



Column #29, July 1997 by Scott Edwards:

IC Temperature Sensors Are Accurate and Flexible

Old-tech sensors are handy where newer devices don't fit

TWO PREVIOUS applications showed how to measure temperature with a lowly thermistor and a clever thermometer-on-a-chip. In both cases, I touted the applications as alternatives to traditional techniques using an analog-to-digital converter and a discrete temperature sensor.

Sometimes the traditional ways are best, however. For instance, neither the thermistor trick featured in the Stamp/Counterfeit application notes (no. 7), nor the DS1620 thermometer chip (*Stamp Applications* no. 2) is suitable for remote use. In both cases the sensors should be at most a foot from the controller.

The thermistor application requires calibration to compensate for component tolerances, and is accurate over a narrow range of temperatures. The DS1620 needs no calibration, and has a wide temperature range, but gobbles up lots of precious program memory.

This month we're going to look at old-fashioned temperature-measurement devices and techniques.

Column #29: IC Temperature Sensors are Accurate and Flexible

LM34 and LM35

If nothing else, you can use these guys to double-check or calibrate other thermometers. LM34 and LM35 IC temperature sensors made by National Semiconductor are simple three-terminal devices. Connect one pin to +5 to +20 volts, and another to ground. The voltage at the output terminal will be proportional to the temperature to the tune of 0.01 volt (10 millivolts) per degree. For example, if the temperature is 68 degrees, the output will be 0.68 V. In the case of the LM34, the output corresponds to degrees Fahrenheit; the LM35, degrees Celsius.

Because of the direct relationship between temperature and voltage, you must have a negative supply voltage or some additional components in order to measure subzero temperatures. The National Semiconductor spec sheets provide suitable circuits. We're going to stick to the basics; you can add bells and whistles later as necessary.

National makes several variants of the LM34 and 35, allowing you to play off the accuracy you want against the price you're willing to pay. Table 29.1 lists your options.

Table 29.1: Characteristics of the LM34/35 sensors

Part	Range	Accuracy	Output
LM34A	-50 to +300°F	±2.0°F	10mV per °F
LM34	-50 to +300°F	±3.0°F	
LM34CA	-40 to +230°F	±2.0°F	
LM34C	-40 to +230°F	±3.0°F	
LM34D	+32 to +212°F	±4.0°F	
LM35A	-55 to +150°C	±1.0°C	10mV per °C
LM35	-55 to +150°C	±1.5°C	
LM35CA	-40 to +110°C	±1.0°C	
LM35C	-40 to +110°C	±1.5°C	
LM35D	0 to +100°C	±2.0°C	

Interfacing to the Stamp

Measuring the voltage output by a temperature sensor requires an analog-to-digital converter (ADC). We've covered a couple of those in the past, including interfacing the 12-bit LTC1298 and using cheap comparators to create your own 8-bit ADC (*Stamp Apps* nos. 4 and 25, respectively). But for temperature measurement, it may make more sense to use the modest ADC0831, shown in BS1 and BS2 application notes (no. 2 in each case).

The ADC0831 is a particularly suitable partner for temperature sensors because you can set both ends of its input-voltage range to match the characteristics of the sensor.

The ADC's V_{in-} pin sets the input voltage that will produce an output of 0 ADC units. The V_{ref} pin sets the voltage corresponding to 255 ADC units. So if you want 1 unit of the ADC's range to correspond to 1°F , tie V_{in-} to ground and V_{ref} to 2.55V. Since the temperature sensor outputs 0.01V (10mV) per degree, an ADC reading of, say 78 corresponds to a temperature of 78° (F or C, depending on whether the sensor is an LM34 or 35).

If you want finer resolution, just bring the endpoints of measurement (voltages at V_{in-} and V_{ref}) closer together and use the Stamp's math capabilities to scale the reading to the correct temperature.

Figure 29.1. Temp sensor and ADC

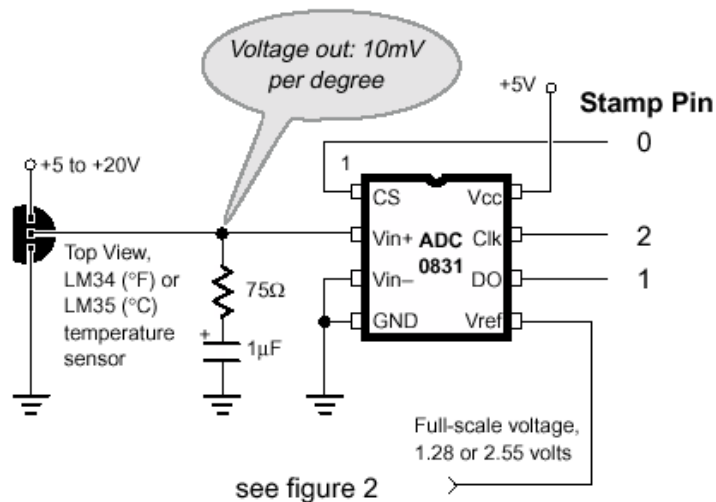


Figure 29.2: Voltage reference options

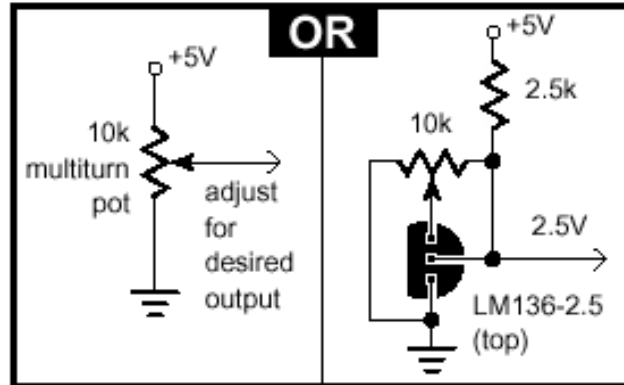


Figure 29.1 shows the hookup for the ADC0831, while Figure 29.2 suggests a couple of options for the reference voltage. The pot option assumes that the +5V supply is reasonably stable, since variations in the supply voltage will affect the reference voltage, and therefore the temperature readings. This is a pretty safe bet with the Stamps and Counterfeits, since they use high-quality regulators.

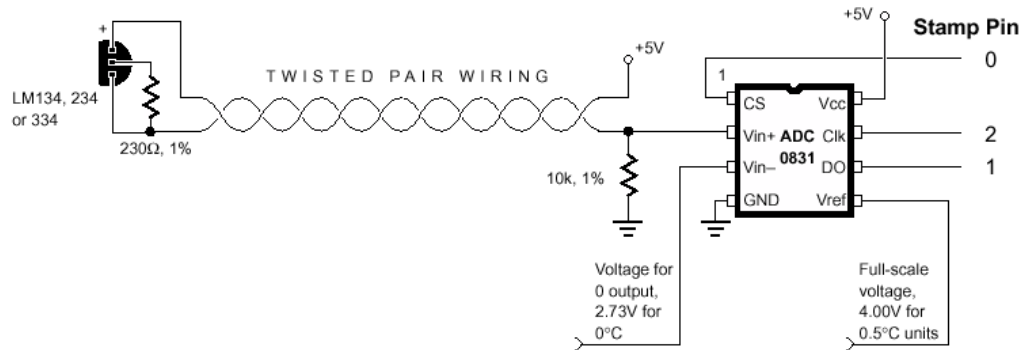
The voltage-reference option is likely to be more accurate and less prone to drift, but is more expensive and less flexible than the pot. The program listing shows the minimal programming required to take temperature readings with this circuit. I used a 2.55V reference, so the result returned by the ADC was the actual temperature in degrees—no adjustment or scaling required.

In my demo setup, I used a 2x16 serial LCD as a display. You can readily modify the code for some other display device if you like. I used one of my company's BS1/LCD kits (blatant plug), installing the ADC0831 right onto the prototyping area.

Remote temperature sensor

Figure 29.3 introduces another temperature sensor—the LM134/234/334 series. Instead of a variable voltage output, this sensor passes a small current that varies with temperature. With the component values shown, that current is 1 microamp (mA) per degree Kelvin. (Kelvin is an absolute scale; its 0 is absolute zero, the lowest possible temperature. To convert °K to °C, just subtract 273.1.)

Figure 29.3: Remote temperature sensor



For measurement purposes, we pass this tiny current through a 10k resistor. By Ohm's Law, the voltage across the resistor is current times resistance = $1 \times 10^{-6} \times 10 \times 10^3 = 10 \times 10^{-3} = 10\text{mV}$ per degree, same as our other sensors. The only difference is the Kelvin scale. If we want the temperature range to start at 0, we have to set the 0 point (Vin-) at 2.73V, since $273^\circ\text{K} = 0^\circ\text{C}$.

To maintain the one-to-one relationship between °C and ADC units, we'd have to place Vref 2.55 volts higher, at $2.73 + 2.55 = 5.28\text{V}$. That's higher than the +5V supply, so it's not allowed. Instead, we can place Vref at 4.00V and divide the ADC result by 2 to get °C.

Now, the big question: Why bother? Why not just put one of the voltage-mode temperature sensors at the end of a long piece of wire? The reason is noise. A current-mode sensor is far less susceptible to noise pickup over long wire runs than a voltage-mode sensor. Those of you who have worked in industrial settings have probably heard of current-loop sensors and control systems—these are used in electrically noisy factory environments where other signaling methods wouldn't work reliably.

So how long a wire run will our current-mode sensor support? National's excellent data book sidesteps this question, and so will I. I have seen this kind of arrangement in use with 100-foot wire runs with apparently satisfactory results.

Column #29: IC Temperature Sensors are Accurate and Flexible

You've seen the movie, now read the book.

The book I'm referring to is the National Semiconductor *Data Acquisition Linear Devices Databook*. If you're connected with a National distributor, you should be able to get this book free for the asking. If not, you can purchase a copy from Jameco (pn 41208) or download individual data sheets from www.national.com.

For those of us who make only occasional forays into analog electronics, data books like National's are a treasure. Browsing the application notes is an education in itself.

```
' Program Listing 29.1: LCDTEMP.BAS (Measure and display the temperature)
' This program uses an ADC0831 to measure the output voltage from
' an LM34 or LM35 temperature sensor. With Vref adjusted to 2.55V,
' The value returned by the ADC is the temperature in degrees--
' no unit conversion required. The program continuously displays
' the temperature on a serial LCD module connected to pin7. The
' prototype was constructed with a BS1/LCD kit from Scott Edwards
' Electronics. The kit integrates a BS1 computer, serial LCD driver,
' and 2x16 supertwist LCD module into a single, compact package.
' The product manual, catalog, and other items of interest are
' available from ftp://ftp.nutsvolts.com/pub/nutsvolts/scott.
' View the file index.txt for a list of available files.

SYMBOL CS = 0           ' ADC chip-select (activates ADC0831).
SYMBOL AD = pin1       ' Data output of ADC.
SYMBOL CLK = 2         ' Clock input to ADC.
SYMBOL S_out = 7       ' Serial output to LCD.
SYMBOL data = b0       ' 8-bit ADC result.
SYMBOL clocks = b2     ' Counter for ADC clock cycles.
SYMBOL I = $FE        ' LCD instruction-prefix.
SYMBOL LCDcls = 1     ' LCD clear-screen instruction.
SYMBOL dg = 223       ' LCD code for degree symbol.
SYMBOL pos = 198      ' LCD position for temp reading.

' Setup: The program starts by setting the initial states of the
' pins and making all but the ADC data line outputs. It then
' clears the LCD and prints a label on the top line. Note that
' the label assumes that an LM34 (Fahrenheit) sensor is used;
' if you're using an LM35, change the F to C.

setup:
  pins = $7b           ' Pins high, except LCD and CLK.
  dirs = $FD           ' S_out, CLK, CS outputs; AD input.
  serout S_out,N2400,(I,LCDcls) ' Clear the LCD screen.
  serout S_out,N2400,(" Temperature ",dg,"F") ' Print fixed label.

' Main: This is where the program spends most of its time. It takes
' a reading via the subroutine 'convert', prints the result to the
' middle of the LCD's second line, pauses a half second, then
' repeats.
main:
```

Column #29: IC Temperature Sensors are Accurate and Flexible

```
gosub convert          ' Get the data.
pause 1000             ' Wait a second for LCD startup.
serout S_out,N2400,(I,pos,#b0," ") ' Write it to the LCD.
pause 500              ' Wait 1/2 second.
goto main              ' Loop continuously.

' Convert: This subroutine gets 8-bit analog readings from the ADC0831.
' With Vin- connected to ground and Vref to 2.55V, the 10mV/degree
' output of the LM34/35 directly translates to degrees.
convert:
  low CS                ' Select ADC.
  pulsout CLK, 1        ' 10 us clock pulse.
  let data = 0          ' Clear data.
  for clocks = 1 to 8   ' Eight data bits.
    let data = data * 2 ' Perform shift left.
    pulsout CLK, 1      ' 10 us clock pulse.
    let data = data + AD ' Put bit in LSB of data.
  next
  high CS               ' Deselect ADC.
Return
```

