ICS 35.160
CCS L 62

团 体 标 准

T/GCC 4002—2026

HiFloat8 数据格式技术规范

Technical specification of HiFloat8 data format

2026-06-23 发布

2026-06-23 实施

全球计算联盟 发布

目　　次

前　言

本文件按照GB/T　1.1—2020《标准化工作导则　第1部分：标准化文件的结构和起草规则》的规定起草。

本文件由全球计算联盟提出并归口。

本文件起草单位：华为技术有限公司、上海人工智能创新中心、科大讯飞股份有限公司、华南理工大学、上海商汤科技开发有限公司、清华大学、北京大学、北京航空航天大学、之江实验室、中国电子技术标准化研究院、中国电信研究院、北京神州鲲泰信息技术有限公司、河南昆仑技术有限公司、天翼云科技有限公司、北京中科加禾智能科技有限公司、希奥端计算（南京）技术有限公司、深圳市遇贤微电子计算有限公司、广州广电五舟科技股份有限公司、杭州天宽科技有限公司、深圳市大数据研究院、飞腾信息技术有限公司、深圳市朗力半导体有限公司、鹏城实验室、全芯智造技术股份有限公司、深圳开源共创科技有限公司。

本文件主要起草人：张浩男、罗元勇、赵昀、王鑫、余子伟、胡桂鹏、马欣达、刘觅、侯杰、王和俊、余姗、钟普、张海俊、汪锦想、陆璐、袁粒、葛云阳、雍洋、谷石桥、唐彦嵩、龚睿昊、刘祥龙、杜金阳、黄丹丹、张琦、李亚玲、李阳、王学聪、王钤、师春雨、高洪福、熊华、唐金龙、张安发、李亮、周明耀、肖军、陈争胜、张帆、韦奕哲、朱光旭、徐方鑫、张士勋、孟晓东、徐建国。

HiFloat8数据格式技术规范

1 范围

本文件规定了二进制HiFloat8数据格式的技术要求，包括格式规范、编码规则及格式转换要求等内容。

本文件适用于基于二进制数据格式的、对动态范围和数值精度具有综合要求且需基于8位固定位宽数据格式进行计算的场景，如人工智能模型的训练与推理，以及基于HiFloat8数据格式的软件开发、测评与性能评估等。

2 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅该日期对应的版本适用于本文件；不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

GB/T 17966—2024 信息技术 微处理器系统 浮点运算

3 术语和定义

GB/T 17966—2024第3章中规定的内容及下列术语和定义适用于本文件。

3.1

数据格式 Data format

计算机系统中用于表示实数值和特殊值的各类字段及其组合方式所构成的编码规则。

3.2

无冗余编码 Non-redundant coding

在固定位宽条件下，通过编码规则确保不同二进制位模式所表示的数值彼此不重叠的一种编码特性。

3.3

阶码 Exponent

浮点数表示中用于表示整数指数的字段，其对应的基数幂次用于确定数值的数量级及可表示范围。

3.4

尾数 Mantissa

浮点数中用于表示有效数字的小数部分。

注：“尾数位”指尾数中用于表示有效数字的小数位。

3.5

点位域 Dot field

指示阶码的存储位宽和编码模式、即时可译的变长前缀码字段。

3.6

锥形精度 Tapered accuracy

数据格式的尾数位数随指数绝对值的增大而逐渐减少的特性。

3.7

非数字 Not a number

表示无效或未定义运算结果的浮点数据特殊编码。

3.8

无穷 Infinity

一种浮点数据特殊数值，表示超出有限数值范围的结果。根据符号位的取值，可分为正无穷和负无穷。

4 缩略语

下列缩略语适用于本文件。

NML：规格化数（Normal）

DML：非规格化数（Denormal）

HiF8：HiFloat8浮点数据格式（HiFloat8 Floating-point Data Format）

Inf：无穷（Infinity）

NaN：非数字（Not a Number）

RHA：远离零半舍入（Round Half Away from Zero）

RTN：舍入到最接近值（Round to Nearest）

SE：阶码符号（Sign of Exponent）

5 二进制格式

5.1 概述

HiFloat8（简称“HiF8”）是一种8位浮点数据格式，应通过动态点位域，指示阶码存储的位宽和DML标志，实现阶码与尾数位宽的动态分配，应该能够在固定8位宽度下支持锥形精度和无冗余编码。

5.2 位字段定义

HiF8编码域段见表1，包含符号域、点位域、阶码域、尾数域四个域段：

a) 符号域：1 位，0 表示数值符号为正，1 表示数值符号为负。

b) 点位域：变长字段（2~4 位），前缀码编码，显式指示当前数值的阶码域 E 的存储位宽 0 到 4，或 DML 编码模式标志。

c) 阶码域：变长字段（0~4 位），当 E 为 0bit 时，表示阶码等于 0；包含阶码符号位，该位为 0 表示正数，1 表示负数；幅值最高位固定为 1 并隐藏不存储。

d) 尾数域：变长字段（1~3 位），NML 模式下，含有 1 位固定为 1 的整数隐藏位，其有效精度为 2 至 4 位，存储有效数字的小数部分，实际存储位宽由 8 位总位宽减去 S、D 和 E 的位宽后动态得出；DML 模式下，尾数域表达空间用于扩展二进制支持的小数值范围，和特殊值零和非数字。

表 1 HiF8 编码域段

域名	**符号域（S）**	**点位域（D）**	**阶码域（E）**	**尾数域（M）**
位宽	1bit	2-4bits	0-4bits	1-3bits

5.3 编码模式与位宽分配

HiF8的编码模式由点位域的取值决定。表2 列出了所有模式下的编码规则及位宽分配情况。

表 2　HiF8 编码模式、位宽分配与指数范围

Dot 值	点位域编码	模式	阶码存储位宽	尾数存储位宽	指数范围
0	0001	NML	0 位	3 位	0
1	001		1 位(SE)	3 位	±1
2	01		2 位(SE+E)	3 位	±[2, 3]
3	10		3 位(SE+E+E)	2 位	±[4, 7]
4	11		4 位(SE+E+E+E)	1 位	±[8, 15]
DML	0000	DML	0 位	3 位	[-22, -16]

当点位域编码为0~4时，数值属于NML模式；否则数值属于DML模式。

a)　NML模式：

1)　若点位域指示阶码存储位宽为 0，则阶码取值为 0。

2)　若点位域指示阶码存储位宽为 1~4，则包含阶码域存储阶码符号位 SE 以及 0~3bits 阶码数值。以 Dot=4 为例，阶码域存储的 4bits 为{SE,E,E,E},解析后的原码二进制为{SE,1,E,E,E}，对应的十进制阶码范围为±[8,15]。

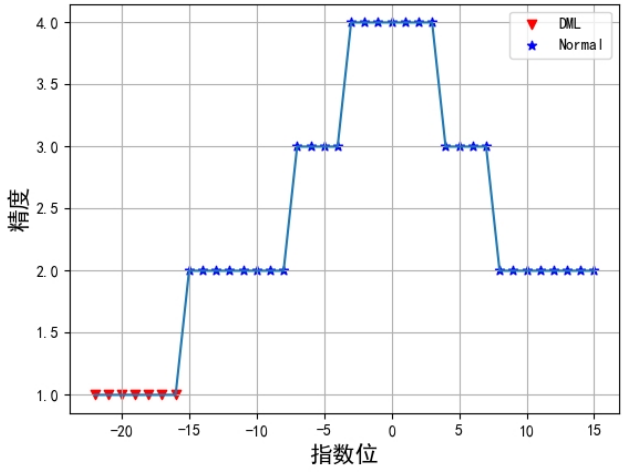
3)　NML 模式下指数范围为[-15,15]。

b)　DML 模式：

1)　阶码域为空。

2)　尾数域 3 位用于扩展 7 个小数值的指数范围；含有 1 位固定值为 1 的整数隐藏位；同时可表示特殊值零和非数字。

3)　DML 模式下指数范围为[-22,-16]。



注：▼代表 DML 模式，★代表 NML 模式。

图 1　HiF8 有效位-指数示意图

HiF8的有效尾数位与指数关系示意见0。不同于域位宽固定的数据格式，HiF8的尾数位数随指数绝对值的增大而逐渐减少，呈现锥形精度特征，指数位大小远离0时有效尾数位数逐步降低。

6　编码规则

6.1 概述

HiF8的编码应包含NML编码和DML编码的两种模式和三种特殊值编码模式。

6.2 NML 编码（Dot=0-4）

当点位域编码取0~4时，HiF8采用NML编码模式。其数值计算遵循二进制科学计数法：

$$X_{NML} = (-1)^S \times 2^E \times 1.M \quad \text{.....(1)}$$

式中：

X_{NML} —— NML编码模式下的编码值；

S —— 符号位（Sign）；

E —— 阶码域解析后的整数值；

$1.M$ —— 1位固定值为1的整数隐藏位（1）和尾数位（M）所表示的值。

HiF8的阶码域采用原码编码，并固定阶码幅值的最高位为隐藏位。当Dot=0时，隐藏位为0，对应阶码值为0；当Dot=1~4时，隐藏位为1，不同Dot值对应的指数范围应连续且无重叠。

6.3 DML 编码（Dot=DML）

当Dot域编码为DML时，HiF8采用DML编码模式。DML编码模式下没有指数域，尾数域直接用于扩展指数范围，其数值计算公式为：

$$X_{DML} = (-1)^S \times 2^{M-23} \times 1.0 \quad \text{.....(2)}$$

式中：

X_{DML} —— DML编码模式下的编码值；

S —— 符号位（Sign）；

M —— 3位无符号定点尾数值（取值1~7）；

0 —— 特殊值定义。

6.4 编码覆盖范围

HiF8在NML和DML模式下的指数范围为[-22,15]，共编码38个二进制指数值。表3 为HiF8编码示意图。编码示意为S_D_E_M，其中，S位符号位，D为点位域，E为阶码，M为尾数位，S_E位用于表示阶码符号（0表示正，1表示负），括号中的数字表示阶码域固定幅值的隐藏位，不占据存储位宽。

表 3 HiF8 编码示意图

点位域取值	HiF8 编码	阶码值(E)
0	S_0001_(0)_MMM	0
1	S_001_S_E(1)_MMM	±1
2	S_01_S_E(1)E_MMM	±[2,3]
3	S_10_S_E(1)EE_MM	±[4,7]
4	S_11_S_E(1)EEE_M	±[8,15]
非规格化	S_0000_MMM	[-22,-16]

6.5 特殊值编码

HiF8定义了三类特殊值，以确保数据格式的完备性。

a) 零（Zero）：$S=0, Dot=0000_2\,(DML), M=000_2$。HiF8 应不区分正负零，采用同一编码表示±0。

b) 非数（NaN）：$S=1, Dot=0000_2\,(DML), M=000_2$。

c) 正负无穷大（±Inf）：$S=X, Dot=11_2, E=0111_2, M=1_2$。

6.6 HiF8 关键编码值

HiF8关键编码值和特征见表4 。

表 4 HiF8 关键编码值和特征

特征	HiF8 编码	数值
无穷大（Infinities）	$S\_11\_0111\_1_2$	±Inf
非数（NaN）	$1\_0000\_000_2$	NaN
零（Zero）	$0\_0000\_000_2$	0
最大 NML	$S\_11\_0111\_0_2$	$\pm 2^{15}$
最小 NML	$S\_11\_1111\_0_2$	$\pm 2^{-15}$
最大 DML	$S\_0000\_111_2$	$\pm 2^{-16}$
最小 DML	$S\_0000\_001_2$	$\pm 2^{-22}$
动态范围（Dynamic Range）	/	38 二进制指数值

7 格式转换要求

7.1 概述

HiF8数据格式应支持由更宽浮点数据格式进行转换得到。更宽浮点数据格式包括：

a) GB/T 17966—2024 定义的 32 位浮点数。

b) GB/T 17966—2024 定义的 16 位浮点数。

c) BF16 数据格式。

转换过程应支持饱和模式与非饱和模式两种转换方式，具体规则见表5。

格式转换可由软件、硬件或软硬件协同方式实现。从更宽浮点类型向HiF8数据格式转换的算法示例见附 录 A。

表 5 更宽浮点数向 HiF8 的转换规则

宽浮点数	HiF8	
	饱和模式	非饱和模式
NAN	NAN	NaN
±INF	$\pm 2^{15}$	±inf
幅值不小于 HiF8 舍入上边界$2^{15}\times 1.25$	$\pm 2^{15}$	±inf
幅值不大于 HiF8 舍入下边界2^{-23}（包括±0）	0	0
幅值处于 HiF8 舍入边界范围内	舍入后数值	舍入后数值

7.2 特殊值转换规则

在格式转换过程中，特殊值的处理应遵循以下规则：

a) 非数字映射为 HiF8 数据格式中定义的非数字表示形式。

b) 无穷大（±inf）根据转换模式进行处理：
   1) 在饱和模式下，映射为 HiF8 的最大 NML（即$\pm 2^{15}$），并保留符号位；
   2) 在非饱和模式下，映射为 HiF8 定义的无穷大，并保留符号位。

7.3 数值幅值转换规则

在格式转换过程中，数值处理应遵循以下规则。

a) 若数据幅值$\geq 2^{15} \times 1.25$（RHA 舍入模式下的 HiF8 舍入上边界值，如果为 RTN 舍入模式，则仅需满足数据幅值$> 2^{15} \times 1.25$）。
   1) 在饱和模式下，映射为 HiF8 的最大 NML（即$\pm 2^{15}$），并保留符号位；
   2) 在非饱和模式下，映射为 HiF8 定义的±inf，并保留符号位。
b) 若数据幅值$\leq 2^{-23}$（HiF8 舍入下边界值，包括±0），映射为 HiF8 的零值。
c) 若数据幅值处于 HiF8 舍入边界范围内，则根据 7.4 定义的舍入规则生成对应的 HiF8 数据表示。在舍入过程中，应考虑进位对阶码 E 与点位域 D 的影响，确保生成正确的二进制编码。转换逻辑参考附　录　A 所示算法示例。

7.4 舍入规则

舍入策略应实现RHA与RTN两种舍入模式，并遵循以下规则：

a) 在 RHA 舍入模式下，查看舍弃位的最高位：
   1) 若该位为 1，则对保留位执行进位操作；
   2) 若该位为 0，则保留位保持不变。
b) 在 RTN 舍入模式下，将舍弃位整体视作二进制小数部分：
   1) 若舍弃位的值大于 0.5，则执行进位；
   2) 若舍弃位的值小于 0.5，则不进位；
   3) 若舍弃位的值等于 0.5，则查看保留位的最低有效位：若该位为 1，则执行进位；若该位为 0，则不进位。

7.5 高精度格式与 HiF8 量化/反量化

高精度浮点格式与HiF8数据格式之间的量化与反量化过程，应满足第7章规定的数值范围、饱和行为、舍入模式及特殊值处理规则。

附 录 A
（资料性）
更宽浮点格式向 HiF8 转换的算法示例

以下示例展示了当输入数据幅值处于HiF8舍入边界范围内时，更宽浮点格式数据向HiF8数据格式进行转换的基本逻辑。

【示例：

1. $E_x = foor(log2(abs(X_{float}))$ //$Ex \in [-23, 15]$
2. $E_x = -22 if\ Ex < -22$ //$Ex \in [-22, 15]$
3. $M_x = abs(x) \times 2^{-Ex}$ //$Mx = (0.5, 2)$
4. $D_x, M_{width} = HiF8\_Encode\_Exp(Ex)$ //舍入前 HiF8 点位域编码和有效位宽
5. $M_{rounded} = Round(M_x, M_{width}, mode = RHA/RTN)$ //舍入操作，$M_{rounded} = [1, 2]$
6. *IF* ($M_{rounded} == 2$): //舍入后更新 HiF8 各位域
7. $M_{rounded} = 1$
8. $Ex = Ex + 1$
9. $D_x = HiF8\_Encode\_Exp(Ex)$
10. *ENDIF*
11. $X_{HiF8} = HiF8\_Encode(Sign(X_{float}), D_x, E_x, M_{rounded})$ //将舍入后的数值编码成 HiF8 格式

算法说明：

a) 第 1~3 行用于提取并预处理更宽浮点格式数据的阶码值和有效位数值：
   1) 第 1 行通过对输入数据绝对值取以 2 为底的对数并向下取整，得到阶码值Ex，其取值范围为[−23,15]；
   2) 第 2 行限制阶码下界，当$Ex <- 22$ 时，将其截断为−22。该处理确保幅值小于 HiF8 最小非零 NML 编码数据，在舍入进位后仍可能被 HiF8 表示，从而扩展 HiF8 的可表达范围；
   3) 第 3 行根据处理后的阶码值计算有效位数值M_x，其取值范围为*(0.5,2)*。

b) 第 4~5 行实现舍入前参数的计算以及舍入操作：
   4) 第 4 行基于更宽浮点数的阶码，计算对应的 HiF8 点位域编码D_x和有效位位宽M_{width}。
      该映射关系需参考表 2 所定义的 HiF8 阶码与点位域、有效位位宽之间的对应规则。
      注：在NML模式下，有效位位宽等于尾数位宽与隐藏位之和；在DML模式下，有效位位宽为1。
   5) 第 5 行基于配置的舍入模式（RHA 或 RTN），计算舍入后的 HiF8 有效位数值。

c) 第 6~10 行用于处理舍入后产生进位的特殊情况（即有效位数值等于 2），此时有效位数值除以 2 变成 1，同时阶码加 1。由于阶码发生了变化，可能引发阶码值跨域 HiF8 点位域指示的边界，因此需参考表格 2，确认更新点位域的编码。

d) 第 11 行基于 HiF8 的编码规则，将舍入后的更宽浮点格式数据编码成二进制的 HiF8 数据并输出。这一步需要输入四个核心要素，包括符号位、点位域编码、阶码数值和有效位数值。】

参 考 文 献

[1] GB/T 17966—2024 信息技术 微处理器系统 浮点运算

[2] IEEE Computer Society, “IEEE Standard for Floating-Point Arithmetic”, in IEEE Std 754-2019 (Revision of IEEE 754-2008), link, June 2019

[3] Google, “The Bfloat16 Numerical Format”, https://cloud.google.com/tpu/docs/bfloat16

——