

Working with Binary Numbers

This application guide describes how to convert between decimal and binary numbers and discusses some of the problems that can occur.

In many Milltronics devices, binary data is stored in 16 bit registers. Depending on what you are using to read this information, access to it will be either very easy or will require some effort.

Typically PLCs handle binary data very easily. For example, in an Allen Brady SLC 500 the 4th bit of register 15 is referenced as "N7:15/4". However, if you are using a PC, accessing binary data can be more difficult depending on the type of software you are using, how it represents data, and how it lets the user access the data.

Description:

After reading a 16 bit register, a particular piece of software can do one of the following:

1. let the user reference the information as bits
2. represent the 16 bit number as an unsigned integer
3. represent the 16 bit number as a signed integer

1. Information as Bits

If the software lets the user reference (read and write) the information as bits, then no conversion is required.

It should be noted though, that there is a known problem with Wonderware InTouch in this situation. If you use the default scaling of -32768 to 32767 for the register, then some of the bits will change due to an internal problem with InTouch. However, if the scaling is changed to be from 0 to 65536, then the data will be correct (for more details on this problem, please see application guide 99001).

Keywords:

Binary, Decimal, Convert, bits, registers, ERS-500, BW-500

2. Information as Unsigned Integer

If the data is represented as an unsigned integer, then the user must convert from an integer to a binary number. This can be done easily by using a conversion formula. For example, you may want to see the 4th bit of Register 41,070 in an ERS-500, which represents the state of the digital input 4. This is done with this formula (using the syntax found in Excel):

$$\text{Bit 4} = \text{Mod} (\text{Int} (\text{R41070} / 2^3), 2)$$

Where:

Mod is the Modulus function and is the remainder after a division.

Int is the Integer function and is the integer part of the number.

If $\text{R41070} = 27$, then $27 / 2^3 = 27 / 8 = 3.375$, and $\text{Int}(3.375) = 3$, and the $\text{Mod}(3, 2)$ is 1. Therefore, Bit 4 would be a 1.

The complete formula is:

Let a be an integer (the number to be converted i.e. R41070 in the example above).

Let $(r_n, r_{n-1}, \dots, r_3, r_2, r_1)$ be the binary equivalent of a , where r_1 is the least significant bit, i.e. the low order of the number (in the example above, r_n would be r_4 and would refer to bit 4).

Then:

$$r_n = \text{Mod} (\text{Int} (a / 2^{n-1}), 2)$$

This is a very useful formula; however, it is important to pay very close attention to how the math is done. Three potential problems are:

1. If the calculation $(a / 2^{n-1})$ is done using integer math, then the wrong number is generated. This needs to be done with floating point math.
2. The Int function can do one of two things, either it rounds to the nearest whole number or it truncates (i.e. chops off the digits after the decimal). For this formula, we need it to truncate.
3. The integer numbers must be unsigned (see below for converting a signed integer into a unsigned integer).

After converting the number into the series of bits, the user is faced with the problem of converting it back to an integer in order to be able to write it back to the device. This is done by virtue of the fact that each bit in a binary number has an integer value. Therefore all you have to do is add up the values. For example, if you have the following number:

011101

Then the integer equivalent is:

$$1 \cdot 2^0 + 0 \cdot 2^1 + 1 \cdot 2^2 + 1 \cdot 2^3 + 1 \cdot 2^4 + 0 \cdot 2^5 = 1 \cdot 1 + 0 \cdot 2 + 1 \cdot 4 + 1 \cdot 8 + 1 \cdot 16 + 0 \cdot 32 = 1 + 4 + 8 + 16 = 29$$

The formula is:

Let N be the decimal equivalent of the binary number:

$$N = \sum r_a \cdot 2^{(a-1)} \quad , \text{ for } a= 1 \text{ to } n$$

3. Information as Signed Integers

If the integer is a signed integer, then to use the formulas stated above the number has to be converted into an unsigned integer. This can be done by recognizing that the only difference between a signed and unsigned 16 bit integer is that the 16th bit is used as a sign in the signed integer.

Therefore, to change a signed integer to an unsigned integer:

$$\text{If } N < 0 \text{ then } N = N + 65536$$

And to convert from unsigned to signed integer:

$$\text{If } N \geq 32768 \text{ then } N = N - 65536$$

Note: The information in this document is intended as a “guide” only. Milltronics assumes no responsibility for its application.