The following should provide you with an understanding of some of the theory and terminology related to DACs (and ADCs).

DAC Terminology

Single Quadrant DAC – The DACs we've been looking at have all used positive (unsigned) binary inputs to produce positive (or negative, if the amplifier is inverting) outputs. In other words, on a coordinate plane, all the results would fit into one quadrant, usually the top right quadrant.

Two-Quadrant DAC – We could level-shift the output so that the results would run from negative to positive, but still with unsigned binary inputs. In this arrangement, it would make sense to make zero volts match the middle of the binary scale. Our results would now occupy the two quadrants on the right side of the coordinate plane.

Four-Quadrant DAC – If we included some logic that converted signed binary inputs into signals that could produce positive to negative output voltages, we would have a DAC that used the entire coordinate plane. Most commercially-available DACs can be configured as Four-Quadrant DACs, and are well-suited to software applications that need to produce full-range output voltages from conventional 2's complement signed binary inputs.

Digitizing – An analog signal has an infinite number of possible voltages over the range between its upper and lower limits. However, a digital signal has discrete output voltages with nothing in between. When we represent an analog signal digitally, we lose some of the accuracy of the signal because we must round each value to the nearest digital equivalent.

Quantization Error – The error introduced will be between 0 and 1 LSB, i.e. the step size. Many DACs offset their values to make the minimum error ±0.5 LSB.

Step Size – The total range is one step less than the reference:

$$V_{step} = \frac{V_{range}}{2^n - 1}$$

or

$$V_{step} = \frac{V_{ref}}{2^n}$$

Sample Rate – For a DAC, the number of digital values converted to an analog output voltage per second is the Sample Rate. The sample rate must be at least twice the highest expected frequency component. (This comes from Nyquist communication theory.)

Resolution – The greater the possible number of digitizing values, the better the signal can be approximated. High resolution means low Quantization Error, but requires more bits.

Linearity – A linear DAC will show a straight-line relationship between the binary input and the voltage output. As a specification, Linearity states how close to a straight line you can expect the output to be.

Monotonicity – If, as the binary input ramps up, the output from a DAC does not always step up as expected, (i.e. some steps actually drop below the previous value), the DAC is not Monotonic.

Answer the following questions about an 8-bit Single Quadrant DAC.

If the output values range from 0 V to +10.0 V, what is the step size? Give your answer to five digits, in mV. 39.216

mV/step

Fill in the following table for selected input values. Provide your answers in volts this time, to four decimal places.

Input value, hexadecimal	Output Voltage, V
0x00	0
0xFF	10
0x80	5.0196

0x93	5.7647

Answer the following questions for an 8-bit Two-Quadrant DAC in which the outputs have been shifted to provide a range from -2.50 V to +2.50 V.

What is the step size for this DAC? Give your answer to 6 digits, in mV. 19.6078 mV/step

Fill in the following table for selected input values. Provide your answers in volts, to four decimal places.

Input value, hexadecimal	Output Voltage, V
0x00	-2.5
0xFF	2.5
0x7F	-0.0098
0x80	0.0098

Hopefully, you just seen one of the conundrums of DACs: If the range of values is nice, the step size will be awkward, and critical values, like zero or half the scale, will not be available. If the step size is nice, the range will be off by a single step at the top end. That's enough to drive any OCD person just a bit crazy!

Here's an example of a Four-Quadrant DAC with a nice step size. Since this is a Four-Quadrant DAC, the binary inputs can be negative or positive, using the 2's Complement system. You'll probably need to convert the hexadecimal values into binary so you can see whether they're negative or positive, then, for the negative ones, you'll have to do a 2's Complement conversion to see what each negative number is.

A particular 12-bit Four-Quadrant DAC has a step size of 2.5 mV/step.

What is the most positive input value, in hexadecimal? 0x7FF What is this, as a decimal number?

2047

What is the expected output voltage, based on the step size of 2.5 mV/step? Give your answer in volts, to four decimal places.

5.1175 V

What is the most negative input value, in hexadecimal? 0x800 What is this, as a decimal number?

-2048

What is the expected output voltage? -5.12 V

Fill in the following table to analyze the predicted outputs for two selected values.

Input, hexadecimal	Decimal equivalent	Output, V
0x42D	1069	2.6725
0xA3C	-1476	-3.69

One of the issues with digitizing analog signals is the amount of memory required for storage. Here's an example to quantify that statement.

Let's analyze one minute of uncompressed video recording that's 1024x768 pixels, each in 32-bit colour, if the sample rate is 24 frames per second.

1024x768 pixels x 32 bits/pixel is just over 25 megabits (Mb) per frame. Multiply that by 24 frames per second to get about 604 Mb per second. Now, multiply by 60 seconds, and you get over 36 Gb of required storage! Divide by 8 bits per byte, and the result is 4.53 GB of required memory for just one minute of uncompressed video, and that's not even HD! In storage values, which are off by a factor of 1.024, this is 4.42 GB. Given that a standard DVD can store 4.7 GB, one minute of uncompressed video of this quality would take essentially one whole DVD!

That brings up a very significant advancement that makes storage and distribution of digitized information possible: Compression.

Compression involves the use of algorithms to reduce the number of bits that need to be stored, while still containing enough information to represent the original data. Compression comes in two general forms: Lossless, where none of the data is lost, and Lossy, where the decompressed information does not contain all of the original data, but is a close approximation. JPEG compression is lossy -- if you zoom in on a JPEG picture, you'll see blocks of pixels that are all the same colour and density -- those regions were determined to have similar enough characteristics that they were treated as identical. In the compression algorithm, the characteristic is recorded once and is assigned to the matching group of pixels, saving a lot of memory. MPEG-4 is also lossy. One of the techniques in this algorithm is only to record when pixels change enough to require an update. So, a blue sky doesn't need to be stored. If you're sitting in front of a stationary camera doing a recording, none of the background needs to be updated -- just the parts of your face that are changing.

So, it's clear there is a lot more that could be learned about digitizing real world events and storing them, but hopefully this has been an informative introduction.