Undersanding Bit Order and DFDL - The MIL-STD-2045 Experience

Status of This Document

Grid Working Document (GWD)

Document Change History

2014-07-23 Created: Split out of other document.

Copyright Notice

Copyright © Open Grid Forum (2014). Some Rights Reserved. Distribution is unlimited.

Abstract

This document describes different notions of bit ordering and how they can arise and the differences among them. The MIL-STD 2045 data format is used as motivation.

Contents

[Undersanding Bit Order and DFDL - The MIL-STD-2045 Experience 1](#_Toc393905631)

[1. Introduction 3](#_Toc393905632)

[2. Terminology and Conventions 4](#_Toc393905633)

[3. Understanding Bit Order 5](#_Toc393905634)

[3.1 The Wire Model 5](#_Toc393905635)

[3.2 The Number Model 6](#_Toc393905636)

[3.3 Contrasting Wire and Number Models 7](#_Toc393905637)

[4. MIL-STD-2045 Uses the Number Model 9](#_Toc393905638)

[5. Security Considerations 10](#_Toc393905639)

[6. Glossary 11](#_Toc393905640)

[7. Contributors 12](#_Toc393905641)

[8. Intellectual Property Statement 13](#_Toc393905642)

[9. Disclaimer 14](#_Toc393905643)

[10. Full Copyright Notice 15](#_Toc393905644)

[11. References 16](#_Toc393905645)

# Introduction

There are a number of military standard binary data formats. The definitions of these formats are not generally available to the public. However, MIL-STD 2045 is a publicly available standard for a binary header used in conjunction with many other binary data formats and it illustrates most of the modeling complexities of the general family of MIL-STD binary formats.

Modeling the MIL-STD-2045 header requires that the bits of bytes are interpreted differently than is typically understood. The term "least significant bit first" is generally used to describe this.

However, this paper points out that there are really two entirely different meanings that can be assigned to "least significant bit first". That phrase, on its own, is ambiguous, and this has led to confusion about the intended format.

# Terminology and Conventions

All indexing of positions in this document are 1-based meaning the first position is position 1.

Bytes are unsigned integers with values from 0 to 255 (decimal)

Hexadecimal numbers will have the prefix "0x".

Decimal numbers will have no prefix when the context is clear, and will be explicitly labeled with (decimal) to resolve ambiguity.

Binary numbers will have the prefix "0b".

The bits of a byte are referred to by their numerical significance as the 2*n* bit, for *n* from 0 to 7. Hence, the byte value 255 = 27 + 26 + 25 + 24 + 23 + 22 + 21 + 20.

Note that the concept of a byte and the 2*n* bit of a byte are mathematical, and independent of any notion of how bits or bytes are represented in a computer system.

Wire: A serial communications link carrying 1 bit at a time.

MSB: Most Significant Bit

LSB: Least Significant Bit

Octet: synonym for Byte

# Understanding Bit Order

Most computer systems today are byte-oriented. However, there are many important data formats where the data orignated on a serial communications link. MIL-STD-2045 describes data of this kind.

When writing a byte of data onto a serial communications link, the data can be written with the least-significant-bit first, or with the most-significant-bit first.

When data is subsequently read from that serial communications link and placed back into bytes of a byte stream, one of three things can happen:

1. the reading can be symmetric with the writing so that the byte value is preserved. We call this scenario the ***Number Model***.
2. the reading can assume most-significant-bit first, so that the byte value is correct if it was written most-significant-bit first. The byte value will not match what was written if it was written least significant bit first, as the bits of the byte will be interpreted with reversed significance. We call this scenario the ***MSB First Wire Model***, or simply the ***Wire Model*** for brevity.
3. the reading can assume least-significant-bit first, so that the byte value is correct if the writing was least-significant bit first. The byte value will not match what was written if it was written most significant bit first, as the bits of the byte will be interpreted with reversed significance.

We will ignore case 3 since it is symmetric with case 2

Data being parsed by DFDL will have been captured using either the Number Model or the Wire Model. Data being unparsed by DFDL will be written out assuming either the Number Model or the Wire Model.

## The Wire Model

In the wire model, the 27 bit of a byte value in a byte stream is assumed to be first (position 1). That is, when examining a sequence of bytes in the wire model, the bytes are assumed to represent bits on a wire with the 27 bit of the byte first, and the 20 bit last.

Consider two bytes. Both have logical value 0xE2, or 0b11100010. Assume the first is written out most-significant-bit first, and the second byte is written out least-significant-bit first.

Numbering bit positions left to right, this data can be visualized as appearing on a conceptual wire. The box below is intended to represent a serial communication channel, which can be thought of as a FIFO queue. Bits are added to the queue on the right. The bits added earlier appear to the left and the first bit added appears left-most:

11100010 01000111

In the above the space is for expository purposes, the conceptual wire carries only 1 or 0 bit values.

Captured into a byte stream, the above data would contain two bytes: 0xE2 and 0x47, which we can call the *physical* bytes.

For MSB first, the physical and logical bytes have the same value.

For LSB first, the physical and logical bytes differ. The logical value is computed by reversing the bits of the physical byte.

In the above, we have underlined bits 3 to 6 and 11 to 14.

To properly construct the values of these 4-bit fields, we must know the bit order used when they were placed onto the wire. Bits 3 to 6 are within the first physical byte, which was written MSB first, so their value is 0x8. Bits 11 to 14 are within physical byte 2 which was written LSB first. To construct the logical value they must be reversed. So the 0001 consecutive bits becomes 0b1000 or 0x8.

The logical values of these two 4-bit wide fields are identical because both are the middle 4 bits of their respective physical byte, and the two bytes had the same logical value, just different representations on the wire.

## The Number Model

In the number model, the data is a sequence of byte values that are integers. The bits can only be referred to unambiguously by their place value. The 27 bit is always the bit with that logical numeric significance. Whether the 27 bit is bit position 1 or bit position 8 depends on the *bit order used to interpret the data.*

Consider two bytes in a byte stream. Both have logical value 0xE2, or 0b11100010.

Assume we interpret the first byte with bit order most-significant-bit first, and the second byte with bit order least-significant-bit first.

This table illustrates the contents of the two bytes in the byte stream, and shows how their bit positions are assigned to the individual bits by their place value:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | byte 1 | | | | | | | | byte 2 | | | | | | | |
| logical value | 0xE2 | | | | | | | | 0xE2 | | | | | | | |
| place value | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 |
| bit value | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| bit position | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 |
| bit order | MSB first | | | | | | | | LSB first | | | | | | | |

This table breaks down the logical values of the bytes into their individual bits by their place value. The bit positions these correspond to are assigned based on the bit order.

* For byte 1, which is MSB first, the bit-position row values increase as the place values of the bits decrease.
* For byte 2, which is LSB first, the bit position row values decrease as the place values of the bits decrease.

Consider bits 3 to 6 and 11 to 14 of this data. The table above is repeated below with these bits and their positions highlighted:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | byte 1 | | | | | | | | byte 2 | | | | | | | |
| logical value | 0xE2 | | | | | | | | 0xE2 | | | | | | | |
| place value | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 |
| bit value | 1 | 1 | **1** | **0** | **0** | **0** | 1 | 0 | 1 | 1 | **1** | **0** | **0** | **0** | 1 | 0 |
| bit position | 1 | 2 | **3** | **4** | **5** | **6** | 7 | 8 | 16 | 15 | **14** | **13** | **12** | **11** | 10 | 9 |
| bit order | MSB first | | | | | | | | LSB first | | | | | | | |

Bits 3 to 6 are 1000 and are interpreted as MSB first, so they correspond to the value 0b1000 or 8. Bits 11, 12, 13, and 14 are 0, 0, 0, and 1 respectively, but these are interpreted as LSB first, so we reverse them to get 0b1000 or 8 which is the correct logical value.

The number model can be thought of as a way of viewing the contents of a byte.

## Contrasting Wire and Number Models

The above illustrations show that there are two different concepts both of which can be considered bit-order. They are indistiguishable for MSB first data.

However, for LSB-first data, they are quite different.

|  |  |  |  |
| --- | --- | --- | --- |
| *Model* | *LSB Summary* | *Logical Value* | *LSB first, it appears in byte stream as:* |
| Wire Model | LSB first data are represented as different values in the byte stream. The bits are reversed. | 0xE2 | 0x47 |
| Number Model | LSB first data are interpreted by accessing the data in the reversed bit-position order.  The bits aren't reversed, rather, the extraction of the bits is done in reversed manner. | 0xE2 | 0xE2 |

The distinction of these two models can also be highlighed by considering fields occupying less than 1 full byte.

In the Wire model, a 5-bit field that is LSB first, can be followed immediately by a 2-bit field that is MSB first. Both occupy bits within the same byte. The positions of the bits are independent of the bit order of the field.

Consider the byte 0xE2. In the wire model:

11100010

The first 5 bits are 11100 in positional order, but LSB first means they must be reversed and the value is 0b00111 or 7.

The next 2 bits (positions 6 and 7) are 01, but MSB first means they need not be reversed so the value is 0b01 or 1.

In summary, in the wire model the bit order is a property of each individual field. The bit positions are absolute, and each field is defined in terms of its bit positions. Only the interpretation of the contents of the field vary with the bit order.

Contrast this with the number model. In the number model, it makes no sense to describe two fields with different bit order if they occupy the same byte. To illustrate why this makes no sense, consider again our single byte 0xE2.

The first field occupies the first 5 bits LSB first. These bits and their positions are shaded below, where the bits are given positions based on LSB first interpretation:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| logical value | 0xE2 | | | | | | | |
| place value | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 |
| bit value | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| bit position | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| bit order | LSB first | | | | | | | |

The second field is supposed to also be within this same byte, and occupies the next two bits (bits 6 and 7), but MSB first. With the bit positions interpreted as MSB-first order, these bits are shaded below:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| logical value | 0xE2 | | | | | | | |
| place value | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 |
| bit value | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| bit position | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| bit order | MSB first | | | | | | | |

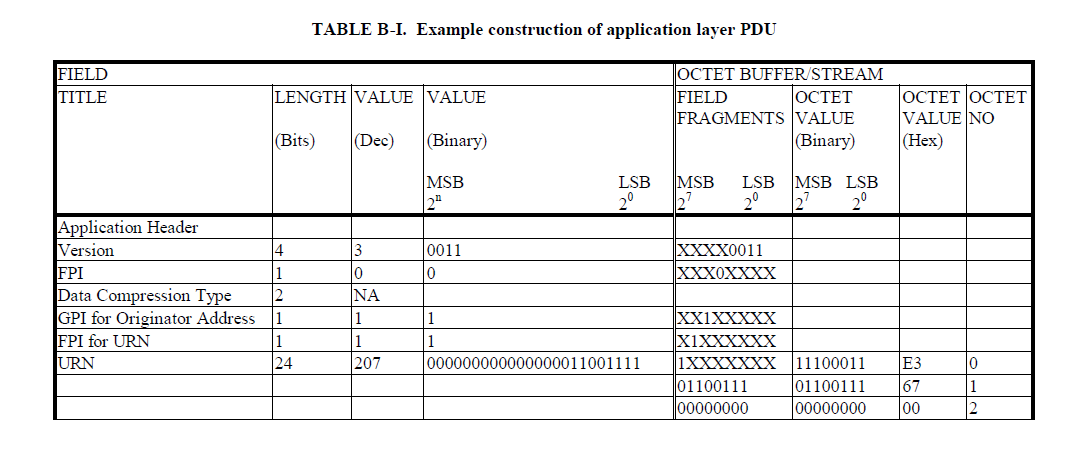
Note however, that we have only 1 byte here, and the bits we are selecting for what are supposedly two distinct adjacent fields, are actually overlapping.

In the number model, the bit order is effectively a property not of an individual field, but of a byte in the data that contains the representation of that field. Each byte of data can only be interpreted one way. Adjacent fields cannot have different bit order unless they appear in different bytes. That is, in the number model, the bit order can only change on a byte boundary.

# MIL-STD-2045 Uses the Number Model

This standard, and many like it, use the number model. All data in this standard is interpreted as LSB first, but the bit values found in the bytes of the data stream are not reversed in their bit significance.

This table is exerpted from the MIL-STD-2045 specification to illustrate example data:



In the table above, the OCTET VALUE column shows the first byte of the data stream contains value 0xE3[[1]](#footnote-2) or binary 0b11100011. This is the actual byte value as would be delivered from a byte-stream of data. This 0xE3 byte value is assembled from the bits specified in the FIELD FRAGMENTS column, where an X represents an unoccupied bit.The first 5 lines of the FIELD FRAGMENTS column are combined, and the lines have X in locations where the lines above or below provides that bit.

The first actual field of data is the Version field which occupies the first 4 bits, using LSB first interpretation. These are the least-significant 4 bits of this first byte, or 0011.

If the model was instead the wire model, the logical 0xE3 byte (0x11100011) would have to have been reversed into 11000111, i.e., value 0xC7. Then the LSB first bits would be the first 4 bits of 0xC7 or 1100, but then these would have to be reversed to get 0011, representing the value 3.

The distinction becomes clearer if we look at the second field, named FPI. This field is a single bit. It is at position 5, LSB first. The place value of the corresponding bit within the first byte is 24.

If the model was instead the wire model, the byte value of 0xC7's position 5 bit would have had place value 23.

# Security Considerations

Only data parsing is discussed in this document. Data serialization, or 'unparsing', raises one critical data security issue which is that when writing data, the contents of all of the data must be specified so as to prevent unused parts of the data being used for unintended purposes.

The DFDL standard (see [DFDL]) specifically provides for this by requiring that a fill byte be specified, the contents of which are used to fill in any unused bits or bytes of the output data.

# Glossary

Byte – 8 bits of data, also called an Octet.

DFDL - Data Format Description Language

MIL-STD-2045 – MIL Standard 2045 47001D with Change 1. See [MILSTD2045] in references.

TDML - Test Data Markup Language

# Contributors

Michael J. Beckerle (Corresponding Author)

Tresys Technology

Columbia, MD

USA

Email: mbeckerle@tresys.com

# Intellectual Property Statement

The OGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the OGF Secretariat.

The OGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights, which may cover technology that may be required to practice this recommendation. Please address the information to the OGF Executive Director.

# Disclaimer

This document and the information contained herein is provided on an “As Is” basis and the OGF disclaims all warranties, express or implied, including but not limited to any warranty that the use of the information herein will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

# Full Copyright Notice

Copyright (C) Open Grid Forum (2014). Some Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included as references to the derived portions on all such copies and derivative works. The published OGF document from which such works are derived, however, may not be modified in any way, such as by removing the copyright notice or references to the OGF or other organizations, except as needed for the purpose of developing new or updated OGF documents in conformance with the procedures defined in the OGF Document Process, or as required to translate it into languages other than English. OGF, with the approval of its board, may remove this restriction for inclusion of OGF document content for the purpose of producing standards in cooperation with other international standards bodies.

The limited permissions granted above are perpetual and will not be revoked by the OGF or its successors or assignees.

# References

|  |  |
| --- | --- |
| [Daffodil] | <https://opensource.ncsa.illinois.edu/confluence/display/DFDL/Daffodil%3A+Open+Source+DFDL> |
| [DFDL] | Michael J Beckerle, Steven M Hanson, Alan W Powell. Data Format Description Language (DFDL) v1.0 Specification. Open Grid Forum. (<http://redmine.ogf.org/dmsf/dfdl-wg>)  Forthcoming Update: GFD-P-R.207 (2014)  Obsolete: GFD-P-R.174. January 2011. |
| [DFDLCharset] | Michael J Beckerle, DFDL-Specific Character Set Encodings, Open Grid Forum 2014  Forthcoming |
| [MILSTD2045] | CONNECTIONLESS DATA TRANSFER APPLICATION LAYER STANDARD, MIL-STD-2045-47001D w/CHANGE 1, 23 June 2008 (available publicly from US Dept. of Defense at <http://assistdocs.com/>) |
| [TDML] | Michael J. Beckerle – Test Data Markup Language. (<http://redmine.ogf.org/dmsf_files/13238>) |

1. The above discussion used 0xE2 because that has an additional asymmetry that 0xE3 lacks, which is that the last two bits of 0xE3 (least significant 2 bits) are 11, so if they are accidently reversed one cannot tell. This discussion of MIL-STD-2045 uses 0xE3 because that is what is used in the MIL-STD-2045 spec document. [↑](#footnote-ref-2)