CHAPTER 5: FUNDAMENTALS OF SAMPLED DATA SYSTEMS

CHAPTER 5: FUNDAMENTALS OF SAMPLED DATA SYSTEMS

INTRODUCTION

To fully understand the specifications for converters it is beneficial to cover the fundamentals of sampling theory.

SECTION 5.1: CODING AND QUANTIZING

Analog-to-digital converters (ADCs) translate analog measurements, which are characteristic of most phenomena in the "real world," to digital language, used in information processing, computing, data transmission, and control systems. Digital-to-analog converters (DACs) are used in transforming transmitted or stored data, or the results of digital processing, back to "real-world" variables for control, information display, or further analog processing. The relationships between inputs and outputs of ADCs and DACs are shown in Figure 5.1.

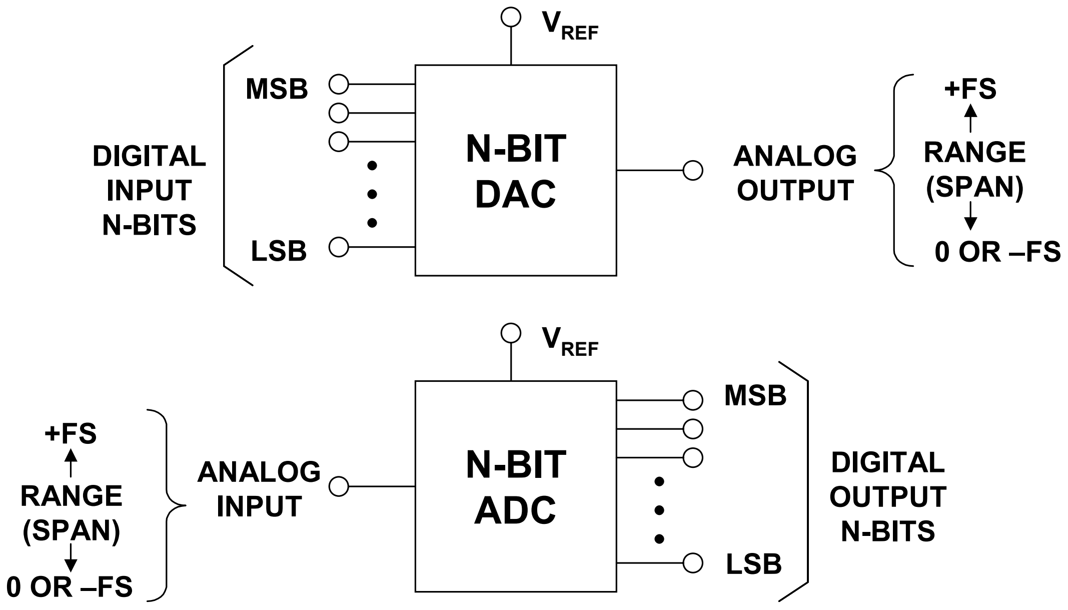


Figure 5.1: *Analog-to-Digital Converter (ADC) and Digital-to-Analog Converter (DAC) Input and Output Definitions*

Analog input variables, whatever their origin, are most frequently converted by transducers into voltages or currents. These electrical quantities may appear as fast or slow "dc" continuous direct measurements of a phenomenon in the time domain, as modulated ac waveforms (using a wide variety of modulation techniques), or in some combination, with a spatial configuration of related variables to represent shaft angles. Examples of the first are outputs of thermocouples, potentiometers on dc references, and analog computing circuitry; of the second, "chopped" optical measurements, ac strain gage or bridge outputs, and digital signals buried in noise; and of the third, synchros and resolvers.

The analog variables to be dealt with in this chapter are those involving voltages or currents representing the actual analog phenomena. They may be either wideband or narrowband. They may be either scaled from the direct measurement, or subjected to some form of analog preprocessing, such as linearization, combination, demodulation, filtering, sample-hold, etc.

As part of the process, the voltages and currents are "normalized" to ranges compatible with assigned ADC input ranges. Analog output voltages or currents from DACs are direct and in normalized form, but they may be subsequently post-processed (e.g., scaled, filtered, amplified, etc.).

Information in digital form is normally represented by arbitrarily fixed voltage levels referred to "ground," either occurring at the outputs of logic gates, or applied to their inputs. The digital numbers used are all basically binary; that is, each "bit," or unit of information has one of two possible states. These states are "off," "false," or "0," and "on," "true," or "1." It is also possible to represent the two logic states by two different levels of current, however this is much less popular than using voltages. There is also no particular reason why the voltages need be referenced to ground—as in the case of emitter coupled logic (ECL), positive emitter coupled logic (PECL) or low voltage differential signaling logic (LVDS) for example.

Words are groups of levels representing digital numbers; the levels may appear simultaneously in *parallel,* on a bus or groups of gate inputs or outputs, *serially* (or in a time sequence) on a single line, or as a sequence of parallel bytes (i.e., "byte-serial") or nibbles (small bytes). For example, a 16-bit word may occupy the 16 bits of a 16-bit bus, or it may be divided into two sequential bytes for an 8-bit bus, or four 4-bit nibbles for a 4-bit bus.

A unique parallel or serial grouping of digital levels, or a *number,* or *code,* is assigned to each analog level which is quantized (i.e., represents a unique portion of the analog range). A typical digital code would be this array:

$$a_7\ a_6\ a_5\ a_4\ a_3\ a_2\ a_1\ a_0 = 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1$$

It is composed of eight bits. The "1" at the extreme left is called the "most significant bit" (MSB, or Bit 1), and the one at the right is called the "least significant bit" (LSB, or bit *N:* 8 in this case). The meaning of the code, as a number, a character, or a representation of an analog variable, is unknown until the *code* and the *conversion relationship* have been defined. It is important not to confuse the designation of a particular bit (i.e., Bit 1, Bit 2, etc.) with the subscripts associated with the "a" array. The subscripts correspond to power of 2 associated with the weight of a particular bit in the sequence.

The best-known code is *natural or straight binary* (base 2). Binary codes are most familiar in representing integers; i.e., in a natural binary integer code having N bits, the LSB has a weight of 2^0 (i.e., 1), the next bit has a weight of 2^1 (i.e., 2), and so on up to the MSB, which has a weight of 2^{N-1} (i.e., $2^N/2$). The value of a binary number is obtained by adding up the weights of all non-zero bits. When the weighted bits are added up, they form a unique number having any value from 0 to 2^{N-1}.

Often, for convenience, a binary number is expressing in *hexadecimal* (base 16). This reduces the length of the word and makes it easier to read. Fig. 5.2 shows the relationship between binary and hexadecimal (commonly referred to as "hex").

BINARY	HEX	BINARY	HEX
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

Figure 5.2: *The Relationship Between Binary and Hexadecimal*

WHOLE NUMBERS:

$$\text{Number}_{10} = \underset{\underset{\text{MSB}}{\uparrow}}{a_{N-1}}2^{N-1} + a_{N-2}2^{N-2} + \ldots + a_1 2^1 + \underset{\underset{\text{LSB}}{\uparrow}}{a_0}2^0$$

$$\text{Example: } 1011_2 = (1\times2^3) + (0\times2^2) + (1\times2^1) + (1\times2^0)$$
$$= 8 + 0 + 2 + 1 = 11_{10}$$

FRACTIONAL NUMBERS:

$$\text{Number}_{10} = \underset{\underset{\text{MSB}}{\uparrow}}{a_{N-1}}2^{-1} + a_{N-2}2^{-2} + \ldots + a_1 2^{-(N-1)} + \underset{\underset{\text{LSB}}{\uparrow}}{a_0}2^{-N}$$

$$\text{Example: } 0.1011_2 = (1\times0.5) + (0\times0.25) + (1\times0.125) + (1\times0.0625)$$
$$= 0.5 + 0 + 0.125 + 0.0625 = 0.6875_{10}$$

Figure 5.3: *Representing a Base-10 Number with a Binary Number (Base-2)*

In converter technology, full-scale (abbreviated *FS*) is independent of the number of bits of resolution, N. A more useful coding is *fractional* binary which is always normalized to full-scale. Integer binary can be interpreted as fractional binary if all integer values are divided by 2^N. For example, the MSB has a weight of ½ (i.e., $2^{(N-1)}/2^N = 2^{-1}$), the next bit has a weight of ¼ (i.e., 2^{-2}, and so forth down to the LSB, which has a weight of $1/2^N$ (i.e., 2^{-N}). When the weighted bits are added up, they form a number with any of 2^N

values, from 0 to $(1 - 2^{-N})$ of full-scale. Additional bits simply provide more fine structure without affecting full-scale range. The relationship between base-10 numbers and binary numbers (base-2) are shown in Figure 5.3 along with examples of each.

Unipolar Codes

In data conversion systems, the coding method must be related to the analog input range (or span) of an ADC or the analog output range (or span) of a DAC. The simplest case is when the input to the ADC or the output of the DAC is always a unipolar positive voltage (current outputs are very popular for DAC outputs, much less for ADC inputs). The most popular code for this type of signal is *straight binary* and is shown in Figure 5.3 for a 4-bit converter. Notice that there are 16 distinct possible levels, ranging from the all-zeros code 0000, to the all-ones code 1111. It is important to note that the analog value represented by the all-ones code is not full-scale (abbreviated FS), but FS – 1 LSB. This is a common convention in data conversion notation and applies to both ADCs and DACs. Figure 5.4 gives the base-10 equivalent number, the value of the base-2 binary code relative to full-scale (FS), and also the corresponding voltage level for each code (assuming a +10 V full-scale converter).

BASE 10 NUMBER	SCALE	+10 V FS	BINARY
+15	+FS – 1 LSB = 15/16 FS	9.375	1111
+14	+7/8 FS	8.750	1110
+13	+13/16 FS	8.125	1101
+12	+3/4 FS	7.500	1100
+11	+11/16 FS	6.875	1011
+10	+5/16 FS	6.250	1010
+9	+9/16 FS	5.625	1001
+8	+1/2 FS	5.000	1000
+7	+7/16 FS	4.375	0111
+6	+3/8 FS	3.750	0110
+5	+5/16 FS	3.125	0101
+4	+1/4 FS	2.500	0100
+3	+3/16 FS	1.875	0011
+2	+1/8 FS	1.250	0010
+1	1 LSB = +1/16 FS	0.625	0001
0	0	0.000	0000

Figure 5.4: *Unipolar Binary Code, 4-Bit Converter*

Figure 5.5 shows the transfer function for an ideal 3-bit DAC with straight binary input coding. Notice that the analog output is zero for the all-zeros input code. As the digital input code increases, the analog output increases 1 LSB (1/8 scale in this example) per code. The most positive output voltage is 7/8 FS, corresponding to a value equal to FS – 1 LSB. The mid-scale output of 1/2 FS is generated when the digital input code is 100.

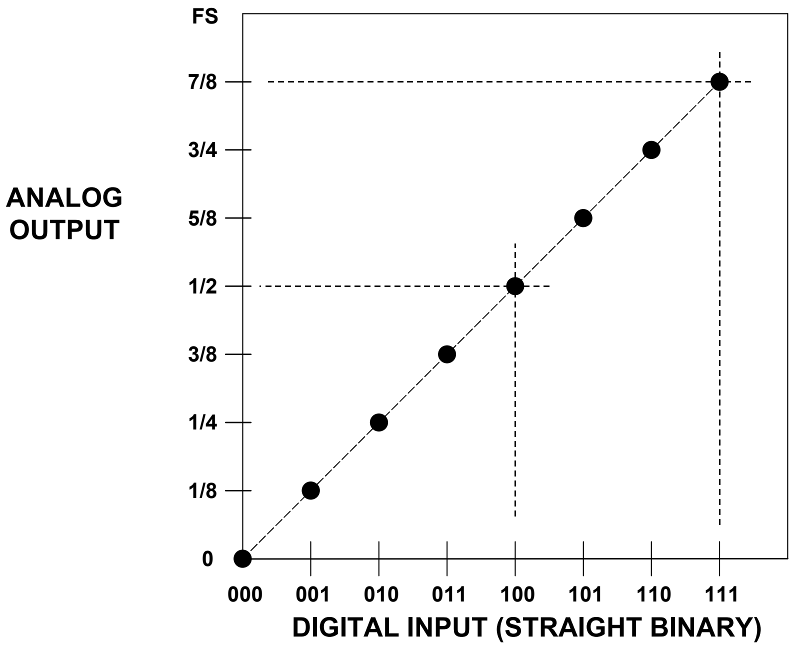


Figure 5.5: *Transfer Function for Ideal Unipolar 3-Bit DAC*

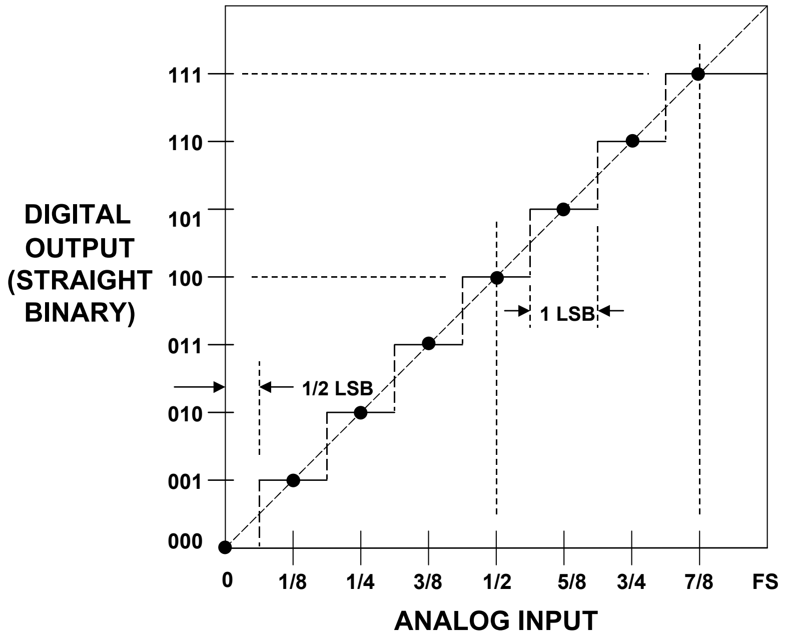


Figure 5.6: *Transfer Function for Ideal 3-Bit Unipolar ADC*

The transfer function of an ideal 3-bit ADC is shown in Figure 5.6. There is a range of analog input voltage over which the ADC will produce a given output code, and this range is the *quantization uncertainty* and is equal to 1 LSB. Note that the width of the transition regions between adjacent codes is zero for an ideal ADC. In practice, however, there is always transition noise associated with these levels, and therefore the width is non-zero. It is customary to define the analog input corresponding to a given code by the *code center* which lies halfway between two adjacent transition regions (illustrated by the black dots in the diagram). This requires that the first transition region occur at ½ LSB. The full-scale analog input voltage is defined by 7/8 FS, (FS – 1 LSB).

Bipolar Codes

In many systems, it is desirable to represent both positive and negative analog quantities with binary codes. Either *offset binary*, *twos complement*, *ones complement*, and *sign magnitude* codes will accomplish this, but offset binary and twos complement are by far the most popular. The relationships between these codes for a 4-bit system is shown in Figure 5.7. Note that the values are scaled for a ±5 V full-scale input/output voltage range.

BASE 10 NUMBER	SCALE	±5V FS	OFFSET BINARY	TWOS COMP.	ONES COMP.	SIGN MAG.
+7	+FS – 1LSB = +7/8 FS	+4.375	1 1 1 1	0 1 1 1	0 1 1 1	0 1 1 1
+6	+3/4 FS	+3.750	1 1 1 0	0 1 1 0	0 1 1 0	0 1 1 0
+5	+5/8 FS	+3.125	1 1 0 1	0 1 0 1	0 1 0 1	0 1 0 1
+4	+1/2 FS	+2.500	1 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0
+3	+3/8 FS	+1.875	1 0 1 1	0 0 1 1	0 0 1 1	0 0 1 1
+2	+1/4 FS	+1.250	1 0 1 0	0 0 1 0	0 0 1 0	0 0 1 0
+1	+1/8 FS	+0.625	1 0 0 1	0 0 0 1	0 0 0 1	0 0 0 1
0	0	0.000	1 0 0 0	0 0 0 0	*0 0 0 0	*1 0 0 0
–1	– 1/8 FS	–0.625	0 1 1 1	1 1 1 1	1 1 1 0	1 0 0 1
–2	– 1/4 FS	–1.250	0 1 1 0	1 1 1 0	1 1 0 1	1 0 1 0
–3	– 3/8 FS	–1.875	0 1 0 1	1 1 0 1	1 1 0 0	1 0 1 1
–4	–1/2 FS	–2.500	0 1 0 0	1 1 0 0	1 0 1 1	1 1 0 0
–5	–5/8 FS	–3.125	0 0 1 1	1 0 1 1	1 0 1 0	1 1 0 1
–6	–3/4 FS	–3.750	0 0 1 0	1 0 1 0	1 0 0 1	1 1 1 0
–7	– FS + 1LSB = –7/8 FS	–4.375	0 0 0 1	1 0 0 1	1 0 0 0	1 1 1 1
–8	– FS	–5.000	0 0 0 0	1 0 0 0		

CODES NOT NORMALLY USED
IN COMPUTATIONS (SEE TEXT)

*		ONES COMP.	SIGN MAG.
	0+	0 0 0 0	0 0 0 0
	0–	1 1 1 1	1 0 0 0

Figure 5.7: *Bipolar Codes, 4-Bit Converter*

For *offset binary*, the zero signal value is assigned the code 1000. The sequence of codes is identical to that of straight binary. The only difference between a straight and offset binary system is the half-scale offset associated with analog signal. The most negative value (–FS + 1 LSB) is assigned the code 0001, and the most positive value (+FS – 1 LSB) is assigned the code 1111. Note that in order to maintain perfect symmetry about mid-scale, the all-zeros code (0000) representing negative full-scale (–FS) is not normally used in computation. It can be used to represent a negative off-range condition or simply assigned the value of the 0001 (–FS + 1 LSB).

The relationship between the offset binary code and the analog output range of a bipolar 3-bit DAC is shown in Figure 5.8. The analog output of the DAC is zero for the zero-value input code 100. The most negative output voltage is generally defined by the 001 code (–FS + 1 LSB), and the most positive by 111 (+FS – 1 LSB). The output voltage for the 000 input code is available for use if desired, but makes the output nonsymmetrical about zero and complicates the mathematics.

The offset binary output code a bipolar 3-bit ADC as a function of its analog input is shown in Figure 5.9. Note that zero analog input defines the center of the mid-scale code 100. As in the case of bipolar DACs, the most negative input voltage is generally defined by the 001 code (–FS + 1 LSB), and the most positive by 111 (+FS – 1 LSB). As discussed above, the 000 output code is available for use if desired, but makes the output nonsymmetrical about zero and complicates the mathematics.

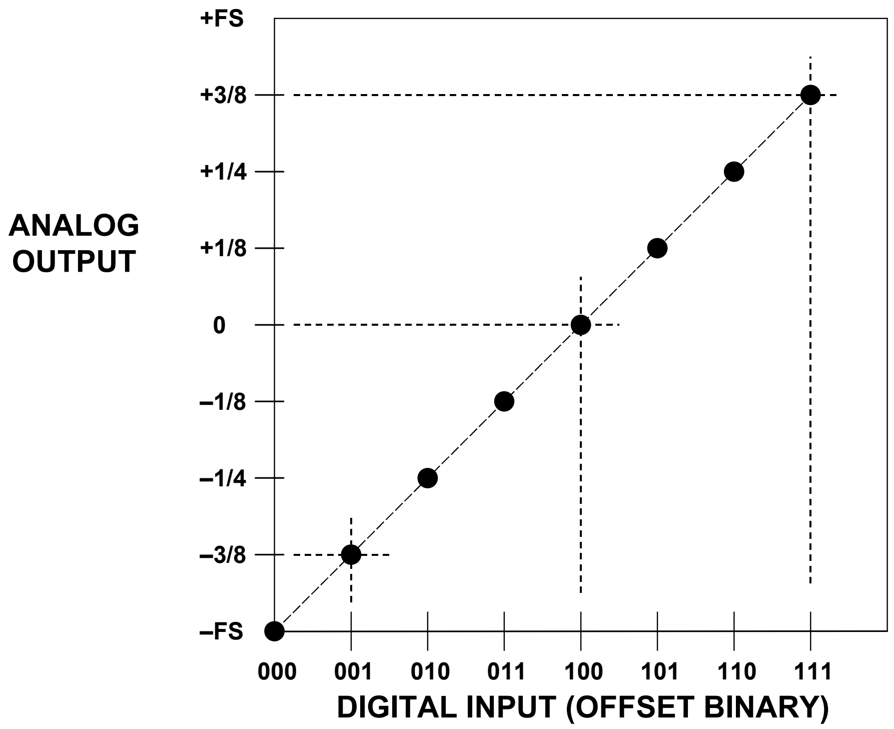


Figure 5.8: *Transfer Function for Ideal Bipolar 3-Bit DAC*

Twos complement is identical to offset binary with the most significant bit (MSB) complemented (inverted). This is obviously very easy to accomplish in a data converter, using a simple inverter or taking the complementary output of a "D" flip-flop. The popularity of twos complement coding lies in the ease with which mathematical operations can be performed in computers. Twos complement, for conversion purposes, consists of a binary code for positive magnitudes (0 sign bit), and the twos complement of each positive number to represent its negative. The twos complement is formed arithmetically by complementing the number and adding 1 LSB. For example, –3/8 FS is obtained by taking the twos complement of +3/8 FS. This is done by first complementing +3/8 FS, 0011 obtaining 1100. Adding 1 LSB, we obtain 1101.

Twos complement makes subtraction easy. For example, to subtract 3/8 FS from 4/8 FS, add 4/8 to –3/8, or 0100 to 1101. The result is 0001, disregarding the extra carry, or 1/8.

Ones complement can also be used to represent negative numbers, although it is much less popular than twos complement and rarely used today. The ones complement is obtained by simply complementing all of a positive number's digits. For instance, the ones complement of 3/8 FS (0011) is 1100. A ones complemented code can be formed by complementing each positive value to obtain its corresponding negative value. This includes zero, which is then represented by either of two codes, 0000 (referred to as 0+) and 1111 (referred to as 0–). This ambiguity must be dealt with mathematically, and presents obvious problems relating to ADCs and DACs for which there is a single code which represents zero.

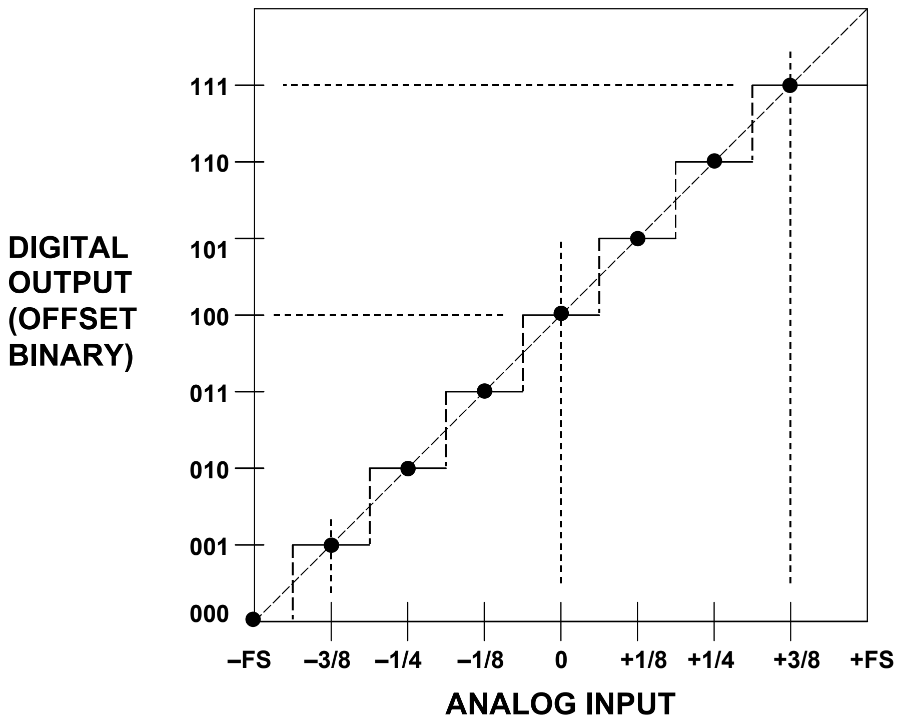


Figure 5.9: *Transfer Function for Ideal 3-Bit Bipolar ADC*

Sign-magnitude would appear to be the most straightforward way of expressing signed analog quantities digitally. Simply determine the code appropriate for the magnitude and add a polarity bit. Sign-magnitude BCD is popular in bipolar digital voltmeters, but has the problem of two allowable codes for zero. It is therefore unpopular for most applications involving ADCs or DACs.

Figure 5.10 summarizes the relationships between the various bipolar codes: offset binary, twos complement, ones complement, and sign-magnitude and shows how to covert between them.

The last code to be considered in this section is *binary-coded-decimal (BCD)*, where each base-10 digit (0 to 9) in a decimal number is represented as the corresponding 4-bit straight binary word as shown in Figure 5.11. The minimum digit 0 is represented as 0000, and the digit 9 by 1001. This code is relatively inefficient, since only 10 of the 16 code states for each decade are used. It is, however, a very useful code for interfacing to decimal displays such as in digital voltmeters.

To Convert From → To ↓	Sign Magnitude	2's Complement	Offset Binary	1's Complement
Sign Magnitude	No Change	If MSB = 1, complement other bits, add 00...01	Complement MSB If new MSB = 1, complement other bits, add 00...01	If MSB = 1, complement other bits
2's Complement	If MSB = 1, complement other bits, add 00...01	No Change	Complement MSB	If MSB = 1, add 00...01
Offset binary	Complement MSB If new MSB = 0 complement other bits, add 00...01	Complement MSB	No Change	Complement MSB If new MSB = 0, add 00...01
1's Complement	If MSB = 1, complement other bits	If MSB = 1, add 11...11	Complement MSB If new MSB = 1, add 11...11	No Change

Figure 5.10: *Relationships among Bipolar Codes*

BASE 10 NUMBER	SCALE	+10V FS	DECADE 1	DECADE 2	DECADE 3	DECADE 4
+15	+FS – 1LSB = +15/16 FS	9.375	1 0 0 1	0 0 1 1	0 1 1 1	0 1 0 1
+14	+7/8 FS	8.750	1 0 0 0	0 1 1 1	0 1 0 1	0 0 0 0
+13	+13/16 FS	8.125	1 0 0 0	0 0 0 1	0 0 1 0	0 1 0 1
+12	+3/4 FS	7.500	0 1 1 1	0 1 0 1	0 0 0 0	0 0 0 0
+11	+11/16 FS	6.875	0 1 1 0	1 0 0 0	0 1 1 1	0 1 0 1
+10	+5/8 FS	6.250	0 1 1 0	0 0 1 0	0 1 0 1	0 0 0 0
+9	+9/16 FS	5.625	0 1 0 1	0 1 1 0	0 0 1 0	0 1 0 1
+8	+1/2 FS	5.000	0 1 0 1	0 0 0 0	0 0 0 0	0 0 0 0
+7	+7/16 FS	4.375	0 1 0 0	0 0 1 1	0 1 1 1	0 1 0 1
+6	+3/8 FS	3.750	0 0 1 1	0 1 1 1	0 1 0 1	0 0 0 0
+5	+5/16 FS	3.125	0 0 1 1	0 0 0 1	0 0 1 0	0 1 0 1
+4	+1/4 FS	2.500	0 0 1 0	0 1 0 1	0 0 0 0	0 0 0 0
+3	+3/16 FS	1.875	0 0 0 1	1 0 0 0	0 1 1 1	0 1 0 1
+2	+1/8 FS	1.250	0 0 0 1	0 0 1 0	0 1 0 1	0 0 0 0
+1	1LSB = +1/16 FS	0.625	0 0 0 0	0 1 1 0	0 0 1 0	0 1 0 1
0	0	0.000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

Figure 5.11: *Binary Coded Decimal (BCD) Code*

Complementary Codes

Some forms of data converters (for example, early DACs using monolithic NPN quad current switches), require standard codes such as natural binary or BCD, but with all bits represented by their complements. Such codes are called *complementary codes*. All the codes discussed thus far have complementary codes which can be obtained by this method.

In a 4-bit complementary-binary converter, 0 is represented by 1111, half-scale by 0111, and FS – 1 LSB by 0000. In practice, the complementary code can usually be obtained by using the complementary output of a register rather than the true output, since both are available.

Sometimes the complementary code is useful in inverting the analog output of a DAC. Today many DACs provide differential outputs which allow the polarity inversion to be accomplished without modifying the input code. Similarly, many ADCs provide differential logic inputs which can be used to accomplish the polarity inversion.

DAC and ADC Static Transfer Functions and DC Errors

The most important thing to remember about both DACs and ADCs is that either the input or output is digital, and therefore the signal is quantized. That is, an N-bit word represents one of 2^N possible states, and therefore an N-bit DAC (with a fixed reference) can have only 2^N possible analog outputs, and an N-bit ADC can have only 2^N possible digital outputs. As previously discussed, the analog signals will generally be voltages or currents.

The resolution of data converters may be expressed in several different ways: the weight of the least significant bit (LSB), parts per million of full-scale (ppm FS), millivolts (mV), etc. Different devices (even from the same manufacturer) will be specified differently, so converter users must learn to translate between the different types of specifications if they are to compare devices successfully. The size of the least significant bit for various resolutions is shown in Figure 5.12.

RESOLUTION N	2^N	VOLTAGE (10V FS)	ppm FS	% FS	dB FS
2-bit	4	2.5 V	250,000	25	– 12
4-bit	16	625 mV	62,500	6.25	– 24
6-bit	64	156 mV	15,625	1.56	– 36
8-bit	256	39.1 mV	3,906	0.39	– 48
10-bit	1,024	9.77 mV (10 mV)	977	0.098	– 60
12-bit	4,096	2.44 mV	244	0.024	– 72
14-bit	16,384	610 μV	61	0.0061	– 84
16-bit	65,536	153 μV	15	0.0015	– 96
18-bit	262,144	38 μV	4	0.0004	– 108
20-bit	1,048,576	9.54 μV (10 μV)	1	0.0001	– 120
22-bit	4,194,304	2.38 μV	0.24	0.000024	– 132
24-bit	16,777,216	596 nV*	0.06	0.000006	– 144

***600nV is the Johnson Noise in a 10kHz BW of a 2.2kΩ Resistor @ 25°C**

Remember: 10-bits and 10V FS yields an LSB of 10mV, 1000ppm, or 0.1%.
All other values may be calculated by powers of 2.

Figure 5.12: *Quantization: The Size of a Least Significant Bit (LSB)*

Before we can consider the various architectures used in data converters, it is necessary to consider the performance to be expected, and the specifications which are important. The following sections will consider the definition of errors and specifications used for data converters. This is important in understanding the strengths and weaknesses of different ADC/DAC architectures.

Figure 5.13 shows the ideal transfer characteristics for a 3-bit unipolar DAC and a 3-bit unipolar ADC. In a DAC, both the input and the output are quantized, and the graph consists of eight points—while it is reasonable to discuss the line through these points, it is very important to remember that the actual transfer characteristic is *not* a line, but a number of discrete points.

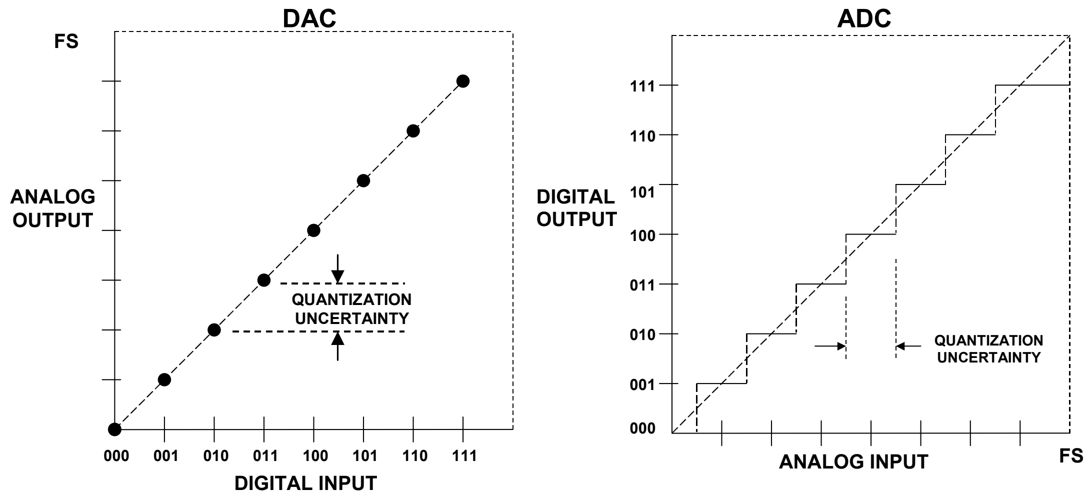


Figure 5.13: *Transfer Functions for Ideal 3-Bit DAC and ADC*

The input to an ADC is analog and is not quantized, but its output is quantized. The transfer characteristic therefore consists of eight horizontal steps. When considering the offset, gain, and linearity of an ADC we consider the line joining the midpoints of these steps—often referred to as the *code centers*.

For both DACs and ADCs, digital full-scale (all 1s) corresponds to 1 LSB below the analog full-scale (FS). The (ideal) ADC transitions take place at ½ LSB above zero, and thereafter every LSB, until 1½ LSB below analog full-scale. Since the analog input to an ADC can take any value, but the digital output is quantized, there may be a difference of up to ½ LSB between the actual analog input and the exact value of the digital output. This is known as the *quantization error* or *quantization uncertainty* as shown in Figure 5.15. In ac (sampling) applications this quantization error gives rise to *quantization noise* which will be discussed in Section 5.3 of this chapter.

As previously discussed, there are many possible digital coding schemes for data converters: *straight binary*, *offset binary*, *1s complement*, *2s complement*, *sign magnitude*, *gray code*, *BCD,* and others. This section, being devoted mainly to the *analog* issues surrounding data converters, will use simple *binary* and *offset binary* in its examples and will not consider the merits and disadvantages of these, or any other forms of digital code.

The examples in Figure 5.13 use *unipolar* converters, whose analog port has only a single polarity. These are the simplest type, but *bipolar* converters are generally more useful in real-world applications. There are two types of bipolar converters: the simpler is merely a unipolar converter with an accurate 1 MSB of negative offset (and many converters are arranged so that this offset may be switched in and out so that they can be used as either unipolar or bipolar converters at will), but the other, known as a *sign-magnitude* converter is more complex, and has N bits of magnitude information and an additional bit which corresponds to the sign of the analog signal. Sign-magnitude DACs are quite rare, and sign-magnitude ADCs are found mostly in digital voltmeters (DVMs). The unipolar, offset binary, and sign-magnitude representations are shown in Figure 5.14.

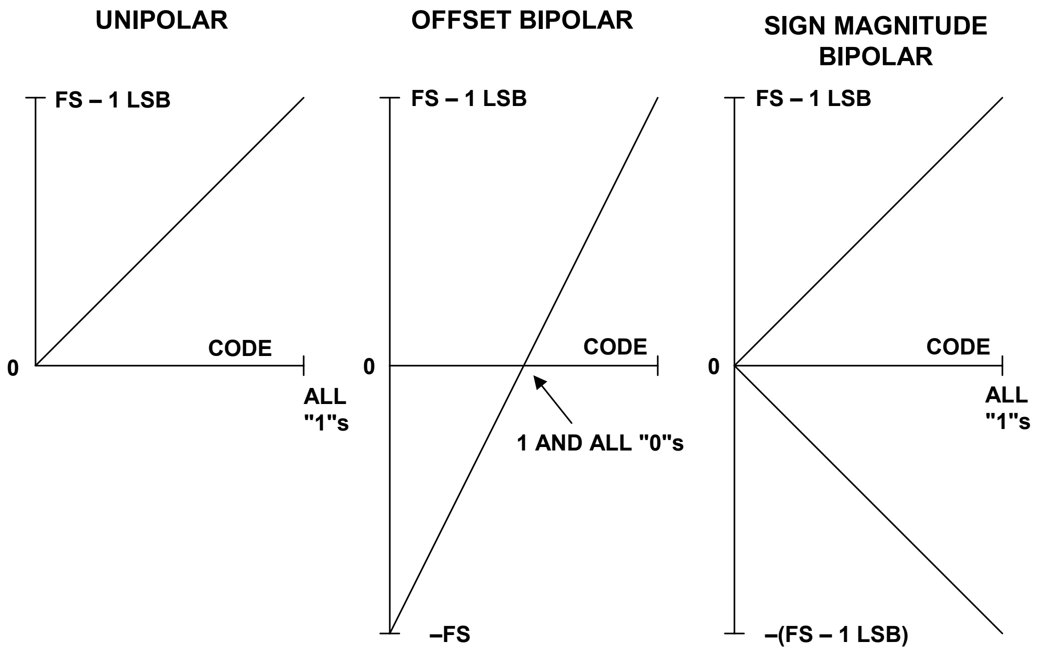


Figure 5.14: *Unipolar and Bipolar Converters*

The four dc errors in a data converter are *offset error*, *gain error*, and two types of *linearity error (differential and integral)*. Offset and gain errors are analogous to offset and gain errors in amplifiers as shown in Figure 5.15 for a bipolar input range. (Though offset error and zero error, which are identical in amplifiers and unipolar data converters, are not identical in bipolar converters and should be carefully distinguished.). The transfer characteristics of both DACs and ADCs may be expressed as D = K + GA, where D is the digital code, A is the analog signal, and K and G are constants. In a unipolar converter, K is zero, and in an offset bipolar converter, it is –1 MSB. The offset error is the amount by which the actual value of K differs from its ideal value.

The gain error is the amount by which G differs from its ideal value, and is generally expressed as the percentage difference between the two, although it may be defined as the gain error contribution (in mV or LSB) to the total error at full-scale. These errors can usually be trimmed by the data converter user. Note, however, that amplifier offset is trimmed at zero input, and then the gain is trimmed near to full-scale. The trim algorithm for a bipolar data converter is not so straightforward.

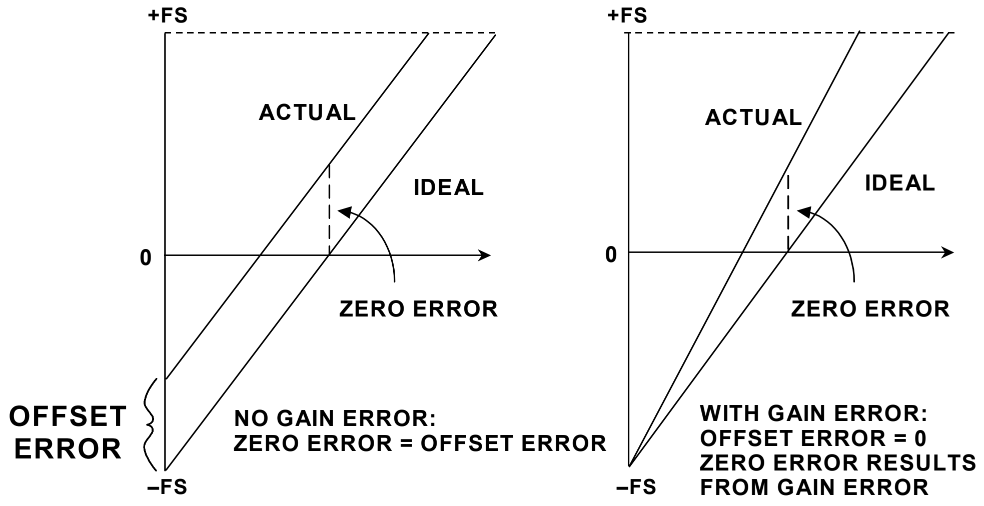


Figure 5.15: *Data Converter Offset and Gain Error*

The integral linearity error of a converter is also analogous to the linearity error of an amplifier, and is defined as the maximum deviation of the actual transfer characteristic of the converter from a straight line, and is generally expressed as a percentage of full-scale (but may be given in LSBs). For an ADC, the most popular convention is to draw the straight line through the midpoints of the codes, or the code centers. There are two common ways of choosing the straight line: *endpoint* and *best straight line* as shown in Figure 5.16.

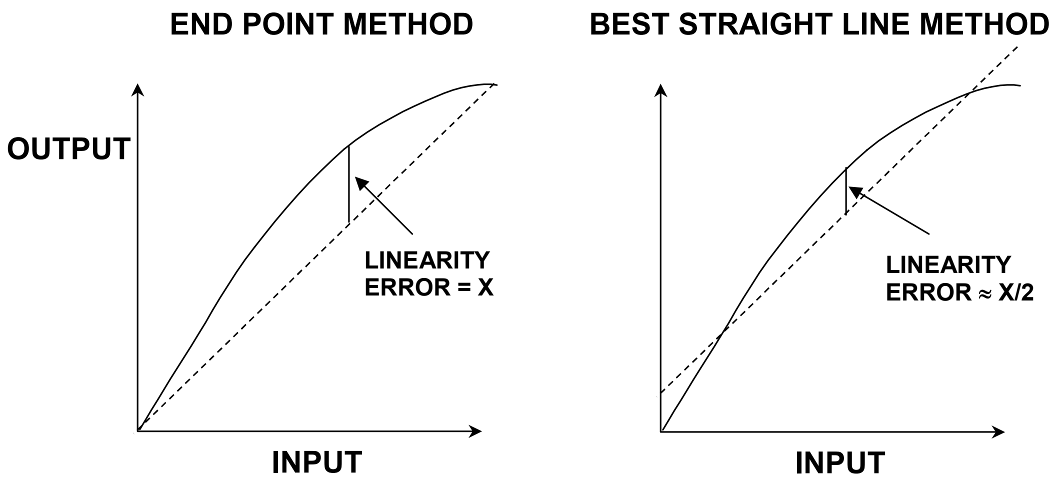


Figure 5.16: *Method of Measuring Integral Linearity Errors (Same Converter on Both Graphs)*

In the *endpoint* system, the deviation is measured from the straight line through the origin and the full-scale point (after gain adjustment). This is the most useful integral linearity measurement for measurement and control applications of data converters (since error budgets depend on deviation from the ideal transfer characteristic, not from some arbitrary "best fit"), and is the one normally adopted by Analog Devices, Inc.

The *best straight line*, however, does give a better prediction of distortion in ac applications, and also gives a lower value of "linearity error" on a data sheet. The best fit straight line is drawn through the transfer characteristic of the device using standard curve fitting techniques, and the maximum deviation is measured from this line. In general, the integral linearity error measured in this way is only 50% of the value measured by endpoint methods. This makes the method good for producing impressive data sheets, but it is less useful for error budget analysis. For ac applications, it is even better to specify distortion than dc linearity, so it is rarely necessary to use the best straight line method to define converter linearity.

The other type of converter nonlinearity is *differential nonlinearity* (DNL). This relates to the linearity of the code transitions of the converter. In the ideal case, a change of 1 LSB in digital code corresponds to a change of exactly 1 LSB of analog signal. In a DAC, a change of 1 LSB in digital code produces exactly 1 LSB change of analog output, while in an ADC there should be exactly 1 LSB change of analog input to move from one digital transition to the next. Differential linearity error is defined as the maximum amount of deviation of any quantum (or LSB change) in the entire transfer function from its ideal size of 1 LSB.

Where the change in analog signal corresponding to 1 LSB digital change is more or less than 1 LSB, there is said to be a DNL error. The DNL error of a converter is normally defined as the maximum value of DNL to be found at any transition across the range of the converter. Figure 5.17 shows the nonideal transfer functions for a DAC and an ADC and shows the effects of the DNL error.

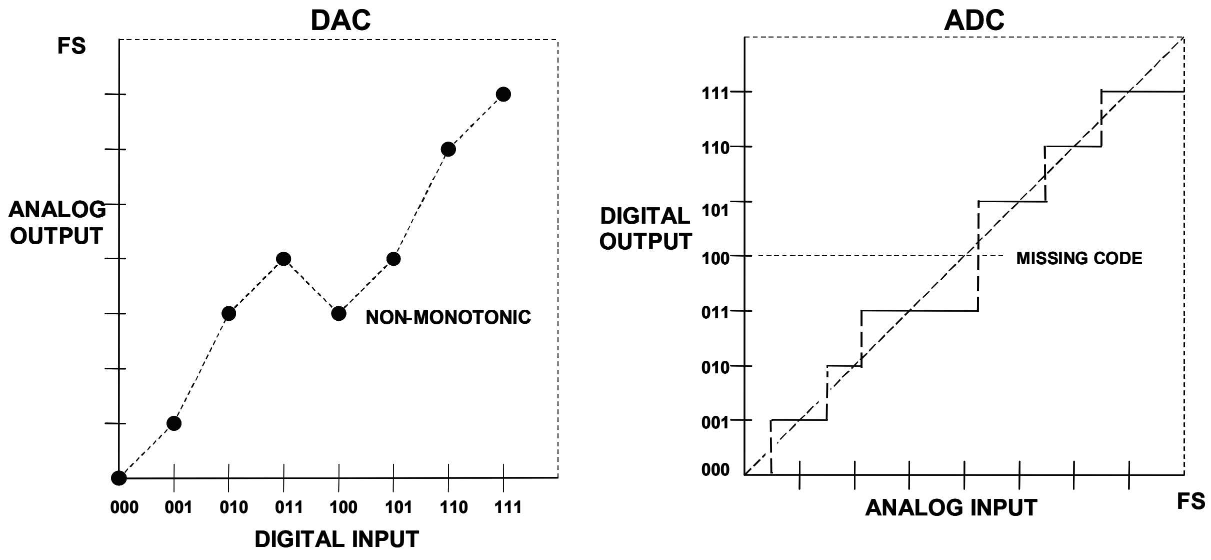


Figure 5.17: *Transfer Functions for Non-Ideal 3-Bit DAC and ADC*

The DNL of a DAC is examined more closely in Figure 5.18. If the DNL of a DAC is less than –1 LSB at any transition, the DAC is *nonmonotonic* i.e., its transfer characteristic contains one or more localized maxima or minima. A DNL greater than +1 LSB does not cause nonmonotonicity, but is still undesirable. In many DAC applications (especially closed-loop systems where nonmonotonicity can change negative feedback to positive feedback), it is critically important that DACs are monotonic. DAC monotonicity is often explicitly specified on data sheets, although if the DNL is guaranteed to be less than 1 LSB (i.e., |DNL| ≤ 1 LSB) then the device must be monotonic, even without an explicit guarantee.

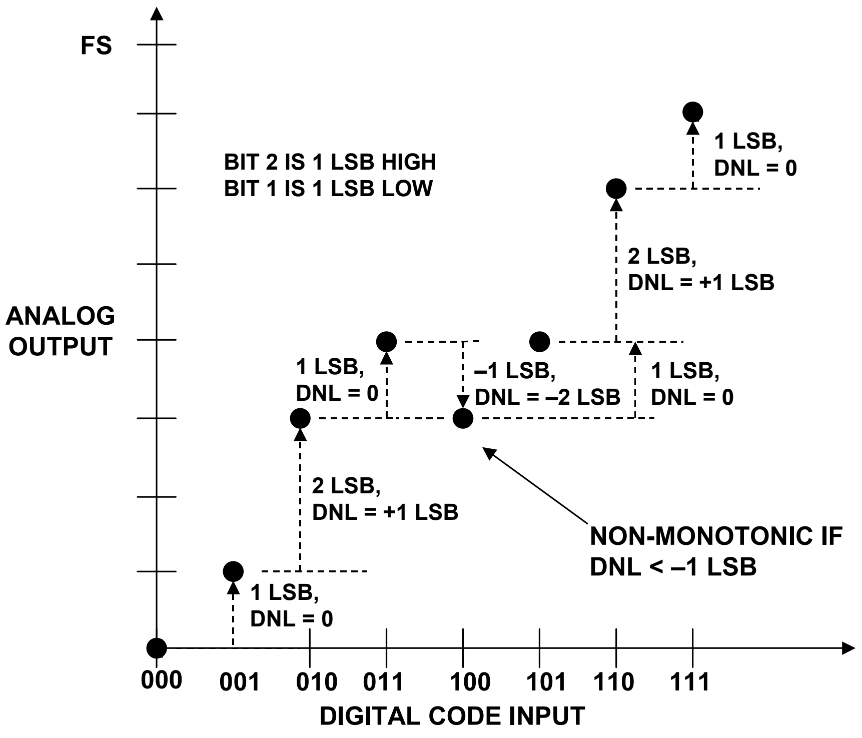


Figure 5.18: *Details of DAC Differential Nonlinearity*

In Figure 5.19, the DNL of an ADC is examined more closely on an expanded scale. ADCs can be nonmonotonic, but a more common result of excess DNL in ADCs is *missing codes*. Missing codes in an ADC are as objectionable as nonmonotonicity in a DAC. Again, they result from DNL < –1 LSB.

Not only can ADCs have missing codes, they can also be nonmonotonic as shown in Figure 5.22. As in the case of DACs, this can present major problems—especially in servo applications.

In a DAC, there can be no missing codes—each digital input word will produce a corresponding analog output. However, DACs can be nonmonotonic as previously discussed. In a straight binary DAC, the most likely place a nonmonotonic condition can develop is at midscale between the two codes: 011…11 and 100…00. If a nonmonotonic condition occurs here, it is generally because the DAC is not properly calibrated or trimmed. A successive approximation ADC with an internal nonmonotonic DAC will generally produce missing codes but remain monotonic. However it is possible for an ADC to be nonmonotonic—again depending on the particular conversion architecture. Figure 5.20 shows the transfer function of an ADC which is nonmonotonic and has a missing code.

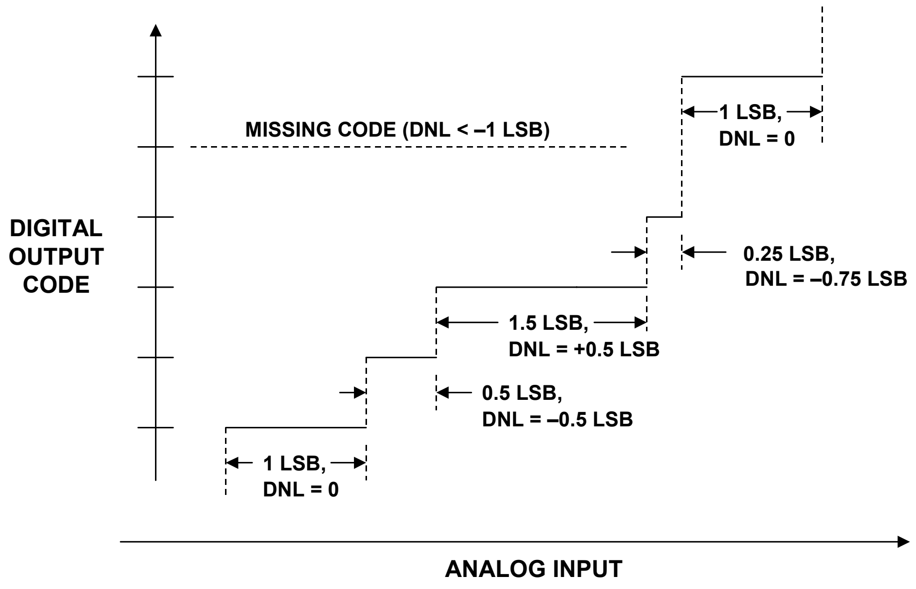


Figure 5.19: *Details of ADC Differential Nonlinearity*

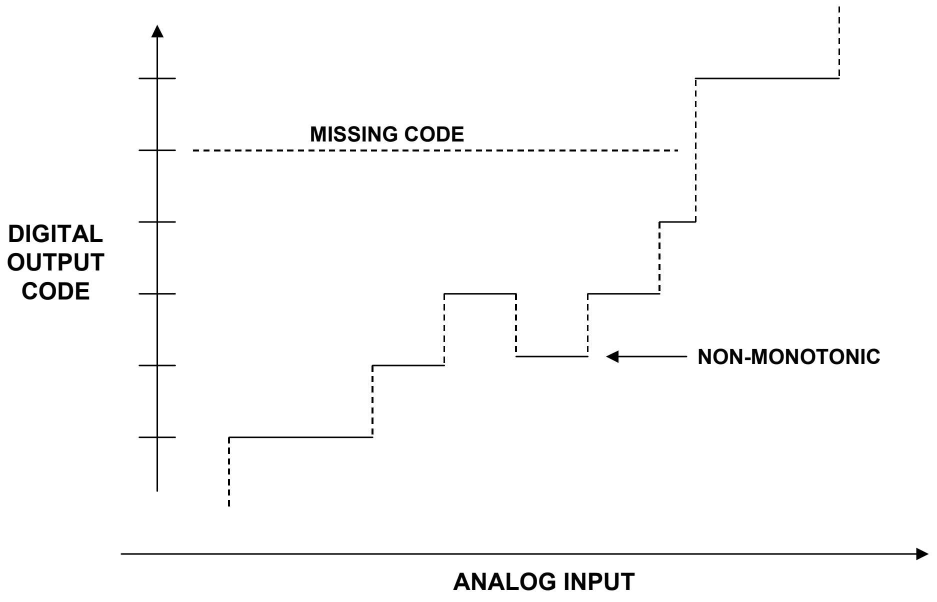


Figure 5.20: *Nonmonotonic ADC with Missing Code*

ADCs which use the *subranging* architecture divide the input range into a number of coarse segments, and each coarse segment is further divided into smaller segments—and ultimately the final code is derived. This process is described in more detail in Chapter 3 of this book. An improperly trimmed subranging ADC may exhibit nonmonotonicity, wide codes, or missing codes at the subranging points as shown in Figure 5.21 A, B, and C, respectively. This type of ADC should be trimmed so that drift due to aging or temperature produces wide codes at the sensitive points rather than nonmonotonic or missing codes.

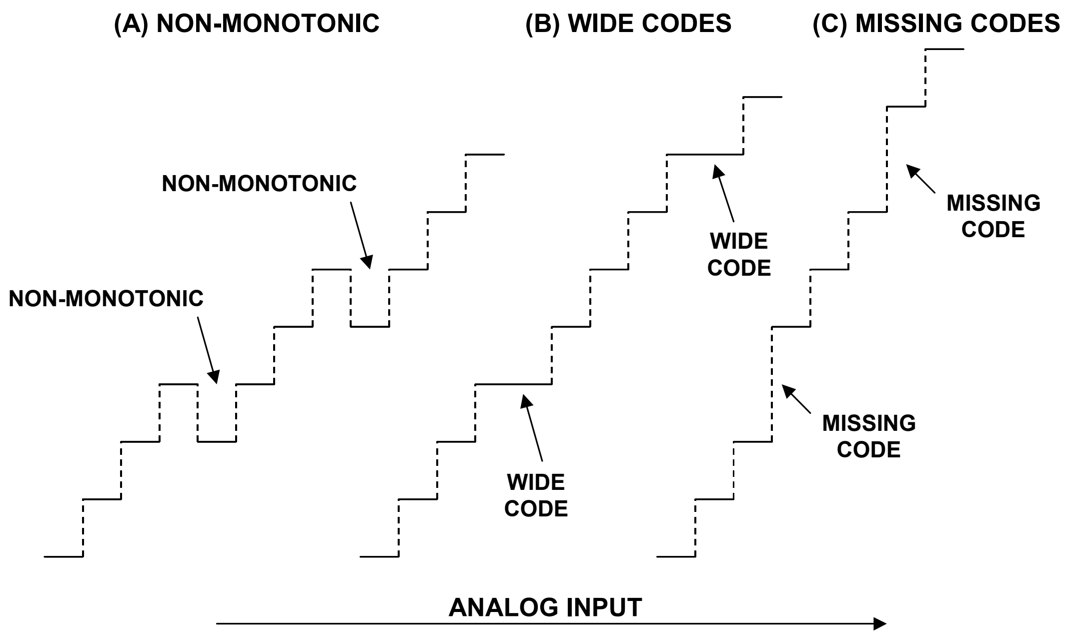


Figure 5.21: *Errors Associated with Improperly Trimmed Subranging ADC*

Defining missing codes is more difficult than defining nonmonotonicity. All ADCs suffer from some inherent transition noise as shown in Figure 5.22 (think of it as the flicker between adjacent values of the last digit of a DVM). As resolutions and bandwidths become higher, the range of input over which transition noise occurs may approach, or even exceed, 1 LSB. High resolution wideband ADCs generally have internal noise sources which can be reflected to the input as effective input noise summed with the signal. The effect of this noise, especially if combined with a negative DNL error, may be that there are some (or even all) codes where transition noise is present for the whole range of inputs. There are therefore some codes for which there is *no* input which will *guarantee* that code as an output, although there may be a range of inputs which will *sometimes* produce that code.

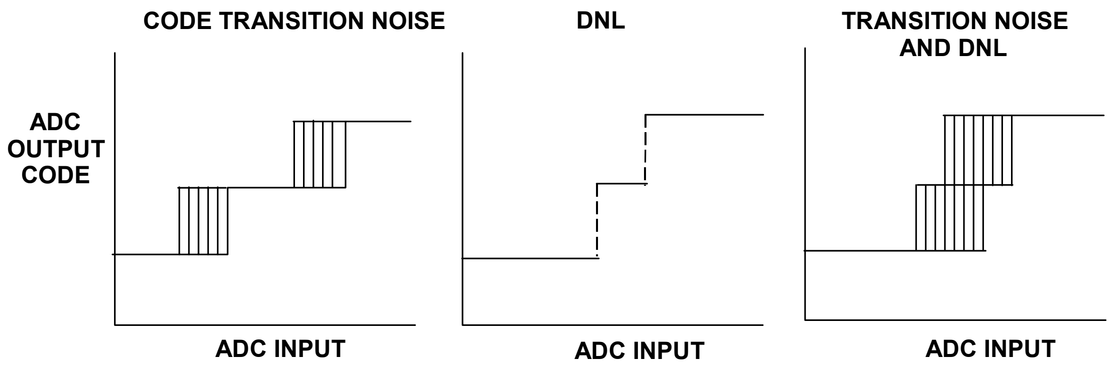


Figure 5.22: *Combined Effects of Code Transition Noise and DNL*

For low resolution ADCs, it may be reasonable to define *no missing codes* as a combination of transition noise and DNL which guarantees some level (perhaps 0.2 LSB) of noise-free code for all codes. However, this is impossible to achieve at the very high resolutions achieved by modern sigma-delta ADCs, or even at lower resolutions in wide bandwidth sampling ADCs. In these cases, the manufacturer must define noise levels and resolution in some other way. Which method is used is less important, but the data sheet should contain a clear definition of the method used and the performance to be expected.

The discussion thus far has not dealt with the most important dc specifications associated with data converters. Other less important specifications require only a definition. There are also AC specifications. Converter specifications are covered in Chapter 3.

REFERENCES: CODING AND QUANTIZATION

1. K. W. Cattermole, **Principles of Pulse Code Modulation**, American Elsevier Publishing Company, Inc., 1969, New York NY, ISBN 444-19747-8. *(An excellent tutorial and historical discussion of data conversion theory and practice, oriented towards PCM, but covers practically all aspects. This one is a must for anyone serious about data conversion! Try internet secondhand bookshops such as http://www.abebooks.com for starters).*

2. Frank Gray, "Pulse Code Communication," **U.S. Patent 2,632,058**, filed November 13, 1947, issued March 17, 1953. *(Detailed patent on the Gray code and its application to electron beam coders).*

3. R. W. Sears, "Electron Beam Deflection Tube for Pulse Code Modulation," **Bell System Technical Journal**, Vol. 27, pp. 44-57, Jan. 1948. *(Describes an electron-beam deflection tube 7-bit, 100 kSPS flash converter for early experimental PCM work).*

4. J. O. Edson and H. H. Henning, "Broadband Codecs for an Experimental 224 Mb/s PCM Terminal," **Bell System Technical Journal**, Vol. 44, pp. 1887-1940, Nov. 1965. *(Summarizes experiments on ADCs based on the electron tube coder as well as a bit-per-stage Gray code 9-bit solid state ADC. The electron beam coder was 9 bits at 12 MSPS, and represented the fastest of its type).*

5. Dan Sheingold, **Analog-Digital Conversion Handbook, 3rd Edition**, Analog Devices and Prentice-Hall, 1986, ISBN-0-13-032848-0. *(The defining and classic book on data conversion).*

SECTION 5.2: SAMPLING THEORY

This section discusses the basics of sampling theory. A block diagram of a typical real-time sampled data system is shown in Figure 5.23. Prior to the actual analog-to-digital conversion, the analog signal usually passes through some sort of signal conditioning circuitry which performs such functions as amplification, attenuation, and filtering. The low-pass/band-pass filter is required to remove unwanted signals outside the bandwidth of interest and prevent aliasing.

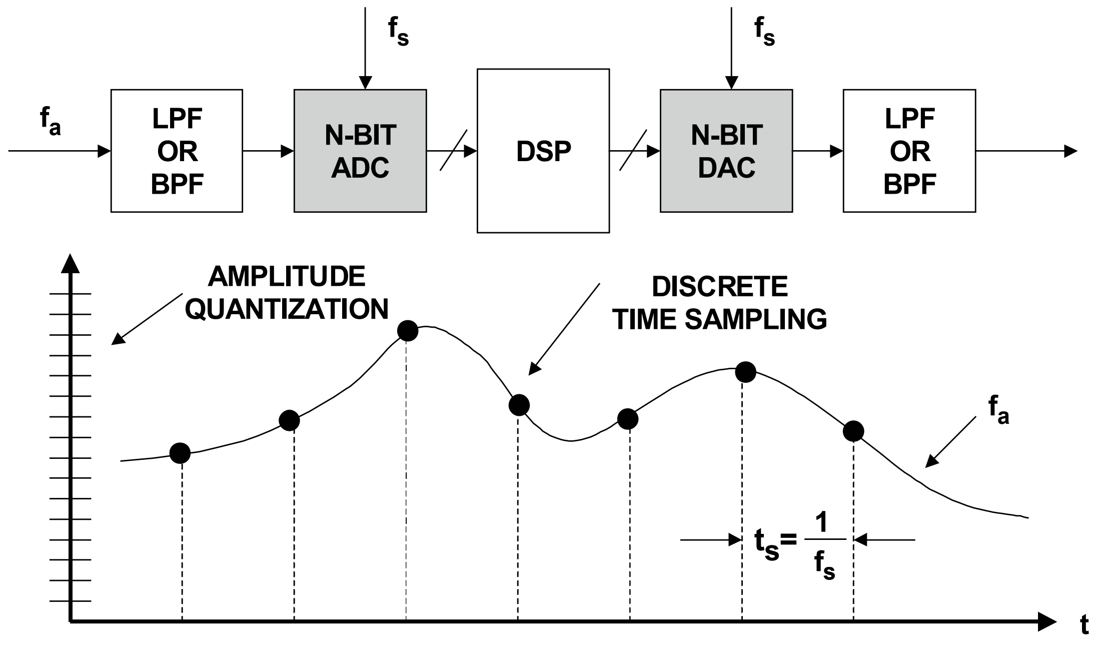


Figure 5.23: *Sampled Data System*

The system shown in Figure 5.23 is a real-time system, i.e., the signal to the ADC is continuously sampled at a rate equal to f_s, and the ADC presents a new sample to the DSP at this rate. In order to maintain real-time operation, the DSP must perform all its required computation within the sampling interval, $1/f_s$, and present an output sample to the DAC before arrival of the next sample from the ADC. An example of a typical DSP function would be a digital filter.

In the case of FFT analysis, a block of data is first transferred to the DSP memory. The FFT is calculated at the same time a new block of data is transferred into the memory, in order to maintain real-time operation. The DSP must calculate the FFT during the data transfer interval so it will be ready to process the next block of data.

Note that the DAC is required only if the DSP data must be converted back into an analog signal (as would be the case in a voice-band or audio application, for example). There are many applications where the signal remains entirely in digital format after the initial A/D conversion. Similarly, there are applications where the DSP is solely responsible for generating the signal to the DAC, such as in CD player electronics. If a DAC is used, it must be followed by an analog anti-imaging filter to remove the image frequencies. Finally, there are slower speed industrial process control systems where sampling rates are much lower—regardless of the system, the fundamentals of sampling theory still apply.

There are two key concepts involved in the actual analog-to-digital and digital-to-analog conversion process: *discrete time sampling* and *finite amplitude resolution due to quantization.* An understanding of these concepts is vital to data converter applications.

The Need for a Sample-and-Hold Amplifier Function

The generalized block diagram of a sampled data system shown in Figure 5.23 assumes some type of ac signal at the input. It should be noted that this does not necessarily have to be so, as in the case of modern digital voltmeters (DVMs) or ADCs optimized for dc measurements, but for this discussion assume that the input signal has some upper frequency limit f_a.

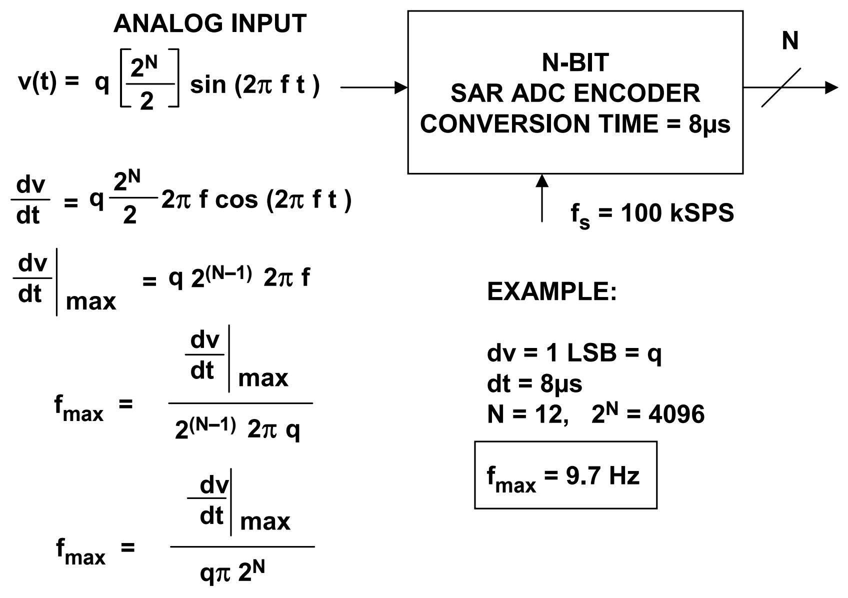

ANALOG INPUT
$v(t) = q \left[\frac{2^N}{2}\right] \sin(2\pi f t)$
N-BIT
SAR ADC ENCODER
CONVERSION TIME = 8µs
N
f_s = 100 kSPS
$\frac{dv}{dt} = q \frac{2^N}{2} 2\pi f \cos(2\pi f t)$
$\left.\frac{dv}{dt}\right|_{max} = q\, 2^{(N-1)}\, 2\pi f$
$f_{max} = \frac{\left.\frac{dv}{dt}\right|_{max}}{2^{(N-1)}\, 2\pi q}$
$f_{max} = \frac{\left.\frac{dv}{dt}\right|_{max}}{q\pi\, 2^N}$
EXAMPLE:
dv = 1 LSB = q
dt = 8µs
N = 12, 2^N = 4096
f_{max} = 9.7 Hz

Figure 5.24: *Input Frequency Limitations of Nonsampling ADC (Encoder)*

Most ADCs today have a built-in sample-and-hold function (SHA), thereby allowing them to process ac signals. This type of ADC is referred to as a *sampling ADC*. However, many early ADCs, such as the Analog Devices' industry-standard AD574, were not of the sampling type, but simply *encoders* as shown in Figure 5.24. If the input signal to a SAR ADC (assuming no SHA function) changes by more than 1 LSB during the conversion time (8 µs in the example), the output data can have large errors, depending on the location of the code. Most ADC architectures are subject to this type of error—

some more, some less—with the possible exception of flash converters having well-matched comparators.

Assume that the input signal to the encoder is a sine wave with a full-scale amplitude ($q2^N/2$), where q is the weight of 1 LSB.

$$v(t) = q\,(2^N/2)\sin(2\pi f t). \qquad \text{Eq. 5.1}$$

Taking the derivative:

$$dv/dt = q\,2\pi f\,(2^N/2)\cos(2\pi f t). \qquad \text{Eq. 5.2}$$

The maximum rate of change is therefore:

$$dv/dt\,|_{max} = q\,2\pi f\,(2^N/2). \qquad \text{Eq. 5.3}$$

Solving for f:

$$f = (dv/dt\,|_{max})/(q\,\pi\,2^N). \qquad \text{Eq. 5.4}$$

If N = 12, and 1 LSB change (dv = q) is allowed during the conversion time (dt = 8 µs), then the equation can be solved for f_{max}, the maximum full-scale signal frequency that can be processed without error:

$$f_{max} = 9.7\ \text{Hz}.$$

This implies any input frequency greater than 9.7 Hz is subject to conversion errors, even though a sampling frequency of 100 kSPS is possible with the 8 µs ADC (this allows an extra 2 µs interval for an external SHA to re-acquire the signal after coming out of the hold mode).

To process ac signals, a sample-and-hold function is added. The ideal SHA is simply a switch driving a hold capacitor followed by a high input impedance buffer. The input impedance of the buffer must be high enough so that the capacitor is discharged by less than 1 LSB during the hold time. The SHA samples the signal in the *sample* mode, and holds the signal constant during the *hold* mode. The timing is adjusted so that the encoder performs the conversion during the hold time. A sampling ADC can therefore process fast signals—the upper frequency limitation is determined by the SHA aperture jitter, bandwidth, distortion, etc., not the encoder. In the example shown, a good sample-and-hold could acquire the signal in 2 µs, allowing a sampling frequency of 100 kSPS, and the capability of processing input frequencies up to 50 kSPS. A complete discussion of the SHA function including these specifications follows later in this chapter.

The Nyquist Criteria

A continuous analog signal is sampled at discrete intervals, $t_s = 1/f_s$, which must be carefully chosen to ensure an accurate representation of the original analog signal. It is clear that the more samples taken (faster sampling rates), the more accurate the digital representation, but if fewer samples are taken (lower sampling rates), a point is reached where critical information about the signal is actually lost. The mathematical basic of sampling was set forth by Harry Nyquist of Bell Telephone Laboratories in two classic papers published in 1924 and 1928, respectively. (See References 1 and 2). Nyquist's original work was shortly supplemented by R. V. L. Hartley (Reference 3). These papers formed the basis for the PCM work to follow in the 1940s, and in 1948, Claude Shannon wrote his classic paper on communication theory (Reference 4).

Simply stated, the Nyquist criteria require that the sampling frequency be at least twice the highest frequency contained in the signal, or information about the signal will be lost. If the sampling frequency is less than twice the maximum analog signal frequency, a phenomenon known as aliasing will occur.

- **A signal with a *maximum BANDWIDTH* f_a must be sampled at a rate $f_a > 2\ f_a$ or information about the signal will be lost because of aliasing**
- **Aliasing occurs whenever $f_a < 2\ f_a$**
- **The concept of aliasing is widely used I communications applications such as direct IF-to-digital conversion**
- **A signal which has frequencyncomponents between f_a and f_b must be sampled at least at a rate $f_a > 2\ (f_b – f_a)$ to prevent alias components from overlapping the signal frequncies.**

Figure 5.25: *Nyquist's Criteria*

In order to understand the implications of *aliasing* in both the time and frequency domain, first consider case of a time domain representation of a single tone sine wave sampled as shown in Figure 5.26. In this example, the sampling frequency f_s is not at least $2f_a$, but only slightly more than the analog input frequency f_a—the Nyquist criteria is violated. Notice that the pattern of the actual samples produces an *aliased* sine wave at a lower frequency equal to $f_s – f_a$.

The corresponding frequency domain representation of this scenario is shown in Figure 5.27B. Now consider the case of a single frequency sine wave of frequency f_a sampled at a frequency f_s by an ideal impulse sampler (see Figure 5.27A). Also assume that $f_s > 2f_a$ as shown. The frequency-domain output of the sampler shows *aliases* or *images* of the original signal around every multiple of f_s, i.e. at frequencies equal to $|\pm Kf_s \pm f_a|$, K = 1, 2, 3, 4, ...

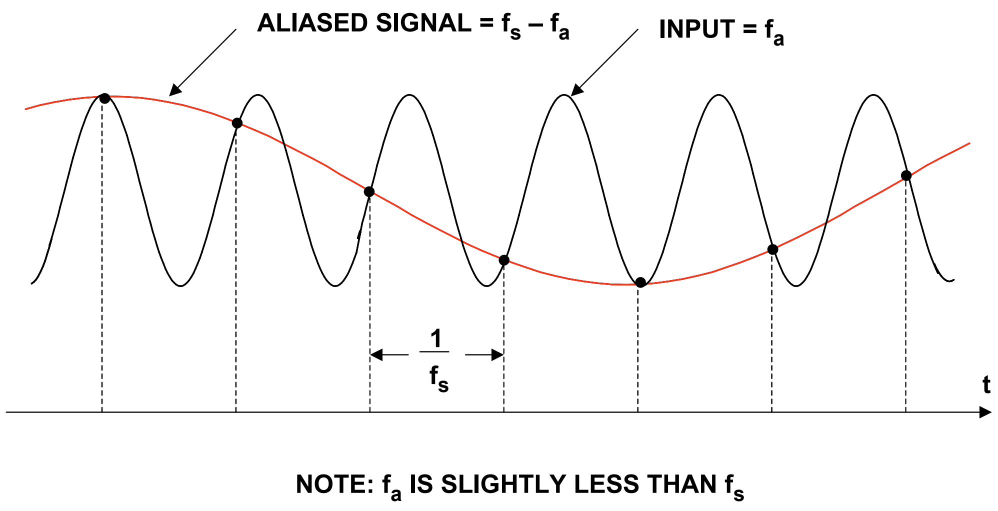

ALIASED SIGNAL = $f_s - f_a$
INPUT = f_a
$\frac{1}{f_s}$
t
NOTE: f_a IS SLIGHTLY LESS THAN f_s

Figure 5.26: *Aliasing in the Time Domain*

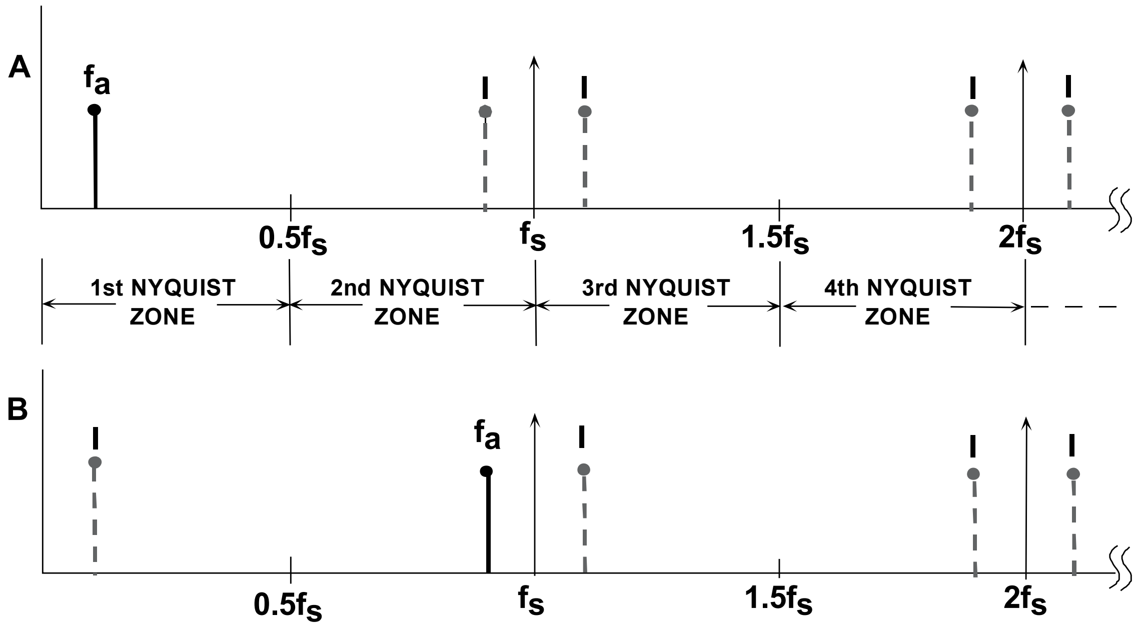


Figure 5.27: *Analog Signal f_a Sampled @ f_s Using Ideal Sampler Has Images (Aliases) at $|\pm Kf_s \pm f_a|$, K = 1, 2, 3, …*

The *Nyquist* bandwidth is defined to be the frequency spectrum from dc to $f_s/2$. The frequency spectrum is divided into an infinite number of *Nyquist zones*, each having a width equal to 0.5 f_s as shown. In practice, the ideal sampler is replaced by an ADC followed by an FFT processor. The FFT processor only provides an output from dc to $f_s/2$, i.e., the signals or aliases which appear in the first Nyquist zone.

Now consider the case of a signal which is outside the first Nyquist zone (Figure 5.27B). The signal frequency is only slightly less than the sampling frequency, corresponding to the condition shown in the time domain representation in Figure 5.26. Notice that even though the signal is outside the first Nyquist zone, its image (or *alias*), $f_s – f_a$, falls inside. Returning to Figure 5.27A, it is clear that if an unwanted signal appears at any of the image frequencies of f_a, it will also occur at f_a, thereby producing a spurious frequency component in the first Nyquist zone.

This is similar to the analog mixing process and implies that some filtering ahead of the sampler (or ADC) is required to remove frequency components which are outside the Nyquist bandwidth, but whose aliased components fall inside it. The filter performance will depend on how close the out-of-band signal is to $f_s/2$ and the amount of attenuation required.

Baseband Antialiasing Filters

Baseband sampling implies that the signal to be sampled lies in the first Nyquist zone. It is important to note that with no input filtering at the input of the ideal sampler, *any frequency component (either signal or noise) that falls outside the Nyquist bandwidth in any Nyquist zone will be aliased back into the first Nyquist zone*. For this reason, an antialiasing filter is used in almost all sampling ADC applications to remove these unwanted signals.

Properly specifying the antialiasing filter is important. The first step is to know the characteristics of the signal being sampled. Assume that the highest frequency of interest is f_a. The antialiasing filter passes signals from dc to f_a while attenuating signals above f_a.

Assume that the corner frequency of the filter is chosen to be equal to f_a. The effect of the finite transition from minimum to maximum attenuation on system dynamic range is illustrated in Figure 5.28A.

Assume that the input signal has full-scale components well above the maximum frequency of interest, f_a. The diagram shows how full-scale frequency components above $f_s – f_a$ are aliased back into the bandwidth dc to f_a. These aliased components are indistinguishable from actual signals and therefore limit the dynamic range to the value on the diagram which is shown as *DR*.

Some texts recommend specifying the antialiasing filter with respect to the Nyquist frequency, $f_s/2$, but this assumes that the signal bandwidth of interest extends from dc to $f_s/2$ which is rarely the case. In the example shown in Figure 5.28A, the aliased components between f_a and $f_s/2$ are not of interest and do not limit the dynamic range.

The antialiasing filter transition band is therefore determined by the corner frequency f_a, the stopband frequency $f_s – f_a$, and the desired stopband attenuation, DR. The required system dynamic range is chosen based on the requirement for signal fidelity.

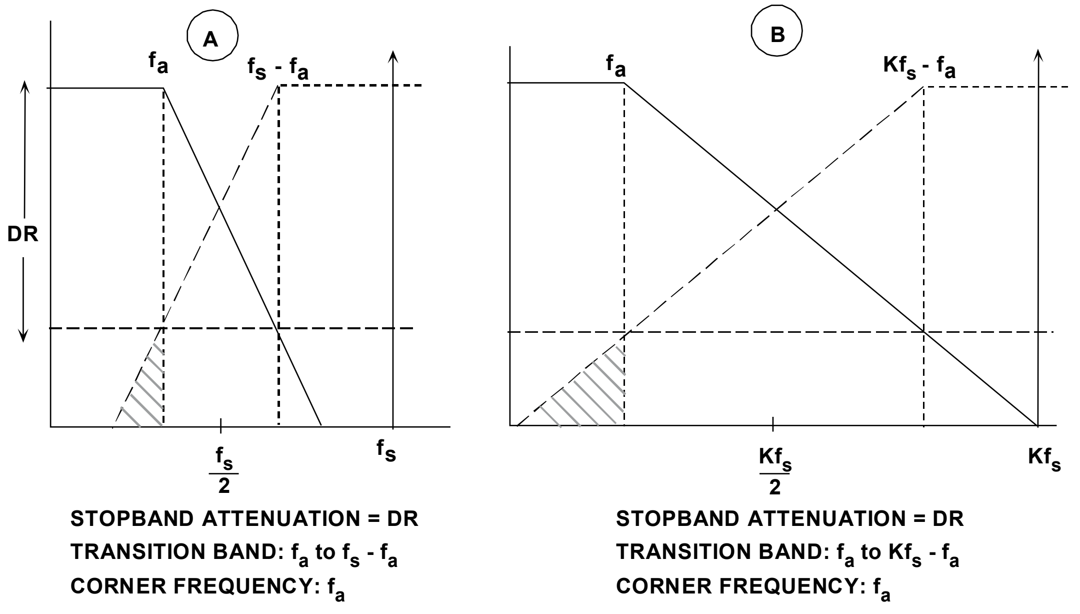

A
f_a
$f_s - f_a$
DR
$\frac{f_s}{2}$
f_s
STOPBAND ATTENUATION = DR
TRANSITION BAND: f_a to $f_s - f_a$
CORNER FREQUENCY: f_a
B
f_a
$Kf_s - f_a$
$\frac{Kf_s}{2}$
Kf_s
STOPBAND ATTENUATION = DR
TRANSITION BAND: f_a to $Kf_s - f_a$
CORNER FREQUENCY: f_a

Figure 5.28: *Oversampling Relaxes Requirements on Baseband Antialiasing Filter*

Filters become more complex as the transition band becomes sharper, all other things being equal. For instance, a Butterworth filter gives 6 dB attenuation per octave for each filter pole. Achieving 60 dB attenuation in a transition region between 1 MHz and 2 MHz (1 octave) requires a minimum of 10 poles—not a trivial filter, and definitely a design challenge.

Therefore, other filter types are generally more suited to high speed applications where the requirement is for a sharp transition band and in-band flatness coupled with linear phase response. Elliptic filters meet these criteria and are a popular choice. There are a number of companies which specialize in supplying custom analog filters. TTE is an example of such a company (Reference 5).

From this discussion, we can see how the sharpness of the antialiasing transition band can be traded off against the ADC sampling frequency. Choosing a higher sampling rate (oversampling) reduces the requirement on transition band sharpness (hence, the filter complexity) at the expense of using a faster ADC and processing data at a faster rate. This is illustrated in Figure 5.28B which shows the effects of increasing the sampling frequency by a factor of K, while maintaining the same analog corner frequency, f_a, and the same dynamic range, DR, requirement. The wider transition band (f_a to $Kf_s – f_a$) makes this filter easier to design than for the case of Figure 5.28A.

The antialiasing filter design process is started by choosing an initial sampling rate of 2.5 to 4 times f_a. Determine the filter specifications based on the required dynamic range and see if such a filter is realizable within the constraints of the system cost and performance. If not, consider a higher sampling rate which may require using a faster ADC. It should be mentioned that sigma-delta ADCs are inherently oversampling converters, and the resulting relaxation in the analog antialiasing filter requirements is therefore an added benefit of this architecture.

The antialiasing filter requirements can also be relaxed somewhat if it is certain that there will never be a full-scale signal at the stopband frequency $f_s - f_a$. In many applications, it is improbable that full-scale signals will occur at this frequency. If the maximum signal at the frequency $f_s - f_a$ will never exceed X dB below full-scale, then the filter stopband attenuation requirement is reduced by that same amount. The new requirement for stopband attenuation at $f_s - f_a$ based on this knowledge of the signal is now only DR – X dB. When making this type of assumption, be careful to treat any noise signals which may occur above the maximum signal frequency f_a as unwanted signals which will also alias back into the signal bandwidth.

Undersampling

Thus far we have considered the case of baseband sampling, i.e., all the signals of interest lie within the first Nyquist zone. Figure 5.29A shows such a case, where the band of sampled signals is limited to the first Nyquist zone, and images of the original band of frequencies appear in each of the other Nyquist zones.

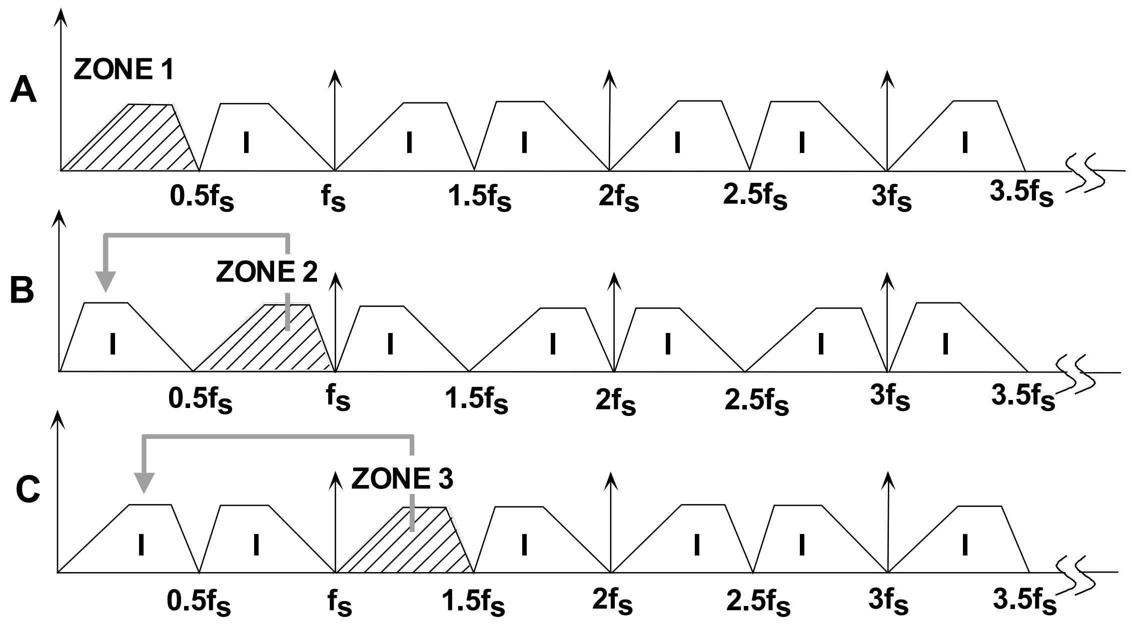


Figure 5.29: *Undersampling and Frequency Translation Between Nyquist Zones*

Consider the case shown in Figure 5.29B, where the sampled signal band lies entirely within the second Nyquist zone. The process of sampling a signal outside the first Nyquist zone is often referred to as *undersampling*, or *harmonic sampling* (*also referred to as band-pass sampling, IF sampling, direct IF to digital conversion*). Note that the first Nyquist zone image contains all the information in the original signal, with the exception of its original location (the order of the frequency components within the spectrum is reversed, but this is easily corrected by re-ordering the output of the FFT).

Figure 5.29C shows the sampled signal restricted to the third Nyquist zone. Note that the first Nyquist zone image has no frequency reversal. In fact, the sampled signal frequencies may lie in *any* unique Nyquist zone, and the first Nyquist zone image is still an accurate representation (with the exception of the frequency reversal which occurs when the signals are located in even Nyquist zones). At this point we can clearly restate the Nyquist criteria:

*A signal must be sampled at a rate equal to or greater than twice its **bandwidth** in order to preserve all the signal information.*

Notice that there is no mention of the precise *location* of the band of sampled signals within the frequency spectrum relative to the sampling frequency. The only constraint is that the band of sampled signals be restricted to a *single* Nyquist zone, i.e., the signals must not overlap any multiple of $f_s/2$ (this, in fact, is the primary function of the antialiasing filter).

Sampling signals above the first Nyquist zone has become popular in communications because the process is equivalent to analog demodulation. It is becoming common practice to sample IF signals directly and then use digital techniques to process the signal, thereby eliminating the need for the IF demodulator and filters. Clearly, however, as the IF frequencies become higher, the dynamic performance requirements on the ADC become more critical. The ADC input bandwidth and distortion performance must be adequate at the IF frequency, rather than only baseband. This presents a problem for most ADCs designed to process signals in the first Nyquist zone, therefore, an ADC suitable for undersampling applications must maintain dynamic performance into the higher order Nyquist zones.

Antialiasing Filters in Undersampling Applications

Figure 5.30 shows a signal in the second Nyquist zone centered around a carrier frequency, f_c, whose lower and upper frequencies are f_1 and f_2. The antialiasing filter is a band-pass filter. The desired dynamic range is DR, which defines the filter stopband attenuation. The upper transition band is f_2 to $2f_s - f_2$, and the lower is f_1 to $f_s - f_1$. As in the case of baseband sampling, the antialiasing filter requirements can be relaxed by proportionally increasing the sampling frequency, but f_c must also be increased so that it is always centered in the second Nyquist zone.

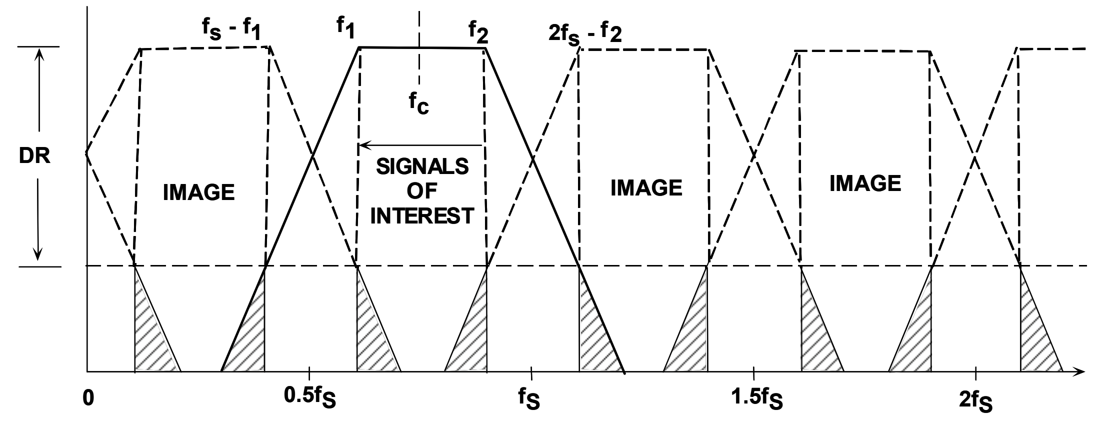
Two key equations can be used to select the sampling frequency, f_s, given the carrier frequency, f_c, and the bandwidth of its signal, Δf. The first is the Nyquist criteria:

$$f_s > 2\Delta f\,. \qquad \text{Eq. 5.5}$$

The second equation ensures that f_c is placed in the center of a Nyquist zone:

$$f_s = \frac{4f_c}{2NZ - 1}, \qquad \text{Eq. 5.6}$$

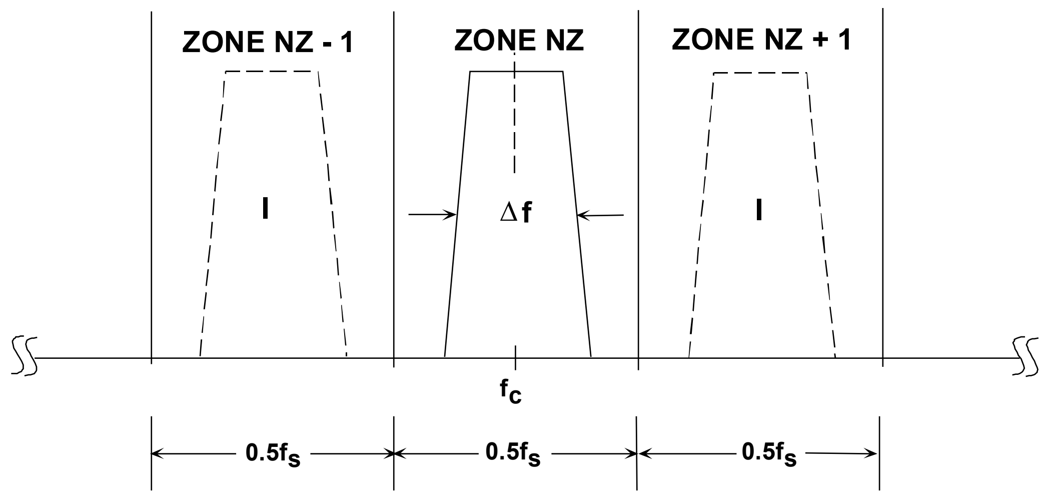
where NZ = 1, 2, 3, 4, ... and NZ corresponds to the Nyquist zone in which the carrier and its signal fall (see Figure 5.36).



BANDPASS FILTER SPECIFICATIONS:
- STOPBAND ATTENUATION = DR
- TRANSITION BAND: f_2 TO $2f_S - f_2$; f_1 TO $f_S - f_1$
- CORNER FREQUENCIES: f_1, f_2

Figure 5.30: *Antialiasing Filter for Undersampling*

NZ is normally chosen to be as large as possible while still maintaining $f_s > 2\Delta f$. This results in the minimum required sampling rate. If NZ is chosen to be odd, then f_c and its signal will fall in an odd Nyquist zone, and the image frequencies in the first Nyquist zone will not be reversed. Trade-offs can be made between the sampling frequency and the complexity of the antialiasing filter by choosing smaller values of NZ (hence a higher sampling frequency).



◆ $f_S > 2\Delta f$

◆ $f_S = \dfrac{4f_C}{2NZ - 1}$, NZ = 1, 2, 3, . . .

Figure 5.31: *Centering an Undersampled Signal within a Nyquist Zone*

As an example, consider a 4 MHz wide signal centered around a carrier frequency of 71 MHz. The minimum required sampling frequency is therefore 8 MSPS. Solving Eq. 5.6 for NZ using f_c = 71 MHz and f_s = 8 MSPS yields NZ = 18.25. However, NZ must be an integer, so we round 18.25 to the next lowest integer, 18. Solving Eq. 5.6 again for f_s yields f_s = 8.1143 MSPS. The final values are therefore f_s = 8.1143 MSPS, f_c = 71 MHz, and NZ = 18.

Now assume that we desire more margin for the antialiasing filter, and we select f_s to be 10 MSPS. Solving Eq. 5.6 for NZ, using f_c = 71 MHz and f_s = 10 MSPS yields NZ = 14.7. We round 14.7 to the next lowest integer, giving NZ = 14. Solving Eq. 5.6 again for f_s yields f_s = 10.519 MSPS. The final values are therefore f_s = 10.519 MSPS, f_c = 71 MHz, and NZ = 14.

The above iterative process can also be carried out starting with f_s and adjusting the carrier frequency to yield an integer number for NZ.

REFERENCES:
SAMPLING THEORY

1. H. Nyquist, “Certain Factors Affecting Telegraph Speed,” **Bell System Technical Journal**, Vol. 3, April 1924, pp. 324-346.

2. H. Nyquist, “Certain Topics in Telegraph Transmission Theory,” **A.I.E.E. Transactions**, Vol. 47, April 1928, pp. 617-644.

3. R.V.L. Hartley, “Transmission of Information,” **Bell System Technical Journal**, Vol. 7, July 1928, pp. 535-563.

4. C. E. Shannon, “A Mathematical Theory of Communication,” **Bell System Technical Journal**, Vol. 27, July 1948, pp. 379-423 and October 1948, pp. 623-656.

5. TTE, Inc., 11652 Olympic Blvd., Los Angeles, CA 90064, http://www.tte.com