APPENDIX H

NUMBERING SYSTEMS

In This Appendix...

Introduction

As almost anyone who uses a computer is somewhat aware, the actual operations of a computer are done with a binary number system. Traditionally, the two possible states for a binary system are represented by the digits "zero" (0) and "one" (1), although "off" and "on" or sometimes "no" and "yes" are closer to what is actually involved. Most of the time a typical PC user has no need to think about this aspect of computers, but every now and then one gets confronted with the underlying nature of the binary system.

A PLC user, specifically the PLC programmer, should be more aware of the binary system. This appendix will provide an explanation of the numbering systems most commonly used by a PLC.

Binary Numbering System

Computers, including PLCs, use the Base 2 numbering system, which is called Binary and often called Decimal. Much the same, a PLC relies on only two valid digits, zero and one, or off and on respectively. You would think that it would be hard to have a numbering system built on Base 2 with only two possible values, but the secret is by encoding using several digits.

Each digit in the base 2 system, when referenced by a computer, is called a bit. When four bits are grouped together, they form what is known as a nibble. Eight bits or two nibbles would be a byte. Sixteen bits or two bytes would be a word (Table 1). Thirty-two bits or two words is a double word.

Word															
Byte								Byte							
Nibble				Nibble				Nibble				Nibble			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1

Binary is not "natural" for us to use since we grow up using the base 10 system. Base 10 uses the numbers 0-9. From now on, the different bases will be shown as a subscripted number following the number. Example; 10 decimal would be 1010.

Table 2 shows how base 2 numbers relate to their decimal equivalents.

A nibble of 1001_2 would be equal to a decimal number 9 ($1*2^3 + 1*2^0$ or $8_{10} + 1_{10}$). A byte of 11010101_2 would be equal to 213 ($1*2^7 + 1*2^6 + 1*2^4 + 1*2^2 + 1*2^0$ or $128_{10} + 64_{10} + 16_{10} + 4_{10} + 1_{10}$).

Binary/Decimal Bit Pattern																
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Power	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Decimal Bit Value	32678	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
Max Value	65535_{10}															

Table 2

Hexadecimal Numbering System

The binary numbering system can be difficult and cumbersome to interpret for some users. Therefore, the hexadecimal numbering system was developed as a convenience for humans since the PLC (computer) only understands pure binary. The hexadecimal system is useful because it can represent every byte (8 bits) as two consecutive hexadecimal digits. It is easier for us to read hexadecimal numbers than binary numbers.

The hexadecimal numbering system uses 16 characters (base 16) to represent values. The first ten characters are the same as our decimal system, 0-9, and the first six letters of the alphabet, A-F. Table 3 lists the first eighteen decimal numbers; 0-17 in the left column and the equivalent hexadecimal numbers are shown in the right column.

Decimal	Hex	Decimal	Hex
0	0	9	9
1	1	10	A
2	2	11	B
3	3	12	C
4	4	13	D
5	5	14	E
6	6	15	F
7	7	16	10
8	8	17	11

Table 3

Note that "10" and "11" in hex are not the same as "10" and "11" in decimal. Only the first ten numbers 0-9 are the same in the two representations. For example, consider the hex number "D8AF." To evaluate this hex number, use the same method used to write decimal numbers. Each digit in a decimal number represents a multiple of a power of ten (base 10). Powers of ten increase from right to left. For example, the decimal number 365 means $3x10^2 + 6x10 + 5$. In hex each digit represents a multiple of a power of sixteen (base 16). Therefore, the hex number D8AF translated to decimal means $13x16^3 + 8x16^2 + 10x16 + 15 = 55471$. However, going through the arithmetic for hex numbers in order to evaluate them is not really necessary. The easier way is to use the calculator that comes as an accessory in Windows. It can convert between decimal and hex when in "Scientific" view.

Note that a hex number such as "365" is not the same as the decimal number "365." Its actual value in decimal terms is $3x16^2 + 6x16 + 5 = 869$. To avoid confusion, hex numbers are often labeled or tagged so that their meaning is clear. One method of tagging hex numbers is to append a lower case "h" at the end. Another method of labeling is to precede the number with 0x. Thus, the hex number "D8AF" can also be written "D8AFh." where the lower case "h" at the end is just a label to make sure we know that it is a hex number. Also, D8AF can be written with a labeling prefix as "0xD8AF."

Octal Numbering System

Many of the early computers used the octal numbering system for compiled printouts. Today, the PLC is about the only device that uses the Octal numbering system. The octal numbering system uses eight values to represent numbers. The values are 0-7 being Base 8. Table 4 shows the first 31 decimal digits in octal. Note that the octal values are 0-7, 10-17, 20-27, and 30-37.

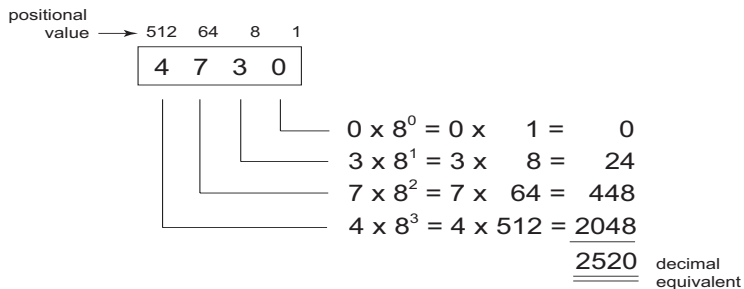| Octal	Decimal	Octal	Decimal
0	0	20	16
1	1	21	17
2	2	22	18
3	3	23	19
4	4	24	20
5	5	25	21
6	6	26	22
7	7	27	23
10	8	30	24
11	9	31	25
12	10	32	26
13	11	33	27
14	12	34	28
15	13	35	29
16	14	36	30
17	15	37	31

Table 4

This follows the DirectLOGIC PLCs. Refer to Chapter 3 bit maps and notice that the memory addresses are numbered in octal, as well as each bit. The octal system is much like counting in the decimal system without the digits 8 and 9 being available.

The general format for four digits of the octal number is:

$$(d \times 8^0) + (d \times 8^1) + (d \times 8^2) + (d \times 8^3)$$

where "d" means digit. This is the same format used in the binary, decimal, or hexadecimal systems except that the base number for octal is 8.

Using the powers of expansion, the example below shows octal 4730 converted to decimal.

```
positional
   value ──→   512   64    8    1

            ┌──────────────────┐
            │  4   7   3   0   │
            └──────────────────┘
                           └──── 0 x 8⁰ = 0 x    1 =      0
                      └───────── 3 x 8¹ = 3 x    8 =     24
                 └────────────── 7 x 8² = 7 x   64 =    448
            └─────────────────── 4 x 8³ = 4 x  512 =   2048
                                                       ─────
                                                        2520   decimal
                                                               equivalent
```

Binary Coded Decimal (BCD) Numbering System

BCD is a numbering system where four bits are used to represent each decimal digit. The binary codes corresponding to the hexadecimal digits A-F are not used in the BCD system. For this reason numbers cannot be coded as efficiently using the BCD system. For example, a byte can represent a maximum of 256 different numbers (i.e., 0-255) using normal binary, whereas only 100 distinct numbers (i.e., 0-99) could be coded using BCD. Also, note that BCD is a subset of hexadecimal and neither one does negative numbers.

BCD Bit Pattern															
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Power	10^3				10^2				10^1				10^0			
Bit Value	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
Max Value	9				9				9				9			

Table 5

One plus for BCD is that it reads like a decimal number, whereas 867 in BCD would mean 867 decimal. No conversion is needed; however, within the PLC, BCD calculations can be performed if numbers are adjusted to BCD after normal binary arithmetic.

Real (Floating Point) Numbering System

The terms Real and floating-point both describe IEEE-754 floating point arithmetic. This standard specifies how single precision (32-bit) and double precision (64-bit) floating point numbers are to be represented as well as how arithmetic should be carried out on them. Most PLCs use the 32-bit format for floating point (or Real) numbers which will be discussed here.

Real (Floating Point 32) Bit Pattern															
Bit #	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Sign	Exponent								Mantissa						
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Mantissa (continues from above)															

Table 6

Floating point numbers which
DirectLOGIC PLCs use have three basic components: sign, exponent and mantissa. The 32-bit word required for the IEEE standard floating point numbers is shown in Table 6. It is represented as a number from 0 to 31, left to right. The first bit (31) is the sign bit, the next eight bits (30-23) are the exponent bits and the final 23 bits (22-0) are the fraction bits.

In summary:

   The sign bit is either "0" for positive or "1" for negative;

   The exponent uses base 2;

   The first bit of the mantissa is typically assumed to be "1.*fff*", where "*f*" is the field of fraction bits.

The Internet can provide a more in-depth explanation of the floating point numbering system. One website to look at is:

   http://www.psc.edu/general/software/packages/ieee/ieee.html

BCD/Binary/Decimal/Hex/Octal - What is the Difference?

Sometimes there is confusion about the differences between the data types used in a PLC. The PLC's native data format is BCD, while the I/O numbering system is octal. Other numbering formats used are binary and Real. Although data is stored in the same manner (0s and 1s), there are differences in the way that the PLC interprets it.

While all of the formats rely on the base 2 numbering system and bit-coded data, the format of the data is dissimilar. Table 7 below shows the bit patterns and values for various formats.

Binary/Decimal Bit Pattern															
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Decimal Bit Value	32678	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
Max Value	65535															

Hexadecimal Bit Pattern															
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Decimal Bit Value	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
Max Value	F				F				F				F			

BCD Bit Pattern															
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Decimal Bit Value	8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
Max Value	9				9				9				9			

Octal Bit Pattern															
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit Value	1	4	2	1	4	2	1	4	2	1	4	2	1	4	2	1
Max Value	1	7			7			7			7			7		

Real (Floating Point 32) Pattern															
Bit #	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Sign	Exponent								Mantissa						
Bit #	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Mantissa (continued from above)															

Table 7

As seen in Table 7, the BCD and hexadecimal formats are similar, although the maximum number for each grouping is different (9 for BCD and F for hexadecimal). This allows both formats to use the same display method. The unfortunate side effect is that unless the data type is documented, it's difficult to know what the data type is unless it contains the letters A-F.

Data Type Mismatch

Data type mismatching is a common problem when using an operator interface. Diagnosing it can be a challenge until you identify the symptoms. Since the PLC uses BCD as the native format, many people tend to think it is interchangeable with binary (unsigned integer) format. This is true to some extent, but not in this case. Table 8 shows how BCD and binary numbers differ.

Data Type Mismatch											
Decimal	0	1	2	3	4	5	6	7	8	9	10	11
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	0001 0000	0001 0001
Binary	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	0000 1010	0000 1011

Table 8

As the table shows, BCD and binary share the same bit pattern up until you get to the decimal number 10. Once you get past 10, the bit pattern changes. The BCD bit pattern for the decimal 10 is actually equal to a value of 16 in binary, causing the number to jump six digits when viewing it as BCD. With larger numbers, the error multiplies. Binary values from 10 to 15 Decimal are actually invalid for the BCD data type.

Looking at a larger number, such as the value shown in Table 9, both the BCD bit pattern and the decimal bit pattern correspond to a base 10 value of 4095₁₀. If bit patterns are read, or interpreted, in a different format than what is used to write them, the data will not be correct. For instance, if the BCD bit pattern is interpreted as a decimal (binary) bit pattern, the result is a base 10 value of 16533₁₀. Similarly, if you try to view the decimal (binary) bit pattern as a BCD value, it is not a valid BCD value at all, but could be represented in hexadecimal as 0xFFF.

Look at the following example and note the same value represented by the different numbering systems.

Base 10 Value	BCD Bit Pattern	Binary Bit Pattern
4095	0100 0000 1001 0101	1111 1111 1111

Table 9

0100	0011		Binary		0001	0010	0011	0100	Binary
6	7		Decimal		4	6	6	0	Decimal
4	3		Hex		1	2	3	4	Hex
0110	0111		BCD		0100	0110	0110	0000	BCD
1	0 3		Octal		1	1 0	6	4	Octal

Signed vs Unsigned Integers

So far, we have dealt with unsigned data types only. Now we will deal with signed data types (negative numbers). The BCD and hexadecimal numbering systems do not use signed data types.

In order to signify that a number is negative or positive, we must assign a bit to it. Usually, this is the Most Significant Bit (MSB) as shown in Table 10. For a 16-bit number, this is bit 15. This means that for 16-bit numbers we have a range of -32767 to 32767.

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

Table 10

There are two ways to encode a negative number: two's complement and Magnitude Plus sign. The two methods are not compatible.

The simplest method to represent a negative number is to use one bit of the PLC word as the sign of a number while the remainder of the word gives its magnitude. It is general convention to use the most significant bit (MSB) as the sign bit: a 1 will indicate a negative, and a 0 will indicate a positive number. Thus, a 16-bit word allows numbers in the range ±32767. Table 11 shows a representation of 100 and a representation of -100 in this

Magnitude Plus Sign	
Decimal	**Binary**
100	0000 0000 0110 0100
-100	1000 0000 0110 0100

Table 11

format.

Two's complement is a bit more complicated. A simple formula for two's complement is to

Two's Complement	
Decimal	Binary
100	0000 0000 0110 0100
-100	1111 1111 1001 1011

Table 12

invert the binary and add one (see Table 12). Basically, 1s are being changed to 0s and all 0s are being changed to 1.

More information about 2's complement can be found on the Internet at the following website:

https://www.cs.cornell.edu/~tomf/notes/cps104/twoscomp.html#fromtwo

AutomationDirect.com Products and Data Types

DirectLOGIC PLCs

The DirectLOGIC PLC family uses the octal numbering system for all addressing which includes: inputs, outputs, internal V-memory locations, timers, counters, internal control relays (bits), etc. Most data in the PLC, including timer and counter current values, is in BCD format by default. User data in V-memory locations may be stored in other data types if it is changed by the programmer, or comes from some external source, such as an operator interface. Any manipulation of data must use instructions appropriate for that data type which includes: Load instructions, Math instructions, Out box instructions, comparison instructions, etc. In many cases, the data can be changed from one data type to another, but be aware of the limitations of the various data types when doing so. For example, to change a value from BCD to decimal (binary), use a BIN instruction box. To change from BCD to a real number, use a BIN and a BTOR instruction box. When using Math instructions, the data types must match. For example, a BCD or decimal (binary) number cannot be added to a real number, and a BCD number cannot be added to a decimal (binary) number. If the data types are mismatched, the results of any math operation will be meaningless.

To simplify number conversions, Intelligent Box (IBox) Instructions are available with DirectSOFT. These instruction descriptions are located in Volume 1, page 5-230, in the Math IBox group.

Most DirectLOGIC analog modules can be set up to give the raw data in decimal (binary) format or in BCD format, so it is necessary to know how the module is being used. DirectLOGIC PID is another area where not all values are in BCD. In fact, nearly all of the PID parameters are stored in the PLC memory as decimal (binary) numbers.

NOTE: *The PID algorithm uses magnitude plus sign for negative decimal (binary) numbers, whereas the standard math functions use two's complement. This can cause confusion while debugging a PID loop.*

When using the Data View in DirectSOFT, be certain that the proper format is selected for the element to be viewed. The data type and length are selected using the drop-down boxes at the top of the Data View window. Also notice that BCD is called BCD/Hex. Remember that BCD is a subset of hexadecimal so they share a display format even though the values may be different. This is where good documentation of the data type stored in memory is crucial.

C-more/C-more Micro-Graphic Panels

In the C-more and C-more Micro-Graphic HMI operator panels, the 16-bit BCD format is listed as "BCD int 16." Binary format is either "Unsigned int 16" or "Signed int 16" depending on whether or not the value can be negative. Real number format is "Floating PT 32."

More available formats are, "BCD int 32", "Unsigned int 32" and "Signed int 32."