



Column #57, January 2000 by Lon Glazner:

Getting the Most Out of Your 12-Bit ADC

In this installment of Stamp Applications, we'll take a look at a couple of simple circuits that allow you to make the most of your ADC.

Analog-to-digital converters (ADC) are ubiquitous components in the world of embedded control design these days. Many microcontrollers have on-board ADCs, and the resolution of these ADCs is increasing on a regular basis. But an increase in resolution does not necessarily translate to an increase in measurement accuracy.

A Bit About Analog-to-Digital Converters

There's a little bit of ground to cover in discussing ADCs in general. I'm not going to go into a dissertation on analog conversion technologies but, for those of you that have never used one before, some of the basics may be helpful.

ADCs come in a wide variety of flavors. When interfacing to a BASIC Stamp, I lean towards those that have serial interfaces. This reduces the number of pins required to control the ADC, and I/O pins are like gold in a BASIC Stamp design. The Serial Peripheral Interface (SPI) seems to provide the simplest interface software, and requires only three I/O lines from the BASIC Stamp.

Column #57: Getting the Most out of your 12-bit ADC

ADCs also come with a variety of “channels.” The term channel refers to the number of ADC measurement points on a specific IC. For instance, a four-channel ADC would have four pins where analog voltages may be measured. The digital interface software determines which channel is being measured for any given data sample.

ADCs also come with a variety of resolution ratings. An eight-bit ADC can return 256 different values for any given sample (including a result of 0); a 10-bit ADC has 1,024 different values; and a 12-bit ADC has 4,096 values. Virtually all ADCs have reference voltage inputs, usually labeled VREF or REF. The voltage at this point determines the full-scale value associated with an ADC measurement. For instance, an eight-bit ADC with a reference voltage of 5.0VDC would return a 255 (binary %11111111, or hex \$FF) if 5.0VDC were present at the channel being sampled.

So, what is the voltage resolution in our eight-bit ADC with a 5.0VDC reference? This can be determined by dividing 5 by the number of possible values the ADC can provide.

$$\text{Or... } 5.0\text{V} / 256 \text{ possible values} = .01953\text{V} / \text{value}$$

Our eight-bit A/D has a resolution of 19.53mV per bit. The accuracy is usually ± 1 least significant bit (LSB), which means that your measurement can be expected to be off by $\pm 19.53\text{mV}$ for any given measurement (ADC measurements are often called “samples”). Engineers will often be required to design analog measurement devices which require greater resolution and better accuracy than described above. For the novice, there is a desire to select a 10- or 12-bit ADC and call it good. This can be a dreadful mistake. After all, even if your ADC has a resolution of 1mV per bit, if there is 8mV of noise on your ADC input, you’ve effectively reduced your ADC to nine bits of resolution. In other words, you’ve paid for the 12-bit device, but your performance is only a little better than an eight-bit ADC.

Don’t throw out the ADC though, there are a few tried and true methods to increase performance. Selecting your reference voltage correctly, and providing appropriate gain and filtering circuits, as well as digital (software) filtering can produce respectable results.

Defining the Design

A good way to get a grasp on some of the particulars of making accurate ADC measurements is by walking through a design. Consider this article ADC-101. There will be areas specific to certain input voltages or signal types that we'll not concern ourselves with, there will also be certain types of noise that we'll ignore for the most part. A great example of this is PCB layout. Poor physical PCB layout, as well as improper grounding, de-coupling, and bypassing techniques have much greater effects on your ADC performance than the improvements granted by the filter circuits we'll cover here. I can't cover all of these issues in one article, but I'll be sure to point out some references for those aspects of ADC measurements that are being omitted.

One last warning before we're off and running. Highly accurate analog measurement systems often have to be calibrated to make full use of the circuitry associated with them. We'll touch on calibration in software, but we won't go into a complete analysis of how to design out component offsets and other error factors through software or hardware calibration.

Design Statement

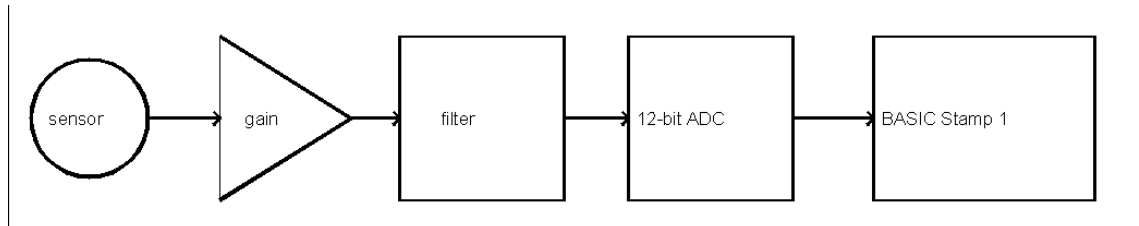
We've been presented with a sensor that outputs a minimum voltage of 1VDC. The full-scale value of the sensor is limited to only 1.1VDC (full-scale is the largest value that will come out of the sensor). It's our job to provide a circuit that can accurately measure the voltage provided by this sensor. To complicate matters, our boss — the grand Pooh-Bah — wants a resolution of 50uV, and an accuracy of $\pm 50\text{uV}$ in our measurements.

There may also be some noise at about 100kHz coming from a component near where our circuit is being placed. We weren't given any specifics on this device, just a heads-up on the potential for some noise at this frequency.

It is pretty clear that just hooking up a 12-bit ADC to the sensor would not provide the kind of resolution that our slave-driver boss requires. But if we provide an amplifier block (or gain block) and some analog filtering, we might just be able to knock this design out with a 12-bit ADC.

Figure 57.1 shows a block diagram of the system that we're designing. The sensor provides a signal that can range from 0-100mV, with a 1V offset. The gain block amplifies the sensor signal so that the full range of the 12-bit ADC can be used. The filter block is a simple two-pole, low pass filter with a cut-off frequency in the area of 10kHz.

Figure 57.1: 12-Bit ADC system block diagram



The Hardware

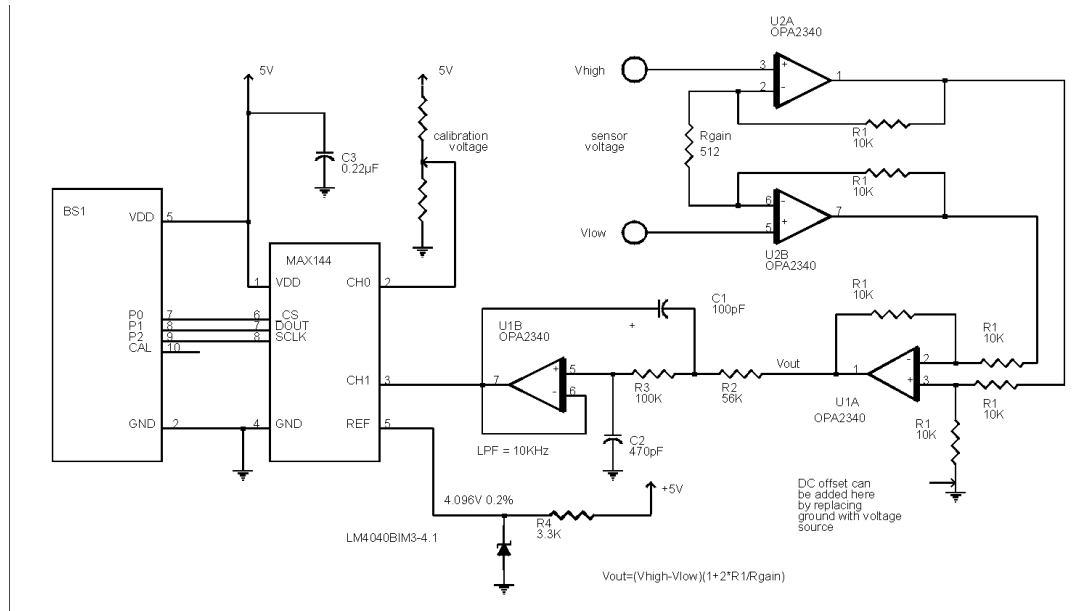
For the hardware design, we'll tackle first-things-first. A two-channel, 12-bit, SPI-ADC would be ideal for this design. The MAX144 (from MAXIM Integrated Products, www.maxim-ic.com) seems to fit the bill, as far as ADCs go. We'll be using a precision voltage reference — the LM4040BIM3-4.1 — from National Semiconductor to provide a voltage reference of 4.096V for the MAX144. There is a significant benefit to making this choice. The 4.096V reference voltage means that every bit measured relates to exactly 1mV (from 4.096V / 4096 possible values).

The gain block and filtering can be done with op-amp circuits. It's a good idea to select op-amps that were designed specifically for instrumentation and ADC interfacing. The OPA2340 from Burr-Brown provides a solid dual op-amp for our purpose. The OPA2340 provides rail-to-rail outputs (to within 1mV) and operates on a single +5V supply. These parts also have a low offset voltage (less than 1mV) which can help reduce measurement errors.

The complete circuit schematic is displayed in Figure 57.2. The gain block is made up of the two op-amps in the U2 package. The gain for this circuit is set to 40. A gain of 40 applied to a full-scale sensor output would generate a $100\text{mV} * 40 = 4.0\text{V}$ output. The additional 0.096V provides us with a little headroom. We can expect the total voltage output to the low-pass filter to remain less than 4.00V, which is less than the full-scale measurement range of the MAX144.

We know that each bit returned by our ADC is related to a 1mV measurement. Since this 1mV resolution is gained up by 40, the actual value per bit is $1\text{mV} / 40 = 25\text{uV}$, which meets the criteria for this design. From the design needs, we were allowed $\pm 50\text{uV}$ accuracy, and that translates into a little room for error.

Figure 57.2: 12-Bit ADC instrumentation circuit

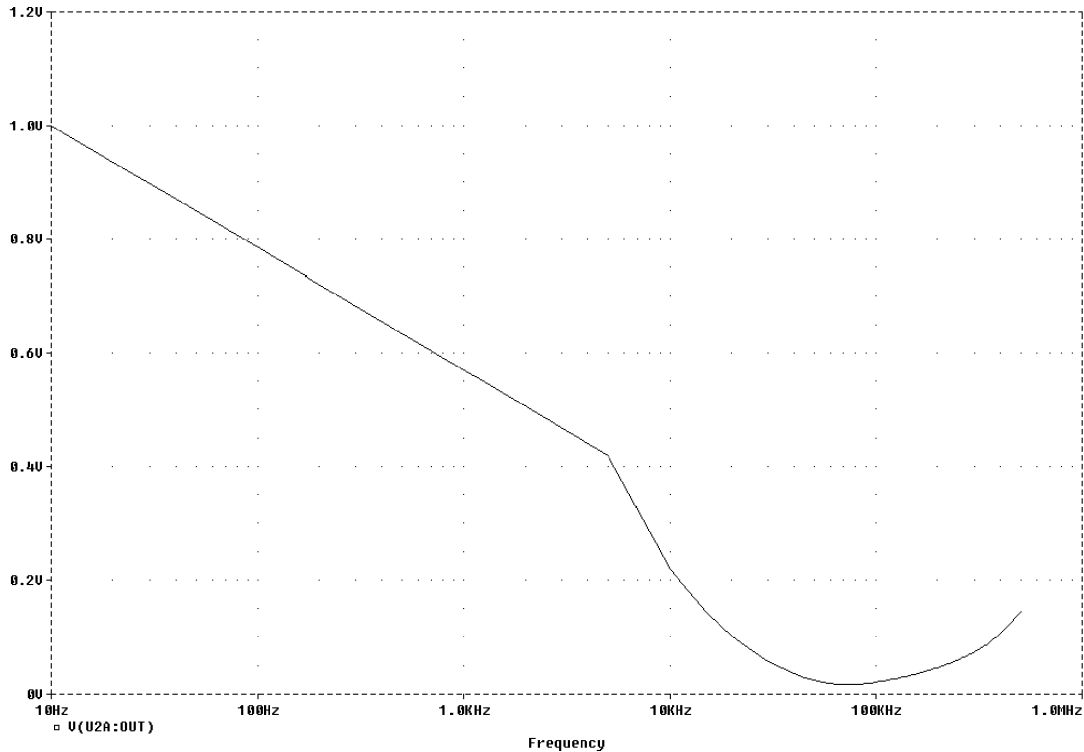


There are a couple of loose strings that should be tied up at this point. The first relates to the OPA2340 parts. I did not show the power or ground pins in the schematic, nor did I show the 0.22µF bypass capacitors which should be located as close as possible to the power pin and connected between the power and ground connections for each op-amp. These capacitors are important and should not be omitted.

Along the same lines, the MAX144 data sheet specifies some layout requirements, which should be read, and adhered to, if you intend to place the MAX144 into any of your designs. Finally, the channel 0 connection was used to measure a calibration voltage provided by a 1K potentiometer configured as a voltage divider. While this is not strictly necessary, it made calibration much easier during initial testing.

To finish up the hardware design, a simulation of the filter circuit was completed to ensure that it would adequately cut off any 100kHz noise that might be coupled onto our sensor signal lines from the noise source our boss had mentioned earlier. The filter output versus frequency is displayed in Figure 57.3.

Figure 57.3: Low-pass filter characteristics



The Software

The software is pretty straightforward for this system. Although, there are a few “gotchas” that I need to mention. One issue of concern is the manner in which the two channels of the ADC are selected. The MAX144 alternates channel 0 and channel 1 with each sample operation that you perform. It starts with channel 0 selected at power-up. This can create some problems.

If you’re like me, you might make a modification to your BASIC Stamp code and then download the software to your Stamp without cycling the power off and then on. This could place your software out of sync with the MAX144. You would be reading channel 1 when you expect to read channel 0.

A simple way around this is to sample the MAX144, and then test the 12th bit in the returned word. If the bit is clear (“0”), then you have just read channel 0. If it is not clear, then you should set the chip select and clock lines high and attempt to read the MAX144 again. The second attempt will provide the sample from channel 0. You only need to do this as part of a software routine for reading channel 0. Once you have successfully read channel, 0 your BASIC Stamp will be in sync with the MAX144.

You’ll notice in the software that the highest nibble of each returned sample is cleared by ANDING the sample result with \$0FFF. This clears the channel information returned with each sample. The resulting data can be summed with other samples to provide a simple averaging routine. In the software listing for this design, there is a 16 sample averaging function being performed. Summing any additional samples would run the risk of overflowing the storage register AD_RESULT. Also keep in mind that any routines that manipulate the data will have some effect on both your accuracy and your measurement resolution. Averaging, in particular, has a tendency to reduce the value of the data that you are manipulating.

In this design, channel 1 of the ADC is actually measuring the sensor-input voltage. Channel 0 is used to read the voltage from a calibration potentiometer. With each pass through the program, the results from channel 1 are added to 2500 (2500mV) and then the result from channel 0 is subtracted from channel 1 + 2500mV. In a system with no errors, you would set your potentiometer to 2500mV. If you found that your system was outputting voltage readings 10mV too high, you would set the potentiometer to 2510mV.

A simple way to get a ball-park calibration value is to tie both sensor inputs to the same point and adjust your calibration potentiometer until the AD_RESULT register reads a “0.” This is a very simple calibration technique that could be adversely affected by component aging, vibration, and temperature, among other things. But it does present one of the more powerful aspects of designing with embedded controllers.

In Closing

In the lab, I was able to calibrate this system and achieve an accuracy of $\pm 50\mu\text{V}$, as well as reach the target resolution of $50\mu\text{V}$. In fact, the sample resolution was good down to $25\mu\text{V}$ steps. I did have some trouble reading values lower than about $300\mu\text{V}$ but, when I got above that threshold, the system was doing its job well.

Column #57: Getting the Most out of your 12-bit ADC

This circuit was built on a solderless breadboard, which is about the most unforgiving place that you can test a design that is highly susceptible to noise. I had intended to work out the basics and move the circuit to a copper board for “dead-bugging” (a prototyping technique where the parts are soldered to a copper ground plane), but the circuit performance was adequate for the testing that I required.

To summarize the techniques used to build accurate ADC-based systems, I think the following five points cover most concerns ...

1. Define your accuracy and resolution requirements before selecting a specific ADC.
2. Select accurate voltage references that are compatible with your ADC. Add gain blocks to make use of the full span of your ADC's measurement range.
3. Provide analog filtering for the signals that you intend to measure.
4. Read the manufacturer's data sheets for layout recommendations and other important insights into any ICs that you expect to use.

For anyone interested in working with this circuit, all of the components mentioned in this article can be purchased from the on-line catalog company Digi-Key (www.digikey.com). See you next month.

Column #57: Getting the Most out of your 12-bit ADC

```
'Program Listing 57.1
'NV100.BAS Interfacing to the MAX144 12-Bit ADC
SYMBOL CS_AD      = 0
SYMBOL          SDATA      = 1
SYMBOL          CLK        = 2
SYMBOL          DATpin     = pin1
SYMBOL          Ad_flag    = bit12

SYMBOL          CLOCKS    = B2
SYMBOL          SAMPLES   = B3
SYMBOL          AD0_CAL   = W0
SYMBOL          AD1_IN    = W2
SYMBOL          WORK      = W3
SYMBOL          AD_RESULT = W4

BEGIN:
    LET DIRS = %11111101
    PAUSE 1000
MAIN:
    GOSUB ANALOG
    GOTO MAIN

| *****
ANALOG:
    HIGH CS_AD
    HIGH CLK
    AD_RESULT = 0
    FOR SAMPLES = 1 TO 16
        LOW CLK
        LOW CS_AD
        LET AD0_CAL = 0
        FOR CLOCKS = 1 TO 16
            LET AD0_CAL = AD0_CAL * 2
            LET AD0_CAL = AD0_CAL + DATpin
            PULSOUT CLK,10
        NEXT
        IF BIT12 = 1 THEN ANALOG
        AD0_CAL = AD0_CAL&$0FFF
        HIGH CS_AD
        HIGH CLK
        LOW CLK
        LOW CS_AD
        LET AD1_IN = 0
        FOR CLOCKS = 1 TO 16
            LET AD1_IN = AD1_IN * 2
            LET AD1_IN = AD1_IN + DATpin
            PULSOUT CLK,10
        NEXT
        AD1_IN = AD1_IN&$0FFF
        HIGH CS_AD
```

Column #57: Getting the Most out of your 12-bit ADC

```
        HIGH    CLK
        WORK = AD1_IN + 2500 - AD0_CAL
        AD_RESULT = AD_RESULT + WORK
NEXT
AD_RESULT = AD_RESULT/16
DEBUG    AD_RESULT, CR, CR
RETURN
!*****
END:
```