

## A discussion of numeric types and their storage in AutomationDirect PLC – specifically the DL06

There are three aspects concerning the use of numeric values. The first is the method of storage in the computer bits in the PLC. The second is the manner of showing those numeric values to you, the human. Finally there are the consequences for the PLC programmer.

### **Value storage, 32 bit:**

V Register (example)	Bit Position		Bit Position	"Binary"	BCD	Real
V2000	0	Accumulator	0	1	1	Fraction
	1		1	2		
	2		2	4		
	3		3	8		
	4		4	16		
	5		5	32		
	6		6	64		
	7		7	128		
	8		8	256		
	9		9	512		
	10		10	1024		
	11		11	2048		
	12		12	4096		
	13		13	8192		
	14		14	16384		
	15		15	32768		
V2001	0		16	65536	10000	Exponent
	1		17	131072	20000	
	2		18	262144	40000	
	3		19	524288	80000	
	4		20	1048576	100000	
	5		21	2097152	200000	
	6		22	4194304	400000	
	7		23	8388608	800000	
	8		24	16777216	1000000	
	9		25	33554432	2000000	
	10		26	67108864	4000000	
	11		27	134217728	8000000	
	12		28	268435456	10000000	
	13		29	536870912	20000000	
	14		30	1073741824	40000000	
	15		31	2147483648	80000000	

This table shows the method of storage of numeric values in 32 bits (two 'V' memory registers). For the 'Binary' and BCD columns the resulting number (seen as a decimal number) is simply the addition of the values shown for each line whose 'bit position' is a '1' (rather than a '0'). (An important restriction of the BCD format is that each group of

four bits can only contain the single digit value 0 – 9. Others yield invalid results.) The calculation of the value for a ‘Real’ given the bit positions that are ‘1’ is beyond the scope of this discussion. Needless to say, it is quite different from the ‘Binary’ or ‘BCD’ formats. The important point to remember is that the usage of the bit positions is very different from one storage format to the others.

For the ‘Binary’ and ‘BCD’ formats there are also single ‘V’ memory formats whose bit values are simply those of bit positions 0 – 15. If you use the double register version of numbers or ‘real’ numbers be sure to provide a nickname for both V memory locations which are utilized to help prevent inadvertent re-use of the second V register location.

### ***Human Representation:***

The representation of these values to the human eye is most true to its actual storage when shown as a string of ones and zeroes. But this doesn’t give much useful information to the viewer.

Some methods of showing the ‘Binary’ format are based on groupings of the bits. Grouping them 3 at a time is the ‘octal’ format. The groupings have the representations from 0 to 7. Another method groups the bits 4 at a time. This is the ‘hexadecimal’ format. The groupings have the representation of 0 to 9 with the addition of the letters ‘a’ through ‘f’. (You may see the letters in their lower-case or upper-case forms but they are equivalent.)

But we are most familiar with our decimal system of numbering. No grouping of the bits naturally results in showing the value in this method. The bit weightings of the ‘on’ bits must be added and presented in the standard ‘decimal’ format.

In ancient computer times (20 years ago) it was common to have input and output devices attached to the industrial computing device in a way that directly encoded individual decimal digits as entered or viewed by an operator. It took four bits to encode the values 0 – 9 for each input or output digit. To store these without conversion a method of dividing up the bits into 4 bit groupings but restricting their values to those from 0 – 9 was developed. This is called Binary Coded Decimal (BCD). The 4 bit groupings read naturally to the human operator as if it were a standard decimal number.

While the storage meant easy interfacing with an operator the computer mathematical hardware demanded that the BCD encoding be converted to the ‘binary’ form before meaningful mathematical operations could be performed. The resultant values must then be converted back to BCD before the result is stored. As you may guess, the time to perform this ‘convert – operate – convert’ function takes much longer than just the operation on numbers that are already in the ‘binary’ form.

The 'real' method of storage can be shown as standard decimal numbers, for the larger values; numbers with a decimal point, for medium to smaller numbers and numbers in exponential format for the extremely large and extremely small numbers.

### ***Ramifications for you as a programmer:***

Some PLC systems do all the heavy lifting for you, converting numbers between representation and storage systems even within one mathematical instruction. The DL06 and its similar systems in the AD PLC lines in the basic instruction set, to put it simply, don't. You are required to pay special attention to the type of storage, the type of representation and the type of mathematical operations being performed at any point in the program. In short, you are required to think about your data.

The basic instructions set (excluding IBoxes) provides for in-accumulator transformations. This means that the original number must be loaded (LD or LDD) into the accumulator (or created in the accumulator by various mathematical operations), the transformation made (BIN, BCD, BTOR, RTOB), then the result utilized or stored. In the case of a 'real' the storage must be by way of the OUTD command. The others may use OUT or OUTD depending on the expected size of the result. Note that the basic instruction set does not provide a direct transformation between BCD and real. This must be performed by way of an intermediate transformation to the binary type.

The mathematical instructions also vary depending on the data type to be utilized. Note that there are no mixed type instructions in the basic set. The operation of addition, for example, has 5 forms: ADD (BCD), ADDD (double register BCD), ADDB (Binary), ADDBD (double register Binary) and ADDR (Real). There are also versions of these in which the second operand is derived from the accumulator stack.

### ***IBoxes:***

The extension of the instruction set by the introduction of IBoxes has eased things for the programmer but, and this is a very personal view, at the expense of programmer control, program space and execution time. No absolute capabilities were added which cannot be accomplished by use of the standard set. Indeed the IBoxes compile to sets of the standard instructions that are executed at run time. The IBoxes do not relieve you of the responsibility to control the V memory storage of the numbers.