Physics 524

Survey of Instrumentation and Laboratory Techniques

2025

University of Illinois at Urbana-Champaign

Unit 5d: Voltage

Goal for this unit

- Learn about analog to digital conversion of voltages, as well as digital-to-analog conversions.

In theory…

DAC

A DAC (digital to analog converter) accepts digital input data and generates the corresponding analog output voltage. Your box of stuff includes an MCP4725 12-bit DAC. It's an I2C device, and uses its power supply input (VIN) and ground as the upper and lower limits on the output voltage it can create. It's a conceptually simple device: increasing the digital value you feed it by one unit changes VOUT by VIN / 4096. The internal circuitry is pretty simple (but I won't discuss it here), and we can run it as fast as I2C will allow. Its output can drive a few milliamps of current.

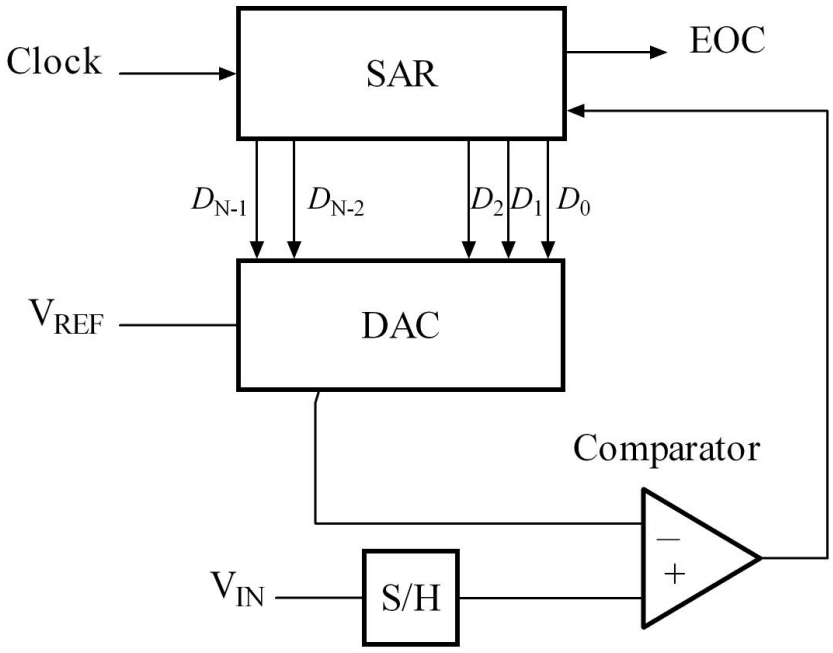
ADC

An ADC (analog to digital converter) accepts analog input data and generates the corresponding digitization of it, kind of like a DAC run backwards. The MEGA 2560 has a single 10 bit ADC shared among 16 analog inputs, while the Adalogger has a 12-bit ADC shared by six inputs. Both microcontrollers allow the user to select the power supply voltage as the upper end of its digitization range, or (optionally) other reference voltages, possibly applied from an external source.

The ADC circuitry is quite sophisticated, allowing the (experienced) user access to a large number of configuration options in which matters of speed, stability, and precision can be balanced against each other to best obtain the desired performance. Take a look at section 32 of the "Atmel SAMD21E / SAMD21G / SAMD21J SMART ARM-Based Microcontroller DATASHEET" linked to this unit on the course website.

The ADC uses a "successive approximation" algorithm to do the digitization. Here's a description from Wikipedia:[1]

[1] https://en.wikipedia.org/wiki/Successive-approximation_ADC



A successive-approximation ADC is a type of analog-to-digital converter that converts a continuous analog waveform into a discrete digital representation using a binary search through all possible quantization levels before finally converging upon a digital output for each conversion.

The successive-approximation analog-to-digital converter circuit typically consists of four chief subcircuits:

- A sample-and-hold circuit to acquire the input voltage V_{in}.
- An analog voltage comparator that compares Vin to the output of the internal DAC and outputs the result of the comparison to the successive-approximation register (SAR).
- A successive-approximation register subcircuit designed to supply an approximate digital code of Vin to the internal DAC.
- An internal reference DAC that, for comparison with V_{ref}, supplies the comparator with an analog voltage equal to the digital code output of the SAR_{in}.
-

The successive approximation register is initialized so that the most significant bit (MSB) is equal to a digital 1. This code is fed into the DAC, which then supplies the analog equivalent of this digital code (Vref/2) into the comparator circuit for comparison with the sampled input voltage. If this analog voltage exceeds Vin, then the comparator causes the SAR to reset this bit; otherwise, the bit is left as 1. Then the next bit is set to 1 and the same test is done, continuing this binary search until every bit in the

SAR has been tested. The resulting code is the digital approximation of the sampled input voltage and is finally output by the SAR at the end of the conversion (EOC).

Mathematically, let Vin = xVref, so x in [−1, 1] is the normalized input voltage. The objective is to approximately digitize x to an accuracy of 1⁄2n. The algorithm proceeds as follows:

1. Initial approximation x0 = 0.
2. ith approximation xi = xi−1 − s(xi−1 − x)/2i, where, s(x) is the signum function (sgn(x) = +1 for x ≥ 0, −1 for x < 0). It follows using mathematical induction that |xn − x| ≤ 1/2n.

As shown in the above algorithm, a SAR ADC requires:

1. An input voltage source Vin.
2. A reference voltage source Vref to normalize the input.
3. A DAC to convert the ith approximation xi to a voltage.
4. A comparator to perform the function s(xi − x) by comparing the DAC's voltage with the input voltage.
5. A register to store the output of the comparator and apply xi−1 − s(xi−1 − x)/2i.

Example: The ten steps to converting an analog input to 10 bit digital, using successive approximation, are shown here for all voltages from 5 V to 0 V in 0.1 V iterations. Since the reference voltage is 5 V, when the input voltage is also 5 V, all bits are set. As the voltage is decreased to 4.9 V, only some of the least significant bits are cleared. The MSB will remain set until the input is one half the reference voltage, 2.5 V.

The binary weights assigned to each bit, starting with the MSB, are 2.5, 1.25, 0.625, 0.3125, 0.15625, 0.078125, 0.0390625, 0.01953125, 0.009765625, 0.0048828125. All of these add up to 4.9951171875, meaning binary 1111111111, or one LSB less than 5.

When the analog input is being compared to the internal DAC output, it effectively is being compared to each of these binary weights, starting with the 2.5 V and either keeping it or clearing it as a result. Then by adding the next weight to the previous result, comparing again, and repeating until all the bits and their weights have been compared to the input, the end result, a binary number representing the analog input, is found.

A four-bit example

It's actually a lot less complicated than it sounds. Let’s run the algorithm by hand for a four-bit digitization, using a four-bit DAC whose reference voltages are 5V and ground. And let’s say that the input to be digitized is 2.9V.

0. ADC analog input is 2.9V

1. set successive approximation register (SAR) high order bit so that SAR = 0b1000 = decimal 8.

2. set DAC output based on current SAR value: DAC output = 5V × 8 / 16 = 2.5V

3. ADC comparator sees its analog input is GREATER than DAC voltage, so it leaves the high order SAR bit as set.

4. Set the next SAR bit to 1 so that SAR = 0b1100 = decimal 12.

5. set DAC output based on current SAR value: DAC output = 5V × 12 / 16 = 3.75V.

6. ADC comparator sees its analog input (2.9V) is LESS than DAC voltage, so it clears the SAR bit to zero. Now SAR = 0b1000 = 8.

7. Set the next SAR bit to 1 so that SAR = 0b1010 = decimal 10.

8. Set DAC output based on current SAR value: DAC output = 5V × 10 / 16 = 3.125V.

9. ADC comparator sees its analog input (2.9V) is LESS than DAC voltage, so it clears the SAR bit to be zero. Now SAR = 0b1000 = 8.

7. Set the next SAR bit to 1 so that SAR = 0b1001 = decimal 9.

8. Set DAC output based on current SAR value: DAC output = 5V × 9 / 16 = 2.8125V.

9. ADC comparator sees its analog input is GREATER than DAC voltage, so it leaves this SAR bit set. Now SAR = 0b10001 = 9.

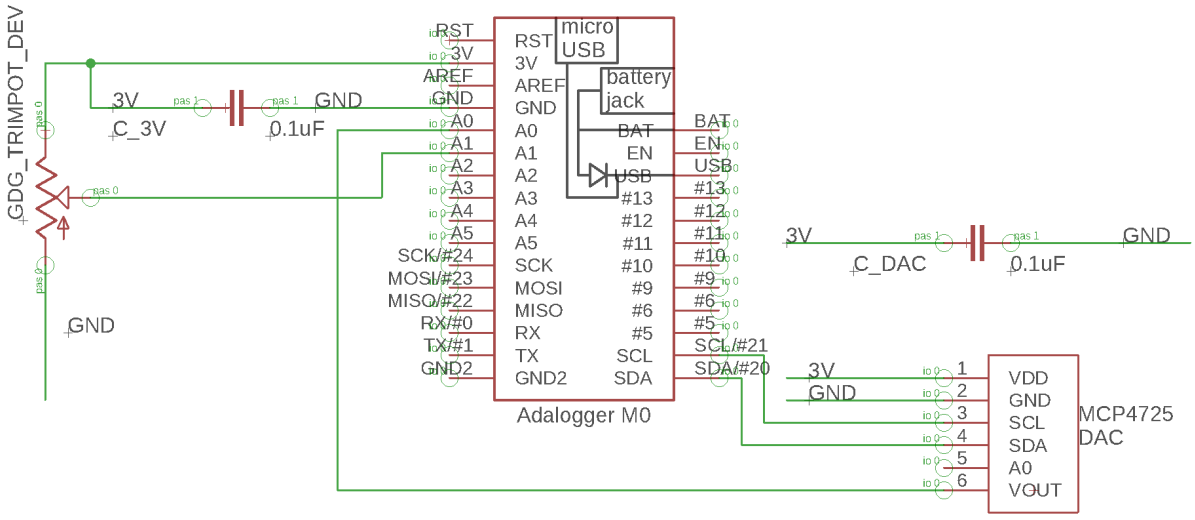
And that's that! Graphically: I could sketch the voltages on the white board for you.

Actual stuff to do in class

Wire up the following breadboard circuit and drive the DAC with code that has the DAC ramp up its voltage output in integer steps, from 0 to 4095. Keep repeating this ramp over and over again. It's OK to make the ramp go as fast as possible.

Get an oscilloscope and display the DAC output voltage ramp and show it off to your instructors.

Have the Adalogger's ADC tell you the value of the trimpot center pin voltage and also, from time to time, the DAC output voltage. (Recall that you can check voltages using your multimeter.)



Homework assignment for this unit

Do this at home, NOT IN CLASS DURING PHYSICS 523, please!

Pretend that the Adalogger uses its ADC inputs primarily to decide whether or not the DAC voltage is greater than the voltage on the trimpot's center pin, but not for much of anything else.

Code up a 12-bit successive approximation ADC algorithm by defining a 12 bit SAR, as described above, and setting or clearing various bits as you hunt for the DAC voltage that's closest to the voltage on the trimpot center pin.