# 7 Digital-to-Analog, Analog-to-Digital Interfacing

## 7.1 Introduction

In any electronic system, two basic types of signals can be generated or measured. They are digital signals which have discrete values, and analog signals which are continuously variable Some examples are pressure, height sound and so on.

In many of the real world systems studied by scientists and engineers the parameters are analog quantities, and when electronic measurement techniques are used, data is derived in analog form from a transducer. It is possible to process, and store analog data using a purely analog system however, this may lead to difficulties in reading and recording.

Digital electronic systems can manipulate and store large amounts of data. The advent of low cost digital microprocessor systems has drastically reduced the cost of implementing digital data processing. In fact, many high integration microprocessors have analog to digital converters built into them. But for the purpose of this course, we are concerned only with devices external to the processor.

Before a digital system can be used, the analog measurements made must first be converted into digital form.

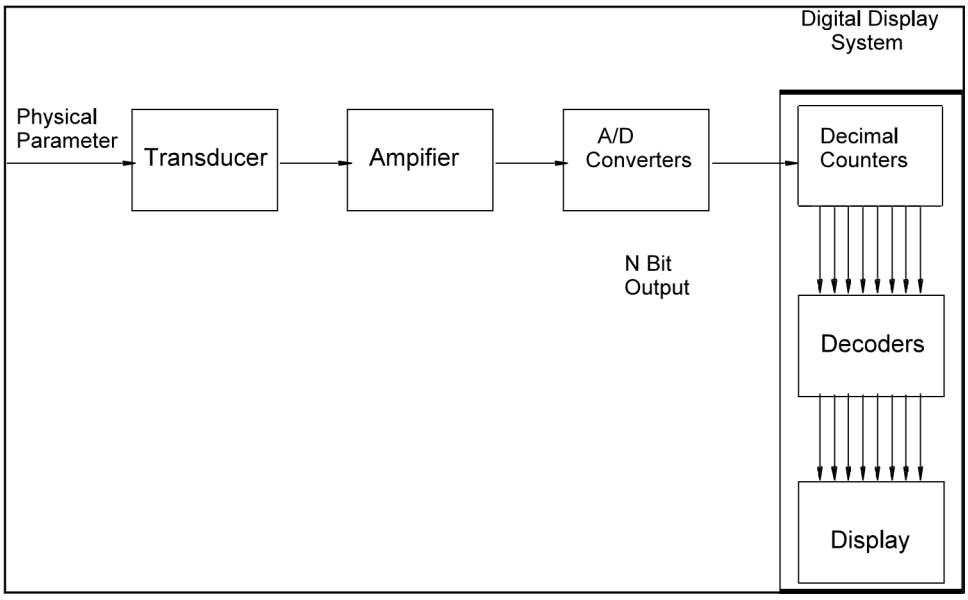



Fig 7.1 A digital instrumentation system

There are some areas where analog techniques are more useful, eg in the presentation of data where a peak or a null is to be determined.

# 7.2 Digital-to-analog Conversion Techniques

A digital-to-analog converter converts a digital value into a analog voltage or current by way of a resistive network. The input could be in binary or BCD, and the output will be a DC voltage or current.

Theoretically, we want the digital value be directly proportional to the to the analog output. For example, say an 8 bit DAC has a value of $255_{10}$ or FFH input to it. This causes an output voltage of 5 volts. An input of $127_{10}$ or 7FH should cause it to output 2.5 V.

A typical DAC is shown in Figure 7.2 consisting of:

a. a resistor network
b. current/voltage switches
c. a reference voltage source
d. an output amplifier for converting current to voltage.

Note that the converter does not know the analog value being output. This is controlled entirely by external hardware and essentially is just a scaling factor.

A more detailed description of a D/A converter can be found in the chapter appendix.

The next diagram shows how a voltage mode R-2R converter may be realised in practice. The accuracy of the DAC is dependent on the resistor network. We need to buffer the inputs and outputs, and provide a stable voltage reference. For more demanding applications, an integrated circuit DAC is needed.

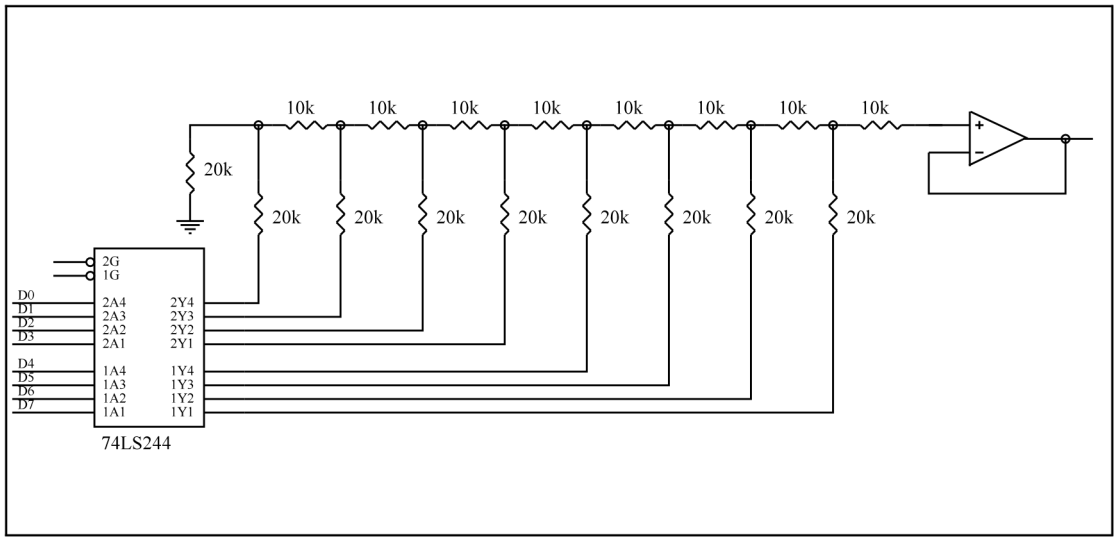



Fig 7.2 - Implementing a Resistance-ladder network

A more expensive option is to use a readily available converter such as the MC1408, shown in Figure 7.3. Note that the chip only takes the place of the ladder network and data switches. You still have to supply the reference voltage, amplifier, and other discrete components. But other DACs will have a data latch built in as well.

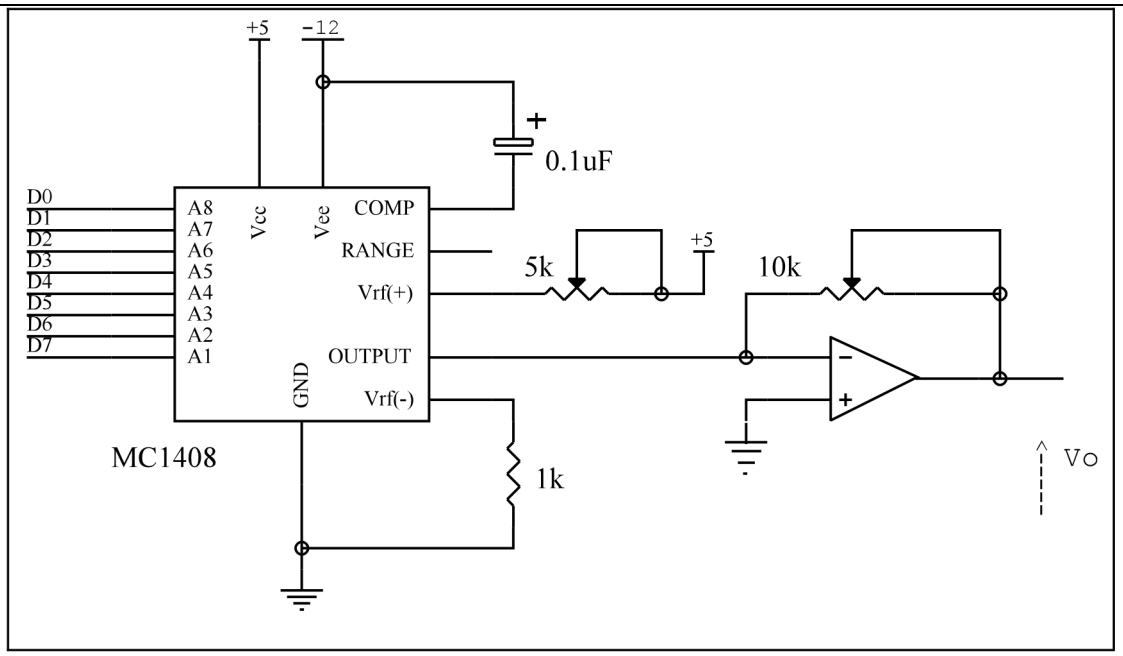



Fig 7.3 Interfacing a MC1408 DAC

Due to the discrete nature of the digital to analog process, it should be noted that the analogue output waveform is a stepped. For example, a sawtooth waveform comes out as a series of steps. Sometimes a low pass filter, also known as a reconstruction filter is used to smooth out the waveform.

## 7.2.1 Generating waveforms

In many applications we have to create an analogue periodic waveform digitally. This process is called synthesis. We then manually compute the digital values and put these into a lookup table, or even put it into an EPROM. There are three factors to consider concerning the waveform:

a) The maximum analogue value
b) The output frequency
c) The number of samples per period

The first factor concerns the quantisation error which determines how much the digital value approximates the analogue equivalent. We want to use the full range of the analogue signal to correspond to the full range of allowable digital values. This minises the quantisation error which determines the scaling factor, which we will discuss shortly. Also, we must know if the DAC is capable of handling negative voltages. If not, we may have to add in an offset voltage so the minimum value of the waveform is zero. For example, sinusoidal waveforms need this offset.

The second factor is limited by the execution speed of the processor. The third factor is a combination of the two.

### 7.2.1.1 Generating a Sine Wave

In theory the DAC accepts inputs from 0 to 255 and outputs proportionally from 0 to the maximum analogue voltage. But because of the components used, there may be clipping after a certain voltage.

Assume we have an 8-bit, DAC in which the supply voltage is 5V. We expect the DAC to produce analogue outputs from 0 to 5V. We want to output a sine wave with a 4V peak to peak voltage. Digitizing with 8 bits does not allow negative values. So we need to offset the sine wave. In this case, we will add an offset so that the minimum voltage is 0V. By doing so, we make full use of the voltage range that can be digitized. This will reduce quantization noise.

For a sine wave, the amplitude is half the peak to peak voltage or 2V. The offset is also 2V. So the equation of the sine wave is:

$$V = 2(1+ \sin \theta)\ V$$

where $\theta$ depends on the number of samples per cycle. More generally, $V= V_{amp}*\sin \theta + V_{offset}$ as shown below.

NOTE: **This discussion is true for SINE type waves only!**
Of course, digitizing other waveform shapes require knowledge of the equation of the waveform.

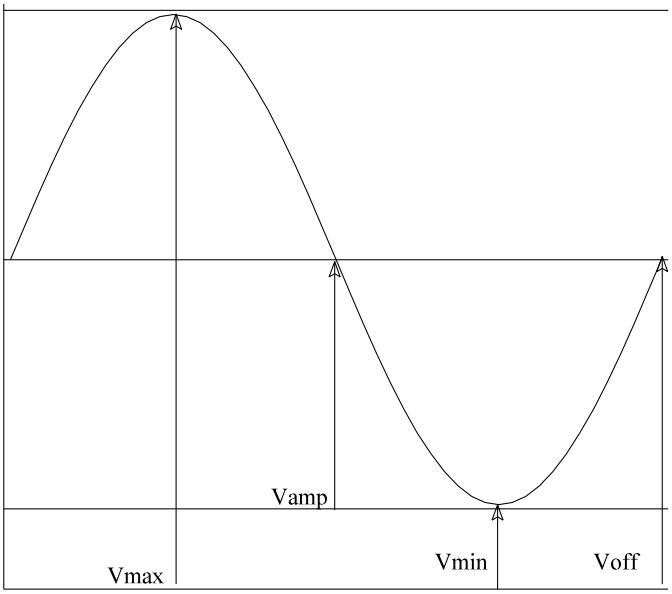



Fig 7.4 Sine Wave

Whenever possible, we should let the lowest value of the waveform be equal to zero, for ease of computation. In some cases this is already true. For example a rectified sine wave does not have an offset, as its lowest value is zero.

Resolution

From the description of the analogue wave, find out its full scale value. This should be equivalent to the largest digital value the D/A converter can handle. In this case, since we have an 8 bit DAC which outputs from 0 to 5V, an input of FFH ($255_{10}$) will result in 5 volts being output. Thus for 1 volt, the equivalent digital value is:

$255/5 = 51_{10}$. We call this the scale factor.

Samples per period

Now let us specify that we want 24 samples per period. We need to compute the value of the sinusoid at each sample point. Note that one period of a sinusoidal wave is equivalent to 360 degrees. So, to compute the value of the sinusoid at each sample point, we will divide 360 by the number of samples. A full treatment of the samples per period will depend things like the Nyquist criterion, which states that the sampling frequency should be twice that of the highest frequency present in the signal.

Here, we will focus on the shape of the waveform and ease of calculation. Less samples per cycle gives a more digital output as mentioned above. In this case let us consider 24 samples as giving us an acceptable likeness to a sine wave. So in one cycle of the sine wave, we need to calculate the values of the sinusoid at 360/24 = 15 degree intervals. It is convenient to do this in the form of a table, as shown below:

| S/no | 2 | sin 2 | V = 2(1 + sin 2) | $F_{scale} * V_{10}$ | Rounded |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 2 | 102 | 102 |
| 1 | 15 | 0.2588 | 2.5176 | 128.3976 | 128 |
| 2 | 30 | 0.5 | 3 | 153 | 153 |
| 3 | 45 | 0.7071 | 3.4142 | 174.1242 | 174 |
| 4 | 60 | 0.866 | 3.732 | 190.332 | 190 |
| 5 | 75 | 0.9659 | 3.9318 | 200.5218 | 201 |
| 6 | 90 | 1 | 4 | 204 | 204 |
| 7 | 105 | 0.9659 | 3.9318 | 200.5218 | 201 |
| 8 | 120 | 0.866 | 3.732 | 190.332 | 190 |
| 9 | 135 | 0.7071 | 3.4142 | 174.1242 | 174 |
| 10 | 150 | 0.5 | 3 | 153 | 153 |
| 11 | 165 | 0.2588 | 2.5176 | 128.3976 | 128 |
| 12 | 180 | 0 | 2 | 102 | 102 |
| 13 | 195 | -0.2588 | 1.4824 | 75.6024 | 76 |
| 14 | 210 | -0.5 | 1 | 51 | 51 |
| 15 | 225 | -0.7071 | 0.5858 | 29.8758 | 30 |
| 16 | 240 | -0.866 | 0.268 | 13.668 | 14 |

| | | | | | |
|---|---|---|---|---|---|
| 17 | 255 | -0.9659 | 0.0682 | 3.4782 | 3 |
| 18 | 270 | -1 | 0 | 0 | 0 |
| 19 | 285 | -0.9659 | 0.0682 | 3.4782 | 3 |
| 20 | 300 | -0.866 | 0.268 | 13.668 | 14 |
| 21 | 315 | -0.7071 | 0.5858 | 29.8758 | 30 |
| 22 | 330 | -0.5 | 1 | 51 | 51 |
| 23 | 345 | -0.2588 | 1.4824 | 75.6024 | 76 |

Output Frequency

This is equal to 1 / (waveform period). The waveform period is the number of samples per period *multiplied* by the time between samples. In the above example, the number of samples per period is 24. If the time delay between each sample is 50 :s for example, the frequency of the sine wave above will be 1 / (24 * 50 :s). If we have a large number of samples per period, we will need a shorter time period between samples in order to get the same frequency. This time period between samples has a minimum value which is determined by the speed of the processor.

The question now is how to send these numbers to the DAC in order for it to produce the waveform. This can be done by either software or hardware number generation.

### 7.2.1.2 Software Generation

One option would be to write a program with the numbers stored as a look up table. Each data would be fetched in turn and sent to a port with the DAC attached to it. The numbers would create the basic profile of a Sine wave, and the total period of the wave would be determined by the time period between fetching and sending each individual number. The maximum frequency of the Sine wave would be limited by the speed of your software. It might be possible to generate the numbers using a software algorithm, however, this would severely limit the rate at which the numbers could be produced, though it might save memory space if the look up table was going to be lengthy and yet the algorithm is simple.

Here is how a C language program would output these values:

```
#define     Count       23    /* number of values in table */
#define     DACPort  *((char *) 0x133          /* DAC Port  */

const unsigned char WaveTable[Count] =
{    102 102,128,153,174,190,201,204,201,190,174,153,128,
      102,76,51,30,14,0,3,14,30,51,76
};

void main (void)
{                          /* main entry for program */
      unsigned char i;
      while 1
```

```
        {
              for (i = 0 ; i <= Count; i++)
              {
                    Sleep(1);        /*delay */
                    DACPort = WaveTblPtr[i]; /* op to data port */
              }
        }
}
```

### 7.2.1.3 Hardware Generation

A ROM based waveform generator is an alternative way of sending the numbers to the DAC. You would program your look up table into an EPROM and use a counter to create the addresses. In the following example, although we have an 8k device, we will only be using the first 256 locations as we only have an 8 bit counter. The content of address 00H would be 102 decimal, address 01H would be 128 decimal and so on. Thus we will use up to 256 samples in the waveform.

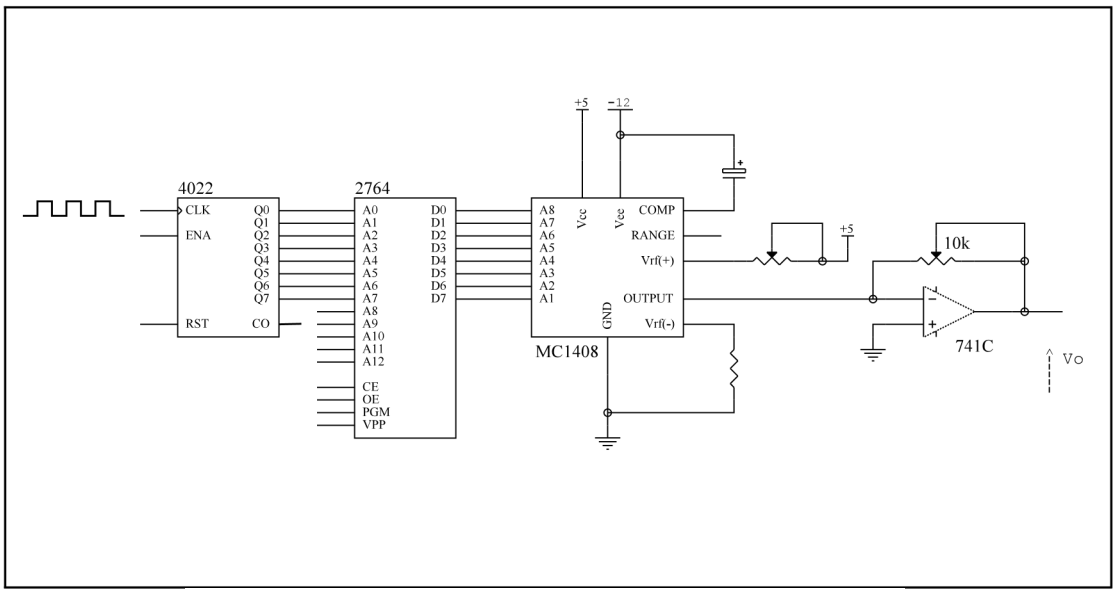



Fig 7.5 A ROM based waveform generator

### 7.2.1.4 Pulse Width Modulation

The previous methods discussed need a latched output with many pins and also analogue resistors which need high precision. Respectively these result in added complexity and manufacturing cost.

For a given sampling period, pulse width modulation (PWM) uses digital voltages and represents an analogue value by the proportion of *time* the voltage is on. In this case, one digital output can produce one analogue voltage. In the figure below, the left side shows a voltage $V_{avg}$ being generated by a fixed digital voltage $V_{hi}$. Assuming $V_{hi}$ is 5 volts and if the duty cycle is 80%, by filtering the waveform, we can get $V_{avg}$ being 4 volts. On figure 7.6, we see the PWM waveform increasing in duty cycle as the analogue waveform ramps up.

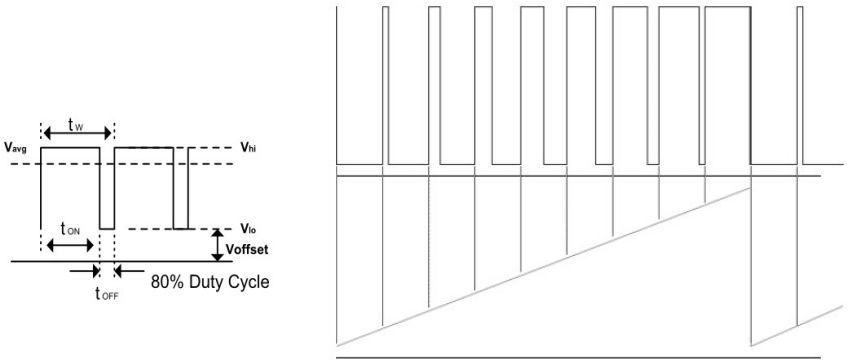



Fig 7.6 Converting a Pulse-width modulation digital output to a DAC output

The benefits are:
- lesser pin count which translates to a smaller size
- lower heat dissipation as the PWM outputs are either On or Off,
- lower cost as there is no need for expensive analogue processes
- better noise immunity because of digital signals
- high switching speeds

However, there is the problem of ripple and digital noise caused by the switching and the need for good filtering techniques.

## 7.3 Analog-to-Digital Conversion Techniques

Analog-to-Digital Converters (ADC) are systems that are used to convert analog signals into digital equivalents. The main techniques are:

- Integrating
- Counter Ramp
- Successive Approximation
- Flash
- Sigma-Delta

### 7.3.1 Integrating Techniques of AD conversion

The principle used by the intergrating techniques is to let an integrator be charged towards a required voltage. Generally, the larger the input voltage the longer the charge time. The output is controls a counter which provides a digital representation of the time required for the conversion, which is directly proportional to the magnitude of the input voltage.

Consider an analog process where switch S2 is first closed to discharge the capacitor C. When the capacitor is discharged, S2 is opened and S1 is closed, at the same time, the counter is reset. The capacitor then charges up via $-V_{ref}$ and a ramp voltage forms at the input of the comparator. This ramp is compared to the input voltage $V_{in}$. When the ramp voltage is larger than $V_{in}$, the comparator switches off the counter. The number of counts is proportional to the magnitude of $V_{in}$.

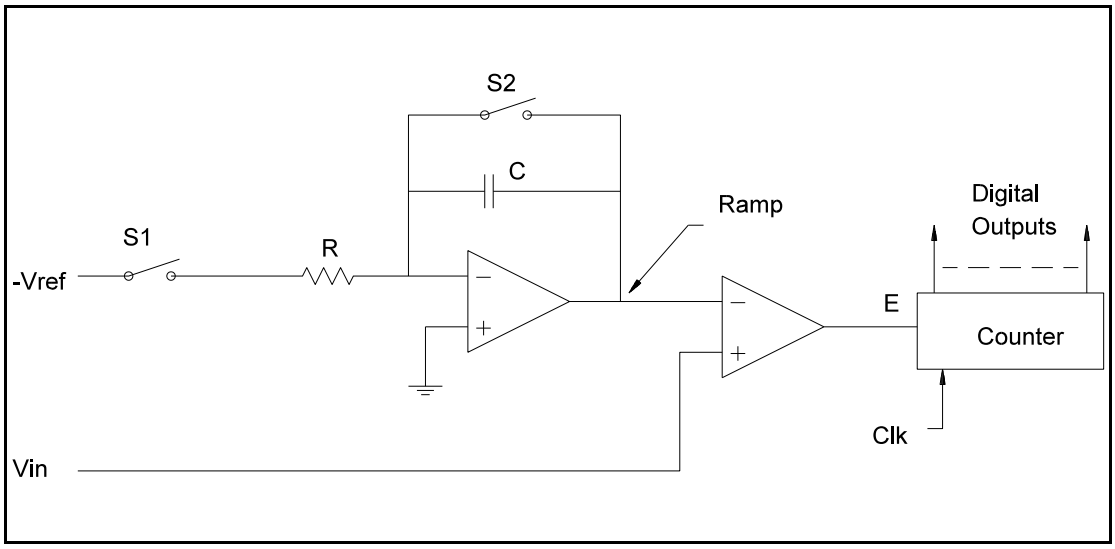



Fig 7.7 Analog integrator

The single ramp ADC has some limitations as its accuracy depends on:

a. the resistor, R which can drift over time and temperature
b. the capacitor, C which can also drift over time and temperature
c. the reference voltage, $-V_{ref}$
d. the device timing process

These problems can be overcome by using a Dual Slope ADC which will not be covered.

## 7.3.2 Counter Ramp ADCs

The simplest in concept is the Counter Ramp ADC. The digital logic circuit is a counter and a comparator. Figure 10-9 shows the block diagram of the Counter Ramp ADC.

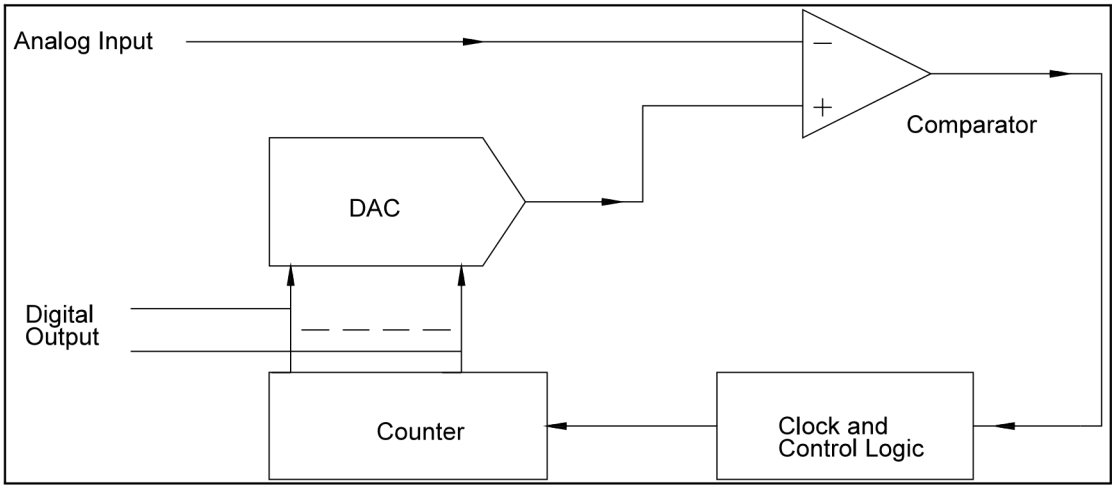



Fig 7.8 A counter ramp ADC

At the start of a conversion, the counter is set to zero and hence forces the DAC output to 0 volts. The non-inverting input of the comparator is thus 0 volts. If the input analog signal at the inverting input is non-zero, the comparator output is set to a logic '0' which enables the clock to pass to the counter. The counter increments in steps, thereby increasing the values output from the DAC.

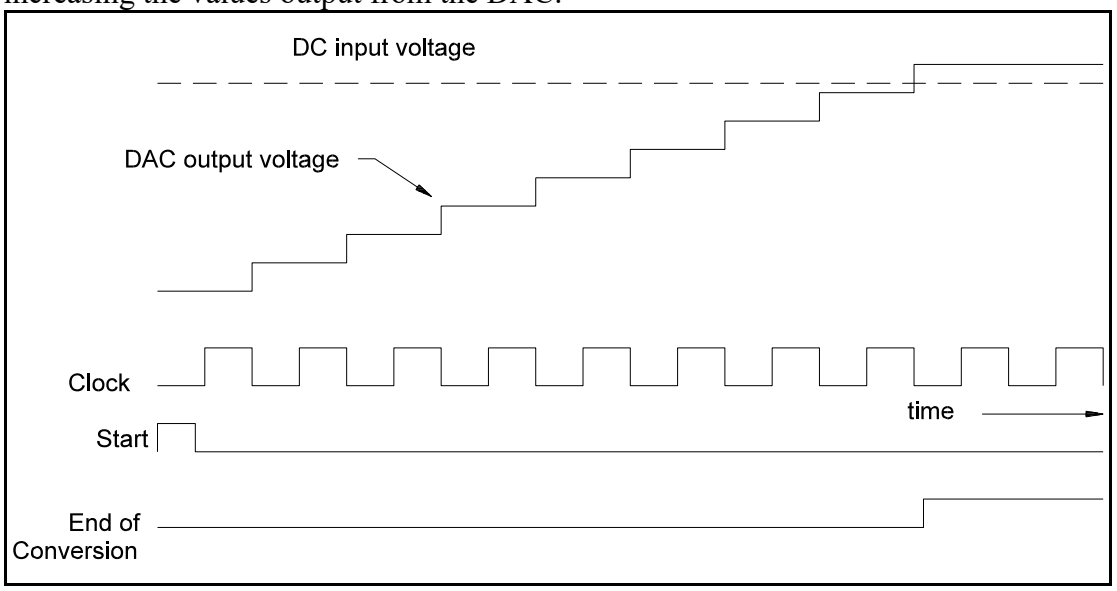



Fig 7.9 Timing Sequence for Counter Ramp ADC

This process continues until the clock is disabled, ie, when the output of the DAC is larger or equal to the analog input. Figure 10-10 shows the timing sequence of the operation.

Note that the counter increments on the rising on the rising clock edge and that the comparator output changes when the DAC output exceeds the input voltage. At the end of the conversion, the comparator outputs a logic "1" which is detected by the control logic. The control logic then generates an **End-of-Conversion** signal to indicate to the digital subsystem that the binary representation of the output can be read from the counter inputs.

A disadvantage of the Counter Ramp is the long conversion time required to count up from the value of 0 volts to the input value. A faster clock may be used to speed up the process but consideration has to be taken regarding the settling time of the clock signal as well as the response of the overall circuitry. Furthermore, smaller input voltages will be converted faster than larger ones. This leads to different conversion times for different voltages.

## 7.3.3 Successive Approximation ADC

The Successive Approximation ADC provides a more rapid conversion than the Counter Ramp ADC. Rather than increasing the DAC output one bit at a time, the digital logic circuitry uses a binary search. This works by taking the value to be found and comparing it with half of the maximum range of possible values. Depending on whether it is larger or smaller than this value, it is accumulated, and another comparison will be done on the lower or higher half of the half range. This is done until only one bit is left to be compared. The number of comparisons is the same as the number of bits.

The following figure shows the block diagram of an eight-bit Successive Approximation ADC. The major difference from the Counter Ramp ADC is that the counter has been replaced by an eight-bit register. The states of the output bits of this register are controlled by the clock and control logic block.

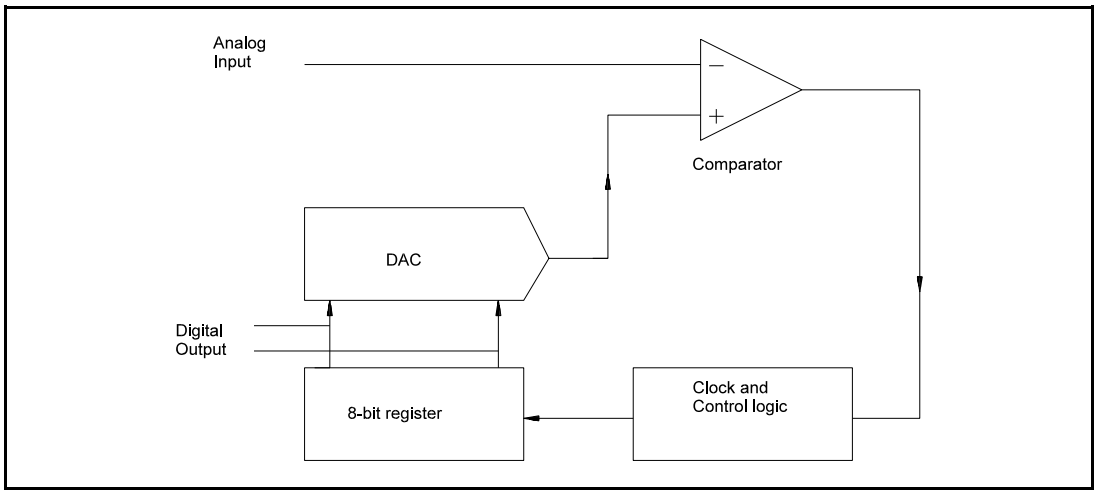



Fig 7.10 A Successive Approximation ADC

As an example, consider a DAC which gives 5 volts is when we input FFH to it. Assume an analogue voltage of 3.57 volts is applied. The value the ADC should give for this voltage is: (3.57/5) *255 = $182_{10}$ or B6H. Let us see how this is value is obtained.

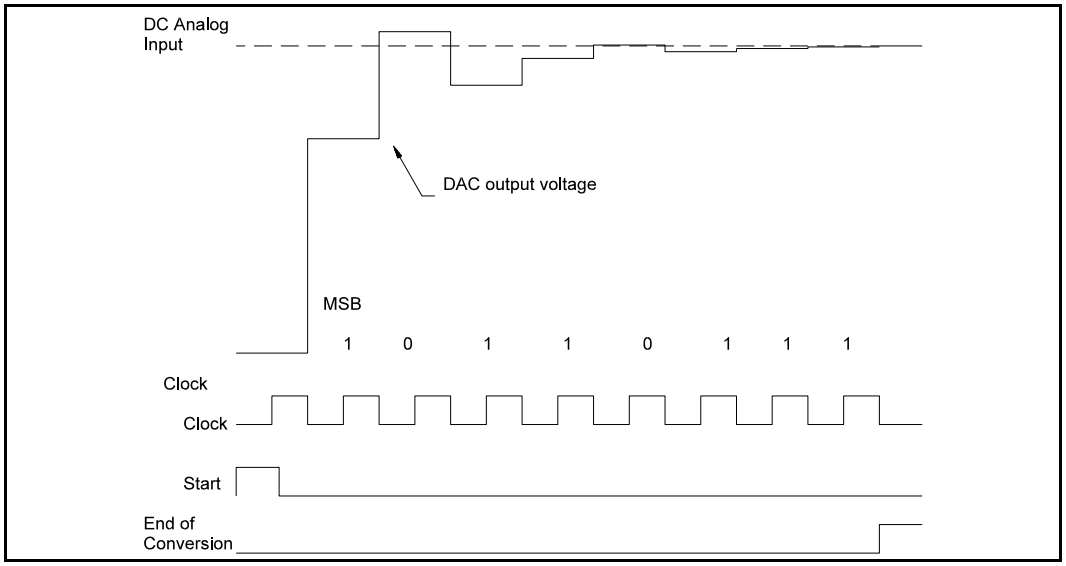



Fig 7.11 Timing Diagram for the Successive Approximation ADC

| Register value-bit ptr | Decimal | DAC voltage | < Input voltage? | Action |
|---|---|---|---|---|
| *1*000 0000 - bit 8 | 128 | 2.51 | Yes | Keep bit |
| 1*1*00 0000 - bit 7 | 192 | 3.76 | No | Clear bit |

| | | | | |
|---|---|---|---|---|
| 10*I*0 0000 - bit 6 | 160 | 3.13 | Yes | Keep bit |
| 101*I* 0000 - bit 5 | 176 | 3.45 | Yes | Keep bit |
| 1011 *I*000 - bit 4 | 184 | 3.61 | No | Clear bit |
| 1011 0*I*00 - bit 3 | 180 | 3.53 | Yes | Keep bit |
| 1011 01*I*0 - bit 2 | 182 | 3.57 | Yes | Keep bit |
| 1011 011*I* - bit 1 | 183 | 3.59 | No | Clear bit |

So the digital value is $182_{10}$ or B6H. In summary, the algorithm is as follows:

1. Clear all bits to zero, place bit pointer to Most Significant Bit
2. Set bit indicated by pointer
3. Use DAC, generate analogue voltage
4. Compare with Input Voltage
5. If Input Voltage > Generated Voltage goto step 7
6. else clear bit indicated by pointer
7. Bit pointer positioned to next lower bit
8. If all bits in register processed then stop, else go to step 2

Of course, we can implement the digital logic and clock as a piece of software from a processor. The comparator and DAC is still needed however.

Parallel/Flash

ADCsThe **Parallel** or **Flash** ADC is the fastest ADC currently available, they use a resistive potential divider to generate the series of voltage levels. There are $2^n$-1 levels for a N-bit conversion.

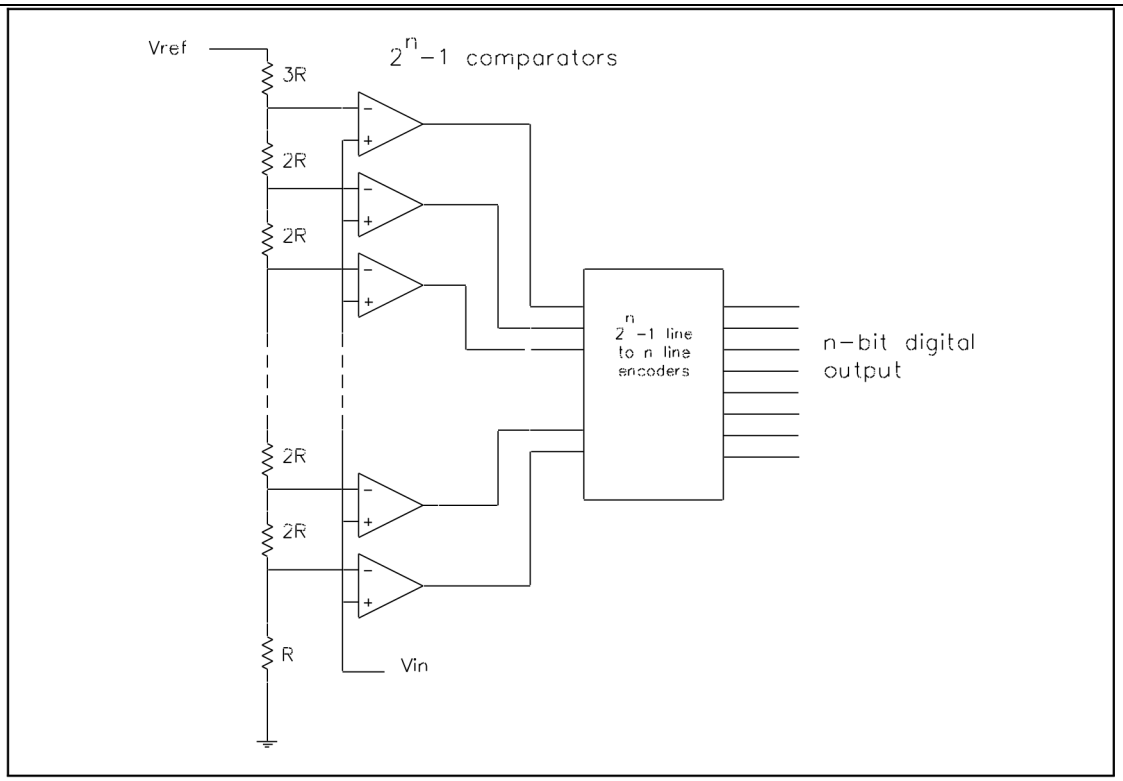



Fig 7.12 A parallel ADC

The parallel conversion principle is conceptually straight forward, but it has the disadvantage of requiring a large amount of circuitry to implement a high resolution conversion. An 8-bit system would require 255 comparators!

## 7.3.4 Sigma Delta converter

This method of conversion is relatively recent, following developments in Integrated Circuit technology which allow for greater amounts of digital circuitry to be included for better signal conversion characteristics. The method is a development of PWM in using one bit for sampling purposes, but at a frequency much higher than that of the input. Instead of turning signals on and off for fixed periods, we can generate a ramp by summing a series of 1's and 0's and approximate any value we want, at a fast enough rate. This is called oversampling, which "spreads out" the quantisation noise (see later) over the higher sampling rate.

Consider a 3 bit representation of a signal which uses $2^3 = 8$ levels over 8 time samples. As before, if we use a Counter Ramp type of conversion (see middle waveform) we can approximate the analogue signal with a series of Pulse Width Modulated (PWM) signals.

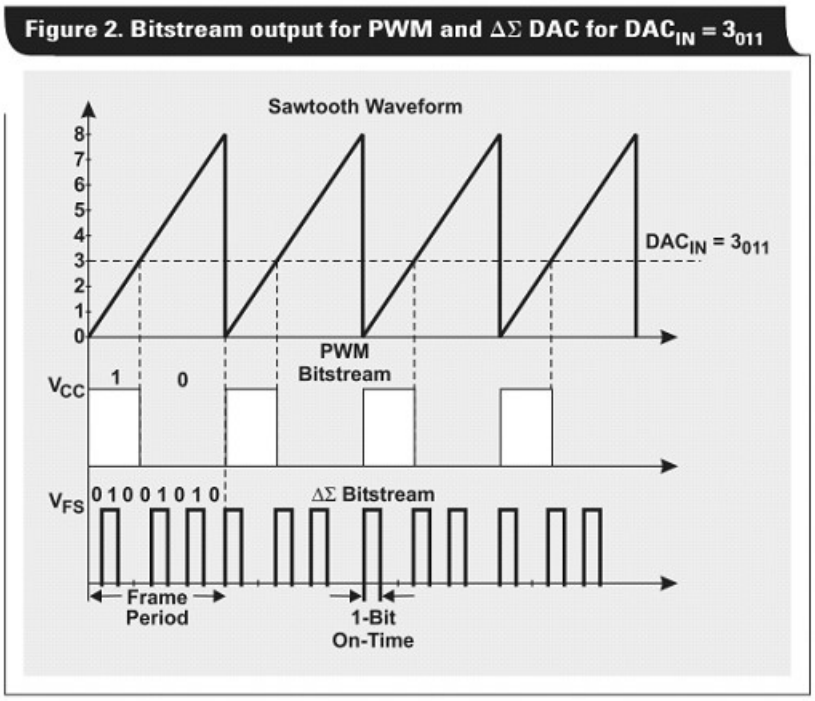



Fig 7.13 Comparison of PWM with Ó-Ä data conversion

In contrast, if now we use a series of pulses (lowest waveform) at a higher frequency to represent the signal over the *same* period of time, we can achieve the same amount of energy transmitted- but we need to *average* over that time period. Consider that if we now have *more* samples in the same time interval, we can more accurately approximate the analogue signal. This requires hardware that can work at a high frequencies but the benefits are that:

- it easier to filter quantisation noise as it occurs at a high frequency
- allows for more pulses to represent a signal by *averaging* it over a higher sampling rate thus achieving higher resolution without the need for high precision analog circuitry - moving more of the quantisation noise to higher frequencies is also known as noise shaping

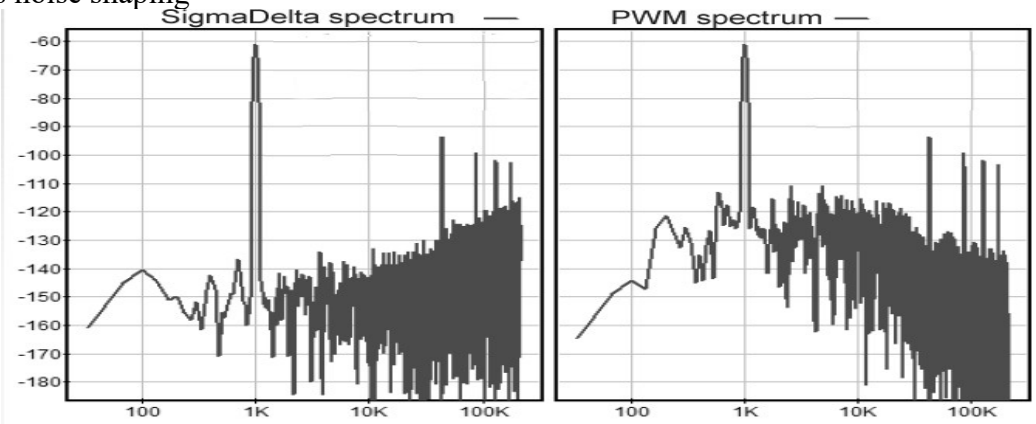



Fig 7.14 Comparison - PWM (left) / Sigma-Delta (right) spectrum of 1 kHz sine wave

From the figure, we see that at the -150 dB point, the Sigma-Delta converter has unwanted signals till about 10kHz when the amplitude of these signals rises, making it harder to low pass filter away. But the PWM has these spurious frequencies at 1-20 kHz

at a higher amplitudes, making it hard to filter away. Together with the noise reduction a higher resolution is achieved, easily giving 16 bits and often 24 bits.

The block diagram shows a first-order Sigma-Delta ADC device, the details of its operation are in the appendix.

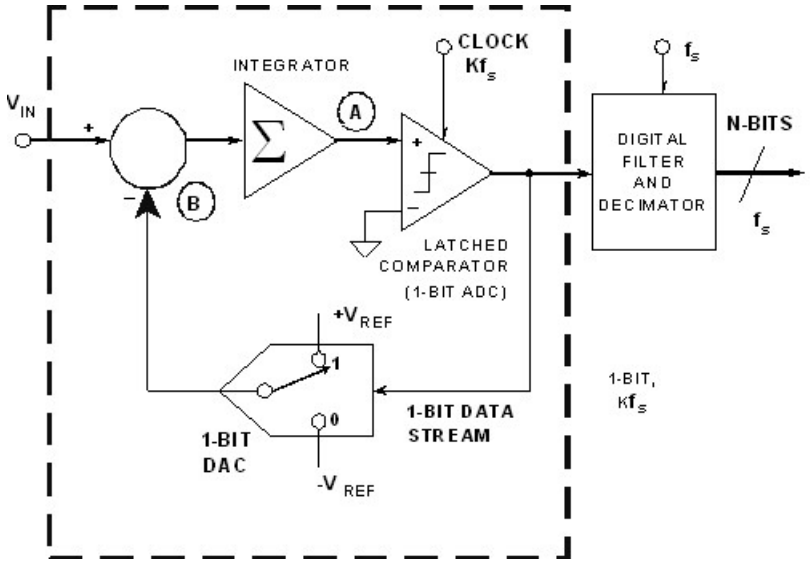



Fig 7.15 Block diagram of a first order Sigma-Delta ADC

**Sample Hold Amplifier**

A sample/hold amplifier is a discrete or integrated device placed in between the input waveform and the ADC. Its purpose is to hold Vin as a constant whilst the conversion is in progress, so that the ADC is not following a changing waveform. Conceptually it is an electronic switch and a capacitor. If no conversion is required the switch is closed and the voltage across the capacitor follows Vin, when conversion starts the switch opens and the voltage across the capacitor will remain constant until conversion is complete.

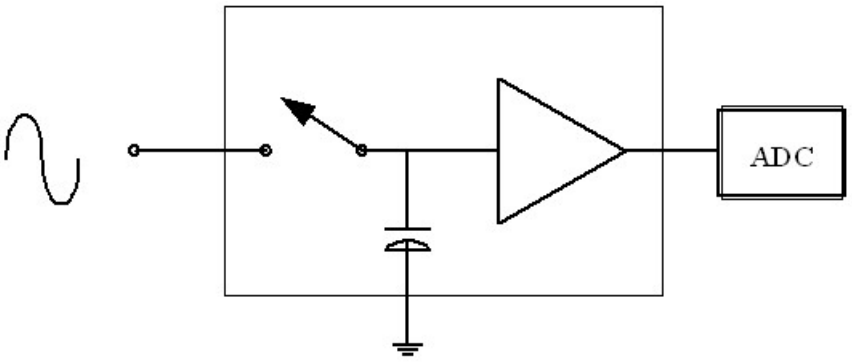



Fig7.16 The inclusion of a Sample/Hold Amplifier

# 7.4 Errors in DA and AD Conversion

There are several sources of errors when converting data. Some are due to the very nature of the process, others are due to impefections in the electronic components.

## 7.4.1 Quantization Errors in D/A Conversion

The analog output of a DAC is not infinitely variable, but is quantized into small but definable steps. Hence the voltage produced at the output of a DAC is seldom able to be exactly the same as the voltage at the output of a true analog system.

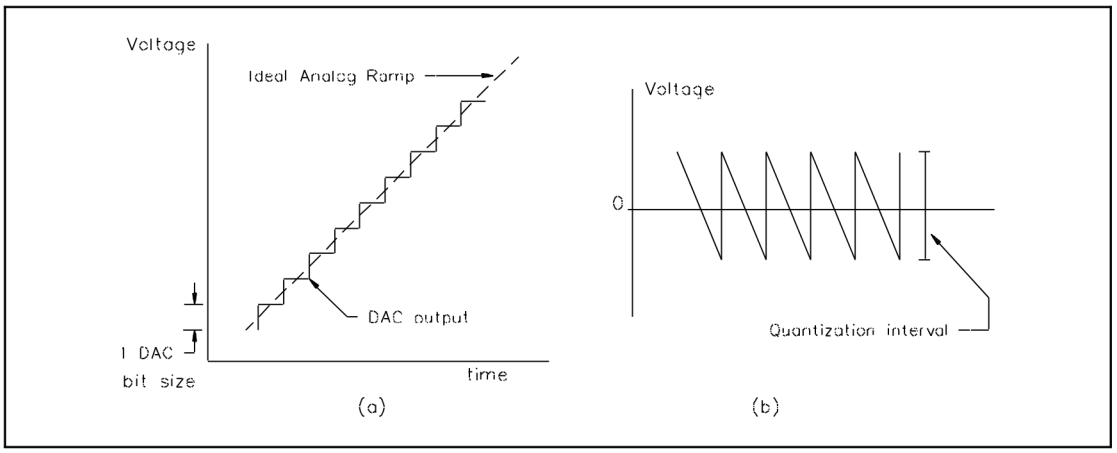



Fig 7.17 Quantization Errors

A portion of the DAC output waveform is superimposed onto an ideal analog waveform. The difference between these two voltages represent an error in the system. As this difference in voltage arises from the quantized nature of the output of the DAC, it is called **Quantization Error**.

To reduce the quantization error, a DAC must be controlled by more input bits. If the full-scale voltage remains the same, the increase in bits will reduce the quantization interval. Quantization error is also often referred to as **Resolution**.

**Resolution** can be expressed in several ways.... eg for an 8-bit A/D:

- as a number of bits ..... 8-bit resolution.
- as a percentage ..... 0.39% ($1/2^8 * 100$)
- as the relative size of the LSB..... 1 in 256
- as an absolute value (say 5v range)... 19mV ($5V/2^8$)

APPENDIX

## Other Sources of Errors

### Gain Error

This is often caused by errors in the feedback resistor on the converter op-amp. The error is usually a linear scale to the expected output.

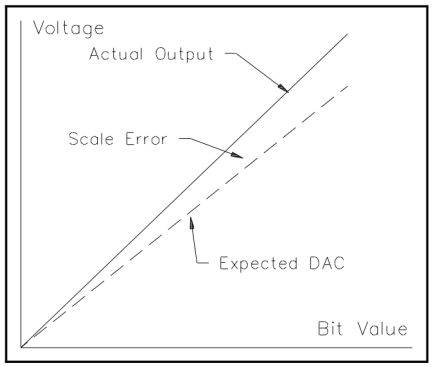



Fig 7.18 Gain Error

### Offset Error

This is when the output is non-zero when a zero input is used. This error can be caused by op-amp errors or leakage currents in the current switches. The error can be corrected by taking readings at zero input value and subtracting from the output.

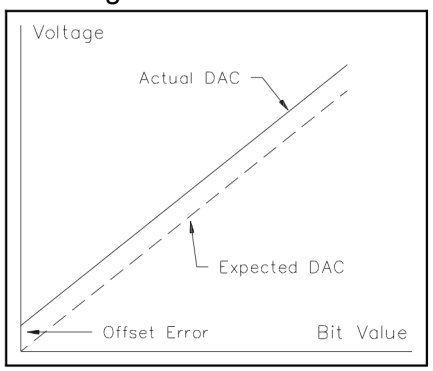



Fig 7.19 Offset Error

### Linearity Errors

This error is usually caused by errors in the current source resistor values. Hence the output loses its proportionality.

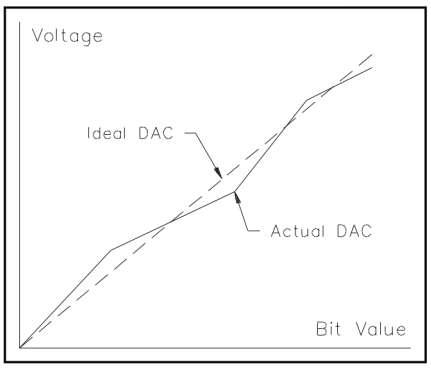



Fig 7.20 Linearity Error

**Internal detail of D/A converters**

The structure of the network is balanced so that the current supplied from any switch is halved at every junction. A binary relationship between each of the switches and hence the current resulting from the switches can be used to represent binary numbers.

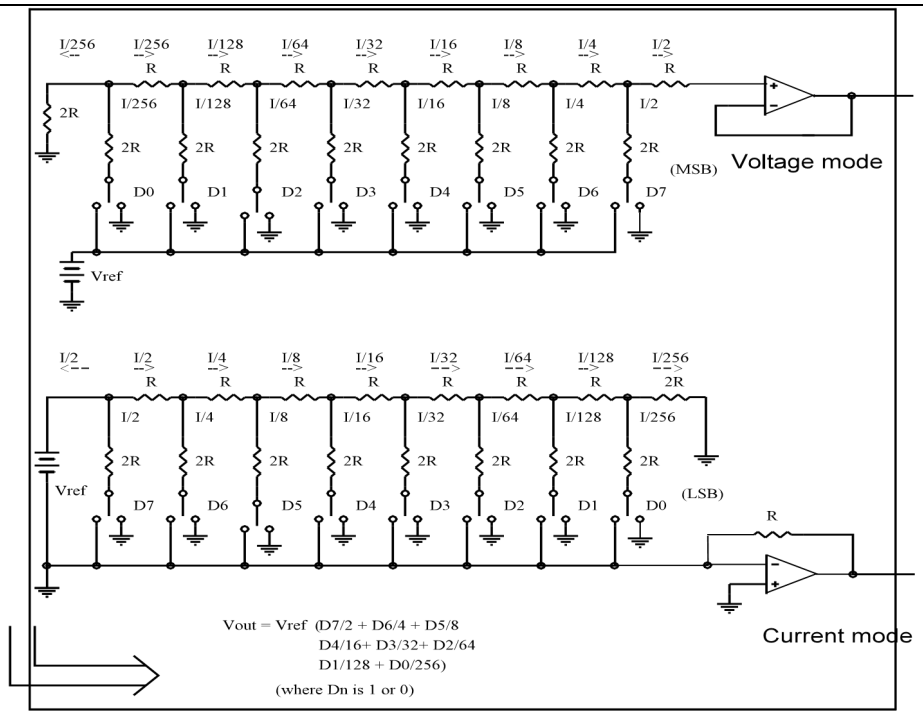



Fig 7.21 A Theoretical Resistance-ladder network

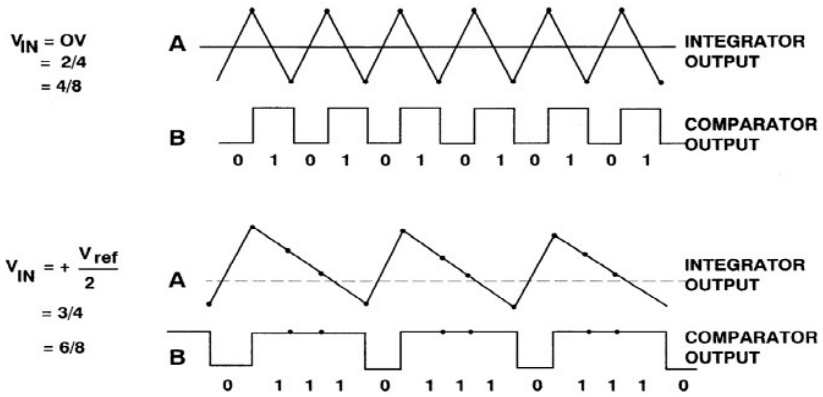
The above figure shows two practical and low cost methods of implementing a DAC which is widely used. The R-2R network is implemented with discrete resistors. A buffer or latch is used to drive the resistor network. An opamp which is capable of 0 volt output is used. This circuit may be used for many applications without much loss in quality. As shown, we can use the network in two modes. The voltage mode is easier to implement and understand. Current mode produces less noise and does not need a high performance op-amp buffer and is preferred in integrated circuit versions of the R-2R ladder.

Details of ∑-Δ (Sigma-Delta) conversion

Assume a DC input at $V_{IN}$. The integrator will be always incrementing or decrementing at point A. The output of the comparator is fed back through a 1-bit DAC to the summing input at point B. This negative feedback will keep the average DC voltage at point B be equal to $V_{IN}$ which is done by controlling the average DAC output. The average DAC output voltage is controlled by the amount of 1's in the 1-bit data stream from the comparator output. As the input signal increases towards $+V_{REF}$ , the number of 1's in the serial bit stream increases, and the number of 0's decreases and vice versa. In this very simple description, this analysis shows that the average value of the input voltage is can be represented by the bit stream out of the comparator. The digital filter and decimator process the serial bit stream and produce the final output data.

Unlike the other types of ADCs considered, only when a large number of 1-bit ADC samples are averaged, will a meaningful value result. The ∑-Δ converter is difficult to analyze in the time domain because of this apparent randomness of the single-bit data output. If the input signal is near positive full-scale, it is clear that there will be more "1"s than "0"s in the bit stream. Likewise, for signals near negative full-scale, there will be more "0"s than "1"s in the bit stream. For signals near midscale, there will be approximately an equal number of "1"s and "0"s. Figure 5 shows the

output of the integrator for two input conditions. The first is for an input of zero (midscale). To decode the output, pass the output samples through a simple digital lowpass filter that averages every four samples. The output of the filter is 2/4. This value represents bipolar zero. If more samples are averaged, more dynamic range is achieved. For example, averaging 4 samples gives 2 bits of resolution, while averaging 8 samples yields 4/8, or 3 bits of resolution. In the bottom waveform of Figure 5, the average obtained for 4 samples is 3/4, and the average for 8 samples is 6/8.




| (Vin-DAC) Adder | Integrator out (V) | Comparator (DAC in) | DAC out (V) |
|---|---|---|---|
| 1 (Vin=1 or $V_{ref}/2$) | 0 | | 0 |
| 1 | 1 (0+1) | 1 (1>=0) | 2 |

| | | | |
|---|---|---|---|
| -1 (1-+2) | 0 (1+-1) | 1 (0>=0) | 2 |
| -1(1-+2) | -1 (0+-1) | 0 (-1<0) | -2 |
| 3(1- -2) | 2 (-1+3) | 1 (2>=0) | 2 |
| -1(1-+2) | 1 (2+-1) | 1 (1>=0) | 2 |
| -1(1-+2) | 0 (1+-1) | 1 (1>=0) | 2 |
| -1(1-+2) | -1 (0+-1) | 0 (-1<0) | -2 |