IEEE 745 Float Transmission by ModBus

By Pat Dixon, PE, PMP (www.DPAS-INC.com)

In many cases in industrial automation, it is necessary to transmit floating point values that exceed the range of a 16-bit signed integer (32767 to -32768). It is desired to have a means of doing this that does not compromise the resolution (precision) of the floating-point value. Unfortunately, many of the common means of transmitting data do not support transmitting floating-point values.

Of these transmission techniques, ModBus may be the most common. ModBus support transmission of 16-bit signed integers, but not floating point.

The common format of a binary representation of a floating point value is the IEEE 745 standard (https://en.wikipedia.org/wiki/IEEE_754-1985). In a 32-bit representation, bit 32 is the sign bit, bits 31-23 are the exponent, and bits 22-1 are the mantissa. This bit pattern can be broken into 2 16-bit parts and transmitted as 2 signed integers over ModBus. Then, the integers need to be reassembled on the receiving end to reconstruct the original floating-point value with all 32 bits intact and in the same order as its origin.

For a 16-bit signed integer, bit 16 is the sign bit and bits 15-1 are the mantissa (magnitude). A complication is that in Rockwell, the 16-bit signed integer is not represented as a sign-magnitude binary pattern.

- In a sign-magnitude integer, the absolute value remains the same regardless of the sign bit. For example, if the value is +1, it is represented as 0 (sign bit positive) 000 0000 0000 0001 (mantissa is 1). If the value is -1, it is represented as 1 (sign bit negative) 000 0000 0000 0001 (same as before).
- Instead, Rockwell uses a 2s-compliment 16-bit signed integer. The value +1 has the same representation as above, but for -1 it is 1 (sign bit negative) 111 1111 1111 1111 (mantissa). This makes it rather non-intuitive and complex, as the bit pattern from the original floating point get re-arranged.

Therefore, to transmit an IEEE 745 floating point by ModBus to Rockwell, if the most significant bit (the sign bit, bit 16) is 1 (negative) we need to recalculate the integer we want to send by ModBus so that the bit pattern that shows up on the other end matches what was in the floating point pattern instead of the 2s-compliment pattern.

To test the algorithm for doing this, I created the Excel spreadsheet float2int_test.xlsm. The Real2Int macro in the Excel spreadsheet takes the entered floating point value and breaks it into the IEEE 745 bit pattern, then calculates the 16-bit sign-magnitude integers, then converts them into 2s-compliment 16-bit signed integers. The DDE macro creates a connection to an RS Linx topic to store the 2s-compliment values to integers in the float2int-convert.ACD logic.

That is the most complex part of the work. Having done this, we can see that the INT values sent from the spreadsheet to the PLC logic will represent the original IEEE 745-bit pattern for a 32-bit floating point. Then, all we need is a COP instruction to designate the first of the 2 INT tags in the array as the source and the REAL tags as its destination. We can then see that the floating-point value in the PLC gets updated with high precision from the full IEEE 745 range with the value we enter on the spreadsheet.