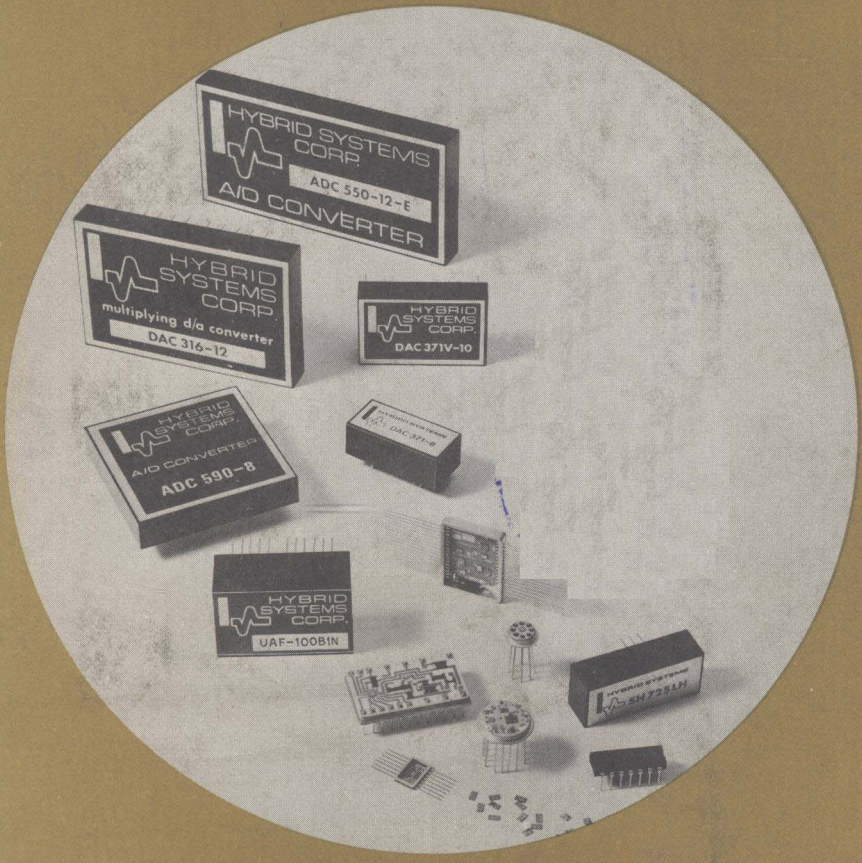
Data Conversion Handbook



• Theory • Specifications • Testing • Circuits

DATA CONVERSION HANDBOOK

by
Donald B. Bruck
Hybrid Systems Corporation

First Edition

Library of Congress Catalog Card Number: 73-87651

FOREWORD

Since 1965, Hybrid Systems Corporation has been a leading manufacturer of analog/digital modules. Our products are used throughout the world in control systems, computer interface systems, graphic display terminals, scientific instruments, test equipment, etc. OEM users incorporate these economical building blocks in their own proprietary products.

Our broad line includes units of all sizes, performance and prices. Attractive quantity and **OEM discounts** are available, as well as annual purchase agreements for your continuing needs. In addition, Hybrid System's staff of qualified, experienced Application Engineers stand ready to provide **technical assistance**; this document being but one example of the continuing flow of information to our customers.

Hybrid Systems is always willing to supply **sample units** on a no-cost evaluation basis. Please do not hesitate to contact us or your H.S.C. area representative. We welcome the opportunity to be of service to you.

The value of a Handbook such as this is enhanced when it has drawn upon broad collective experiences. I thus wish to thank the many talented engineers of Hybrid Systems' Engineering and Applications Departments for the suggestions and contributions which I have gratefully incorporated in this publication.

It is with great pleasure that we offer this sequel to our original Digital-to-Analog Converter Handbook and look foward to providing you with continued and increased products and services.

Donald B. Bruck
President

INTRODUCTION

The development and subsequent rapid reduction in cost of digital intergrated circuits is already legendary. It has resulted in an explosion in applications of digital processing techniques that have exceeded even the most optimistic projections. The myriad uses of digital processing have generated equally diverse performance requirements of digital-to-analog converters (DAC's) and analog-to-digital converters (ADC's), since a need for these elements appears wherever a digital system interfaces with the ever-present analog world.

Like the operational amplifier, the DAC's and ADC's are now available as practical and economic "building blocks" featuring various combinations of performance characteristics. A DAC which is ideally suited for computer graphic display terminals is probably far from optimum for process control. An ADC which is perfect for voice encoding is very likely a poor choice for use in a biological data acquisition. For every intended application, there is an optimum permutation of performance characteristics. It therefore behooves the user to appropriately understand the techniques and specifications embodied in such data conversion modules.

It is the purpose of this Handbook to assist in this regard by attempting to shed some light on the various types of modules, the limitations and features of each, the meaning of the specifications, methods of testing, etc.

INDEX

INDEX (Continued)

Page

INDEX (Continued)

Page

1. CODES

1.1 INTRODUCTION

The object of A/D and D/A conversion is to implement a means of producing a unique but consistent relationship between a voltage or current and a digital code. From a theoretical encoding point of view, there need not be any rhyme nor reason to the assignment of codes to voltage level, as long as it is truly single-valued. From a practical point of view, however, if one wants to conveniently instrument same and/or do calculations directly on the code, arbitrary assignment is abandoned in favor of mathematically defined codes. In deference to those of us who are too old to have studied the "new math" in elementary school, it is probably worthwhile at this point to briefly review some basic concepts of binary number systems and the fundamentals of binary arithmetic.

1.2 BINARY NUMBER SYSTEMS

Inasmuch as a consistent number system can be to any base greater than one, there is nothing sacrosanct about the commonly used decimal (base ten) system other than the coincidence (?) that humans were endowed with 10 fingers and 10 toes. The conclusion is probably that the criteria for selection of an appropriate number base should be mainly a question of convenience. The flip-flop, and its well known 2-state logic, made binary encoding very convenient to achieve electronically. This author vaguely recalls reading a mathematical proof that a number system to the base e (2.71828 . . .) would be most efficient from some esoteric point of view, but ignores the trivial (?) detail of instrumenting same. Hopefully, all copies of that writing have been destroyed, and this writer shall assume that we are in mutual agreement that this Handbook should be confined to only binary codes.

The two states of a binary digit, actually referred to as a bit*, are generally represented by a ONE ("**1**") or ZERO ("**0**"); the former a contribution from that bit and the latter no contribution. In common numerical systems, a number is represented by a string of digits, each contributing k times the next lowest digit (left to right); where k is the number base.

**Alleged to be a contraction of "binary digit"*

As in decimal where the *ten's* are ten times the units, and the *hundreds* ten times the *tens*, and so forth, in a binary system each bit carries twice the "weight" of the preceding bit.

Thus, binary weighting is . . . . 32, 16, 8, 4, 2, 1 and 1/2, 1/4, 1/8, 1/16, 1/32, ... It might be useful for the reader to convince himself that we can indeed represent (or generate) any integer by a combination of some or all of the elements of such a series. See Figure 1-1 for examples which include the binary representation denoting the status of each bit. The total of numbers spanned by n digits to the base k is $k \cdot k \cdot k \cdot \cdot \cdot k = k^n$. For example, three decimal digits (k=10, n =3) can "count" to 1000*. Similarly, n binary bits can count to 2^n. Stated more professionally, an n-bit binary number has 2^n different states (codes).

	WEIGHTS									
	2^4		2^3		2^2		2^1		2^0	
	=		=		=		=		=	
	16		8		4		2		1	
13 =	0	+	8	+	4	+	0	+	1	
	0		**1**		**1**		**0**		**1**	BINARY
12 =	0	+	8	+	4	+	0	+	0	
	0		**1**		**1**		**0**		**0**	BINARY
11 =	0	+	8	+	0	+	2	+	1	
	0		**1**		**0**		**1**		**1**	BINARY

Figure 1-1. Binary Representation.

*000 thru 999

1.3 BINARY ARITHMETIC

We all suffered through memorizing the addition tables of the decimal system (1+1=2, 1+2=3, . . . , 9+1=0 and carry 1, etc.). Fortunately, such suffering is minimal wth the binary system since each bit can have only two values, thus reducing the permutations; which are:

```
  0    1    0    1
+ 0  + 0  + 1  + 1
  0    1    1    0  CARRY 1
```

Counting in binary (adding 1 to the previous total) becomes simply as shown in Figure 1-2 for 4 bits. A definition is worth noting at this time.

1 1 1 1	15
1 1 1 0	14
1 1 0 1	13
1 1 0 0	12
1 0 1 1	11
1 0 1 0	10
1 0 0 1	9
1 0 0 0	8
1 1 1	7
1 1 0	6
1 0 1	5
1 0 0	4
1 1	3
1 0	2
1	1
0	0

Figure 1-2. Binary Counting.

Specifically, with regard to the two states of a binary bit ("**0**" and "**1**"), each is called the COMPLEMENT of the other, and the verb *COMPLEMENTING* refers to the act of reversing the state (i.e. "**1**" → "**0**" or "**0**" → "**1**"). The purpose of same will be evident later.

1.4 UNIPOLAR CODES

NATURAL BINARY

This is the best known and most easily understood of the various binary codes. Although from a strict definition point of view, *all* two-state codes are binary, the term "binary code" is often used interchangeably with natural binary. Natural binary code will be discussed at length because, in this author's opinion, a thorough understanding at this level provides the necessary background for understanding bipolar codes, and makes mastering of converters in general much easier.

The natural binary code is simply the ensemble of states achieved by counting one step at a time in binary from all "ZEROS" to all "ONES". As previously mentioned, n binary bits will produce 2^n counts, or codes (states). For obvious reasons, the bit with the greatest weight is called the *Most Significant Bit*, or MSB, and similarly, the one with the smallest contribution, the *Least Significant Bit*, or LSB.

In the case of a DAC, for example, the MSB is often weighted to be 5.000 Volts; the next strongest contributor (next Most Significant Bit) is 2.500 Volts, and so on, till we get to the Least Significant Bit (LSB) whose weight would be $5.000/2^{n-1}$ for an n-bit converter. A voltage can then be generated by appropriately including or excluding its respective contribution to the total. If the contribution of a particular bit is to be included, we define it as a "**1**"; if it is to be excluded, we indicate a "**0**". Figure 1-3 illustrates the 8 possible voltage levels generated using a "3 bit" code ($2^3 = 8$) with a 5.000 Volt MSB.

MSB		LSB	MSB LSB	
1	**1**	**1**	5.000 + 2.500 + 1.250 = 8.750 volts	
1	**1**	**0**	5.000 + 2.500 + 0 = 7.500 volts	3/4 FS
1	**0**	**1**	5.000 + 0 + 1.250 = 6.250 volts	
1	**0**	**0**	5.000 + 0 + 0 = 5.000 volts	1/2 FS
0	**1**	**1**	0 + 2.500 + 1.250 = 3.750 volts	
0	**1**	**0**	0 + 2.500 + 0 = 2.500 volts	1/4 FS
0	**0**	**1**	0 + 0 + 1.250 = 1.250 volts	
0	**0**	**0**	0 + 0 + 0 = 0.000 volts	ZERO

Figure 1-3. 3-Bit Natural Binary Conversion.

One point worth noting at this time is that the "steps", of which there are 2^n-1, are 1.250 Volts, which can be calculated by dividing 10.000 Volts "FULL SCALE" by the $2^3 = 8$ possible levels. If more levels within the above range were desired, additional bits would have to be used. One additional bit, making it a 4 bit code, would halve the LSB step size to .625 Volts, two additional bits would make it one-quarter to .3124 Volts, and so on. Thus, the degree to which we desire to resolve any arbitrary number within the range is *theoretically* limited solely by the number of bits used in the code. Figure 1-4 illustrates an n-bit natural binary code. An explanation as to why the maximum code (**11 . . .11**) is called FS − 1 LSB follows.

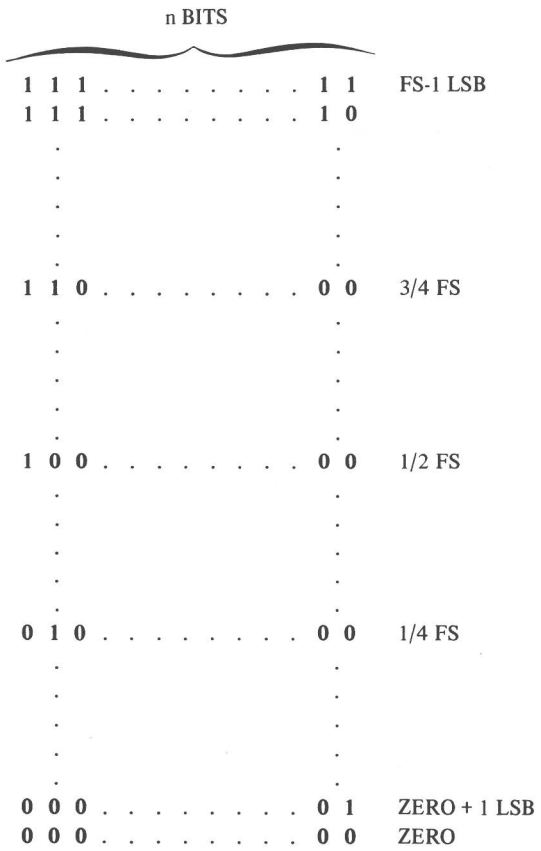


Figure 1-4. n-Bit Natural Binary Code.

FULL SCALE

Examining the previous example, we notice that the maximum voltage which we can generate with an MSB of 5.000 Volts is not the "FULL SCALE" 10.000 Volts as defined by twice "*HALF-SCALE*" (MSB), but rather one LSB worth less! Had it been necessary or desirable to actually reach 10.000 Volts, the bit contributions would have had to been 5.7144, 2.8572, and 1.4286* Volts vis-a-vis 5.000, 2.500, and 1.250. Thus, we see that by alternatively electing a step size to produce 10.000 Volts when all bits are on ("**1**"), we loose the ability to generate "HALF SCALE" of exactly 5.000 Volts.

The advantage in defining the **100 . . . 0** code as HALF SCALE (5.00 V, in the example) and calling the unreachable 10.000 Volts as FULL SCALE, is one of convenience in that there is indeed a code state for 3/4 FS, 1/2 FS, 1/4 FS, etc. Furthermore, the contributions of the bits are nice even values like 5.000, 2.500, etc., with which we tend to be at ease and this makes testing, trimming, and calibration much easier than had they been 5.7144, 2.8572, etc., as previously calculated. The disadvantage is simply that one cannot then achieve twice the HALF SCALE amount, or 10.000 V.

Instead of setting HALF SCALE or FULL SCALE to be convenient values, a valid alternative is selecting the magnitude of the LSB for convenience, such as 10.0 mV, for example. The disadvantage of this approach lies in the ineffficiency resulting from the fact that there will probably be extra unused codes existing above the maximum desired output (input) voltage. A case in point would be a 10 bit code with each step equal to 10.0 mV. Thus, the actual maximum would be 10.23 volts, and any outputs in excess of 10.00 volts might be wasted.

In summary, there are three potential, and commonly used, methods of "calibrating" a D/A converter for all codes. These are:

a. SELECT THE MAGNITUDE OF THE MSB (HALF SCALE) TO BE SOME CONVENIENT VALUE. This is the calibration method now generally used in procedures supplied by the manufacturers, In this alternative, an LSB = FULL SCALE/2^n, (Note: 2^n is the number of levels and FULL SCALE is *considered* twice HALF SCALE.)

*Calculated by dividing 10.00 Volts by the $2^n - 1 = 7$ *steps*