RFID Encoding Notes

This is a set of notes on the encoding schemes for the Variable Length Alphanumeric encoding scheme.

There are six encoding schemes used for Variable Length Alphanumeric option. It is used for Serial Numbers, Batch/Lot Numbers, and other fields you will likely not use. The six encoding schemes are:

0 – Binary – A numeric value encoded as binary [0..9] An approximately 3.2 bits per character encoding.

1 – Lower-case hexadecimal – A 4 bit per character encoding. [0..9,a..f]

2 – Upper-case hexadecimal – A 4 bit per character encoding. [0..9,A..F]

3 – Code 40 – An approximately 5 bit per character encoding [0..9,A..Z,colon,dash,comma]. The encoding is in triplets, that is, three characters at a time, the space character is used for padding.

4 – Base 64 – A 6 bit per character encoding [0..9,A..Z,a-z,dash,underscore]

5 – ASCII – A 7 bit per character encoding. Most printable 7- bit ASCII characters

The number at the beginning of each line is the numerical encoding identifier used before the field length value. It is converted to a 3-bit binary number and combined with a 5-bit binary value representing the number of characters in the input data. The combined value comes right before the field data as two nibbles. Since the encoding value is shifted left one bit, the value of 1, for example, will appear as a 2 in the EPC string.

The encodings, except for the two hexadecimal encodings, are not an integer number of nibbles (4-bit values that are expressed as a hexadecimal character). The hexadecimal encodings have the additional advantage of appearing in the EPC (RFID) string exactly as they do in the raw form. For these reasons, **students are encouraged to use hexadecimal encoding**. The encoding value cannot be specified in the demo application. GS1 prefers that the encoding software look at the input data and determine the encoding that uses the least number of bits. To force hexadecimal encoding, use only numeric values and at least one character from the sets [A..F] or [a..f]. For example, A1234 or 1f2e3d.

As an example of how complicated encoding can be, let's take a binary example. The value of 15 (0b111) is encoded as 0b000111pp or 0x1E. The pp at the end of the binary string are padding bits that are empty (not used). The next field must be shifted two places to the left to fill these padding bits. The padding at the end of the next field must be increased by these two bits. If the total number of padding bits exceeds 4, then the number of padding bits is reduced to modulo 4. The three 0 bits at the beginning of the binary string are needed because the length information is in source characters and, since a 9 takes 4 bits and is one character, a 1 must be stretched to 4 bits also. It is impossible to know what the value of the first digit is, and therefore how many bits it uses. Various values will result in different amounts of left padding.

The other encodings are less complicated than binary but more complicated than hex.