
100BASE-TX Decoding Instructions

Note: These instructions assume the following:

- You have downloaded the descrambler key binary sequence file.
- You have printed the 100BASE-TX simulated frame (PDF document)

General Instructions

Decoding the 100BASE-TX waveform should not be attempted until you have first successfully decoded one or more of the 10Mbps waveforms. Performing a 10Mbps waveform decode will prepare you for the additional steps required with 100BASE-TX, so that you don't have to learn all the processes at once.

As with the 10Mbps exercises, use a pencil. It is nearly impossible to do this without error the first time.

A short summary of the steps required is presented first, and then a detailed example follows.

1. Use the MLT-3 encoding rules to derive a binary sequence from the waveform.
2. Align the descrambler key sequence with the waveform-derived binary, such that Idle code groups are discovered and decoded. See Steps 3 and 4. If Idle code groups are not found, move the descrambler key sequence over one bit and try again until you are successful.
3. Perform a logical XOR operation on the waveform-derived binary and the descrambler key sequence to produce bits that are used to group into 5-bit symbols.
4. Verify descrambler key alignment; trying again if necessary.
5. Referring to the 5B/4B Symbol decoding table, convert the 5-bit symbols into 4-bit nibbles.
6. Reorder groups of two nibbles back into MAC Layer octets.
7. Count octet groups into Ethernet Fields. For example, the Preamble is 7 octets, followed by the single-octet Start Frame Delimiter. Refer to the Ethernet 802.3 Standard frame field definitions for octets required for each field.

After the Ethernet frame is decoded, identify the protocol indicated by the Ethernet Length/Type field, and then look up the field definitions for that protocol. After the protocol identified by the Ethernet Length/Type field code is divided into fields and decoded, repeat the process using the next protocol identifier. This may continue until the entire frame is divided into the correct fields and decoded.

Generally, decoding up through OSI Layer 4 is comparatively rapid and satisfying. Decoding OSI Layers 5–7 protocols is more challenging, and might not provide enough benefit to be worthwhile.

Detailed Decoding Instructions

The brief summary decoding summary above is expanded to provide detailed directions and explanation in the following text.

Step 1. Use the MLT-3 encoding rules to derive binary values from the waveform.

MLT-3 encoding is accomplished by changing the voltage level by 1 volt for each binary 1, and by remaining at the same voltage for each binary 0. The voltage changes occur between +1 and –1 volts, and only a change of voltage is important—not the voltage level itself. Each timing period in the MLT-3 encoding examples in Figure 1 is accompanied by the proper binary decoded value below the waveform.

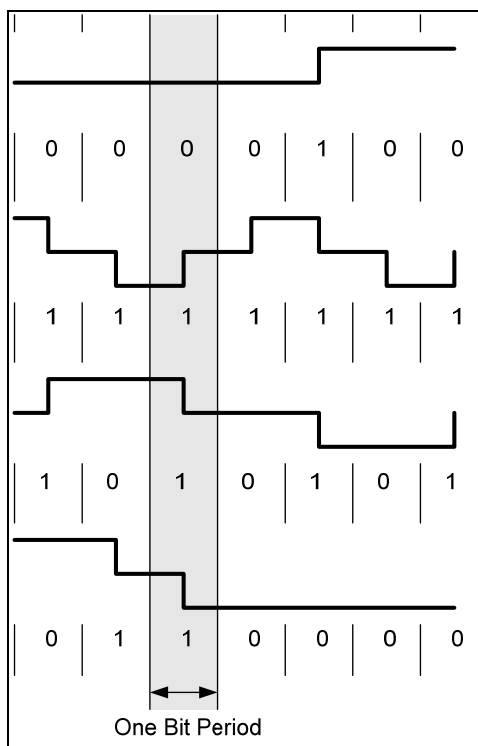


Figure 1 *MLT-3 decoding examples.*

Apply the encoding/decoding example information in Figure 1 to the waveform to derive binary values for each timing interval.

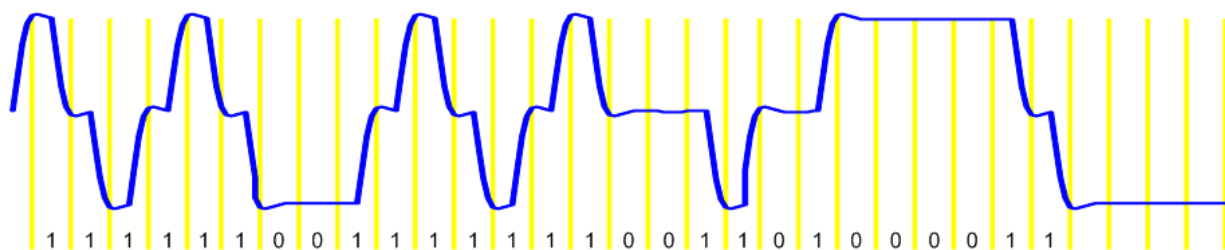


Figure 2 Derive the binary values for the MLT-3 encoded waveform.

Step 2. Align the descrambler key with the derived binary data.

Once a reasonable sample of binary is available, take the descrambler key data and align it with the binary (above or below your derived binary) so that the next operation may be performed. Because it is almost certain that the descrambler key will have to move over, be careful to annotate the waveform document with the descrambler bits using pencil.

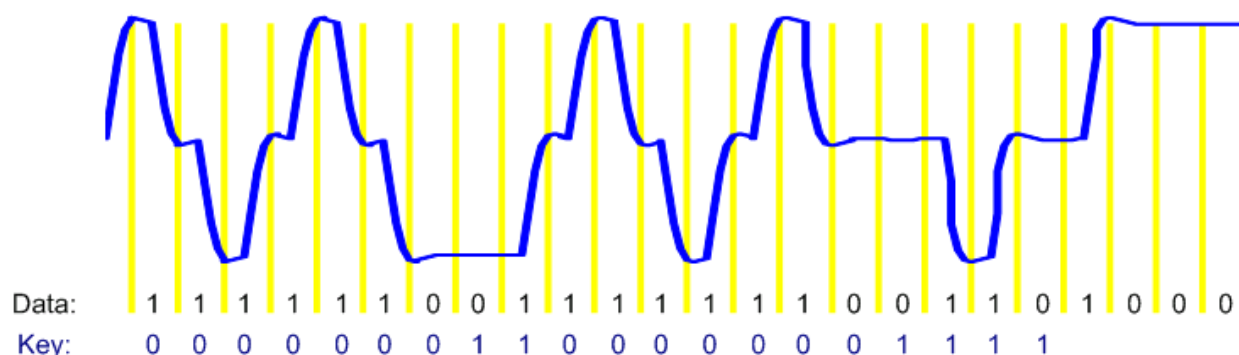


Figure 3 Descrambler key aligned with derived binary (either above or below; below in this example) in preparation for the next step.

Step 3. Apply a logical Exclusive-OR (XOR) conversion to each derived data bit and descrambler key bit.

After you have a reasonably lengthy set of derived binary aligned with the descrambler key, use a logical XOR operation to descramble the MLT-3 binary back into 4B/5B encoded 5-bit symbols.


		
0 0		0
0 1		1
1 0		1
1 1		0

Figure 4 Logical XOR table of results.

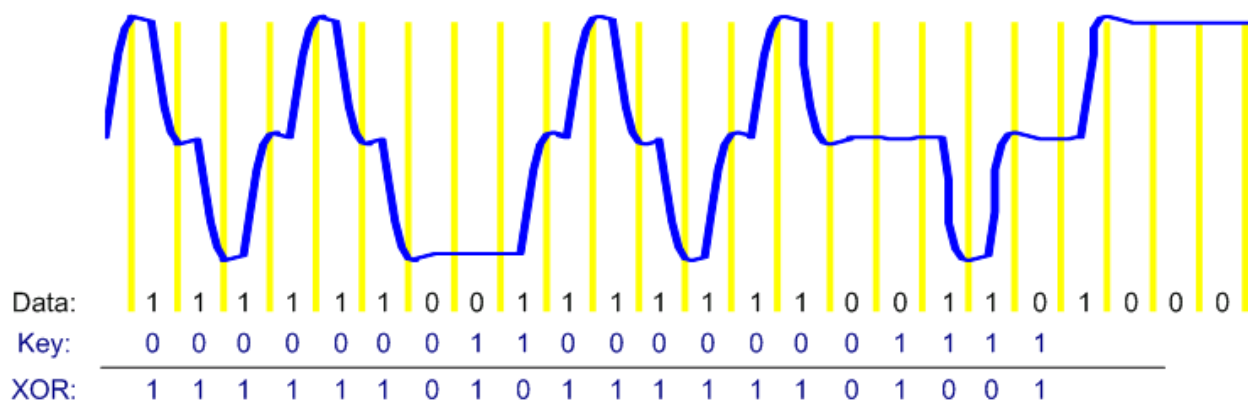


Figure 5 Logical XOR results annotated below the derived binary and the descrambler key for a short sample of the waveform.

Step 4. Look up the resulting symbol code to see whether the datastream converts to Idle symbols.

A table of symbols for 4B/5B encoding may be found in 802.3 Clause 24.2.2.1 (PDF Section 2), and also at <http://en.wikipedia.org/wiki/4B5B>. As long as the pilot program is funded, the 802.3 standard may be downloaded from <http://standards.ieee.org/getieee802/>.

This simulated waveform sample begins with Idle symbols, so you should not have to slide your set of descrambler bits over more than five times to discover an Idle symbol. If this was raw data captured from a network, you could find yourself sliding the descrambler key sequence over as many as 8,000 times (assuming a full-sized 2,000-octet Ethernet envelope frame was being transmitted at the exact moment the data sample was gathered). You will know that you are properly aligned only if you decode Idle symbols, followed by a J/K symbol sequence. During the encoding process before transmission, the first two symbols from the Preamble are substituted for the J and K symbols (see Figure 6).

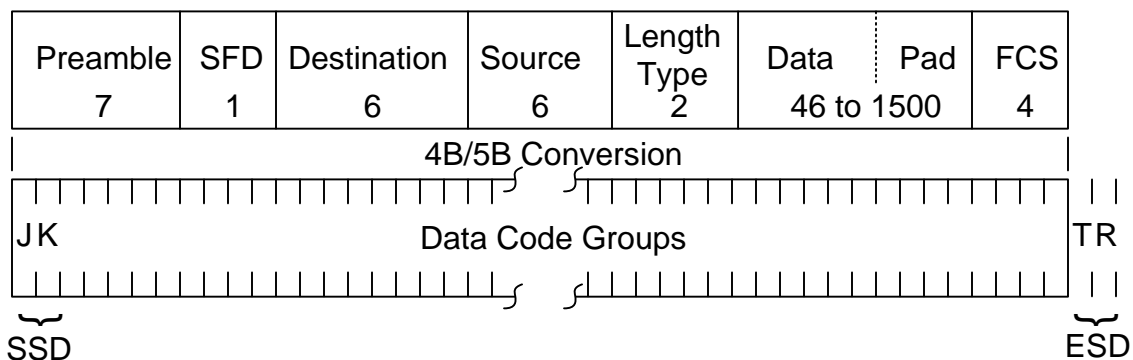


Figure 6 802.3 Ethernet frame fields, and the location of the J and K symbol substitutions.

Between data transmissions, the 100BASE-TX link is filled with Idle symbols to maintain synchronization between link partners.

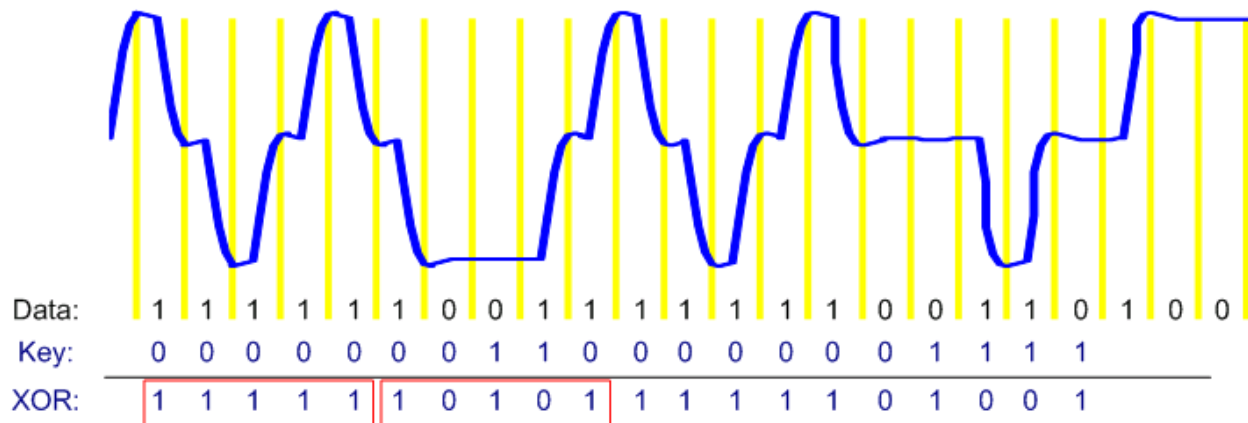


Figure 7 Symbol groupings and table lookup is performed by groups of five bits.

The proper sequence that should be seen is a series of some number of Idle symbols (11111 in binary), followed by the J and K symbols (11000 = J, 10001 = K in binary). If these symbols are not found, in that order, move the descrambler key data over one bit and try again.

In Figure 7 the first symbol appears to be an Idle, but the second symbol is neither an Idle nor a J-symbol, so the descrambler key is misaligned. Move the descrambler key bits right one bit and try again. Repeat until a J-symbol is found to be following some number of Idles.

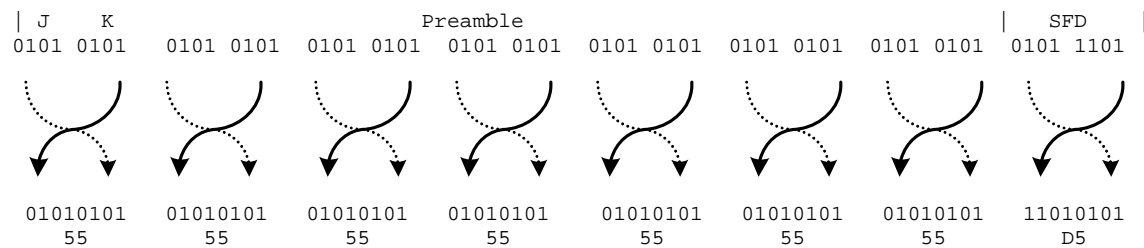
Step 5. Convert the symbols to 4-bit nibbles.

When the descramble key is properly aligned, the resulting sequence of 5-bit symbols for the entire waveform may be remapped back to 4-bit codes by symbol table lookup. The descrambler key repeats, so when all the bits of the descrambler key have been used, start over with the first bit again.

```
11000 10001 01011 01011 01011 01011 01011 01011 01011 01011 01011 01011 01011 01011 01011 11011
<Table lookup>
0101  0101  0101  0101  0101  0101  0101  0101  0101  0101  0101  0101  0101  0101  0101  1101
```

Step 6. Reorder the 4-bit nibbles and convert back to hexadecimal.

The process of encoding 4B/5B data to symbols reverses the order of the 4-bit nibbles within each octet. To restore the correct order, swap the position of each pair of 4-bit nibbles, starting with the conversion results of the J and K symbols. This will have no apparent effect until you reach the Start Frame Delimiter.



Step 7. Group according to the 802.3 field size definitions.

Refer to Figure 6 for a listing of the number of octets in each Ethernet field. Group the octets in the decoded waveform accordingly. At this time, the decoded data represents the MAC Layer request. Use the value found in the Length/Type field to discover what the next layer protocol is, and which fields are to be decoded next.

The process for converting the remainder of the Ethernet frame is described in the *Network Maintenance and Troubleshooting Guide*, Appendix B (ISBN 978-0-321-64741-2).