**Binary**
**Numbering System**

Computers including PLC's use the Base 2 numbering system, which is called Binary or Boolean. There are only two valid digits in Base 2, 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, as we are all well aware. From now on, the different Bases will be shown as a subscripted number following the number. Example; 10 decimal would be $10_{10.}$

*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 # | 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**

As you have probably noticed, the Binary numbering system is not very easy to interpret. For a few bits, it is easy, but larger numbers tend to take up a lot of room writing them down and are very hard to keep track of what the bit position is while doing the conversion. That is where using an alternate numbering system comes in. One of the first numbering systems to be used was Hexadecimal or Hex for short.

Hex is a numbering system that uses Base 16. The numbers $0\text{-}9_{10}$ are represented normally, but the numbers $10_{10}$ through $15_{10}$ are represented by the letters A through F respectively( *Table 3*). This works well with the Binary system as each nibble ($1111_2$) is equal to $15_{10}$. Therefore, for a sixteen-bit word you could have a possible Hex value of $FFFF_{16}$. See *Table 4* for an example.

| Decimal | Hex |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |

| Decimal | Hex |
|---|---|
| 8 | 8 |
| 9 | 9 |
| 10 | A |
| 11 | B |
| 12 | C |
| 13 | D |
| 14 | E |
| 15 | F |

***Table 3***

| | **Hexadecimal** | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit # | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Power | $16^3$ | | | | $16^2$ | | | | $16^1$ | | | | $16^0$ | | | |
| Bit Value | 8 | 4 | 2 | 1 | 8 | 4 | 2 | 1 | 8 | 4 | 2 | 1 | 8 | 4 | 2 | 1 |
| **Max Value** | **F** | | | | **F** | | | | **F** | | | | **F** | | | |

***Table 4***

Hex to decimal conversions work in much the same way as Binary. $C2_{16}$ would be equal to $194_{10}$ ($12*16^1 + 2*16^0$ or $192 + 2$.) $A6D4_{16}$ would be equal to $42708_{10}$ ($10*16^3 + 6*16^2 + 13*16^1 + 4*16^0$ or $40960_{10} + 1536_{10} + 208_{10} + 4_{10}$.)

**Octal**
**Numbering System**

The Octal numbering system is similar to the Hexadecimal numbering system in the interpretation of the bits (*Table 5*). This big difference is that the maximum value for Octal is 7 since it is Base 8.

| | **Octal** | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit # | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Power | $8^5$ | $8^4$ | | | $8^3$ | | | $8^2$ | | | $8^1$ | | | $8^0$ | | |
| 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** | | |

***Table 5***

You should be getting the hang of how this works by now. $63_8$ is equal to $51_{10}$ ($6*8^1 + 3*8^0$ or $48_{10} + 3_{10}$.)

**BCD**
**Numbering System**

Again, the BCD system is like the Octal and Hexadecimal numbering systems. It also relies on bit-coded data(*Table 6*). It is Base 10, or Decimal, but it is Binary Coded Decimal. There is a big difference in BCD and Binary, as we will see in a bit.

| | **BCD** | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 6***

The somewhat good part about BCD is that it reads like a Decimal number, IE 867 BCD would mean 867 Decimal. No conversion needed. However, as with all things related to computers there are snags to worry about.

## Real (Floating Point)
## Numbering System

The terms Real and floating-point both describe IEEE-754 floating-point numbers numbers. Most plcs' use the 32-bit format for floating point (or Real) numbers (*Table 7*).

The formula and layout of the number is as follows.

$Number=1.M * 2^{(E-127)}$

Number: The number to be converted to floating-point
M: Mantissa
E: Exponent

| | **Real (Floating-Point 32)** | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 | | | | | | | | | | | | | | | |

***Table 7***

Calculating the Real number format is very complex. For that very reason, we will not attempt it here. If you are interested in the conversion process there are numerous papers on the internet that go into specific detail on how to do it.

You may notice that there is not a minimum or maximum value given for the Real number format. The range is from negative infinity to positive infinity. Having said this, and having noticed that there are only thirty-two bits possible to create every number from negative infinity to positive infinity, it is easy to surmise that not all numbers can be represented. This is in fact the case. There is an inherent amount of error with the Real format. Again, there are numerous documents on the internet, which can explain this phenomenon better than this paper could.

I'm sure that you are wondering how much error can there be and if there is a lot of error why is this format used? To try and give a short answer, it depends on the application. For most plc applications unless you are trying to be 100% accurate with absolutely no possible error then the Real format will not pose many problems. So most of the time the inherent error can be ignored, but it is important to know that it exists.

**BCD/Binary/Decimal/Hex/Octal**
**What is the difference?**

There is sometimes confusion about the differences between the data types used in a plc. The plcs' native data format is BCD while the I/O numbering system is Octal. Other numbering systems used in plcs' are binary and real. Although data is stored in the same manner (0's and 1's), there are differences in the way that the plc interprets it.

While all of the formats rely on the base two numbering system and bit coded data, the format of the data is dissimilar. *Table 8* below shows the bit patterns and values for Binary, Hexadecimal, BCD, and Octal. We will cover the Real numbering system later.

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

***Table 8***

**Data Type Mismatch**

Data type mismatching is a common problem when using an operator interface. Diagnosing it can be a challenge until you realize 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 is not the case. *Table 9* shows how BCD and Binary numbers differ.

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

***Table 9***

As the table shows, BCD and Binary share the same bit pattern up until you get to the decimal number 10, so there are similarities. Once you get past 10 things change drastically! The BCD bit pattern for the decimal 10 is actually equal to a value of 16 in Binary causing you to jump six digits by looking at it with the BCD data type! With larger numbers, the error multiplies. The Binary values from 10 to 15 decimal are actually invalid for the BCD data type.

Let's look at a larger number shown in *Table 10*.

| Decimal | BCD | Binary |
|---|---|---|
| 4096 | 0100 0000 1001 0110 | 0001 0000 0000 0000 |

***Table 10***

As a Decimal number, the value is 4096. If we interpret the BCD number as Binary, the Decimal number would be 16534. Similarly, if we interpret the Binary number as BCD, the Decimal number would be 1000. This leaves quite a bit of difference between the two numbers.

As you can see from *Table 8*, The BCD and Hexadecimal formats are similar, although the maximum number for each grouping is different (9 for BCD, F for Hexadecimal.) This allows both formats to use the same display method. The unfortunate side effect of this is that unless the data type is documented, you really do not know what the data type is unless it contains the letters A-F.

## Sign vs. Unsigned Integers

So far, we have dealt with unsigned data types only. Now we will deal with negative numbers and signed data types. The BCD data type cannot be used for 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 MSB or most significant bit (*Table 11*). 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 11***

Of course, it cannot be as simple as that. We have two ways of encoding a negative number, two's complement and Magnitude Plus sign. The two methods are not compatible.

As long as the value is positive or bit 15 is not set then the rules work the same as before. If bit 15 is set then we must know which encoding scheme was used.

Magnitude plus sign is the easiest to decode. Basically, it is the same format as the positive number with bit 15 set (*Table 12*).

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

***Table 12***

Two's Complement is slightly more difficult. The formula is to invert the binary value and add one (*Table 13*).

| **Two's Complement** | |
|---|---|
| Decimal | Binary |
| 100 | 0000 0000 0110 0100 |
| -100 | 1111 1111 1001 1100 |

***Table 13***

**AutomationDirect.com Products**
**And**
**Data Types**

## DirectLogic PLC's

The DirectLogic plc family uses the Octal numbering system for I/O modules. All of the internal V-Memory is stored in BCD format unless specifically changed by the programmer. This means that all math should be done using BCD math functions. You have the option of using Binary or Real numbers as well as Double-Word BCD values in some of the plc models. To change data types you must use a function box. To change from BCD to Binary, you would use a BIN box. To change from BCD to Real you would use a BIN and then a BTOR box. You cannot add a BCD or Binary number to a Real number, or a BCD number to a Binary number and get a correct result. The data types must match. An analogy would be trying to make orange juice out of lemons, they are both citrus fruits, but the result just isn't going to taste right.

There are a couple of points to be aware of as far as all V-Memory being BCD by default. Analog cards can be setup to give Binary results as well as BCD, so you have to know how the card is being used. PID is another place where everything is not BCD. In fact, nearly all of the PID parameters are stored in the plc as Binary numbers.

An interesting point is that the PID algorithm uses Magnitude Plus Sign for negative binary numbers, whereas the regular math functions use Two's Complement. This is worth noting as it can really cause confusion while debugging a PID loop.

The last point to be aware of is in data view mode. As strange as it may sound Binary, Hex, and Decimal are all stored the same way in the plc and are all called Binary format. The only difference is when you look at them in data view mode. Be sure you select the proper format from the drop down box. In addition, you will notice that BCD is called BCD/Hex. As you may remember from a few pages ago, they are actually the same as long as you do not have the letters A through F, so they share a display format even though they are very different values. This is where good documentation of the data type stored in memory is crucial.

## EZ-Touch/EZ-Text

In the EZ-Touch and EZ-Text, 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_Point_32.