Floating Point Formats for Machine Learning

IEEE Working Group P3109 interim report
on 8-bit binary floating-point formats.

First public release: 18 September 2023

This version: 4 October 2023

Questions and comments by email to the P3109 Secretary <jeffrey.sarnoff@ieee.org>,
or via GitHub issues at https://github.com/P3109/Public

V1.0.3 revised 4 Oct 2023

Table of Contents

1. Introduction 3
2. Typographical conventions and notation 3
3. Values 4
4. Subnormals 5
5. Not a number (NaN) 5
6. Zero 5
7. Infinities 6
8. Extremal Values 6
9. Classification Operators 7
10. Comparison Operators 7

Appendix A: Numerical Examples 9

Appendix B: Comparison table 10

Appendix C: Value Tables 11

Value table: binary8p3 12

Value table: binary8p4 13

Value table: binary8p5 14

Bibliography 15

1. Introduction

This document represents the results of discussions and decisions made by IEEE working group P3109, "Standard for Arithmetic Formats for Machine Learning". The Project Authorization Request (PAR) for P3109 defines the scope, need, and stakeholders as follows:

> ***Scope of proposed standard**: This standard defines a binary arithmetic and data format for machine learning-optimized domains. It also specifies the default handling of exceptions occurring in this arithmetic. This standard provides a consistent and flexible arithmetic framework optimized for Machine Learning Systems (MLS) in hardware and/or software implementations to minimize the work required to make MLS interoperable with each other as well as other dependent systems. This standard is aligned with the IEEE Std 754-2019 for Floating-Point Arithmetic.*
>
> ***Need for the Project**: Machine Learning Systems have different arithmetic requirements from most other domains. Precisions tend to be lower, and accuracy is measured in dimensions other than just numerical (e.g. inference accuracy). Furthermore, Machine Learning Systems often are integrated into mission-critical and safety-critical systems. With no standards specifically addressing these needs, Machine Learning Systems are built with inconsistent expectations and assumptions that hinder the compatibility and reuse of machine learning hardware, software, and training data.*
>
> ***Stakeholders for the Standard**: System developers, vendors, and users of machine learning applications across many industries and interests including but not limited to compute, storage, medical, telecommunications, e-commerce, fleet-management, automotive, robotics, and security.*

The scope of this interim release is interchange formats only. The working group continues to deliberate on the specification of operations.

2. Typographical conventions and notation

Bold text describes the decisions and specifications of this document.

Non-bold text is background material, typically providing rationale and arguments representing the discussions of the WG leading to a decision and specification.

This document specifies 8-bit floating-point interchange formats (binary formats) and associated operations. Binary formats are parameterized by their width, the number of bits spanned in memory (here, 8); and their precision (p), the number of bits spanned by the true significand (one more than the number of explicit "mantissa" bits).

The formats defined herein shall be referred to as "binary8" formats, and further parametrized by precision p yielding names "binary8pp".

For example, "binary8p3" is a format with 3 bits of precision, hence a 2-bit mantissa and a 5-bit exponent field.

3. Values

This section describes the set of values that a binary8 format shall represent. The universe of values in existing floating point usage encompasses some finite real numerical values, the nonfinite numerical values positive and negative infinity (`-Inf`, `+Inf`), the non-numeric not-a-number values (`NaN`, $\texttt{NaN}_1$, …), and negative zero (`-0`). The value set for each binary8 format specifies the set of values that are available in that format.

Each binary format shall be associated with a unique encoding. An 8-bit binary encoding is a mapping from 8-bit strings to values. Some of these mappings are included as an Appendix.

The four special values (0, +Inf, -Inf, NaN) have encodings that are shared by all binary8 formats.

Table 1 – Encoding Special Values

Value	Hexadecimal Encoding	Bit Sequence
Zero	0x00	0000 0000
Positive Infinity (+Inf)	0x7F	0111 1111
Negative Infinity (-Inf)	0xFF	1111 1111
Not a Number (NaN)	0x80	1000 0000

The set of finite floating-point numbers representable with a binary format is determined by two parameters:

- Precision p, the number of digits in the significand including the implicit leading bit
- Maximum exponent $emax$, the exponent of the largest finite value

IEEE-754 2019 includes the radix b and $emin$ in the list of format parameters. This document covers binary (radix 2) formats only. The parameter $emin$ is determinable from other parameters, so is not a format-defining parameter.

P3109 formats shall define emax = $2^{8-p-1} - 1$. In IEEE-754, $emax$ was consistently chosen across formats to be $2^{w-1} - 1$, where w is the exponent field width in bits. This choice has the consequence that the binary8pp value sets are subsets of the IEEE-754 binary16 value set for $p > 2$, and a near-symmetric distribution of values below and above the value 1.

Table 2 – Parameters for Binary Formats *

Parameter	binary8pp	binary8p5	binary8p4	binary8p3	binary16	binary32	binary64
k, storage width in bits	**8**	**8**	**8**	**8**	**16**	**32**	**64**
p, precision in bits	**p**	**5**	**4**	**3**	**11**	**24**	**53**
$emax$, max exponent	**$2^{8-p-1} - 1$**	**3**	**7**	**15**	**15**	**127**	**1023**
Derived parameters:							
w, exponent field width	$8 - p$	3	4	5	5	8	11
Exponent bias	$emax + 1$	4	8	16	15	127	1023
Sign bit		1	1	1	1	1	1

t, trailing significand field width in bits	$p-1$	4	3	2	10	23	52

* Adapted from table 3.5 of IEEE-754 (2019), and extended to include proposed binary8 formats.

4. Subnormals

Binary8 value sets shall include subnormals.

The IEEE-754 value sets include subnormals. A value with trailing significand field m and exponent e is interpreted as $1.m \times 2^{e-b}$ except when all bits of the exponent bitfield are 0, in which case, the value is $0.m \times 2^{e-b+1}$.

When training models, it is common to represent near-zero values for gradients. Subnormal numbers induce equal quantization steps around zero; this expands the reach of binary8 trainable models. In statistical applications, the subnormal range is useful for uniform-like distributions, being uniform around zero. This also supports working with Gaussian-like distributions where numbers around zero are more probable.

5. Not a number (NaN)

Binary8 value sets shall include exactly one NaN, which shall not signal.

Other floating point formats define several NaN values, denoted (`NaN`, $\texttt{NaN}_1$, …). NaNs are returned from operations with results outside the set of values. For example, `DIV(0,0)`, or `ADD(Inf, -Inf)`. Multiple NaN encodings are used in other formats to allow different exceptional conditions to be distinguished.

In the context of machine learning systems, uses of NaN include:

- Debugging of code running on the accelerator. In AI accelerators, exceptions may be difficult or expensive to convey back to user code, so it is common practice to allow NaN values to propagate through calculations in order to indicate that an error has occurred.
- Usage as a "notable value" indicator. In some datasets, for example tabular data, values may be missing. It is useful to use a value outside of the normal numeric range to indicate the position of these values. Particularly where memory usage is a concern, as may be expected in float8 applications, the use of a separate "mask" array, or a list of indices, imposes an extra memory overhead. In some cases, an Inf can be used as a missing value, but given the restricted range of float8, it is likely that infinity shall be used as a separate indicator of "larger/smaller than +/- MaxFloat".
- The use of multiple NaN payloads is not unknown in statistical code (e.g. the R system has NaN and N/A), but it is not widely used, and in the context of float8, multiple NaNs imposes either additional hardware complexity (using only a subset of the significand range), or a large reduction in encoding space (e.g. 8 codes for E5, 16 codes for E4, 32 codes for E3).

6. Zero

Binary8 formats shall have exactly one zero. This zero value is non-negative.

The inclusion of negative zero would incur the cost of an additional code point. Given the decision to encode only a single NaN, placing that NaN at the negative zero code point enables the strictly positive and strictly negative number ranges to be symmetric.

A key rationale for inclusion of -0 in IEEE-754 was the consistent implementation of branch cuts in the atan2 function [1] [2]. Although the atan function is common in deep learning, it is used as an activation function, rather than a

trigonometric operation, and the $\mathrm{atan2}$ function is not known in deep learning applications. Furthermore, it is not expected that this standard shall define either atan or $\mathrm{atan2}$.

A secondary rationale for inclusion of -0 is the hardware simplification offered by its presence in the implementation of sign/magnitude arithmetic. However, the existence of in-market implementations is evidence that the small hardware simplification has not been sufficient to balance the loss of one code point.

It might be considered that the use of integer comparisons in sorting would argue against the placing of NaN at the negative zero code point. For example, the JAX machine learning framework is known to sort using integer comparison [link]. However such sorting still requires $O(n)$ preprocessing and postprocessing steps to enable the use of twos-complement integer comparison, and already has special treatment of NaN and -0, so eliminating -0 and placing NaN in the -0 position imposes negligible additional burden. As an aside, it is noted that sorting using comparison operations, as typically defined, is undefined in presence of NaN, however the existing practice is to sort NaN (e.g. using totalOrder) to the end of the array, and this remains permitted, at no additional cost.

7. Infinities

Binary8 formats shall include positive and negative infinities.

This decision causes a reduction in dynamic range (252 values rather than 254), but offers improved numerical robustness in important machine learning use cases. Two generic classes of such usage are:

- mask values, for example, in Transformer models in machine learning, [ref].
- representation of overflow.

As illustrated in Appendix A, both usages are facilitated by the presence of infinity.

8. Extremal Values

It is practical to offer extremal finite values for supported 8-bit binary interchange formats. Following IEEE 754-2019 naming patterns, we adopt: maxNormal(T), minNormal(T), minSubnormal(T) where T is a binary8 format. For example: maxNormal(binary8p4) == 7/4 * (2^7), minNormal(binary8p5) == 1 / (2^3).

	maxNormal	minNormal	minSubnormal
binary8p3	3/2 * (2^15)	1 / (2^15)	1 / (2^17)
binary8p4	7/4 * (2^7)	1 / (2^7)	1 / (2^10)
binary8p5	15/8 * (2^3)	1 / (2^3)	1 / (2^7)
binary8p6	31/16 * (2^1)	1 / (2^1)	1 / (2^6)

Table 3: Extremal Values

9. Classification Operators

Conforming implementations shall provide these classification predicates and the classifier function. The classification predicates and the classifier function shall not signal exceptions.

predicate	Definition	predicate	definition
isZero	Iff *x* is 0	isNaN	Iff *x* is NaN
isInfinite	iff *x* is infinite	isFinite	Iff *x* is zero, subnormal or normal
isNormal	Iff *x* is normal	isSubnormal	Iff *x* is subnormal
isSignMinus	Iff *x* has a negative sign [1]		
isCanonical	True [2]	isSignaling	False [3]

Table 4: Classification Predicates

[1] isSignMinus(NaN) is True: all binary8 formats encode NaN as 0x80 (0b10000000).
[2] There are no non-canonical binary8 interchange formats.
[3] All binary8 formats have one NaN; it does not signal.

The Classifier function

enum class(*x*)
    NaN
    Zero
    positiveInfinity
    positiveNormal
    positiveSubnormal
    negativeInfinity
    negativeNormal
    negativeSubnormal
end

10. Comparison Operators

Conforming implementations shall provide these comparison operators and the totalOrder(*x*, *y*) function.

Comparison operators are two argument predicates and their negations that return { True, False }. Comparisons shall not raise exceptions. Comparisons are either ordered or unordered. A comparison is unordered iff either argument is NaN. All other comparisons are ordered.

For { $=$, $>$, $\geq$, $<$, $\leq$, $\lessgtr$ }, if any argument is NaN the result is False.
For { $\neq$, $\not>$, $\not\geq$, $\not<$, $\not\leq$, $\not\lessgtr$ }, if any argument is NaN the result is True.
Otherwise, the result of a comparison shall match the mathematical result.

Table 5: Comparison Predicates and Negations

math symbol	**predicate** *true relations*	**math** symbol	**negation** *true relations*
=	CompareEqual *equal*	≠, NOT =	CompareNotEqual *less than, greater than, unordered*
>	CompareGreater *greater than*	≯, NOT >	CompareNotGreater *less than, equal, unordered*
≥	CompareGreaterEqual *equal, greater than*	≱, NOT ≥	CompareLessUnordered *less than, unordered*
<	CompareLess *less than*	≮, NOT <	CompareNotLess *greater than, equal, unordered*
≤	CompareLessEqual *less than, equal*	≰, NOT ≤	CompareGreaterUnordered *greater than, unordered*
⩽	CompareOrdered *less than, equal, greater than*	⩽̸, NOT ⩽	CompareUnordered *unordered*

The totalOrder predicate

totalOrder(*x, y*) provides a total ordering over each binary8 format's value set. It shall not raise any exceptions. totalOrder(*x*, *y*) shall return { True, False } in accord with the logic given below.

```
boolean totalOrder( x, y )
   if ! ( isNaN(x) || isNaN(y) )
      return compareLessEqual( x,  y )
   else
      return isNaN( x ) [1]
   end
end
```

[1] All binary8 formats encode NaN as 0x80. The most significant bit is set, so, following 754, it is as -NaN.

Logical operations used within totalOrder()

! is the logical negation operator: !true == false, !false == true.
|| is the short-circuiting, left-associative logical OR.
- if a is true, a || b returns true without evaluating b.
- if a evaluates as false, a || b returns the evaluation of b.
- (a || b || c) evaluates as (a || b) || c.

Appendix A: Numerical Examples

Mask Values

A common use for ∞ is to create masks, for example, in Transformer models in machine learning, [ref]. These values, assembled in mask matrix M with values $M_{ij} \in \{0, -\infty\}$ are typically be added to computed values A, in a computation such as:

$$\log\left(\text{sum}\left(\exp(\tau * (A + M))\right)\right)$$

where τ is a "temperature" or "base" parameter [ref]. This calculation depends on the property that $\exp(\tau * A_{ij} - \infty) = 0$. It is clear that where M_{ij} is a large float (e.g. 480), then $\exp(-480)$ is an extremely small number, clearly much closer to zero than to any other value. However, careful implementations do not execute the calculation as written, and instead fuse the $\log(\text{sum}(\exp(\boldsymbol{v})))$ operation into a single operation $\text{logsumexp}(\boldsymbol{v})$, whose implementation makes use of the identity

$$\text{logsumexp}(\boldsymbol{v}) = \text{logsumexp}(\boldsymbol{v} - \max(\boldsymbol{v})) + \max(\boldsymbol{v})$$

Without the "sticky" properties of Inf, this would produce incorrect answers. For example, in a format where MaxFloat=240 without Inf, and MaxFloat=224 with Inf:

$$\text{logsumexp}(\tau * [-224, -\infty]) \rightarrow \text{logsumexp}(\tau * [0, -\infty])$$

while

$$\text{logsumexp}(\tau * [-224, -240]) \rightarrow \text{logsumexp}(\tau * [0, -16])$$

If $\tau = 1$ and all calculations are done in 8-bit floating point, then the answer will be the same, as $\exp(-16) = 0$, but if τ is small, or calculations are done in mixed precision, as is common with 8-bit floating point, the loss of "stickiness" shall silently yield unexpected answers. It is not expected that the full calculation shall be done in 8-bit floating point, but the subtraction of the maximum value (and computation of the maximum) might reasonably be in 8-bit floating point.

Overflow to Infinity

A second use of infinity is to indicate overflow on conversion to the binary8 type. Existing implementations offer several behaviours on overflow: overflow to infinity, saturation to MaxFloat, and overflow to NaN. The existence of a code point for infinity allows any of these options to be implemented in a given instantiation, while removing the code point removes the possibility of implementing the first.

Appendix B: Comparison table

This table summarizes the points of difference and agreement between the formats proposed in this document and a number of existing formats, some of which have hardware implementations.

OCP: Open Compute Platform [3], describing hardware implementations including nVidia, Intel, and ARM.
AGQ: AMD, Graphcore, Qualcomm [4], implemented in Graphcore's C600 product.
TSL: Tesla Dojo Technology, A Guide to Tesla's Configurable Floating Point Formats & Arithmetic

Format	P3109			OCP		AGQ		TSL	
Subformat	P3	P4	P5	E5	E4	E5	E4	E4	E5
Special values shared by all subformats	Y			N		Y		N	
Exactly one NaN	Y			N		Y		Y	
Positive and negative infinity	Y			N	Y	N		N	
Include negative zero	N			Y		N		N	
Max exponent $emax$	15	7	3	15	8	15	7	N/A	N/A

Appendix C: Value Tables

Value tables mapping 8-bit strings to value sets are provided in this section.

A typical entry is of the form:

```
HEX  BINARY     = BINARY_FLOAT  = DECIMAL
0x01 0_00000_01 = +0b0.01*2^-15 = 7.62939453125e-06
```

Where the fields are interpreted as follows:

HEX	Hexadecimal encoding of the code point
BINARY	Binary expansion of the code point, with underscores separating sign_exponent_significand
BINARY_FLOAT	The precise float value as a binary fraction followed by 2^*e* with decimal exponent *e*
DECIMAL	The decimal expansion of the value

10.1. Value table: binary8p3

0x00 = 0_00000_00 = +0b0.00*2^-15 = 0	0x40 = 0_10000_00 = +0b1.00*2^0 = 1	0x80 = 1_00000_00 = NaN	0xc0 = 1_10000_00 = -0b1.00*2^0 = -1
0x01 = 0_00000_01 = +0b0.01*2^-15 = 7.62939e-06	0x41 = 0_10000_01 = +0b1.01*2^0 = 1.25	0x81 = 1_00000_01 = -0b0.01*2^-15 = -7.62939e-06	0xc1 = 1_10000_01 = -0b1.01*2^0 = -1.25
0x02 = 0_00000_10 = +0b0.10*2^-15 = 1.52588e-05	0x42 = 0_10000_10 = +0b1.10*2^0 = 1.5	0x82 = 1_00000_10 = -0b0.10*2^-15 = -1.52588e-05	0xc2 = 1_10000_10 = -0b1.10*2^0 = -1.5
0x03 = 0_00000_11 = +0b0.11*2^-15 = 2.28882e-05	0x43 = 0_10000_11 = +0b1.11*2^0 = 1.75	0x83 = 1_00000_11 = -0b0.11*2^-15 = -2.28882e-05	0xc3 = 1_10000_11 = -0b1.11*2^0 = -1.75
0x04 = 0_00001_00 = +0b1.00*2^-15 = 3.05176e-05	0x44 = 0_10001_00 = +0b1.00*2^1 = 2	0x84 = 1_00001_00 = -0b1.00*2^-15 = -3.05176e-05	0xc4 = 1_10001_00 = -0b1.00*2^1 = -2
0x05 = 0_00001_01 = +0b1.01*2^-15 = 3.8147e-05	0x45 = 0_10001_01 = +0b1.01*2^1 = 2.5	0x85 = 1_00001_01 = -0b1.01*2^-15 = -3.8147e-05	0xc5 = 1_10001_01 = -0b1.01*2^1 = -2.5
0x06 = 0_00001_10 = +0b1.10*2^-15 = 4.57764e-05	0x46 = 0_10001_10 = +0b1.10*2^1 = 3	0x86 = 1_00001_10 = -0b1.10*2^-15 = -4.57764e-05	0xc6 = 1_10001_10 = -0b1.10*2^1 = -3
0x07 = 0_00001_11 = +0b1.11*2^-15 = 5.34058e-05	0x47 = 0_10001_11 = +0b1.11*2^1 = 3.5	0x87 = 1_00001_11 = -0b1.11*2^-15 = -5.34058e-05	0xc7 = 1_10001_11 = -0b1.11*2^1 = -3.5
0x08 = 0_00010_00 = +0b1.00*2^-14 = 6.10352e-05	0x48 = 0_10010_00 = +0b1.00*2^2 = 4	0x88 = 1_00010_00 = -0b1.00*2^-14 = -6.10352e-05	0xc8 = 1_10010_00 = -0b1.00*2^2 = -4
0x09 = 0_00010_01 = +0b1.01*2^-14 = 7.62939e-05	0x49 = 0_10010_01 = +0b1.01*2^2 = 5	0x89 = 1_00010_01 = -0b1.01*2^-14 = -7.62939e-05	0xc9 = 1_10010_01 = -0b1.01*2^2 = -5
0x0a = 0_00010_10 = +0b1.10*2^-14 = 9.15527e-05	0x4a = 0_10010_10 = +0b1.10*2^2 = 6	0x8a = 1_00010_10 = -0b1.10*2^-14 = -9.15527e-05	0xca = 1_10010_10 = -0b1.10*2^2 = -6
0x0b = 0_00010_11 = +0b1.11*2^-14 = 0.000106812	0x4b = 0_10010_11 = +0b1.11*2^2 = 7	0x8b = 1_00010_11 = -0b1.11*2^-14 = -0.000106812	0xcb = 1_10010_11 = -0b1.11*2^2 = -7
0x0c = 0_00011_00 = +0b1.00*2^-13 = 0.00012207	0x4c = 0_10011_00 = +0b1.00*2^3 = 8	0x8c = 1_00011_00 = -0b1.00*2^-13 = -0.00012207	0xcc = 1_10011_00 = -0b1.00*2^3 = -8
0x0d = 0_00011_01 = +0b1.01*2^-13 = 0.000152588	0x4d = 0_10011_01 = +0b1.01*2^3 = 10	0x8d = 1_00011_01 = -0b1.01*2^-13 = -0.000152588	0xcd = 1_10011_01 = -0b1.01*2^3 = -10
0x0e = 0_00011_10 = +0b1.10*2^-13 = 0.000183105	0x4e = 0_10011_10 = +0b1.10*2^3 = 12	0x8e = 1_00011_10 = -0b1.10*2^-13 = -0.000183105	0xce = 1_10011_10 = -0b1.10*2^3 = -12
0x0f = 0_00011_11 = +0b1.11*2^-13 = 0.000213623	0x4f = 0_10011_11 = +0b1.11*2^3 = 14	0x8f = 1_00011_11 = -0b1.11*2^-13 = -0.000213623	0xcf = 1_10011_11 = -0b1.11*2^3 = -14
0x10 = 0_00100_00 = +0b1.00*2^-12 = 0.000244141	0x50 = 0_10100_00 = +0b1.00*2^4 = 16	0x90 = 1_00100_00 = -0b1.00*2^-12 = -0.000244141	0xd0 = 1_10100_00 = -0b1.00*2^4 = -16
0x11 = 0_00100_01 = +0b1.01*2^-12 = 0.000305176	0x51 = 0_10100_01 = +0b1.01*2^4 = 20	0x91 = 1_00100_01 = -0b1.01*2^-12 = -0.000305176	0xd1 = 1_10100_01 = -0b1.01*2^4 = -20
0x12 = 0_00100_10 = +0b1.10*2^-12 = 0.000366211	0x52 = 0_10100_10 = +0b1.10*2^4 = 24	0x92 = 1_00100_10 = -0b1.10*2^-12 = -0.000366211	0xd2 = 1_10100_10 = -0b1.10*2^4 = -24
0x13 = 0_00100_11 = +0b1.11*2^-12 = 0.000427246	0x53 = 0_10100_11 = +0b1.11*2^4 = 28	0x93 = 1_00100_11 = -0b1.11*2^-12 = -0.000427246	0xd3 = 1_10100_11 = -0b1.11*2^4 = -28
0x14 = 0_00101_00 = +0b1.00*2^-11 = 0.000488281	0x54 = 0_10101_00 = +0b1.00*2^5 = 32	0x94 = 1_00101_00 = -0b1.00*2^-11 = -0.000488281	0xd4 = 1_10101_00 = -0b1.00*2^5 = -32
0x15 = 0_00101_01 = +0b1.01*2^-11 = 0.000610352	0x55 = 0_10101_01 = +0b1.01*2^5 = 40	0x95 = 1_00101_01 = -0b1.01*2^-11 = -0.000610352	0xd5 = 1_10101_01 = -0b1.01*2^5 = -40
0x16 = 0_00101_10 = +0b1.10*2^-11 = 0.000732422	0x56 = 0_10101_10 = +0b1.10*2^5 = 48	0x96 = 1_00101_10 = -0b1.10*2^-11 = -0.000732422	0xd6 = 1_10101_10 = -0b1.10*2^5 = -48
0x17 = 0_00101_11 = +0b1.11*2^-11 = 0.000854492	0x57 = 0_10101_11 = +0b1.11*2^5 = 56	0x97 = 1_00101_11 = -0b1.11*2^-11 = -0.000854492	0xd7 = 1_10101_11 = -0b1.11*2^5 = -56
0x18 = 0_00110_00 = +0b1.00*2^-10 = 0.000976562	0x58 = 0_10110_00 = +0b1.00*2^6 = 64	0x98 = 1_00110_00 = -0b1.00*2^-10 = -0.000976562	0xd8 = 1_10110_00 = -0b1.00*2^6 = -64
0x19 = 0_00110_01 = +0b1.01*2^-10 = 0.0012207	0x59 = 0_10110_01 = +0b1.01*2^6 = 80	0x99 = 1_00110_01 = -0b1.01*2^-10 = -0.0012207	0xd9 = 1_10110_01 = -0b1.01*2^6 = -80
0x1a = 0_00110_10 = +0b1.10*2^-10 = 0.00146484	0x5a = 0_10110_10 = +0b1.10*2^6 = 96	0x9a = 1_00110_10 = -0b1.10*2^-10 = -0.00146484	0xda = 1_10110_10 = -0b1.10*2^6 = -96
0x1b = 0_00110_11 = +0b1.11*2^-10 = 0.00170898	0x5b = 0_10110_11 = +0b1.11*2^6 = 112	0x9b = 1_00110_11 = -0b1.11*2^-10 = -0.00170898	0xdb = 1_10110_11 = -0b1.11*2^6 = -112
0x1c = 0_00111_00 = +0b1.00*2^-9 = 0.00195312	0x5c = 0_10111_00 = +0b1.00*2^7 = 128	0x9c = 1_00111_00 = -0b1.00*2^-9 = -0.00195312	0xdc = 1_10111_00 = -0b1.00*2^7 = -128
0x1d = 0_00111_01 = +0b1.01*2^-9 = 0.00244141	0x5d = 0_10111_01 = +0b1.01*2^7 = 160	0x9d = 1_00111_01 = -0b1.01*2^-9 = -0.00244141	0xdd = 1_10111_01 = -0b1.01*2^7 = -160
0x1e = 0_00111_10 = +0b1.10*2^-9 = 0.00292969	0x5e = 0_10111_10 = +0b1.10*2^7 = 192	0x9e = 1_00111_10 = -0b1.10*2^-9 = -0.00292969	0xde = 1_10111_10 = -0b1.10*2^7 = -192
0x1f = 0_00111_11 = +0b1.11*2^-9 = 0.00341797	0x5f = 0_10111_11 = +0b1.11*2^7 = 224	0x9f = 1_00111_11 = -0b1.11*2^-9 = -0.00341797	0xdf = 1_10111_11 = -0b1.11*2^7 = -224
0x20 = 0_01000_00 = +0b1.00*2^-8 = 0.00390625	0x60 = 0_11000_00 = +0b1.00*2^8 = 256	0xa0 = 1_01000_00 = -0b1.00*2^-8 = -0.00390625	0xe0 = 1_11000_00 = -0b1.00*2^8 = -256
0x21 = 0_01000_01 = +0b1.01*2^-8 = 0.00488281	0x61 = 0_11000_01 = +0b1.01*2^8 = 320	0xa1 = 1_01000_01 = -0b1.01*2^-8 = -0.00488281	0xe1 = 1_11000_01 = -0b1.01*2^8 = -320
0x22 = 0_01000_10 = +0b1.10*2^-8 = 0.00585938	0x62 = 0_11000_10 = +0b1.10*2^8 = 384	0xa2 = 1_01000_10 = -0b1.10*2^-8 = -0.00585938	0xe2 = 1_11000_10 = -0b1.10*2^8 = -384
0x23 = 0_01000_11 = +0b1.11*2^-8 = 0.00683594	0x63 = 0_11000_11 = +0b1.11*2^8 = 448	0xa3 = 1_01000_11 = -0b1.11*2^-8 = -0.00683594	0xe3 = 1_11000_11 = -0b1.11*2^8 = -448
0x24 = 0_01001_00 = +0b1.00*2^-7 = 0.0078125	0x64 = 0_11001_00 = +0b1.00*2^9 = 512	0xa4 = 1_01001_00 = -0b1.00*2^-7 = -0.0078125	0xe4 = 1_11001_00 = -0b1.00*2^9 = -512
0x25 = 0_01001_01 = +0b1.01*2^-7 = 0.00976562	0x65 = 0_11001_01 = +0b1.01*2^9 = 640	0xa5 = 1_01001_01 = -0b1.01*2^-7 = -0.00976562	0xe5 = 1_11001_01 = -0b1.01*2^9 = -640
0x26 = 0_01001_10 = +0b1.10*2^-7 = 0.0117188	0x66 = 0_11001_10 = +0b1.10*2^9 = 768	0xa6 = 1_01001_10 = -0b1.10*2^-7 = -0.0117188	0xe6 = 1_11001_10 = -0b1.10*2^9 = -768
0x27 = 0_01001_11 = +0b1.11*2^-7 = 0.0136719	0x67 = 0_11001_11 = +0b1.11*2^9 = 896	0xa7 = 1_01001_11 = -0b1.11*2^-7 = -0.0136719	0xe7 = 1_11001_11 = -0b1.11*2^9 = -896
0x28 = 0_01010_00 = +0b1.00*2^-6 = 0.015625	0x68 = 0_11010_00 = +0b1.00*2^10 = 1024	0xa8 = 1_01010_00 = -0b1.00*2^-6 = -0.015625	0xe8 = 1_11010_00 = -0b1.00*2^10 = -1024
0x29 = 0_01010_01 = +0b1.01*2^-6 = 0.0195312	0x69 = 0_11010_01 = +0b1.01*2^10 = 1280	0xa9 = 1_01010_01 = -0b1.01*2^-6 = -0.0195312	0xe9 = 1_11010_01 = -0b1.01*2^10 = -1280
0x2a = 0_01010_10 = +0b1.10*2^-6 = 0.0234375	0x6a = 0_11010_10 = +0b1.10*2^10 = 1536	0xaa = 1_01010_10 = -0b1.10*2^-6 = -0.0234375	0xea = 1_11010_10 = -0b1.10*2^10 = -1536
0x2b = 0_01010_11 = +0b1.11*2^-6 = 0.0273438	0x6b = 0_11010_11 = +0b1.11*2^10 = 1792	0xab = 1_01010_11 = -0b1.11*2^-6 = -0.0273438	0xeb = 1_11010_11 = -0b1.11*2^10 = -1792
0x2c = 0_01011_00 = +0b1.00*2^-5 = 0.03125	0x6c = 0_11011_00 = +0b1.00*2^11 = 2048	0xac = 1_01011_00 = -0b1.00*2^-5 = -0.03125	0xec = 1_11011_00 = -0b1.00*2^11 = -2048
0x2d = 0_01011_01 = +0b1.01*2^-5 = 0.0390625	0x6d = 0_11011_01 = +0b1.01*2^11 = 2560	0xad = 1_01011_01 = -0b1.01*2^-5 = -0.0390625	0xed = 1_11011_01 = -0b1.01*2^11 = -2560
0x2e = 0_01011_10 = +0b1.10*2^-5 = 0.046875	0x6e = 0_11011_10 = +0b1.10*2^11 = 3072	0xae = 1_01011_10 = -0b1.10*2^-5 = -0.046875	0xee = 1_11011_10 = -0b1.10*2^11 = -3072
0x2f = 0_01011_11 = +0b1.11*2^-5 = 0.0546875	0x6f = 0_11011_11 = +0b1.11*2^11 = 3584	0xaf = 1_01011_11 = -0b1.11*2^-5 = -0.0546875	0xef = 1_11011_11 = -0b1.11*2^11 = -3584
0x30 = 0_01100_00 = +0b1.00*2^-4 = 0.0625	0x70 = 0_11100_00 = +0b1.00*2^12 = 4096	0xb0 = 1_01100_00 = -0b1.00*2^-4 = -0.0625	0xf0 = 1_11100_00 = -0b1.00*2^12 = -4096
0x31 = 0_01100_01 = +0b1.01*2^-4 = 0.078125	0x71 = 0_11100_01 = +0b1.01*2^12 = 5120	0xb1 = 1_01100_01 = -0b1.01*2^-4 = -0.078125	0xf1 = 1_11100_01 = -0b1.01*2^12 = -5120
0x32 = 0_01100_10 = +0b1.10*2^-4 = 0.09375	0x72 = 0_11100_10 = +0b1.10*2^12 = 6144	0xb2 = 1_01100_10 = -0b1.10*2^-4 = -0.09375	0xf2 = 1_11100_10 = -0b1.10*2^12 = -6144
0x33 = 0_01100_11 = +0b1.11*2^-4 = 0.109375	0x73 = 0_11100_11 = +0b1.11*2^12 = 7168	0xb3 = 1_01100_11 = -0b1.11*2^-4 = -0.109375	0xf3 = 1_11100_11 = -0b1.11*2^12 = -7168
0x34 = 0_01101_00 = +0b1.00*2^-3 = 0.125	0x74 = 0_11101_00 = +0b1.00*2^13 = 8192	0xb4 = 1_01101_00 = -0b1.00*2^-3 = -0.125	0xf4 = 1_11101_00 = -0b1.00*2^13 = -8192
0x35 = 0_01101_01 = +0b1.01*2^-3 = 0.15625	0x75 = 0_11101_01 = +0b1.01*2^13 = 10240	0xb5 = 1_01101_01 = -0b1.01*2^-3 = -0.15625	0xf5 = 1_11101_01 = -0b1.01*2^13 = -10240
0x36 = 0_01101_10 = +0b1.10*2^-3 = 0.1875	0x76 = 0_11101_10 = +0b1.10*2^13 = 12288	0xb6 = 1_01101_10 = -0b1.10*2^-3 = -0.1875	0xf6 = 1_11101_10 = -0b1.10*2^13 = -12288
0x37 = 0_01101_11 = +0b1.11*2^-3 = 0.21875	0x77 = 0_11101_11 = +0b1.11*2^13 = 14336	0xb7 = 1_01101_11 = -0b1.11*2^-3 = -0.21875	0xf7 = 1_11101_11 = -0b1.11*2^13 = -14336
0x38 = 0_01110_00 = +0b1.00*2^-2 = 0.25	0x78 = 0_11110_00 = +0b1.00*2^14 = 16384	0xb8 = 1_01110_00 = -0b1.00*2^-2 = -0.25	0xf8 = 1_11110_00 = -0b1.00*2^14 = -16384
0x39 = 0_01110_01 = +0b1.01*2^-2 = 0.3125	0x79 = 0_11110_01 = +0b1.01*2^14 = 20480	0xb9 = 1_01110_01 = -0b1.01*2^-2 = -0.3125	0xf9 = 1_11110_01 = -0b1.01*2^14 = -20480
0x3a = 0_01110_10 = +0b1.10*2^-2 = 0.375	0x7a = 0_11110_10 = +0b1.10*2^14 = 24576	0xba = 1_01110_10 = -0b1.10*2^-2 = -0.375	0xfa = 1_11110_10 = -0b1.10*2^14 = -24576
0x3b = 0_01110_11 = +0b1.11*2^-2 = 0.4375	0x7b = 0_11110_11 = +0b1.11*2^14 = 28672	0xbb = 1_01110_11 = -0b1.11*2^-2 = -0.4375	0xfb = 1_11110_11 = -0b1.11*2^14 = -28672
0x3c = 0_01111_00 = +0b1.00*2^-1 = 0.5	0x7c = 0_11111_00 = +0b1.00*2^15 = 32768	0xbc = 1_01111_00 = -0b1.00*2^-1 = -0.5	0xfc = 1_11111_00 = -0b1.00*2^15 = -32768
0x3d = 0_01111_01 = +0b1.01*2^-1 = 0.625	0x7d = 0_11111_01 = +0b1.01*2^15 = 40960	0xbd = 1_01111_01 = -0b1.01*2^-1 = -0.625	0xfd = 1_11111_01 = -0b1.01*2^15 = -40960
0x3e = 0_01111_10 = +0b1.10*2^-1 = 0.75	0x7e = 0_11111_10 = +0b1.10*2^15 = 49152	0xbe = 1_01111_10 = -0b1.10*2^-1 = -0.75	0xfe = 1_11111_10 = -0b1.10*2^15 = -49152
0x3f = 0_01111_11 = +0b1.11*2^-1 = 0.875	0x7f = 0_11111_11 = +Inf	0xbf = 1_01111_11 = -0b1.11*2^-1 = -0.875	0xff = 1_11111_11 = -Inf

10.2. Value table: binary8p4

0x00 = 0_0000_000 = +0b0.000*2^-7 = 0	0x40 = 0_1000_000 = +0b1.000*2^0 = 1	0x80 = 1_0000_000 = NaN	0xc0 = 1_1000_000 = -0b1.000*2^0 = -1
0x01 = 0_0000_001 = +0b0.001*2^-7 = 0.000976562	0x41 = 0_1000_001 = +0b1.001*2^0 = 1.125	0x81 = 1_0000_001 = -0b0.001*2^-7 = -0.000976562	0xc1 = 1_1000_001 = -0b1.001*2^0 = -1.125
0x02 = 0_0000_010 = +0b0.010*2^-7 = 0.00195312	0x42 = 0_1000_010 = +0b1.010*2^0 = 1.25	0x82 = 1_0000_010 = -0b0.010*2^-7 = -0.00195312	0xc2 = 1_1000_010 = -0b1.010*2^0 = -1.25
0x03 = 0_0000_011 = +0b0.011*2^-7 = 0.00292969	0x43 = 0_1000_011 = +0b1.011*2^0 = 1.375	0x83 = 1_0000_011 = -0b0.011*2^-7 = -0.00292969	0xc3 = 1_1000_011 = -0b1.011*2^0 = -1.375
0x04 = 0_0000_100 = +0b0.100*2^-7 = 0.00390625	0x44 = 0_1000_100 = +0b1.100*2^0 = 1.5	0x84 = 1_0000_100 = -0b0.100*2^-7 = -0.00390625	0xc4 = 1_1000_100 = -0b1.100*2^0 = -1.5
0x05 = 0_0000_101 = +0b0.101*2^-7 = 0.00488281	0x45 = 0_1000_101 = +0b1.101*2^0 = 1.625	0x85 = 1_0000_101 = -0b0.101*2^-7 = -0.00488281	0xc5 = 1_1000_101 = -0b1.101*2^0 = -1.625
0x06 = 0_0000_110 = +0b0.110*2^-7 = 0.00585938	0x46 = 0_1000_110 = +0b1.110*2^0 = 1.75	0x86 = 1_0000_110 = -0b0.110*2^-7 = -0.00585938	0xc6 = 1_1000_110 = -0b1.110*2^0 = -1.75
0x07 = 0_0000_111 = +0b0.111*2^-7 = 0.00683594	0x47 = 0_1000_111 = +0b1.111*2^0 = 1.875	0x87 = 1_0000_111 = -0b0.111*2^-7 = -0.00683594	0xc7 = 1_1000_111 = -0b1.111*2^0 = -1.875
0x08 = 0_0001_000 = +0b1.000*2^-7 = 0.0078125	0x48 = 0_1001_000 = +0b1.000*2^1 = 2	0x88 = 1_0001_000 = -0b1.000*2^-7 = -0.0078125	0xc8 = 1_1001_000 = -0b1.000*2^1 = -2
0x09 = 0_0001_001 = +0b1.001*2^-7 = 0.00878906	0x49 = 0_1001_001 = +0b1.001*2^1 = 2.25	0x89 = 1_0001_001 = -0b1.001*2^-7 = -0.00878906	0xc9 = 1_1001_001 = -0b1.001*2^1 = -2.25
0x0a = 0_0001_010 = +0b1.010*2^-7 = 0.00976562	0x4a = 0_1001_010 = +0b1.010*2^1 = 2.5	0x8a = 1_0001_010 = -0b1.010*2^-7 = -0.00976562	0xca = 1_1001_010 = -0b1.010*2^1 = -2.5
0x0b = 0_0001_011 = +0b1.011*2^-7 = 0.0107422	0x4b = 0_1001_011 = +0b1.011*2^1 = 2.75	0x8b = 1_0001_011 = -0b1.011*2^-7 = -0.0107422	0xcb = 1_1001_011 = -0b1.011*2^1 = -2.75
0x0c = 0_0001_100 = +0b1.100*2^-7 = 0.0117188	0x4c = 0_1001_100 = +0b1.100*2^1 = 3	0x8c = 1_0001_100 = -0b1.100*2^-7 = -0.0117188	0xcc = 1_1001_100 = -0b1.100*2^1 = -3
0x0d = 0_0001_101 = +0b1.101*2^-7 = 0.0126953	0x4d = 0_1001_101 = +0b1.101*2^1 = 3.25	0x8d = 1_0001_101 = -0b1.101*2^-7 = -0.0126953	0xcd = 1_1001_101 = -0b1.101*2^1 = -3.25
0x0e = 0_0001_110 = +0b1.110*2^-7 = 0.0136719	0x4e = 0_1001_110 = +0b1.110*2^1 = 3.5	0x8e = 1_0001_110 = -0b1.110*2^-7 = -0.0136719	0xce = 1_1001_110 = -0b1.110*2^1 = -3.5
0x0f = 0_0001_111 = +0b1.111*2^-7 = 0.0146484	0x4f = 0_1001_111 = +0b1.111*2^1 = 3.75	0x8f = 1_0001_111 = -0b1.111*2^-7 = -0.0146484	0xcf = 1_1001_111 = -0b1.111*2^1 = -3.75
0x10 = 0_0010_000 = +0b1.000*2^-6 = 0.015625	0x50 = 0_1010_000 = +0b1.000*2^2 = 4	0x90 = 1_0010_000 = -0b1.000*2^-6 = -0.015625	0xd0 = 1_1010_000 = -0b1.000*2^2 = -4
0x11 = 0_0010_001 = +0b1.001*2^-6 = 0.0175781	0x51 = 0_1010_001 = +0b1.001*2^2 = 4.5	0x91 = 1_0010_001 = -0b1.001*2^-6 = -0.0175781	0xd1 = 1_1010_001 = -0b1.001*2^2 = -4.5
0x12 = 0_0010_010 = +0b1.010*2^-6 = 0.0195312	0x52 = 0_1010_010 = +0b1.010*2^2 = 5	0x92 = 1_0010_010 = -0b1.010*2^-6 = -0.0195312	0xd2 = 1_1010_010 = -0b1.010*2^2 = -5
0x13 = 0_0010_011 = +0b1.011*2^-6 = 0.0214844	0x53 = 0_1010_011 = +0b1.011*2^2 = 5.5	0x93 = 1_0010_011 = -0b1.011*2^-6 = -0.0214844	0xd3 = 1_1010_011 = -0b1.011*2^2 = -5.5
0x14 = 0_0010_100 = +0b1.100*2^-6 = 0.0234375	0x54 = 0_1010_100 = +0b1.100*2^2 = 6	0x94 = 1_0010_100 = -0b1.100*2^-6 = -0.0234375	0xd4 = 1_1010_100 = -0b1.100*2^2 = -6
0x15 = 0_0010_101 = +0b1.101*2^-6 = 0.0253906	0x55 = 0_1010_101 = +0b1.101*2^2 = 6.5	0x95 = 1_0010_101 = -0b1.101*2^-6 = -0.0253906	0xd5 = 1_1010_101 = -0b1.101*2^2 = -6.5
0x16 = 0_0010_110 = +0b1.110*2^-6 = 0.0273438	0x56 = 0_1010_110 = +0b1.110*2^2 = 7	0x96 = 1_0010_110 = -0b1.110*2^-6 = -0.0273438	0xd6 = 1_1010_110 = -0b1.110*2^2 = -7
0x17 = 0_0010_111 = +0b1.111*2^-6 = 0.0292969	0x57 = 0_1010_111 = +0b1.111*2^2 = 7.5	0x97 = 1_0010_111 = -0b1.111*2^-6 = -0.0292969	0xd7 = 1_1010_111 = -0b1.111*2^2 = -7.5
0x18 = 0_0011_000 = +0b1.000*2^-5 = 0.03125	0x58 = 0_1011_000 = +0b1.000*2^3 = 8	0x98 = 1_0011_000 = -0b1.000*2^-5 = -0.03125	0xd8 = 1_1011_000 = -0b1.000*2^3 = -8
0x19 = 0_0011_001 = +0b1.001*2^-5 = 0.0351562	0x59 = 0_1011_001 = +0b1.001*2^3 = 9	0x99 = 1_0011_001 = -0b1.001*2^-5 = -0.0351562	0xd9 = 1_1011_001 = -0b1.001*2^3 = -9
0x1a = 0_0011_010 = +0b1.010*2^-5 = 0.0390625	0x5a = 0_1011_010 = +0b1.010*2^3 = 10	0x9a = 1_0011_010 = -0b1.010*2^-5 = -0.0390625	0xda = 1_1011_010 = -0b1.010*2^3 = -10
0x1b = 0_0011_011 = +0b1.011*2^-5 = 0.0429688	0x5b = 0_1011_011 = +0b1.011*2^3 = 11	0x9b = 1_0011_011 = -0b1.011*2^-5 = -0.0429688	0xdb = 1_1011_011 = -0b1.011*2^3 = -11
0x1c = 0_0011_100 = +0b1.100*2^-5 = 0.046875	0x5c = 0_1011_100 = +0b1.100*2^3 = 12	0x9c = 1_0011_100 = -0b1.100*2^-5 = -0.046875	0xdc = 1_1011_100 = -0b1.100*2^3 = -12
0x1d = 0_0011_101 = +0b1.101*2^-5 = 0.0507812	0x5d = 0_1011_101 = +0b1.101*2^3 = 13	0x9d = 1_0011_101 = -0b1.101*2^-5 = -0.0507812	0xdd = 1_1011_101 = -0b1.101*2^3 = -13
0x1e = 0_0011_110 = +0b1.110*2^-5 = 0.0546875	0x5e = 0_1011_110 = +0b1.110*2^3 = 14	0x9e = 1_0011_110 = -0b1.110*2^-5 = -0.0546875	0xde = 1_1011_110 = -0b1.110*2^3 = -14
0x1f = 0_0011_111 = +0b1.111*2^-5 = 0.0585938	0x5f = 0_1011_111 = +0b1.111*2^3 = 15	0x9f = 1_0011_111 = -0b1.111*2^-5 = -0.0585938	0xdf = 1_1011_111 = -0b1.111*2^3 = -15
0x20 = 0_0100_000 = +0b1.000*2^-4 = 0.0625	0x60 = 0_1100_000 = +0b1.000*2^4 = 16	0xa0 = 1_0100_000 = -0b1.000*2^-4 = -0.0625	0xe0 = 1_1100_000 = -0b1.000*2^4 = -16
0x21 = 0_0100_001 = +0b1.001*2^-4 = 0.0703125	0x61 = 0_1100_001 = +0b1.001*2^4 = 18	0xa1 = 1_0100_001 = -0b1.001*2^-4 = -0.0703125	0xe1 = 1_1100_001 = -0b1.001*2^4 = -18
0x22 = 0_0100_010 = +0b1.010*2^-4 = 0.078125	0x62 = 0_1100_010 = +0b1.010*2^4 = 20	0xa2 = 1_0100_010 = -0b1.010*2^-4 = -0.078125	0xe2 = 1_1100_010 = -0b1.010*2^4 = -20
0x23 = 0_0100_011 = +0b1.011*2^-4 = 0.0859375	0x63 = 0_1100_011 = +0b1.011*2^4 = 22	0xa3 = 1_0100_011 = -0b1.011*2^-4 = -0.0859375	0xe3 = 1_1100_011 = -0b1.011*2^4 = -22
0x24 = 0_0100_100 = +0b1.100*2^-4 = 0.09375	0x64 = 0_1100_100 = +0b1.100*2^4 = 24	0xa4 = 1_0100_100 = -0b1.100*2^-4 = -0.09375	0xe4 = 1_1100_100 = -0b1.100*2^4 = -24
0x25 = 0_0100_101 = +0b1.101*2^-4 = 0.101562	0x65 = 0_1100_101 = +0b1.101*2^4 = 26	0xa5 = 1_0100_101 = -0b1.101*2^-4 = -0.101562	0xe5 = 1_1100_101 = -0b1.101*2^4 = -26
0x26 = 0_0100_110 = +0b1.110*2^-4 = 0.109375	0x66 = 0_1100_110 = +0b1.110*2^4 = 28	0xa6 = 1_0100_110 = -0b1.110*2^-4 = -0.109375	0xe6 = 1_1100_110 = -0b1.110*2^4 = -28
0x27 = 0_0100_111 = +0b1.111*2^-4 = 0.117188	0x67 = 0_1100_111 = +0b1.111*2^4 = 30	0xa7 = 1_0100_111 = -0b1.111*2^-4 = -0.117188	0xe7 = 1_1100_111 = -0b1.111*2^4 = -30
0x28 = 0_0101_000 = +0b1.000*2^-3 = 0.125	0x68 = 0_1101_000 = +0b1.000*2^5 = 32	0xa8 = 1_0101_000 = -0b1.000*2^-3 = -0.125	0xe8 = 1_1101_000 = -0b1.000*2^5 = -32
0x29 = 0_0101_001 = +0b1.001*2^-3 = 0.140625	0x69 = 0_1101_001 = +0b1.001*2^5 = 36	0xa9 = 1_0101_001 = -0b1.001*2^-3 = -0.140625	0xe9 = 1_1101_001 = -0b1.001*2^5 = -36
0x2a = 0_0101_010 = +0b1.010*2^-3 = 0.15625	0x6a = 0_1101_010 = +0b1.010*2^5 = 40	0xaa = 1_0101_010 = -0b1.010*2^-3 = -0.15625	0xea = 1_1101_010 = -0b1.010*2^5 = -40
0x2b = 0_0101_011 = +0b1.011*2^-3 = 0.171875	0x6b = 0_1101_011 = +0b1.011*2^5 = 44	0xab = 1_0101_011 = -0b1.011*2^-3 = -0.171875	0xeb = 1_1101_011 = -0b1.011*2^5 = -44
0x2c = 0_0101_100 = +0b1.100*2^-3 = 0.1875	0x6c = 0_1101_100 = +0b1.100*2^5 = 48	0xac = 1_0101_100 = -0b1.100*2^-3 = -0.1875	0xec = 1_1101_100 = -0b1.100*2^5 = -48
0x2d = 0_0101_101 = +0b1.101*2^-3 = 0.203125	0x6d = 0_1101_101 = +0b1.101*2^5 = 52	0xad = 1_0101_101 = -0b1.101*2^-3 = -0.203125	0xed = 1_1101_101 = -0b1.101*2^5 = -52
0x2e = 0_0101_110 = +0b1.110*2^-3 = 0.21875	0x6e = 0_1101_110 = +0b1.110*2^5 = 56	0xae = 1_0101_110 = -0b1.110*2^-3 = -0.21875	0xee = 1_1101_110 = -0b1.110*2^5 = -56
0x2f = 0_0101_111 = +0b1.111*2^-3 = 0.234375	0x6f = 0_1101_111 = +0b1.111*2^5 = 60	0xaf = 1_0101_111 = -0b1.111*2^-3 = -0.234375	0xef = 1_1101_111 = -0b1.111*2^5 = -60
0x30 = 0_0110_000 = +0b1.000*2^-2 = 0.25	0x70 = 0_1110_000 = +0b1.000*2^6 = 64	0xb0 = 1_0110_000 = -0b1.000*2^-2 = -0.25	0xf0 = 1_1110_000 = -0b1.000*2^6 = -64
0x31 = 0_0110_001 = +0b1.001*2^-2 = 0.28125	0x71 = 0_1110_001 = +0b1.001*2^6 = 72	0xb1 = 1_0110_001 = -0b1.001*2^-2 = -0.28125	0xf1 = 1_1110_001 = -0b1.001*2^6 = -72
0x32 = 0_0110_010 = +0b1.010*2^-2 = 0.3125	0x72 = 0_1110_010 = +0b1.010*2^6 = 80	0xb2 = 1_0110_010 = -0b1.010*2^-2 = -0.3125	0xf2 = 1_1110_010 = -0b1.010*2^6 = -80
0x33 = 0_0110_011 = +0b1.011*2^-2 = 0.34375	0x73 = 0_1110_011 = +0b1.011*2^6 = 88	0xb3 = 1_0110_011 = -0b1.011*2^-2 = -0.34375	0xf3 = 1_1110_011 = -0b1.011*2^6 = -88
0x34 = 0_0110_100 = +0b1.100*2^-2 = 0.375	0x74 = 0_1110_100 = +0b1.100*2^6 = 96	0xb4 = 1_0110_100 = -0b1.100*2^-2 = -0.375	0xf4 = 1_1110_100 = -0b1.100*2^6 = -96
0x35 = 0_0110_101 = +0b1.101*2^-2 = 0.40625	0x75 = 0_1110_101 = +0b1.101*2^6 = 104	0xb5 = 1_0110_101 = -0b1.101*2^-2 = -0.40625	0xf5 = 1_1110_101 = -0b1.101*2^6 = -104
0x36 = 0_0110_110 = +0b1.110*2^-2 = 0.4375	0x76 = 0_1110_110 = +0b1.110*2^6 = 112	0xb6 = 1_0110_110 = -0b1.110*2^-2 = -0.4375	0xf6 = 1_1110_110 = -0b1.110*2^6 = -112
0x37 = 0_0110_111 = +0b1.111*2^-2 = 0.46875	0x77 = 0_1110_111 = +0b1.111*2^6 = 120	0xb7 = 1_0110_111 = -0b1.111*2^-2 = -0.46875	0xf7 = 1_1110_111 = -0b1.111*2^6 = -120
0x38 = 0_0111_000 = +0b1.000*2^-1 = 0.5	0x78 = 0_1111_000 = +0b1.000*2^7 = 128	0xb8 = 1_0111_000 = -0b1.000*2^-1 = -0.5	0xf8 = 1_1111_000 = -0b1.000*2^7 = -128
0x39 = 0_0111_001 = +0b1.001*2^-1 = 0.5625	0x79 = 0_1111_001 = +0b1.001*2^7 = 144	0xb9 = 1_0111_001 = -0b1.001*2^-1 = -0.5625	0xf9 = 1_1111_001 = -0b1.001*2^7 = -144
0x3a = 0_0111_010 = +0b1.010*2^-1 = 0.625	0x7a = 0_1111_010 = +0b1.010*2^7 = 160	0xba = 1_0111_010 = -0b1.010*2^-1 = -0.625	0xfa = 1_1111_010 = -0b1.010*2^7 = -160
0x3b = 0_0111_011 = +0b1.011*2^-1 = 0.6875	0x7b = 0_1111_011 = +0b1.011*2^7 = 176	0xbb = 1_0111_011 = -0b1.011*2^-1 = -0.6875	0xfb = 1_1111_011 = -0b1.011*2^7 = -176
0x3c = 0_0111_100 = +0b1.100*2^-1 = 0.75	0x7c = 0_1111_100 = +0b1.100*2^7 = 192	0xbc = 1_0111_100 = -0b1.100*2^-1 = -0.75	0xfc = 1_1111_100 = -0b1.100*2^7 = -192
0x3d = 0_0111_101 = +0b1.101*2^-1 = 0.8125	0x7d = 0_1111_101 = +0b1.101*2^7 = 208	0xbd = 1_0111_101 = -0b1.101*2^-1 = -0.8125	0xfd = 1_1111_101 = -0b1.101*2^7 = -208
0x3e = 0_0111_110 = +0b1.110*2^-1 = 0.875	0x7e = 0_1111_110 = +0b1.110*2^7 = 224	0xbe = 1_0111_110 = -0b1.110*2^-1 = -0.875	0xfe = 1_1111_110 = -0b1.110*2^7 = -224
0x3f = 0_0111_111 = +0b1.111*2^-1 = 0.9375	0x7f = 0_1111_111 = +Inf	0xbf = 1_0111_111 = -0b1.111*2^-1 = -0.9375	0xff = 1_1111_111 = -Inf

10.3. Value table: binary8p5

0x00 = 0_000_0000 = +0b0.0000*2^-3 = 0	0x40 = 0_100_0000 = +0b1.0000*2^0 = 1	0x80 = 1_000_0000 = NaN	0xc0 = 1_100_0000 = -0b1.0000*2^0 = -1
0x01 = 0_000_0001 = +0b0.0001*2^-3 = 0.0078125	0x41 = 0_100_0001 = +0b1.0001*2^0 = 1.0625	0x81 = 1_000_0001 = -0b0.0001*2^-3 = -0.0078125	0xc1 = 1_100_0001 = -0b1.0001*2^0 = -1.0625
0x02 = 0_000_0010 = +0b0.0010*2^-3 = 0.015625	0x42 = 0_100_0010 = +0b1.0010*2^0 = 1.125	0x82 = 1_000_0010 = -0b0.0010*2^-3 = -0.015625	0xc2 = 1_100_0010 = -0b1.0010*2^0 = -1.125
0x03 = 0_000_0011 = +0b0.0011*2^-3 = 0.0234375	0x43 = 0_100_0011 = +0b1.0011*2^0 = 1.1875	0x83 = 1_000_0011 = -0b0.0011*2^-3 = -0.0234375	0xc3 = 1_100_0011 = -0b1.0011*2^0 = -1.1875
0x04 = 0_000_0100 = +0b0.0100*2^-3 = 0.03125	0x44 = 0_100_0100 = +0b1.0100*2^0 = 1.25	0x84 = 1_000_0100 = -0b0.0100*2^-3 = -0.03125	0xc4 = 1_100_0100 = -0b1.0100*2^0 = -1.25
0x05 = 0_000_0101 = +0b0.0101*2^-3 = 0.0390625	0x45 = 0_100_0101 = +0b1.0101*2^0 = 1.3125	0x85 = 1_000_0101 = -0b0.0101*2^-3 = -0.0390625	0xc5 = 1_100_0101 = -0b1.0101*2^0 = -1.3125
0x06 = 0_000_0110 = +0b0.0110*2^-3 = 0.046875	0x46 = 0_100_0110 = +0b1.0110*2^0 = 1.375	0x86 = 1_000_0110 = -0b0.0110*2^-3 = -0.046875	0xc6 = 1_100_0110 = -0b1.0110*2^0 = -1.375
0x07 = 0_000_0111 = +0b0.0111*2^-3 = 0.0546875	0x47 = 0_100_0111 = +0b1.0111*2^0 = 1.4375	0x87 = 1_000_0111 = -0b0.0111*2^-3 = -0.0546875	0xc7 = 1_100_0111 = -0b1.0111*2^0 = -1.4375
0x08 = 0_000_1000 = +0b0.1000*2^-3 = 0.0625	0x48 = 0_100_1000 = +0b1.1000*2^0 = 1.5	0x88 = 1_000_1000 = -0b0.1000*2^-3 = -0.0625	0xc8 = 1_100_1000 = -0b1.1000*2^0 = -1.5
0x09 = 0_000_1001 = +0b0.1001*2^-3 = 0.0703125	0x49 = 0_100_1001 = +0b1.1001*2^0 = 1.5625	0x89 = 1_000_1001 = -0b0.1001*2^-3 = -0.0703125	0xc9 = 1_100_1001 = -0b1.1001*2^0 = -1.5625
0x0a = 0_000_1010 = +0b0.1010*2^-3 = 0.078125	0x4a = 0_100_1010 = +0b1.1010*2^0 = 1.625	0x8a = 1_000_1010 = -0b0.1010*2^-3 = -0.078125	0xca = 1_100_1010 = -0b1.1010*2^0 = -1.625
0x0b = 0_000_1011 = +0b0.1011*2^-3 = 0.0859375	0x4b = 0_100_1011 = +0b1.1011*2^0 = 1.6875	0x8b = 1_000_1011 = -0b0.1011*2^-3 = -0.0859375	0xcb = 1_100_1011 = -0b1.1011*2^0 = -1.6875
0x0c = 0_000_1100 = +0b0.1100*2^-3 = 0.09375	0x4c = 0_100_1100 = +0b1.1100*2^0 = 1.75	0x8c = 1_000_1100 = -0b0.1100*2^-3 = -0.09375	0xcc = 1_100_1100 = -0b1.1100*2^0 = -1.75
0x0d = 0_000_1101 = +0b0.1101*2^-3 = 0.101562	0x4d = 0_100_1101 = +0b1.1101*2^0 = 1.8125	0x8d = 1_000_1101 = -0b0.1101*2^-3 = -0.101562	0xcd = 1_100_1101 = -0b1.1101*2^0 = -1.8125
0x0e = 0_000_1110 = +0b0.1110*2^-3 = 0.109375	0x4e = 0_100_1110 = +0b1.1110*2^0 = 1.875	0x8e = 1_000_1110 = -0b0.1110*2^-3 = -0.109375	0xce = 1_100_1110 = -0b1.1110*2^0 = -1.875
0x0f = 0_000_1111 = +0b0.1111*2^-3 = 0.117188	0x4f = 0_100_1111 = +0b1.1111*2^0 = 1.9375	0x8f = 1_000_1111 = -0b0.1111*2^-3 = -0.117188	0xcf = 1_100_1111 = -0b1.1111*2^0 = -1.9375
0x10 = 0_001_0000 = +0b1.0000*2^-3 = 0.125	0x50 = 0_101_0000 = +0b1.0000*2^1 = 2	0x90 = 1_001_0000 = -0b1.0000*2^-3 = -0.125	0xd0 = 1_101_0000 = -0b1.0000*2^1 = -2
0x11 = 0_001_0001 = +0b1.0001*2^-3 = 0.132812	0x51 = 0_101_0001 = +0b1.0001*2^1 = 2.125	0x91 = 1_001_0001 = -0b1.0001*2^-3 = -0.132812	0xd1 = 1_101_0001 = -0b1.0001*2^1 = -2.125
0x12 = 0_001_0010 = +0b1.0010*2^-3 = 0.140625	0x52 = 0_101_0010 = +0b1.0010*2^1 = 2.25	0x92 = 1_001_0010 = -0b1.0010*2^-3 = -0.140625	0xd2 = 1_101_0010 = -0b1.0010*2^1 = -2.25
0x13 = 0_001_0011 = +0b1.0011*2^-3 = 0.148438	0x53 = 0_101_0011 = +0b1.0011*2^1 = 2.375	0x93 = 1_001_0011 = -0b1.0011*2^-3 = -0.148438	0xd3 = 1_101_0011 = -0b1.0011*2^1 = -2.375
0x14 = 0_001_0100 = +0b1.0100*2^-3 = 0.15625	0x54 = 0_101_0100 = +0b1.0100*2^1 = 2.5	0x94 = 1_001_0100 = -0b1.0100*2^-3 = -0.15625	0xd4 = 1_101_0100 = -0b1.0100*2^1 = -2.5
0x15 = 0_001_0101 = +0b1.0101*2^-3 = 0.164062	0x55 = 0_101_0101 = +0b1.0101*2^1 = 2.625	0x95 = 1_001_0101 = -0b1.0101*2^-3 = -0.164062	0xd5 = 1_101_0101 = -0b1.0101*2^1 = -2.625
0x16 = 0_001_0110 = +0b1.0110*2^-3 = 0.171875	0x56 = 0_101_0110 = +0b1.0110*2^1 = 2.75	0x96 = 1_001_0110 = -0b1.0110*2^-3 = -0.171875	0xd6 = 1_101_0110 = -0b1.0110*2^1 = -2.75
0x17 = 0_001_0111 = +0b1.0111*2^-3 = 0.179688	0x57 = 0_101_0111 = +0b1.0111*2^1 = 2.875	0x97 = 1_001_0111 = -0b1.0111*2^-3 = -0.179688	0xd7 = 1_101_0111 = -0b1.0111*2^1 = -2.875
0x18 = 0_001_1000 = +0b1.1000*2^-3 = 0.1875	0x58 = 0_101_1000 = +0b1.1000*2^1 = 3	0x98 = 1_001_1000 = -0b1.1000*2^-3 = -0.1875	0xd8 = 1_101_1000 = -0b1.1000*2^1 = -3
0x19 = 0_001_1001 = +0b1.1001*2^-3 = 0.195312	0x59 = 0_101_1001 = +0b1.1001*2^1 = 3.125	0x99 = 1_001_1001 = -0b1.1001*2^-3 = -0.195312	0xd9 = 1_101_1001 = -0b1.1001*2^1 = -3.125
0x1a = 0_001_1010 = +0b1.1010*2^-3 = 0.203125	0x5a = 0_101_1010 = +0b1.1010*2^1 = 3.25	0x9a = 1_001_1010 = -0b1.1010*2^-3 = -0.203125	0xda = 1_101_1010 = -0b1.1010*2^1 = -3.25
0x1b = 0_001_1011 = +0b1.1011*2^-3 = 0.210938	0x5b = 0_101_1011 = +0b1.1011*2^1 = 3.375	0x9b = 1_001_1011 = -0b1.1011*2^-3 = -0.210938	0xdb = 1_101_1011 = -0b1.1011*2^1 = -3.375
0x1c = 0_001_1100 = +0b1.1100*2^-3 = 0.21875	0x5c = 0_101_1100 = +0b1.1100*2^1 = 3.5	0x9c = 1_001_1100 = -0b1.1100*2^-3 = -0.21875	0xdc = 1_101_1100 = -0b1.1100*2^1 = -3.5
0x1d = 0_001_1101 = +0b1.1101*2^-3 = 0.226562	0x5d = 0_101_1101 = +0b1.1101*2^1 = 3.625	0x9d = 1_001_1101 = -0b1.1101*2^-3 = -0.226562	0xdd = 1_101_1101 = -0b1.1101*2^1 = -3.625
0x1e = 0_001_1110 = +0b1.1110*2^-3 = 0.234375	0x5e = 0_101_1110 = +0b1.1110*2^1 = 3.75	0x9e = 1_001_1110 = -0b1.1110*2^-3 = -0.234375	0xde = 1_101_1110 = -0b1.1110*2^1 = -3.75
0x1f = 0_001_1111 = +0b1.1111*2^-3 = 0.242188	0x5f = 0_101_1111 = +0b1.1111*2^1 = 3.875	0x9f = 1_001_1111 = -0b1.1111*2^-3 = -0.242188	0xdf = 1_101_1111 = -0b1.1111*2^1 = -3.875
0x20 = 0_010_0000 = +0b1.0000*2^-2 = 0.25	0x60 = 0_110_0000 = +0b1.0000*2^2 = 4	0xa0 = 1_010_0000 = -0b1.0000*2^-2 = -0.25	0xe0 = 1_110_0000 = -0b1.0000*2^2 = -4
0x21 = 0_010_0001 = +0b1.0001*2^-2 = 0.265625	0x61 = 0_110_0001 = +0b1.0001*2^2 = 4.25	0xa1 = 1_010_0001 = -0b1.0001*2^-2 = -0.265625	0xe1 = 1_110_0001 = -0b1.0001*2^2 = -4.25
0x22 = 0_010_0010 = +0b1.0010*2^-2 = 0.28125	0x62 = 0_110_0010 = +0b1.0010*2^2 = 4.5	0xa2 = 1_010_0010 = -0b1.0010*2^-2 = -0.28125	0xe2 = 1_110_0010 = -0b1.0010*2^2 = -4.5
0x23 = 0_010_0011 = +0b1.0011*2^-2 = 0.296875	0x63 = 0_110_0011 = +0b1.0011*2^2 = 4.75	0xa3 = 1_010_0011 = -0b1.0011*2^-2 = -0.296875	0xe3 = 1_110_0011 = -0b1.0011*2^2 = -4.75
0x24 = 0_010_0100 = +0b1.0100*2^-2 = 0.3125	0x64 = 0_110_0100 = +0b1.0100*2^2 = 5	0xa4 = 1_010_0100 = -0b1.0100*2^-2 = -0.3125	0xe4 = 1_110_0100 = -0b1.0100*2^2 = -5
0x25 = 0_010_0101 = +0b1.0101*2^-2 = 0.328125	0x65 = 0_110_0101 = +0b1.0101*2^2 = 5.25	0xa5 = 1_010_0101 = -0b1.0101*2^-2 = -0.328125	0xe5 = 1_110_0101 = -0b1.0101*2^2 = -5.25
0x26 = 0_010_0110 = +0b1.0110*2^-2 = 0.34375	0x66 = 0_110_0110 = +0b1.0110*2^2 = 5.5	0xa6 = 1_010_0110 = -0b1.0110*2^-2 = -0.34375	0xe6 = 1_110_0110 = -0b1.0110*2^2 = -5.5
0x27 = 0_010_0111 = +0b1.0111*2^-2 = 0.359375	0x67 = 0_110_0111 = +0b1.0111*2^2 = 5.75	0xa7 = 1_010_0111 = -0b1.0111*2^-2 = -0.359375	0xe7 = 1_110_0111 = -0b1.0111*2^2 = -5.75
0x28 = 0_010_1000 = +0b1.1000*2^-2 = 0.375	0x68 = 0_110_1000 = +0b1.1000*2^2 = 6	0xa8 = 1_010_1000 = -0b1.1000*2^-2 = -0.375	0xe8 = 1_110_1000 = -0b1.1000*2^2 = -6
0x29 = 0_010_1001 = +0b1.1001*2^-2 = 0.390625	0x69 = 0_110_1001 = +0b1.1001*2^2 = 6.25	0xa9 = 1_010_1001 = -0b1.1001*2^-2 = -0.390625	0xe9 = 1_110_1001 = -0b1.1001*2^2 = -6.25
0x2a = 0_010_1010 = +0b1.1010*2^-2 = 0.40625	0x6a = 0_110_1010 = +0b1.1010*2^2 = 6.5	0xaa = 1_010_1010 = -0b1.1010*2^-2 = -0.40625	0xea = 1_110_1010 = -0b1.1010*2^2 = -6.5
0x2b = 0_010_1011 = +0b1.1011*2^-2 = 0.421875	0x6b = 0_110_1011 = +0b1.1011*2^2 = 6.75	0xab = 1_010_1011 = -0b1.1011*2^-2 = -0.421875	0xeb = 1_110_1011 = -0b1.1011*2^2 = -6.75
0x2c = 0_010_1100 = +0b1.1100*2^-2 = 0.4375	0x6c = 0_110_1100 = +0b1.1100*2^2 = 7	0xac = 1_010_1100 = -0b1.1100*2^-2 = -0.4375	0xec = 1_110_1100 = -0b1.1100*2^2 = -7
0x2d = 0_010_1101 = +0b1.1101*2^-2 = 0.453125	0x6d = 0_110_1101 = +0b1.1101*2^2 = 7.25	0xad = 1_010_1101 = -0b1.1101*2^-2 = -0.453125	0xed = 1_110_1101 = -0b1.1101*2^2 = -7.25
0x2e = 0_010_1110 = +0b1.1110*2^-2 = 0.46875	0x6e = 0_110_1110 = +0b1.1110*2^2 = 7.5	0xae = 1_010_1110 = -0b1.1110*2^-2 = -0.46875	0xee = 1_110_1110 = -0b1.1110*2^2 = -7.5
0x2f = 0_010_1111 = +0b1.1111*2^-2 = 0.484375	0x6f = 0_110_1111 = +0b1.1111*2^2 = 7.75	0xaf = 1_010_1111 = -0b1.1111*2^-2 = -0.484375	0xef = 1_110_1111 = -0b1.1111*2^2 = -7.75
0x30 = 0_011_0000 = +0b1.0000*2^-1 = 0.5	0x70 = 0_111_0000 = +0b1.0000*2^3 = 8	0xb0 = 1_011_0000 = -0b1.0000*2^-1 = -0.5	0xf0 = 1_111_0000 = -0b1.0000*2^3 = -8
0x31 = 0_011_0001 = +0b1.0001*2^-1 = 0.53125	0x71 = 0_111_0001 = +0b1.0001*2^3 = 8.5	0xb1 = 1_011_0001 = -0b1.0001*2^-1 = -0.53125	0xf1 = 1_111_0001 = -0b1.0001*2^3 = -8.5
0x32 = 0_011_0010 = +0b1.0010*2^-1 = 0.5625	0x72 = 0_111_0010 = +0b1.0010*2^3 = 9	0xb2 = 1_011_0010 = -0b1.0010*2^-1 = -0.5625	0xf2 = 1_111_0010 = -0b1.0010*2^3 = -9
0x33 = 0_011_0011 = +0b1.0011*2^-1 = 0.59375	0x73 = 0_111_0011 = +0b1.0011*2^3 = 9.5	0xb3 = 1_011_0011 = -0b1.0011*2^-1 = -0.59375	0xf3 = 1_111_0011 = -0b1.0011*2^3 = -9.5
0x34 = 0_011_0100 = +0b1.0100*2^-1 = 0.625	0x74 = 0_111_0100 = +0b1.0100*2^3 = 10	0xb4 = 1_011_0100 = -0b1.0100*2^-1 = -0.625	0xf4 = 1_111_0100 = -0b1.0100*2^3 = -10
0x35 = 0_011_0101 = +0b1.0101*2^-1 = 0.65625	0x75 = 0_111_0101 = +0b1.0101*2^3 = 10.5	0xb5 = 1_011_0101 = -0b1.0101*2^-1 = -0.65625	0xf5 = 1_111_0101 = -0b1.0101*2^3 = -10.5
0x36 = 0_011_0110 = +0b1.0110*2^-1 = 0.6875	0x76 = 0_111_0110 = +0b1.0110*2^3 = 11	0xb6 = 1_011_0110 = -0b1.0110*2^-1 = -0.6875	0xf6 = 1_111_0110 = -0b1.0110*2^3 = -11
0x37 = 0_011_0111 = +0b1.0111*2^-1 = 0.71875	0x77 = 0_111_0111 = +0b1.0111*2^3 = 11.5	0xb7 = 1_011_0111 = -0b1.0111*2^-1 = -0.71875	0xf7 = 1_111_0111 = -0b1.0111*2^3 = -11.5
0x38 = 0_011_1000 = +0b1.1000*2^-1 = 0.75	0x78 = 0_111_1000 = +0b1.1000*2^3 = 12	0xb8 = 1_011_1000 = -0b1.1000*2^-1 = -0.75	0xf8 = 1_111_1000 = -0b1.1000*2^3 = -12
0x39 = 0_011_1001 = +0b1.1001*2^-1 = 0.78125	0x79 = 0_111_1001 = +0b1.1001*2^3 = 12.5	0xb9 = 1_011_1001 = -0b1.1001*2^-1 = -0.78125	0xf9 = 1_111_1001 = -0b1.1001*2^3 = -12.5
0x3a = 0_011_1010 = +0b1.1010*2^-1 = 0.8125	0x7a = 0_111_1010 = +0b1.1010*2^3 = 13	0xba = 1_011_1010 = -0b1.1010*2^-1 = -0.8125	0xfa = 1_111_1010 = -0b1.1010*2^3 = -13
0x3b = 0_011_1011 = +0b1.1011*2^-1 = 0.84375	0x7b = 0_111_1011 = +0b1.1011*2^3 = 13.5	0xbb = 1_011_1011 = -0b1.1011*2^-1 = -0.84375	0xfb = 1_111_1011 = -0b1.1011*2^3 = -13.5
0x3c = 0_011_1100 = +0b1.1100*2^-1 = 0.875	0x7c = 0_111_1100 = +0b1.1100*2^3 = 14	0xbc = 1_011_1100 = -0b1.1100*2^-1 = -0.875	0xfc = 1_111_1100 = -0b1.1100*2^3 = -14
0x3d = 0_011_1101 = +0b1.1101*2^-1 = 0.90625	0x7d = 0_111_1101 = +0b1.1101*2^3 = 14.5	0xbd = 1_011_1101 = -0b1.1101*2^-1 = -0.90625	0xfd = 1_111_1101 = -0b1.1101*2^3 = -14.5
0x3e = 0_011_1110 = +0b1.1110*2^-1 = 0.9375	0x7e = 0_111_1110 = +0b1.1110*2^3 = 15	0xbe = 1_011_1110 = -0b1.1110*2^-1 = -0.9375	0xfe = 1_111_1110 = -0b1.1110*2^3 = -15
0x3f = 0_011_1111 = +0b1.1111*2^-1 = 0.96875	0x7f = 0_111_1111 = +Inf	0xbf = 1_011_1111 = -0b1.1111*2^-1 = -0.96875	0xff = 1_111_1111 = -Inf

Bibliography

[1] W. Kahan, "Branch Cuts for Complex Elementary Functions or Much Ado About Nothing's Sign Bit," in *Inst. Math. Appl. Conf. Ser. New Ser.*, 1987.

[2] W. Kahan and J. W. Thomas, "Augmenting a Programming Language with Complex Arithmetic," EECS Department, University of California, Berkeley, 1991.

[3] P. Micikevicius, S. Oberman, P. Dubey, M. Cornea, A. Rodriguez, I. Bratt, R. Grisenthwaite, N. Jouppi, C. Chou, A. Huffman, M. Schulte, R. Wittig, D. Jani and S. Deng, "OCP 8-bit Floating Point Specification (OFP8)," opencompute.org, 2023.

[4] B. Noune, P. Jones, D. Justus, D. Masters and C. Luschi, "8-bit Numerical Formats for Deep Neural Networks," cs.LG, 2022.