Stamp Applications no. 25 (March '97):

Analog-to-Digital Conversion The Old-Fashioned Way

Using comparators to measure voltage and a total-shutdown power supply

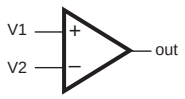
by Scott Edwards

IN THE OLDEN DAYS, say 20 years ago, analog-to-digital converters (ADCs) were rare and expensive. Designers who needed to measure voltages generally built their own ADCs using a building block called a *comparator*.

Even now that ADCs are no longer exotic, comparator-based tricks are still popular. Comparator ADCs are much cheaper than comparable packaged ADCs, and they're a good education for those who wonder what's under the hood of ADC chips.

This month, we'll look at comparators and the ADC circuits you can build with them. As an added bonus, we'll design a power supply that turns a system on at the touch of a button and off at the flip of a bit.

The Hotter/Colder Game, on a Chip. As the name suggests, a comparator is a circuit that compares two inputs. It's a close relative of the operational amplifier (op amp) and uses the same schematic symbol. See figure 1.



IF V1 > V2 THEN out = 1
IF V1 < V2 THEN out = 0

Figure 1. Comparator symbol and operation.

A comparator takes two input voltages and compares them. If the voltage at the + input (called the *noninverting* input) is higher, then the output is 1; if the – (*inverting*) input is higher, the output is 0.

The rules of comparator operation don't say what happens when the voltages are equal. In practical circuits, there's almost no such thing as "equal" where comparators are concerned. The comparator's gain is so high and tiny noise voltages so unavoidable that "equal" ends up looking like nervous twitching between the greater-than and less-than conditions.

Imagine how you might use a comparator to measure an unknown voltage. Let's assume you had a variable voltage source with a calibrated dial. You'd connect that to the inverting input of the comparator, and the unknown voltage to the noninverting input. You'd also connect an LED to the comparator output so you could see the result of the comparison (1 or 0; on or off).

Starting at one end of the dial, you'd adjust the voltage source until the comparator output changed state. For example, suppose you start at 0 volts. The comparator outputs 1 because the unknown voltage is higher. You dial upward gradually, stopping as soon as the comparator outputs 0. Checking the dial, you find that this happens at about 1.5V. So you know that the unknown voltage is very close to 1.5V.

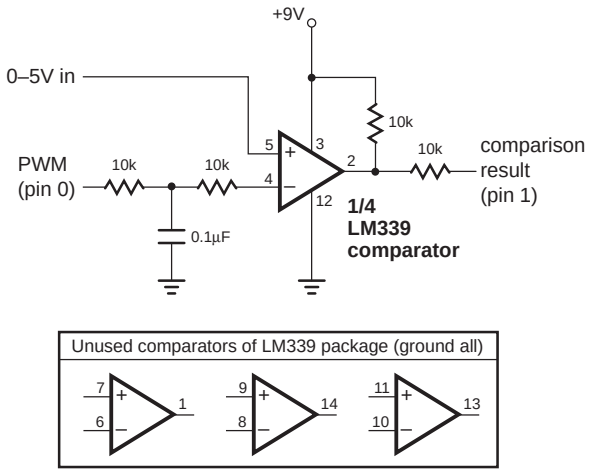


Figure 2. Hookup for Stamp-comparator ADC.

Figure 2 and listing 1 shown how a BS1 or Counterfeit controller can be used to make that kind of voltage measurement using one section of an LM339 comparator. The Stamp's PWM output serves as the calibrated variable voltage, which the program adjusts up from 0V while watching the comparator output. The result is expressed as a number from 0 to 255, where 0 is 0V and 255 is 5V. Each unit is approximately 19.6 millivolts (5/255).

If you run the program, or just think about the example, you'll recognize a weakness in this strategy. Since the variable voltage starts upward from 0, the further the unknown voltage is from 0, the more trials (and therefore time) it takes to complete a measurement. In the listing-1 example, measurements of 10 units or less take a fraction of a second, while measurements of 200+ units take a couple of seconds.

There's a commonsense alternative that's easy to understand: Suppose we were playing a guessing game in which I pick a number between 0 and 255, and you have to guess that number. As a hint, I tell you whether each guess is higher or lower than my number.

If you wanted to win the game quickly, you would *not* start at 0 and guess *1, 2, 3, 4...* until you reached the correct number. More than likely, you'd start with a number in the middle of the range, say 128, and use the higher/lower clue to guide your next guess. For instance, if you said "128" and I replied "higher," you could eliminate the whole range of 0 through 128 from further guessing!

Now you could split the remaining range of 129 to 255 in half, guessing 192. I say "lower" and your choices narrow to 129 to 191. By continuing to split and narrow the range with each guess, you'd be sure to have the correct number in just eight guesses.

That's pretty much the way listings 2 and 3 (BS1 and BS2) work. The systematic divide-and-conquer approach is much faster than listing 1. On a BS1 it runs at 13 conversions per second; a BS2 gets 37.

Speaking of speed, note that the comparator ADC has a weakness in common with most low-end packaged ADCs—it's not good with rapidly changing signals. If the signal changes while the ADC is trying to measure it, the result is not valid. It's like taking a photo of a fast-moving car using a slow shutter—the image is blurred. To make an ADC work correctly with fast-changing signals, you need the electronic equivalent of a fast shutter, a circuit called a *sample-and-hold*. As the name implies, this circuit grabs a sample of the input signal, then the ADC measures that. No blur.

There are plenty of applications, like temperature sensing, that move slowly enough not to require a sample-and-hold circuit. A rule of thumb is that if you can accurately measure the voltage with a digital meter (good for only a few samples a second), then the comparator ADC will be just fine. If the signal would be more appropriately viewed on an oscilloscope, you definitely need a sample-and-hold. Building a discrete sample-and-hold circuit would be an excellent educational experience, but wouldn't make much practical sense, since it might add a dozen components to the circuit. If you need a sample-and-hold, go ahead and buy a packaged ADC that includes one. See issue 4 of this column (available from the *N&V* web site) for a description of the LTC1298, a 12-bit ADC with sample-and-hold capability, or see Sources for the LTC1298 AppKit.

Going Further. Now that you have seen how a comparator works, you may think of other applications. For example, if you want an indication when a voltage is above or below a

reference level (e.g., low-battery warning), the basic comparator circuit is a ready-made answer. Scouring textbooks and application notes will suggest other uses: zero-crossing detectors for AC, Schmitt triggers to clean up slow/noisy signals, level detectors, square-wave generators, etc.

One obvious modification of the example would be to use the remaining sections of the LM339 to build a 4-channel ADC. You'd connect all of the inverting inputs (–) together, and the outputs to separate Stamp pins. To take a measurement from a particular channel would require looking at only the appropriate output, ignoring the others.

Total Shutdown. The Stamps have Nap and Sleep modes that reduce their current draw during periods of inactivity. That's fine for the Stamp, but what about external circuitry—how about turning it off too?

That's the idea behind figure 3 and listing 4, which arose from a question posed by a reader. He wanted his project to turn on at the touch of a button, and completely off after a period of inactivity. Many commercial products work this way, saving a lot of batteries from an early grave.

My answer is to use a National Semiconductor LP2951. This is an efficient, low-dropout regulator with a shutdown pin (pin 3 in the figure, labeled *Off*). When this pin is high, the regulator shuts down; when it's low, the regulator turns on. When the user presses the ON button shown in the schematic, the LP2951 supplies regulated 5 volts to the system.

Since the Stamp is powered by this 5V supply, it wakes up about 20 milliseconds after the button is pressed. It immediately applies a low to the shutdown pin to hold the power supply on. When the user releases the button, the circuit remains powered.

When the Stamp's work is done, it applies a high to the shutdown pin, immediately cutting 5V power to the circuit. In shutdown, the regulator draws just a few 10s of microamps, mostly through the 47k resistor on the shutdown pin itself. It still puts about 0.7V onto the 5V supply rail, resulting in a tiny leakage current through the rest of the circuit. In most cases, this won't cause any problems; I just mention it for those who might poke around with a meter.

The LEDs at the bottom of the schematic are to help you see the operation of the demo; they can be omitted in your final application.

After you download the demo program, remove the Stamp programming cable. Otherwise, leakage currents from the cable will prevent complete shutdown.

One capability of the LP2951 hinted at in the schematic but not shown in the demo is its error output (pin 5, marked *Err*). If the LP2951 experiences a problem, such as overheating, excess current draw or inadequate input voltage to maintain regulation, it will output a low on the error pin. You might experiment with monitoring this input with the Stamp as an early warning of power loss. But in most cases the Stamp is likely to lose consciousness before it can do anything about the warning. If you don't use the error-detect feature, you can omit the pair of 220k resistors.

Sources. For more information on the BASIC Stamp, contact Parallax Inc., 3805 Atherton Road no. 102, Rocklin, CA 95765; phone 916-624-8333; http://www.parallaxinc.com.

The ICs mentioned in this article are available from Jameco Electronic Components, 1355 Shoreway Road, Belmont, CA 94002-4100; phone 415-592-8097 or 800-831-4242; fax 415-592-2503 or 800-237-6948. Part numbers are 107203 (LP2951) and 23851 (LM339).

Scott Edwards Electronics carries the LTC1298 AppKit mentioned in this article. This package shows users of Stamps (1 and 2) and PICs (using Parallax assembly language) how to interface the LTC1298 12-bit ADC. It includes printed documentation, source code on disk, and an LTC1298 chip for $25.

For a catalog of serial LCDs and Stamp-related products, contact Scott Edwards Electronics, PO Box 160, Sierra Vista, AZ 85636-0160; phone 520-459-4802; fax 520-459-0623; Internet at ftp.nutsvolts.com in /pub/nutsvolts/scott; e-mail 72037.2612@compuserve.com.

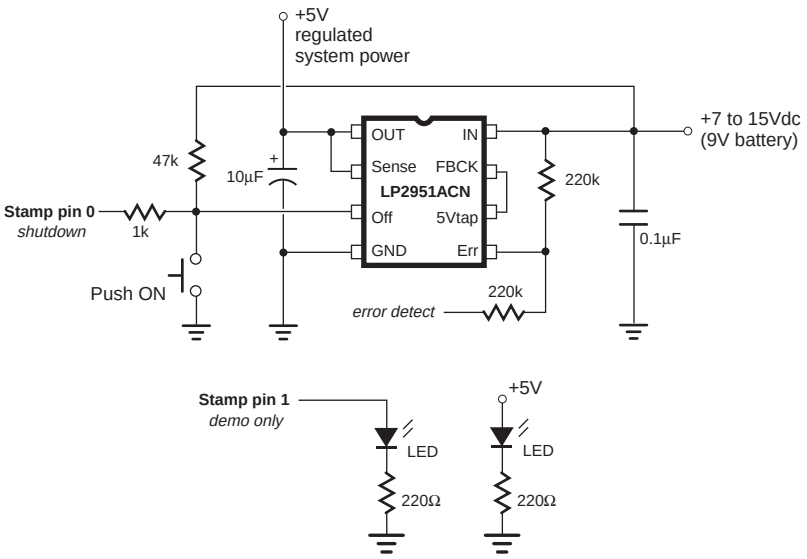


Figure 3. Total-shutdown power supply.

Listing 1. Simple (but dumb) Comparator ADC for BS1

```
' Program: COMP_AD1.BAS
' (Single-slope ADC with the BS1 and a comparator)
' This program implements a single-slope ADC with a comparator.
' The unknown voltage goes to the comparator's + input and the
' Stamp's PWM output to the - input. The Stamp incrementally
' increases the PWM output to the comparator reference until
' the reference exceeds the input voltage. This is a simple
' (albeit not very intelligent) way to make an ADC. One major
' drawback of this approach is that the higher the unknown
' voltage, the longer the conversion takes. See the program
' COMP_AD2.BAS for a vastly improved version.

SYMBOL refOut  =       0        ' Comparator reference.
SYMBOL compIn  =       pin1     ' Comparator output.
SYMBOL ADCres  =       b2       ' Analog-to-digital result.

' Demonstration loop: take a conversion, display it, and loop.
again:
  gosub ADconvert      ' Perform conversion.
  debug ADCres         ' Display it.
goto again             ' Do it again.
```

```
' ADC conversion routine.
ADconvert:
  ADCres = 0                    ' Start at 0 volts.
convLoop:
  PWM refOut,ADCres,1           ' Output 1 PWM cycle.
  if compIn = 0 then done       ' If reference > unknown, done.
  ADCres = ADCres + 1           ' Otherwise, increase by 1.
  if ADCres <> 0 then convLoop  ' If rollover from 255 to 0, quit.
done:
return                          ' Return to program.
```

Listing 2. Improved Comparator ADC for BS1

```
' Program: COMP_AD2.BAS
' (Binary-search ADC with the BS1 and a comparator)
' This program implements an ADC using a comparator.
' The unknown voltage goes to the comparator's + input and the
' Stamp's PWM output to the - input. The Stamp systematically
' searches for the unknown voltage by splitting the possible
' range of voltages in half, seeing whether the unknown is
' higher or lower, then splitting that range in half...
' This approach assures that the conversion is always
' finished in the shortest time possible. The conversion
' subroutine presented here runs at about 13 conversions/second.

SYMBOL refOut  =       0        ' Comparator reference.
SYMBOL comp_p  =       pin1     ' Comparator output (pin).
SYMBOL ADCres  =       b2       ' Analog-to-digital result.
SYMBOL pwrTwo  =       b3       ' Power-of-2 to add to ADCres.

' Demonstration loop: take a conversion, display it, and loop.
again:
  gosub ADconvert      ' Perform conversion.
  debug ADCres         ' Display it.
goto again             ' Do it again.

' ADC conversion routine.
ADconvert:
  ADCres = 0: pwrTwo = 128
convLoop:
  ADCres = ADCres + pwrTwo      ' Add current power-of-2 to ADCres
  PWM refOut,ADCRes,1           ' ..and output that voltage via PWM.
  if comp_p = 1 then skip1      ' If unknown voltage is lower, then
  ADCRes = ADCres-pwrTwo        ' subtract power-of-2 from ADCres.
skip1:
  pwrTwo = pwrTwo/2             ' Try next lower power-of-2..
  if pwrTwo <> 0 then convLoop  ' ..until power-of-2 = 0
return
```

Listing 3. Improved Comparator ADC for BS2

```
' Program: COMP_AD2.BS2
' (Binary-search ADC with the BS2 and a comparator)
' This program implements an ADC using a comparator.
' The unknown voltage goes to the comparator's + input and the
' Stamp's PWM output to the - input. The Stamp systematically
' searches for the unknown voltage by splitting the possible
' range of voltages in half, seeing whether the unknown is
' higher or lower, then splitting that range in half...
' This approach assures that the conversion is always
' finished in the shortest time possible. The conversion
' subroutine presented here runs at about 37 conversions/second.

refOut  con    0       ' Comparator reference.
comp_p  var    in1     ' Comparator output (pin).
ADCres  var    byte    ' Analog-to-digital result.
pwrTwo  var    byte    ' Power-of-2 to add to ADCres.

' Demonstration loop: take a conversion, display it, and loop.
again:
  gosub ADconvert      ' Perform conversion.
  debug ? ADCres       ' Display it.
goto again             ' Do it again.

' ADC conversion routine.
ADconvert:
  ADCres = 0: pwrTwo = 128
convLoop:
  ADCres = ADCres + pwrTwo      ' Add current power-of-2 to ADCres
  PWM refOut,ADCRes,1           ' ..and output that voltage via PWM.
  if comp_p = 1 then skip1      ' If unknown voltage is lower, then
  ADCRes = ADCres-pwrTwo        ' subtract power-of-2 from ADCres.
skip1:
  pwrTwo = pwrTwo >> 1          ' Try next lower power-of-2..
  if pwrTwo <> 0 then convLoop  ' ..until power-of-2 = 0
return
```

Listing 4. Total-Shutdown Voltage Regulator for BS1

```
' Program: TURNOFF.BAS
' (BS1 controls LP2951 regulator for system shutdown)
' This program demonstrates how the Stamp can use a shutdown-
' capable power supply to provide push-on/auto-off power
' control. In this type of operation, the user presses a button,
' putting a low on the LP2951 shutdown pin and supplying 5 volts
' to the system. This starts up the Stamp, which immediately
' puts a low on the shutdown pin itself. This latches the power
' supply on after the user releases the button. The process takes
' only 20 ms, so even a brief button press will do. When the
' Stamp's work is done, it shuts itself (and everything else
' on the same power supply) off by putting a high on the shutdown
' pin. The supply remains off until the button is pressed again.
' If you have the Stamp powered by the LP2951, remember that you
' will have to hold the ON button down throughout downloading.
' Also, you may find that this program will not start up normally
' unless the Stamp programming cable is removed.

SYMBOL powerControl    =        0       ' LP2951 shutdown pin.
SYMBOL LEDoutput       =        7       ' LED output for demo.

PowerOn:
  low powerControl     ' Latch power supply ON.

'=====================================================
' Substitute your own code for the LED flasher below.
for b2 = 1 to 20       ' Flash LED 10 on/off cycles.
  toggle LEDoutput     ' Toggle the LED.
  w2 = 700/b2          ' W2 sets delay that decreases..
  pause w2             ' ..with each cycle.
next
'=====================================================

PowerOff:
  high powerControl    ' Turn power (and Stamp) OFF.
```