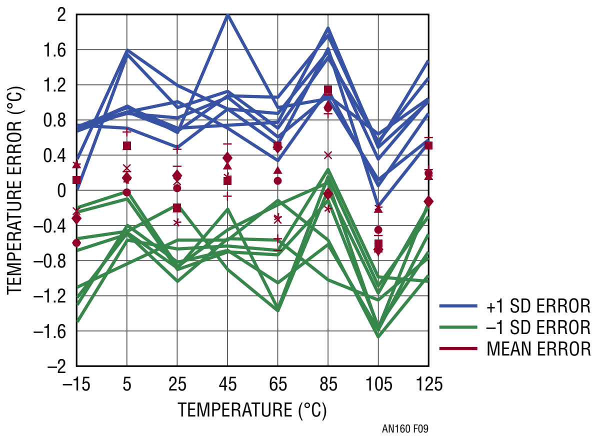

Figure 9. Temperature Error Statistics Across a Temperature Sweep

The distributions of the traces in Figure 9 are not repeatable. They are random. Notice that there are some temperatures at which the errors are larger. This is due to the fact that these temperatures produce voltages where the ADC is close to a bit boundary, and very sensitive to noise. At other temperatures the voltages are far away from bit boundaries, so less sensitive to noise. We will see that a combination of aliasing and quantization produce a wandering of the signal, even after averaging, between the limits of one ADC LSB. This is an unavoidable artifact of the ADC step size.

THEORY

For a bipolar junction transistor (BJT), V_{BE} depends upon the magnitude of the current flowing and upon temperature. V_{BE} is a direct function of current, $I_C = \beta I_B$, and an inverse function of temperature.

Figure 10 shows the relationship between V_{BE} and I_C in an ideal BJT. Actual BJT performance will be limited by leakage on the low end (seen as a flattening of the curve), and by resistances at the high end. Notice that at a given current (horizontal line), V_{BE} changes in inverse proportion to temperature. Conveniently, we can rely on the well known fact that at a particular current, $I_C(0)$, the transistor V_{BE} will change linearly with temperature (at approximately –2mV/°C). At first glance this property is not obvious from the basic transistor current equation:

$$I_C = I_S(T) \bullet \left[\exp\left(\frac{q}{nkT} \bullet V_{BE}\right) - 1\right] \quad (2)$$

which we can simplify for typical values of I_C:

$$I_C = I_S(T) \bullet \exp\left(\frac{q}{nkT} \bullet V_{BE}\right) \quad (3)$$

Using this equation, as we increase temperature we would expect to see higher V_{BE}, not lower. The fact is that $I_S(T)$ is not constant. It is a strong function of temperature, among other things, and it dominates the temperature dependence of the device. Additionally, the "other things" that affect $I_S(T)$ are manufacturing related, and cause part-to-part variability.

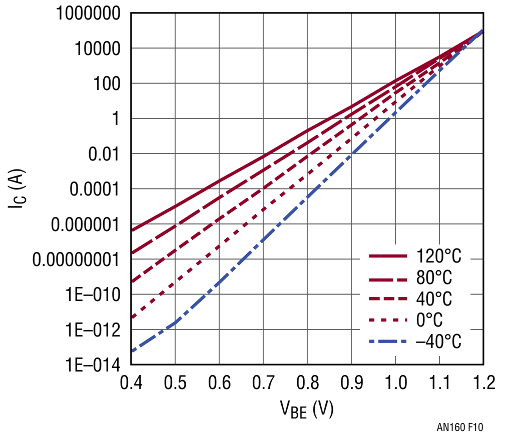

Figure 10. BJT Collector Current vs Base-Emitter Voltage

A better way to visualize the situation is shown in Figure 11. It shows the effects of temperature on V_{BE} for several values of current. Notice that multiplying the current in the transistor increases V_{BE} voltage, giving positive ΔV_{BE}. Also notice that the change is smaller at lower temperatures than at higher temperatures. This is a useful insight! Driving the transistor with two different currents, the difference between V_{BE} values at high temperature is larger than at lower temperature. ΔV_{BE} is a direct function of temperature.

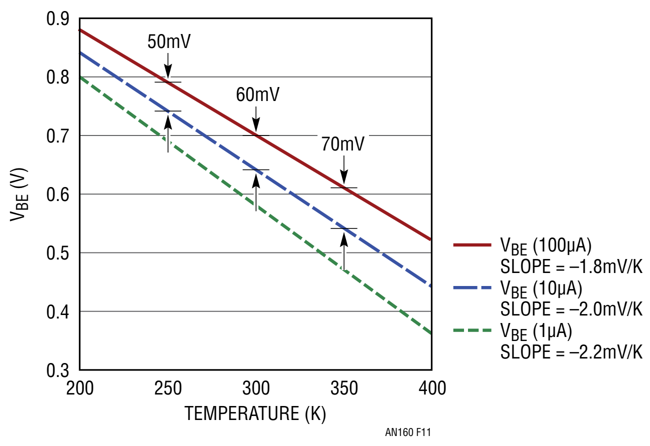

Figure 11. V_{BE} vs Temperature in a BJT

Figure 12 shows two different transistors operating with two different bias currents, and having different V_{BE} voltages. We can either use two transistors as shown, or drive one transistor with two different currents. Each approach has its advantages.

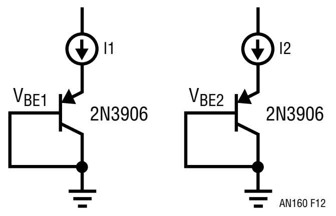

Figure 12. Sensing Temperature Using BJT ΔV_{BE}

It is simple to solve the transistor equation to derive the temperature relationship:

$$V_{BE} = \frac{nkT}{q} \ln\left(\frac{I_C}{I_S(T)}\right) \tag{4}$$

$$\Delta V_{BE} = V_{BE\_HIGH} - V_{BE\_LOW} = \frac{nkT}{q}\ln\left(\frac{I_{HIGH}}{I_S}\right) - \frac{nkT}{q}\ln\left(\frac{I_{LOW}}{I_S(T)}\right) \tag{5}$$

$$\Delta V_{BE} = \frac{nkT}{q}\left[\ln\left(\frac{I_{HIGH}}{I_S(T)}\right) - \ln\left(\frac{I_{LOW}}{I_S(T)}\right)\right] \tag{6}$$

$$\Delta V_{BE} = \frac{nkT}{q} \bullet \ln\left(\frac{I_{HIGH}}{I_{LOW}}\right) \tag{7}$$

$$T = \Delta V_{BE} \bullet \frac{q}{nk} \bullet \frac{1}{\ln\left(\frac{I_{HIGH}}{I_{LOW}}\right)} \tag{8}$$

Quantization Errors

The BJT gives a continuous analog voltage, but we must measure it with an analog-to-digital converter that quantizes both voltage and time. As we saw in the measured data, this has implications for the accuracy of our temperature measurements.

An ADC is a finite-resolution device. The output is a finite-precision number that is close to, but not exactly equal to the input voltage. It takes steps of a finite size (Δ), and cannot represent voltages more precisely than the size of its steps. The difference between the actual voltage and the measured voltage is called quantization error, shown in Figure 13. We need to understand how this quantization error affects the temperature that we measure.

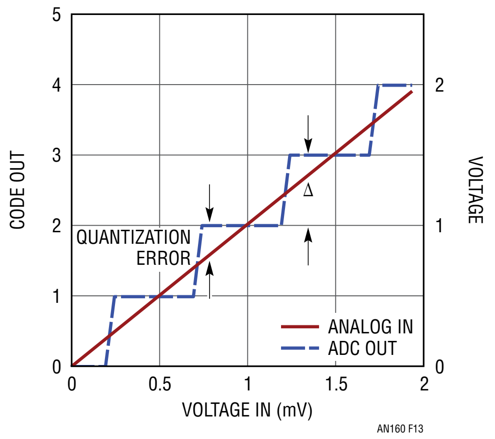

Figure 13. ADC Quantization Error

The fundamental equation that we solve to convert transistor V_{BE} voltage into temperature is Equation 8. The calculation is sensitive to each of the parameters that we measure with the ADC. Errors or uncertainties in each measurement affect the calculated temperature, depending on the sensitivity to each parameter. Sensitivity is defined as the slope of the curve in response to changes in a parameter. The charge on an electron and Boltzmann's constant do not change, so the ratio q/k is constant. The value of n for the transistor is taken as a constant for a given device, but it can vary from device to device, and certainly depends upon the device family (2N3904, 2N3906, etc.). The temperature equation is also sensitive to the value of the transistor non-ideality factor, n. The values of these constants are:

$q = 1.60217662 \bullet 10^{-19}$ coulombs

$k = 1.38064852 \bullet 10^{-23}\ m^2\ kg\ s^{-2}\ K^{-1}$

$n = 1.016$ (typical for a 2N3906)

ΔV_{BE} is the difference between V_{BE} at two different bias currents: V_{BE_LOW} and V_{BE_HIGH}. The two bias currents are I_{LOW} and I_{HIGH}, and are well known, but their precise ratio is more important than their absolute value.

The finite precision of the ADC causes finite steps in the temperature calculation. Each voltage measurement has a different sensitivity. We can calculate sensitivity by taking the derivative of the Equation 8 with respect to the variable of interest.

Sensitivity to measured ΔV_{BE} is given by:

$$\frac{\partial T}{\partial \Delta V_{BE}} = \frac{q}{n \bullet k} \bullet \frac{1}{\ln\left(\frac{I_{HIGH}}{I_{LOW}}\right)} \quad (9)$$

ΔV_{BE} at room temperature is in the neighborhood of 66mV with I_{HIGH} and I_{LOW} of 255µA and 20µA respectively. The ratio of currents is 255µA/20µA = 12.75. The natural log is ln(12.75) = 2.5455. Assuming a nominal value of n=1.016, this sensitivity is:

$$\frac{\partial T}{\partial \Delta V_{BE}} = 11.60452 \bullet 10^3 \bullet \frac{1}{\ln(12.75)} \quad (10)$$

$$\approx 4559°C/V$$

The LTC2970 ADC can resolve steps of 500µV. Multiplying by a ΔV_{BE} 1-LSB step (change in either V_{BE} high or low) of 500µV gives: 2.28°C/LSB (See Table 1).

Similarly, the calculation is sensitive to measured currents, I_{HIGH} and I_{LOW}. Because I_{HIGH} is approximately $10 \bullet I_{LOW}$ we will find that the temperature calculation is approximately 10× more sensitive to changes in I_{LOW} than I_{HIGH}. In general, sensitivity to the *ratio* of currents (Δr) is:

$$\frac{\partial T}{\partial \Delta r} = \Delta V_{BE} \bullet \frac{q}{n \bullet k} \bullet \frac{I_{LOW}}{I_{HIGH}} \bullet \frac{1}{\ln^2\left(\frac{I_{HIGH}}{I_{LOW}}\right)} \quad (11)$$

Assuming a nominal ΔV_{BE} in the neighborhood of 66mV, then we have a sensitivity of approximately:

$$\frac{\partial T}{\partial \Delta r} \approx 0.066 \bullet 11605 \bullet \frac{1}{12.75} \bullet \frac{1}{6.5} \approx 9.24°C \quad (12)$$

Note that Equation 12 represents sensitivity to the *ratio* of currents.

Sensitivity to only the larger current (I_{HIGH}) is:

$$\frac{\partial T}{\partial I_{HIGH}} = \Delta V_{BE} \bullet \frac{q}{n \bullet k} \bullet \frac{I_{LOW}}{I_{HIGH}} \bullet \frac{1}{\ln^2\left(\frac{I_{HIGH}}{I_{LOW}}\right)} \bullet \frac{1}{I_{LOW}} \quad (13)$$

Where I_{LOW} is actually a measured voltage across a resistor, so the measured value is $I_{LOW\_ACTUAL} \bullet R$.

Assuming I_{LOW} = 25µA • 10kΩ = 0.25, and a 500µV ADC LSB step we have temperature steps of 0.0236°C/LSB (See Table 1).

Similarly, sensitivity to the small current (I_{LOW}) is:

$$\frac{\partial T}{\partial I_{LOW}} = \Delta V_{BE} \bullet \frac{q}{n \bullet k} \bullet \frac{I_{LOW}}{I_{HIGH}} \bullet \frac{1}{\ln^2\left(\frac{I_{HIGH}}{I_{LOW}}\right)} \bullet \frac{-I_{HIGH}}{I_{LOW}^2} \quad (14)$$

Assuming I_{HIGH} = 255µA • 10kΩ = 2.55 and a 500µV ADC LSB step, we have temperature steps of –0.241°C/LSB (See Table 1).

All sensitivity results are assembled in Table 1 on page 10. These are the parameters that the ADC measures, so sensitivity is in degrees Centigrade per volt. Currents are measured as voltage across the sense resistor.

Circuit Accuracy

Temperature sensor accuracy is limited by several factors. Fundamentally Equation 8 is sensitive to the value of the non-ideality factor of the transistor, the measured values of voltage, and measured values of current. There are also several hidden nonlinearities and sensitivities that do not show up in Equation 8. These include errors in the ADC measurements – integral nonlinearity (INL), differential nonlinearity (DNL), gain, and offset — as well as non-constant transistor β.

ADC ERRORS

The ADC has several types of errors, listed in the data sheet. These are errors in the measured values caused by analog imperfections in the ADC.

ADC Gain:

ADC gain should ideally be 1.0. The LTC2970 data sheet states that gain error is < 0.4%. At worst this gives a gain of 1.004. For V_{BE} measurements of 0.574V and 0.640V (at room temperature), the measured voltages with gain error would be:

$$V_{BE} = 0.574 \bullet 1.004 = 0.57630$$

$$V_{BE} = 0.640 \bullet 1.004 = 0.64256$$

This changes ΔV_{BE} from 66.0mV to 66.264mV. The 264µV difference is about half of 1LSB of 500µV. This is equivalent to about 1.2°C temperature error.

ADC Offset:

ADC offset is effectively a fixed value added to every code. It is the ADC output when the input is 0. For the temperature calculation, the most sensitive measurement, by far, is ΔV_{BE}. Because ΔV_{BE} is a differential measurement, constants like ADC offset are eliminated by subtraction:

$$(V_1 + V_{OFFSET}) - (V_2 + V_{OFFSET}) = V_1 - V_2 \qquad (15)$$

Offsets do affect the ratiometric current measurement; however, the effect is small because sensitivity to the current ratio is low.

ADC INL:

The LTC2970 data sheet shows INL as a function of input voltage, with typical values of 1LSB for voltages near 1V, and nearing 0LSB for input voltages near 0V. This implies that for V_{BE} measurements near 0.6V we should expect ~0.6LSB INL error, and a change of approximately 1LSB/V, or 500µV/V. See Figure 14. The ADC INL curve has the same shape for every part, so we can treat INL as a gain error, with an order of magnitude smaller effect than the actual ADC gain error above. This should result in temperature errors less than 0.13°C.

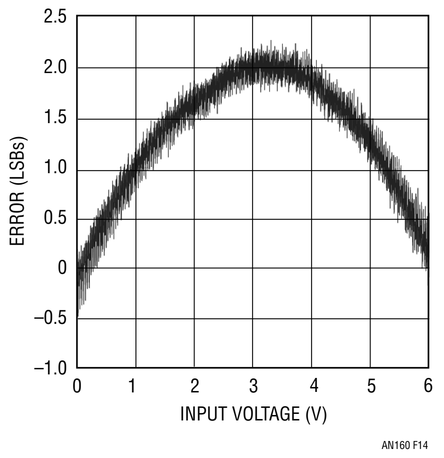

Figure 14. LTC2970 ADC Integral Nonlinearity (INL)

ADC DNL:

The LTC2970 data sheet specifies DNL (code-to-code step size errors) less than 0.5LSB, or 250µV. This is of the same order as the ADC gain term, resulting in temperature errors <1.2°C. These errors are caused by random mismatch, and amenable to scrambling with a dithering source that changes the applied BJT current, and subsequent removal through averaging.

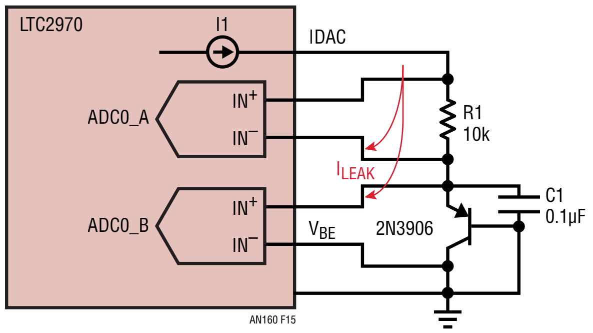

Figure 15. ADC Input Leakage Path

ADC Input Leakage:

The LTC2970 ADC input pins are not infinite impedance. They leak a little. This leakage causes current to flow in the sense resistor that does not flow in the BJT. This sensed current is primarily an issue for the current ratio, and not for the measured V_{BE}. The magnitude of the leakage currents is < 0.1µA. There are two ADC input channels connected to one end of the resistor, so we use the maximum of 0.2µA. The error in the current ratio is:

$$ERR_{RATIO} = \frac{255}{20} - \frac{255.2}{20.2} = 0.116 \quad (16)$$

The resulting temperature error is –0.85°C.

The LTC2970 data sheet does not state it, but this error is worse at high temperatures, so it amounts to a nonlinear temperature gain error.

TRANSISTOR PROPERTIES

Variations in n:

Non-ideality factor (n) is essentially constant in a given transistor, but varies from transistor to transistor. Within a given family (2N3906), the device-to-device variations in n are small. Between families (2N3906 vs 2N3904) the differences are larger. It is important to use an accurate value for n in calculations and C code. Sensitivity to small errors in the non-ideality factor of the transistor is given by this equation:

$$\frac{\partial T}{\partial n} = \Delta V_{BE} \bullet \frac{q}{n^2 \bullet k} \bullet \frac{1}{\ln\left(\frac{I_{HIGH}}{I_{LOW}}\right)} \quad (17)$$

$$\frac{\partial T}{\partial n} \approx 0.066 \bullet 11605 \bullet \frac{-1}{1.016^2} \bullet \frac{1}{12.75} \bullet \frac{1}{2.55} \quad (18)$$
$$\approx -22.8$$

For a change in n of 0.001 we expect a change of –0.0228°C.

Non-Constant β:

The transistor equation at the root of all of our calculations is:

$$I_C = I_S e^{\left[\frac{V_{BE}}{nV_T}\right]} = \beta \bullet I_B \quad (19)$$

But I_C is not the same as input current, I_{IN}. A portion of the input current flows to the base of the transistor, and the remainder flows through the collector. The sum, the input current, is the emitter current.

$$I_{IN} = I_E = I_C + I_B \quad (20)$$

$$I_E = \frac{(\beta + 1)}{\beta} \bullet I_C \quad (21)$$

We are measuring the current going into the BJT, and calculating the ratio of measured currents (the ratio of I_E):

$$\frac{\frac{(\beta_{HIGH} + 1)}{\beta_{HIGH}} \bullet I_{CHIGH}}{\frac{(\beta_{LOW} + 1)}{\beta_{LOW}} \bullet I_{CLOW}} \quad (22)$$

To the extent that β does not change with applied current, the ratio of currents will remain unchanged, but if $\beta_{LOW} \neq \beta_{HIGH}$ then the ratio will be off, distorting the temperature measurement. If $\beta_{HIGH} = 44$ and $\beta_{LOW} = 40$ then the calculated temperature will be off by ~0.33°C. For larger values of β the temperature error will be smaller.

The complete set of temperature inaccuracies is given in Table 1. These are specific to the LTC2970 operating with a 2N3906 transistor.

Table 1. Sensitivity of Calculated Temperature to Parameters**

Parameter	Sensitivity (°C/V)	Error: OC per ADC LSB
ΔV_{BE} (either $V_{BE(LOW)}$ or $V_{BE(HIGH)}$)	4559	2.28 (°C/LSB)
I_{HIGH}	47.2	0.0236 (°C/LSB)
I_{LOW}	–482	–0.241 (°C/LSB)
Parameter	**Sensitivity (°C/unit)**	**Error (°C)**
ADC Gain	–	< 1.2
ADC INL	–	< 0.13
ADC DNL	–	< 1.2
ADC Input Leakage	–	> –0.85
Non-ideality Factor (n)	–	> –0.0228
Transistor β	–	< 0.33

** NOTE: Assumes given biasing currents.

QUANTIZATION AND NOISE

In general, ADC sampling is subject to two kinds of sampling errors: time quantization and voltage quantization. Time quantization because the signal is only sampled at discrete points in time, not continuously measured. Voltage quantization because the ADC can only represent a finite number of levels, and the input voltage is always found between two levels, never precisely at one level.

Voltage Quantization

A typical method of representing the output of an ADC is as the signal value plus some added noise:

$$\text{ADC Code}[i] = \text{Gain} \bullet (\text{Input Voltage} + e[i]) \quad (23)$$

Where Gain = $2N/V_{FULL\_SCALE}$, and e[i] is the error at the sample number i. The quantization error term, e[i], represents the voltage difference between the input voltage and the nearest ADC code voltage. This error can be as large as ±Δ/2, when the input is midway between two ADC codes. See Figure 13.

For a noiseless DC input, e[i] is the same for all values of i. All of the ADC outputs will be identical and have identical errors. No amount of post-processing will reduce the error. Taking the average of 100 samples will give a result with the same error as taking one sample.

If there is noise present, then there is a finite probability that the ADC will choose a different code than it would in the noiseless case. When this happens, ADC Code[i] ≠ ADC Code[i – 1] and averaging several ADC samples together can give a more precise answer, effectively averaging out some of the quantization error. In order for averaging to be effective, the noise must be dynamic over time, uncorrelated to the input (truly random) and have symmetrical distribution around zero mean. It should also be larger in magnitude than one LSB of the ADC. When this is true, then averaging a suitably large number of samples (N) can reduce the noise floor to below one least significant bit (LSB) of the ADC, such that the averaged minimum step size approaches:

$$V_{STEP} \rightarrow \frac{V_{LSB}}{N} \quad (24)$$

This is possible because the noise causes the ADC output to "chatter" between adjacent codes, but the noise itself has zero mean, so it drops out of the average. The remaining average value will be more precise than the noiseless measurement.

To see why this is true, examine Figure 16. The noise margin on the Y-axis is the magnitude of noise required to produce a different ADC output given the DC input on the X-axis. If the input is sitting at Δ/2 away from the nearest ADC threshold voltage, then a noise event greater than Δ/2 is required to change the code. If the input voltage is sitting directly on top of the code transition then any noise will cause the code to change.

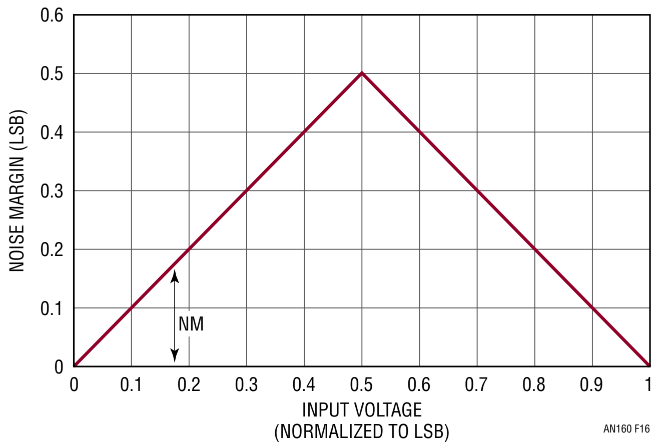

Figure 16. Noise Margin for ADC Input Voltage (Normalized to One ADC LSB)

We can calculate the probability of a noise event causing an ADC output change by applying this noise margin to the probability density function of the noise (if it is known). Probability of disruption is the integrated area under the curve outside of the window defined as $\mu + NM$ and $\mu - NM$, where μ is the mean (presumably $\mu = 0$). See Figure 17.

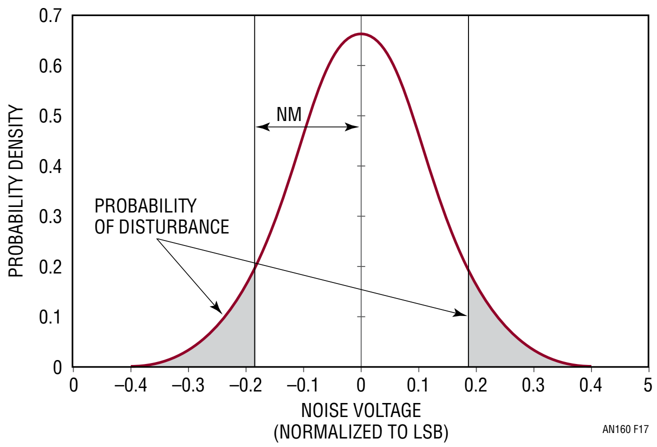

Figure 17. Noise Distribution and Probability of Disturbance

Sweeping the ADC input voltage over a range produces a repeating “ripple” of probability that a noise event will disturb the ADC reading. Near the code transitions the probability is high, and near the center of the code the probability is at a minimum. Note that the precise shape depends upon the noise statistics. Here we have depicted it as Gaussian, but it is more likely to have a multi-modal distribution due to discrete noise frequencies. In any case, the repetitive nature shown in Figure 18 will still hold, even if its exact shape is different.

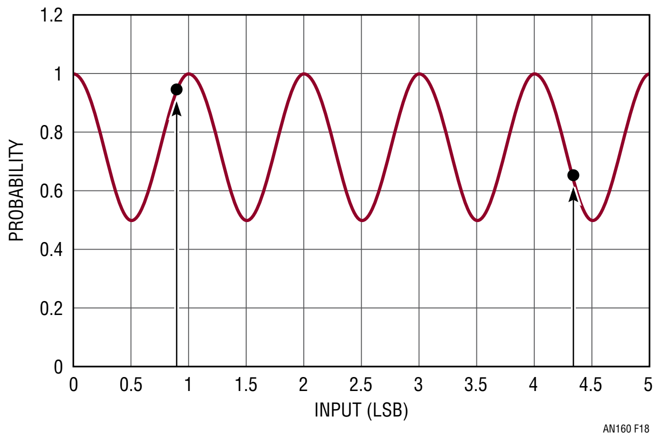

Figure 18. Probability of a Noise Disturbance vs ADC Input Voltage

Applying two different DC input voltages to the ADC (for the temperature measurement), produces two different probabilities of disruption to the measured value (the two dots in Figure 18). Since the temperature measurement is composed of multiple ADC readings we expect that some temperatures are more sensitive than others, and as we sweep the temperature across a range, each ADC reading will become noisy in turn as it approaches its closest ADC code threshold.

Time Quantization

Additional measurement noise arises because the ADC takes samples at discrete time points, and it must necessarily sample a signal that is NOT band limited within the Nyquist bandwidth. To see why this is true, note that the temperature measurement algorithm makes discrete steps between high and low current, and that the ADC must accurately measure the settled voltages at each step. In order for the voltage to settle between ADC readings, there must be a delay of at least 9 time constants (dictated by the 500µV resolution of the ADC); the RC time constant of the circuit must be smaller than 1/9th of the time between ADC samples, T_{SAMPLE}. The bandwidth of this circuit is, then:

$$BW = \frac{1}{2\pi\tau} > \frac{9}{2\pi T_{SAMPLE}} \qquad (25)$$

Sampling theory tells us that energy at frequencies greater than

$$\frac{F_S}{2} = \frac{1}{2 \bullet T_{SAMPLE}}$$

will alias down in frequency and become part of the sampled signal in the baseband. It becomes impossible to distinguish input noise near DC from noise near multiples of the sampling frequency. This means that all of the circuit noise that cannot be filtered before sampling will show up in the sampled signals. A digital filter following such an aliasing sampler cannot remove the noise. Figure 19 shows this.

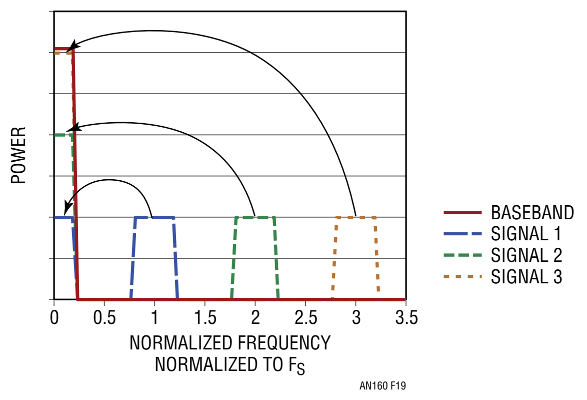

Figure 19. Noise Aliasing in a Sampled Signal

Quantization and Filtering

Because a temperature measurement is a combination of four ADC readings, two at low current and two at high current, there is a combined probability that one or more of the measurements will be perturbed by noise. Averaging several measurements at a given temperature will smooth out the noise to some extent.

The resulting waveform from the ADC quantizer output with noise and aliasing at its input may look something like the example in Figure 20.

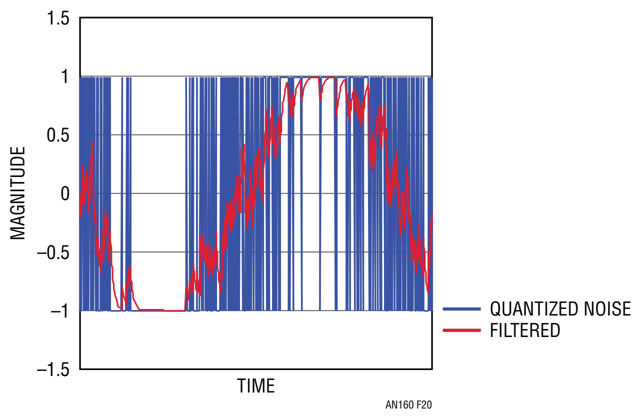

Figure 20. Quantized Noise and Filtered Result

The ADC output steps between adjacent codes more often during more sensitive times, and less often, or not at all during less sensitive times. Noise from high frequencies is aliased down in frequency and summed with baseband signals. The ADC chooses the code that is closest to the (noisy) voltage at its input. When the ADC remains at one code for many output samples, the filter settles to one state. If instead the ADC output chatters between two different codes the filter assumes some intermediate value.

Limitations of Averaging

There is a reason that dedicated temperature sensor ICs like the LTC2983 use 24-bit ADCs to measure analog values. The signals are very small, and often buried in noise. In this circuit we are using the LTC2970, with a 14-bit ADC having 500μV/LSB. This is good enough for reasonably accurate temperature calculations, but we must understand the limitations. We calculated earlier that a single bit represents ~2.3°C, so stepping from one ADC code to the next increments the temperature by that much.

The plot below shows repeated individual measurements taken from a sensor held at a constant 27°C. The data is not averaged, so we can see the ADC transition steps. Not surprisingly, this plot appears similar to the example in Figure 20.

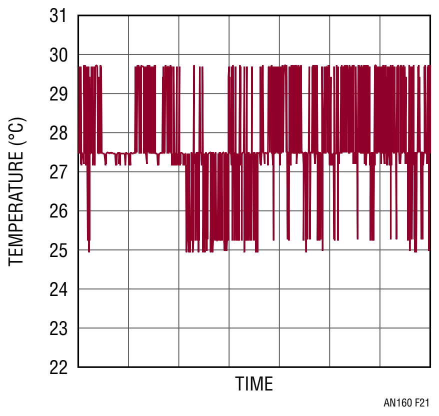

Figure 21. Temperature Output with No Processing

In a noiseless world, we could take 10 measurements, and they would all be identical, though all off by up to 2.3°C. In a system with noise, we have the potential for the voltages to cross bit boundaries between measurements, so sequential measurements at identical temperature differ by ~2.3°C. To reduce the effects of noise we can take multiple measurements and average (or otherwise filter) them. This reduces the difference from one measurement to the next, but it does not conquer the effects of aliasing, which can produce noise frequencies near DC that the digital lowpass filter does not remove.

Figure 22 shows the moving average of reported temperature for another sensor at room temperature. The moving average is computed over 40 samples with an update on every sample. The data is less noisy on a sample-by-sample basis than Figure 21, but over time it still wanders.

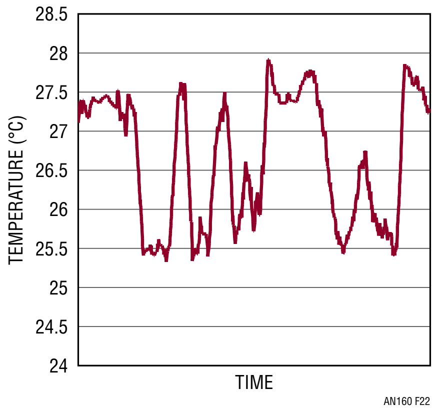

Figure 22. Temperature Moving Average at Room Temperature

Beyond simple averaging, one approach to further smoothing out this ripple is to "dither" the IDAC currents at which the V_R and V_{BE} measurements are taken, changing the IDAC output current by one or two LSBs at each reading. Taking advantage of averaging, we can dynamically move the ADC inputs up and down, scrambling the probability distribution from Figure 18 and flattening out the combined probability so that every temperature has a more equal probability of noise. While this does not remove noise from the measurements, it makes all temperatures more equally noisy.

If we add a dithering component to the IDAC current to scramble the measured values at the ADC input, we can make the digital filter somewhat more effective. We intentionally dither the DAC current through the transistor so that the V_{BE} measurement changes slightly with every sample. The mean of the dither over the averaging interval should be zero, so that we do not add offset into the calculations.

Figure 23 shows a Butterworth lowpass digital filter smoothing the temperature measurements from the sensor with dithering. The blue trace shows the individual temperature measurements, and the red trace is the output of the Butterworth filter. Note the wider span of individual temperature measurements as dithering changes the influence of the circuit noise. Filtering with dithering reduces the span of the output temperature readings to about ±1.25°C, in this case.

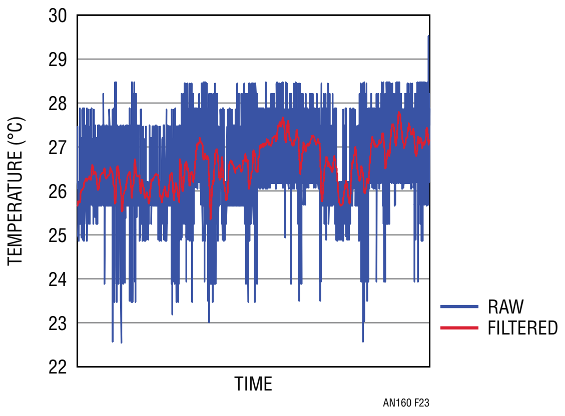

Figure 23. Dithered Temperature Measurements Processed by a Butterworth Filter

The primary differences between a moving average filter and a Butterworth, or other infinite-impulse-response digital filter are that the moving average requires more microprocessor memory to store historical values but less high precision math, and the IIR filter requires less memory but high precision math (either floating-point or large integer). These differences, among others, affect the selection and design of the digital filter.

Averaging and filtering do hold promise, but the results are still sensitive to noise, which depends on the system in which the part is operating. Narrow-band noise can be anticipated and filtered, but broadband noise cannot. It is important to consider the sources of noise and aliasing when designing a filter to smooth the temperature readings.

DETECTING CIRCUIT FAILURES

An open circuit or a short circuit will cause one or more ADC readings to be distorted. The obvious cases are when the DAC becomes disconnected, or when one ADC channel is open. In these cases the ADC reading will be far away from the expected value. To detect circuit failure, the C code can individually compare each measured value to a known-good range (V_{BE} should be around 0.6V; V_R should be a known current times a known resistance: 255µA • 10k = 2.55V). More subtle circuit faults can be caught if we compare the ΔV_{BE} and current ratio to expected values (ΔV_{BE} should be around 60mV, and the current ratio should be well known because it is programmed (e.g. 12.75). If any of these values is outside of its range, the measurement can be assumed bogus. A series of bogus measurements indicates circuit failure. Simple modifications to the C code can catch these errors and indicate failure to the user.

INTERESTING VARIATIONS

In noisy environments it may be advantageous to raise the V_{BE} voltage by supplying more current than the 255µA. The simplest way to do this is to use the voltage DAC output of the LTC2970 instead of the IDAC. The voltage DAC output is a voltage-buffered version of the IDAC pin voltage. The IDAC voltage is set by a fixed resistor at the IDAC output. A 1mA BJT current can be achieved using a 3.6V full-scale voltage DAC output (by tying a 14.2kΩ resistor from the IDAC output pin to GND) and using a 3kΩ current sense resistor in series with the BJT.

The 2.3°C temperature step size is limited by the ΔV_{BE} measurement, which is a small value, and is dictated by the BJT, not by the magnitude of current flowing. A simple way to increase ΔV_{BE} is to stack two BJTs one atop the other. This doubles the V_{BE} voltage and also ΔV_{BE}. Doubling the voltage cuts the temperature step size in half by doubling the signal-to-quantization-noise. Figure 24 shows the modified circuit.

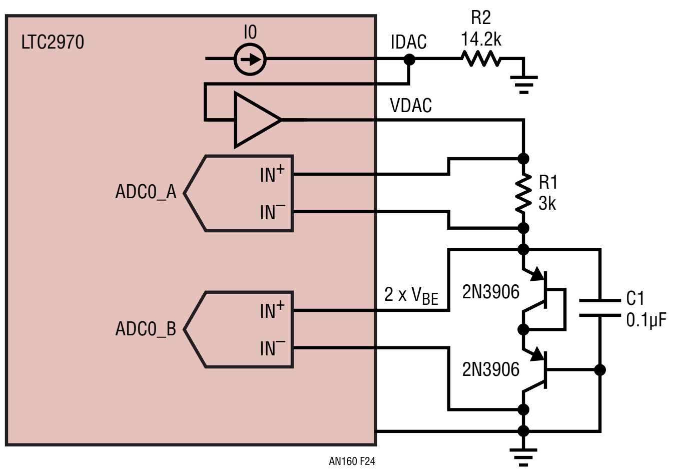

Figure 24. Modified LTC2970 External Temperature Sense Circuit

An important factor in the length of time required by each measurement is the delay necessary to settle out analog transients after changing the IDAC current. There are several opportunities for improvement here. The first is to decrease settling time. Decreasing the capacitor across the BJT (C1 in Figure 1 and Figure 24), decreasing the current sense resistor value, and decreasing the small signal impedance of the BJT by running more current through it, all contribute to lower setting times. We already mentioned using the voltage DAC instead of the IDAC, which would accomplish both raising the current and decreasing the current sense resistor value. Remember that the biggest issue with decreasing the settling time is that raising the circuit bandwidth raises the potential for noise.

The most important opportunities for saving measurement time involve cutting the ADC loop latency. First, toggle continuously between high current and low current without setting the current to zero between measurements. Second, take differential measurements first from high current to low, then from low to high. These save several delay periods in the C code. Third, program the LTC2970 to ignore unused inputs, thus shortening the polling loop that the ADC makes. Use the ADC_MON register to deselect any channels that don't need to be sampled by the ADC. Each channel deselected saves 33.3ms. In addition to shortening the sampling time, these time-saving measures also mitigate a potential source of error: temperature drift. If the temperature of the system changes at a rate comparable to the time between V_{BE} measurements, then significant temperature error will result. Shortening the sampling time improves the temperature drift error.

CONCLUSION

The LTC2970 is versatile beyond its intended data sheet applications. When a system requires only one channel of power supply management, the second channel offers the opportunity to sense external temperature in a properly-connected transistor. A few lines of microcontroller code drive the LTC2970, read ADC measurements, and calculate temperature to within better than ±2.5°C. The application is nearly free, and provides additional options for an LTC2970.

Linear Technology Corporation
1630 McCarthy Blvd., Milpitas, CA 95035-7417
(408) 432-1900 • FAX: (408) 434-0507 • www.linear.com

LT 0716 • PRINTED IN USA