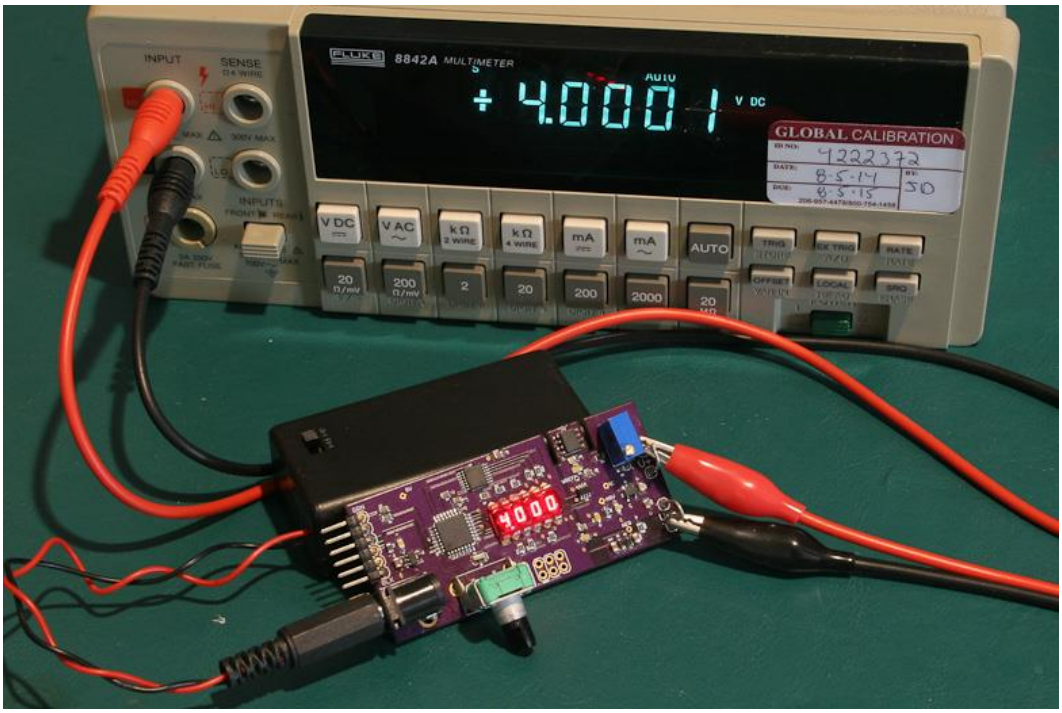


Programmable Voltage Reference with 1 mV resolution and 100 uV accuracy

Over the last few months I've done a few projects that made use of the analog inputs on the Arduino/ATmega processor. I found that trying to debug the software at the same time as debugging and calibrating the hardware was cumbersome (to put it mildly). The more I dealt with this, the more I thought that a programmable voltage reference would simplify things. With a programmable reference I could simulate an analog sensor input and debug the software without worrying about whether the sensor hardware is working or not. The voltage reference makes it easier to manipulate the input, test boundary conditions or check the accuracy of the ADC.

Consider this example of a simple temperature probe. The probe uses an ATtiny85 8-pin processor connected to an LMT84 analog temperature sensor and two LEDs, one red and one green. As long as the temperature stays in range the green LED flashes once every 15 seconds. If the temperature goes out of range, either high or low, for more than a few samples the red LED flashes instead until the unit is reset. The whole thing is powered by a coin cell and physically is about the size of the cell holder and twice as thick. The battery needs to last for a few months.

The design is simple, but has a couple of tricky aspects. How do you debug a processor that has no serial port and is asleep most of the time? There weren't even any spare pins to send out debug signals. The processor controls the power to the sensor so that's not on all the time either. The sensor is putting out a signal only one millisecond out of every 15,000. Measuring that is a tough job for a DMM or for my old analog scope. And even if you assume the sensor is working perfectly how do you manipulate the sensor accurately to test the range thresholds?

By substituting a voltage reference for the sensor the development is simplified. By supplying a known voltage to the ADC input I can debug the software by writing test code to flash the LEDs and get the ADC

working right. Once I know I'm getting good readings I can then vary the reference input through a range of values to check that the software handles all input values correctly and triggers when it should. Once I know the ADC is reading correctly and the software is doing the right thing with the ADC values, I can put the sensor back in the circuit and verify it using the software.

So having decided that a voltage reference would be a good thing to have, I looked around on eBay and other places. I don't seem to have [url= http://www.eevblog.com/2011/10/24/eevblog-210-krohn-hite-dc-voltage-standard-teardown-calibration/]Dave Jones' eBay luck[/url]. Everything I could find was hundreds of dollars. I don't have that much to spend, so instead I decided to build my own.

EDC MV-106G2 DC VOLTAGE STANDARD & CALIBRATOR KROHN-HITE

$550.00

or Best Offer

Example Voltage Standard posting on eBay

I need to go on a tangent here for a moment. I keep using the term voltage reference. If you do a search for voltage reference on eBay, or your favorite parts supplier, you will get lots of hits and what you find will cost only a few dollars. However, most of these provide only a single voltage and have accuracies in the 0.05% range. That means that at 4 volts they could be off by as much as 2 mV. I'd like something that is accurate to at least 100 uV or about twenty times better. The things that I'm talking about and what Dave Jones has are called voltage standards and can be set for many different values. The difference is that a voltage standard has been calibrated to a particular accuracy specification using an instrument that has been certified to have an even higher accuracy (typically ten times higher accuracy). And that instrument has been calibrated by an instrument certified to higher accuracy yet. This "chain of trust" goes back all the way to some standards body that holds the absolute gold standard for that particular chain. The best meter that I own is a 5 ½ digit (200,000 count) Fluke. It has been calibrated by a certified calibration lab so I have access to that chain of trust. And while I think I can achieve the level of accuracy I want, it might be presumptuous of me to call my device a voltage standard. So I'll keep calling it a reference even though I hope it behaves like and I can use it like a standard.

Getting back to the design of the device: The goal would be to set voltages with the same or better resolution than the ATmega's ADC input across a reasonable input range. Since the ADC is 10 bit it has 1024 levels. If you choose the band-gap reference for the ADC, that's roughly 1 millivolt resolution. If instead we choose AVcc as the reference the resolution would be ~5 mV (on a 5 V system), but let's shoot for 1 mV across the whole range. A nice round number for the range would be 4.095 volts. That's 12-bits worth of 1 mV steps. For accuracy, I'd like when I set the reference to 1.234 mV that it be significantly closer to that value than to 1.233 or 1.235. Same thing when I set the value to 0.005 or

4.095. Let's call it within 100 uV of the target value. Of course, accuracy and resolution are no good if there's too much noise, so it must be low noise as well. This is a voltage reference, not a power supply, so a limit of a few milliamps is acceptable. The last spec is budget. As I said voltage standards on eBay go for hundreds of dollars. I am targeting a budget of $50 parts cost.

A voltage reference is really a simple device. In fact it could be as simple as a potentiometer and an OpAmp. Hook the output of the OpAmp to a good voltmeter and set the voltage level you need using the pot. But that ties up the meter and setting the pot to millivolt values across a multi-volt scale is at best fiddly. So let's complicate things a bit. Let's make the selection digital and include a 4-digit LED display. If we build the control around an Arduino core, the software development is simpler, plus we can re-use the programming serial port for remote control.

Interestingly enough, though I've finished project already, I just recently watched [url=http://ianjohnston.com/index.php?option=com_content&view=article&id=108]Ian Johnson's video[/url] on hacking a voltage standard he bought on eBay. Inside his EDC 501J I expected to see a high end integrated precision DAC. Instead what was inside was a DAC built from discrete resistors and many more trim pots than I expected. The expense must have been in trimming all those resistors!

I want to be able to calibrate as well, but let's hold that to a single trim pot and do the rest in software. For the core of my design I chose the REF5040ID voltage reference chip paired with the AD5060BRJZ DAC. I chose the 5040 because it starts with good accuracy, good temperature stability, good aging and is trimmable. The 5040 is accurate to 0.05% or about 2 mV, but has a trim input. By using a trimmable reference I should be able to get the error well under 1 mV. The 5060 is a 16-bit DAC with plus or minus 2 LSB relative accuracy (INL) and 1 LSB differential nonlinearity (DNL). If you recall, the output resolution needs to be 12 bits. By using a 16-bit DAC the output can be set to the proper 12-bit value and then calibrated with another 4 bits. If the 12-bit step is 1 mV then 4 more bits is 1/16 or 0.0625 mV which should enable me to calibrate to well within the spec. Note that together these two parts are over $20. They are relatively expensive compared to the microcontroller, display or even the PCB, but since they also are the heart of the device I think they are worth it.

All of the rest of the circuitry is there to provide the control interface and to support those two chips. There is a voltage-follower OpAmp driving the output. The OpAmp doesn’t have any more drive than the DAC itself, but it’s better to expose the output of a $2 part rather than a $13 part. The display is a 4 digit 7 segment LED that I have used before. The control knob is a rotary encoder with push button. Turning the encoder right or left steps the output up or down by 1 mV. Pressing the encoder knob push button jumps the output by 250 mV. The output steps up to a max value of 4.095 mV and then rolls around to its lowest setting of 1 mV. In the other direction stepping down past 1 mV rolls around again to 4.095 V. This allows any valid value to be set without too much work. The device will also accept an ASCII string on the serial port to set the voltage level.

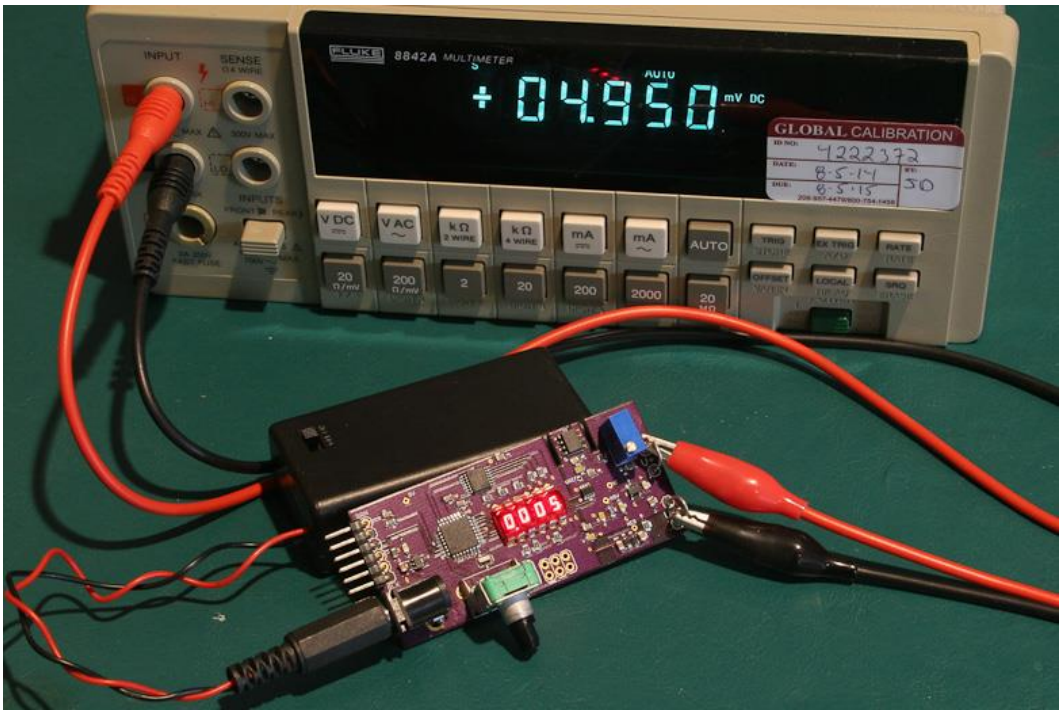
To answer some anticipated questions:

- Why doesn't the output go to 0 V? I wanted to power the device from battery or USB and to keep the cost down I wanted it to be a single-ended supply. But a single-ended supply means that devices that

drive voltage have a hard time getting all the way to ground so I allowed a 1 mV minimum to ease the design. Besides if you really need 0 volts you can remove the voltage reference and ground that signal. You can even ground the output with the voltage reference still in the circuit (the output will not be damaged according to the OpAmp spec) but I advise against it.

- Why doesn't the output go to 5 V? Most of the sensors I'm working with are powered by 3.3 V, so a range of 4.095 V is more than adequate. By powering the voltage reference from USB I have only 5V (plus or minus 10%) available and the reference chip needs a few hundred millivolts of headroom to do its work. Plus by using a 4.096 volt reference, the math for the project is simplified. I could have used a boost regulator but that would have added more cost. If I had chosen a 5 V reference I would have also needed to go to a more expensive 17 or 18 bit DAC to meet the resolution and accuracy spec.

The circuit is burned in for 250 hours before calibrating and the reference is trimmed to as close to 4.096 V as I can make it. Then to calibrate the DAC I wrote a program to set each valid 12-bit value on the DAC and then read it back from the Fluke's GPIB port. From these I compute a correction factor for that value which gets stored in the microcontrollers EEPROM memory. Since there is not enough EEPROM to store all 4096 correction factors they are run-length encoded before being stored.



PVR set to 5 mV shows accurate to 50 uV

That's about it. You can see a fair number of capacitors and a couple inductors in the schematic. Those are to prevent and reduce signal noise. The rest of the circuit is pretty straight forward.

So how well does it work? Let's review the specs.

- Range: 1 to 4095 mV. By design. Check

- Resolution: 1 mV steps. By design. Check
- Accuracy: For values in the range of 1 to 1.999 V I'm measuring 50 uV accuracy. Unfortunately at that point my 200,000 count meter hits its own limits and I can only measure to plus or minus 100 uV, but I believe with a better meter I could calibrate to better than 50 uV across the full range. Even at plus or minus 100 uV the spec has been met.
- Noise: Jitter on the meter is always limited to the least significant digit even when the meter range is down in the microvolts. I don't have a quantitative measure, but the output seems pretty quiet.
- Current: Loading the output with a 1k resistor does not change the output voltage but heavier loading quickly starts affecting the output. A few milliamps of drive is all we designed in so it's not surprising that's all we get.
- Cost: the total parts cost including PCB was $43.89.

Is it useful? Unquestionably yes. I find myself reaching for this tool whenever I'm building a circuit with an ADC or writing the associated code. It is very valuable to have a reliable reference that I can check against or that can fill in for a sensor or other signal source in a controllable, reliable and repeatable way. I highly recommend it.

As usual you can find the schematic, board layout, parts list and source code on my Github site. Or I'm contemplating selling (assembled, tested and calibrated) a batch of these to earn myself a new scope and/or better DMM. If you are interested in having one, let me know.

Good luck with your projects!

- Chip