



NPU RuntimeAPI

参考指南

版本号: 2.1
发布日期: 2024.04.11

版本历史

版本号	日期	制/修订人	内容描述
1.0	2022.09.14	AWA1911	初始版本，给出 NPU 函数 API 说明。
2.0	2024.02.22	AWA1382	1) 新增第 1 章前言； 2) 第 2 章，新增一般调用流程说明和高阶接口说明； 3) 新增第 3 章 libawnn 说明
2.1	2024.04.11	AWA1382	第 2 章，新增 vip_query_input/output。



目 录

1	前言	1
1.1	文档简介	1
1.2	目标读者	1
1.3	适用范围	1
1.4	文档约定	1
1.4.1	名词解释	1
1.4.2	标志说明	2
2	VIPLiteAPI 说明	3
2.1	一般调用流程说明	3
2.2	重要数据结构说明	4
2.2.1	vip_buffer_create_params_t	4
2.3	基础接口说明	4
2.3.1	vip_init	4
2.3.2	vip_create_network	5
2.3.3	vip_prepare_network	5
2.3.4	vip_set_input/output	5
2.3.5	vip_run_network	6
2.3.6	vip_finish_network	6
2.3.7	vip_destroy_network	6
2.3.8	vip_destroy	6
2.3.9	vip_create_buffer()	7
2.3.10	vip_destroy_buffer()	7
2.3.11	vip_map_buffer()	7
2.3.12	vip_unmap_buffer()	8
2.3.13	vip_query_input/output()	8
2.4	高阶接口说明	8
2.4.1	网络相关接口说明	8
2.4.1.1	vip_set_network()	8
2.4.1.2	vip_trigger_network()	8
2.4.1.3	vip_wait_network()	9
2.4.1.4	vip_dup_network()	9
2.4.2	buffer 处理接口说明	10
2.4.2.1	vip_create_buffer_from_handle()	10
2.4.2.2	vip_create_buffer_from_physical()	10
2.4.2.3	vip_create_buffer_from_fd()	10
2.4.2.4	vip_create_buffer_cache()	11
2.5	返回值错误码说明	11



1 前言

1.1 文档简介

本文档对 NPU 端侧运行所用的 API 进行说明。

1.2 目标读者

本文档（本指南）主要适用于以下人员：

- 技术支持工程师
- AI 软件开发工程师
- AI 算法开发工程师

1.3 适用范围

硬件平台：V85x、R853、MR527、T527、AI985、MR536、T536

软件平台：Tina 系统、Android13 及以上系统

1.4 文档约定

1.4.1 名词解释

术语	解释说明
----	------

NPU	Neural Network Processing Unit，神经网络处理器。
-----	---

NBG	Network Binary Graph，驱动可识别的网络二进制文件。
-----	-------------------------------------

1.4.2 标志说明

本文中出现的符号如下：

注意

- 提醒操作中应注意的事项。不当的操作可能会损坏器件，影响可靠性、降低性能等。

说明

为准确理解文中指令、正确实施操作而提供的补充或强调信息。



2 VIPLiteAPI 说明

2.1 一般调用流程说明

VIPLite 版本的 NPU 端侧运行流程大致如下图所示：

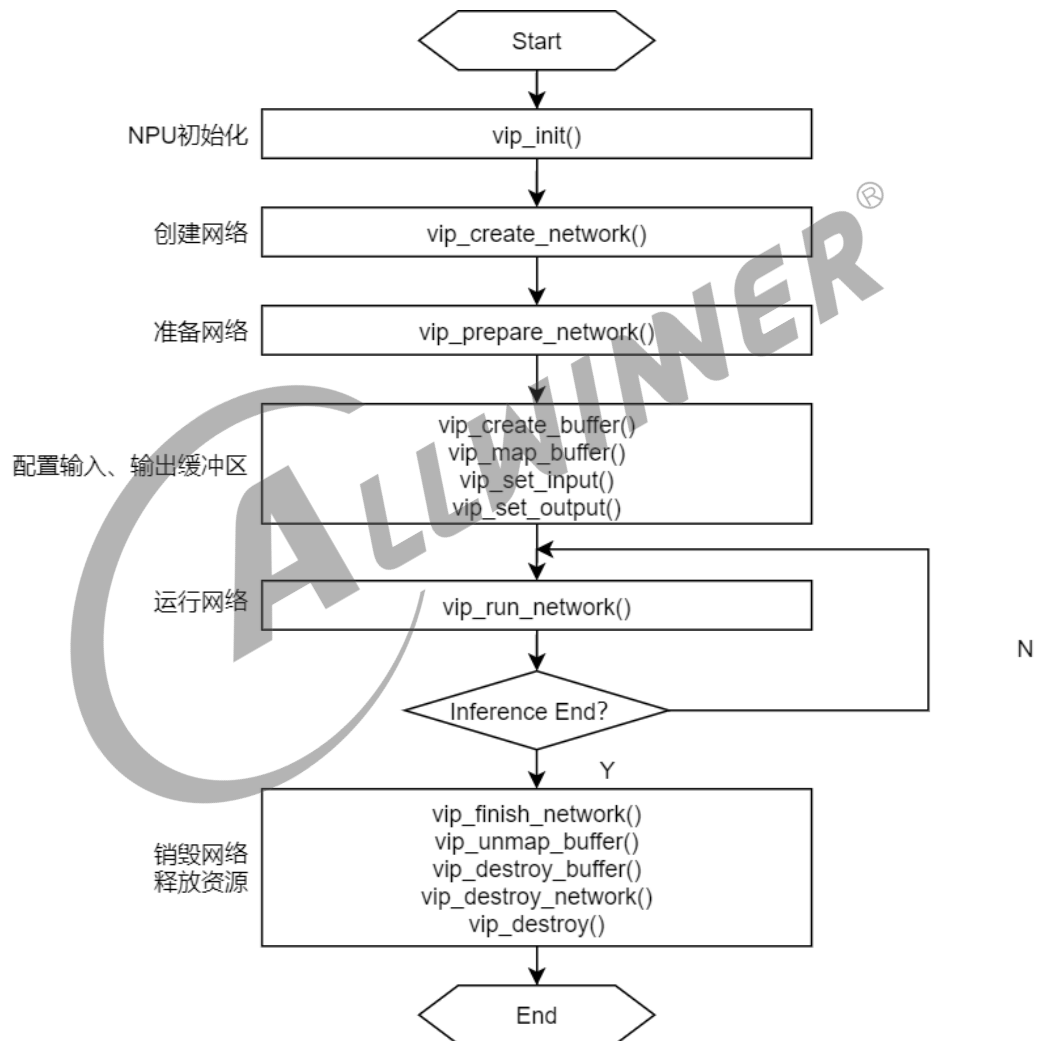


图 2-1: NPU 端侧调用流程

NPU 初始化后，创建网络并准备网络，然后给网络配置输入、输出缓冲区。准备工作完成后，开始运行网络，可反复运行。不需网络后释放资源和销毁网络。

2.2 重要数据结构说明

本节介绍重要数据结构，其他详见ai-sdk/viplite/include/vip_lite.h

2.2.1 vip_buffer_create_params_t

此结构体表示 NPU buffer 的信息，结构体的定义如下：

成员变量	数据类型	含义
num_of_dims	vip_uint32_t	shape 维度
sizes[6]	vip_uint32_t	shape
data_format	vip_buffer_format_e	数据格式
quant_format	vip_buffer_quantize_format_e	量化格式
quant_data	union	量化信息
memory_type	vip_uint32_t	应用程序的 memory 类型

说明

vip_buffer_create_params_t 结构体的信息要通过vip_query_input/output从 NBG 文件中获取，保持一致。

2.3 基础接口说明

2.3.1 vip_init

- **函数原型：** vip_status_e vip_init(void)
- **功能描述：** 初始化 NPU 硬件和 VIPLite 软件环境，即完成 NPU 模块上下层资源的初始化。
- **参数说明：** 无
- **返回值：** VIP_SUCCESS，初始化成功；其他，失败
- **其他说明：**

1. 可多次调用，但是vip_destroy()的调用数量要匹配vip_init()的数量，只有第一个vip_init()调用和最后一个vip_destroy()才会真正执行调用，他们之间的其他调用不会触发初始化或销毁。
2. 多线程中，只需调用一次。

注意

在 V85x、R853 平台，vip_init有参数，函数原型：vip_status_e vip_init(vip_uint32_t video_mem_size)，参数 video_mem_size 是额外指定分配内存堆的大小，用于 NPU 分配内存使用。

2.3.2 vip_create_network

- **函数原型：**vip_status_e vip_create_network(IN const void data, IN vip_uint32_t size_of_data, IN vip_enum type, OUT vip_network network)
- **功能描述：**根据传入的 NBG 文件，进行创建网络。
- **参数说明：**data, NBG 文件路径或 NBG 的缓冲区；size_of_data, NBG 的大小；type, NBG 文件的获取方式，参考 vip_create_network_type_e；network, 返回的网络句柄，失败时返回 VIP_NULL。
- **返回值：**VIP_SUCCESS, 创建网络成功；其他，失败
- **其他说明：**

1. 参考《NPU_模型部署_开发指南》在ai-sdk/example/vpm_run获取 vpm_run 源码，有从 memory、flash、file 获取 NBG 文件创建网络的参考代码。

2.3.3 vip_prepare_network

- **函数原型：**vip_status_e vip_prepare_network(IN vip_network network)
- **功能描述：**准备和验证网络的配置，进行运行前的准备。
- **参数说明：**network, 网络句柄。
- **返回值：**VIP_SUCCESS, 准备网络成功；其他，失败
- **其他说明：**

1. 此 API 将内部存储器资源分配给网络，包括在内部内存池中分配命令缓冲区（command buffer）并对命令进行修正。
2. vip_set_network()在此 API 前调用，进行网络配置。

2.3.4 vip_set_input/output

- **函数原型：**vip_status_e vip_set_input/output(IN vip_network network, IN vip_uint32_t index, IN vip_buffer input/output)
- **功能描述：**配置网络的输入/输出缓冲区。
- **参数说明：**network, 网络句柄；index, 网络的输入/输出的索引；input/output, 网络的输入/输出缓冲区。
- **返回值：**VIP_SUCCESS, 绑定输入/输出缓冲区成功；其他，失败
- **其他说明：**

1. 绑定输入/输出 buffer 时，驱动会修正 command buffer。
2. 可重复调用此 API 更新输入/输出 buffer。
3. 需在vip_prepare_network()接口之后调用。

4. 确保在vip_run_network()接口调用之前已经绑定有效的输入/输出 buffer，否则报错VIP_ERROR_MISSING_INPUT_OUTPUT。

2.3.5 vip_run_network

- **函数原型：** vip_status_e vip_run_network(IN vip_network network)
- **功能描述：** 提交运行网络的任务，开始运行网络。
- **参数说明：** network，网络句柄。
- **返回值：** VIP_SUCCESS，运行网络成功；其他，失败
- **其他说明：**

1. NPU 硬件执行在已提交的任务中最高优先级的任务。
2. 若希望不等待执行完成就立即返回状态，请使用vip_trigger_network()和vip_wait_network()。
3. 要设置网络优先级，请使用vip_set_network()。
4. 确保在此 API 调用之前已经绑定有效的输入/输出 buffer，否则报错VIP_ERROR_MISSING_INPUT_OUTPUT。

2.3.6 vip_finish_network

- **函数原型：** vip_status_e vip_finish_network(IN vip_network network)
- **功能描述：** 释放准备好的网络资源。
- **参数说明：** network，网络句柄。
- **返回值：** VIP_SUCCESS，释放网络资源成功；其他，失败
- **其他说明：**

1. 与vip_prepare_network()是对应的。
2. 仅当网络不再使用时调用，如果仍需使用，不要调用此 API，因为准备网络非常耗时。

2.3.7 vip_destroy_network

- **函数原型：** vip_status_e vip_destroy_network(IN vip_network network)
- **功能描述：** 销毁网络。释放分配给该网络的相关资源。
- **参数说明：** network，网络句柄。
- **返回值：** VIP_SUCCESS，销毁网络成功；其他，失败

2.3.8 vip_destroy

- **函数原型：** vip_status_e vip_destroy(void)

- **功能描述：**终止驱动程序，释放vip_init()请求的资源，并关闭 NPU 硬件。
 - **参数说明：**无
 - **返回值：**VIP_SUCCESS，摧毁成功；其他，失败
 - **其他说明：**
1. vip_destroy()的调用数量要匹配vip_init()的数量，只有第一个vip_init()调用和最后一个vip_destroy()才会真正执行调用，他们之间的其他调用不会触发初始化或销毁。
 2. 执行此 API 后，在调用任何其他 API 之前需先调用vip_init()。

2.3.9 vip_create_buffer()

- **函数原型：** vip_status_e vip_create_buffer(IN vip_buffer_create_params_t create_param, IN vip_uint32_t size_of_param, OUT vip_buffer buffer);
- **功能描述：**创建特定大小的 NPU 的 buffer。
- **参数说明：**create_param，参数结构体；size_of_param，参数的大小；buffer，返回的 buffer 句柄，失败时返回 VIP_NULL。
- **返回值：**VIP_SUCCESS，创建 buffer 成功；其他，失败
- **其他说明：**

1. buffer 中，行、切片、batch 间无 padding

2.3.10 vip_destroy_buffer()

- **函数原型：** vip_status_e vip_destroy_buffer(IN vip_buffer buffer);
- **功能描述：**销毁 NPU 的 buffer，和释放该内存。
- **参数说明：**buffer，句柄。
- **返回值：**VIP_SUCCESS，销毁 buffer 成功；其他，失败

2.3.11 vip_map_buffer()

- **函数原型：** void * vip_map_buffer(IN vip_buffer buffer);
- **功能描述：**映射指向 NPU buffer 的虚拟地址，供应用程序使用。
- **参数说明：**buffer，句柄。
- **返回值：**应用程序可访问的指针。

2.3.12 vip_unmap_buffer()

- **函数原型：** vip_status_e vip_unmap_buffer(IN vip_buffer buffer);
- **功能描述：** 取消映射 NPU buffer 的虚拟地址。
- **参数说明：** buffer, 句柄。
- **返回值：** VIP_SUCCESS, 取消映射成功; 其他, 失败

2.3.13 vip_query_input/output()

- **函数原型：** vip_status_e vip_query_input(IN vip_network network, IN vip_uint32_t index, IN vip_enum property, OUT void *value);
- **功能描述：** 查询指定网络的输入/输出信息。
- **参数说明：** network, 网络句柄; index, 网络的输入/输出的索引; property, 信息类型; value, 返回的信息的指针。
- **返回值：** VIP_SUCCESS, 获取信息成功; 其他, 失败
- **其他说明：**

1. property 参考 vip_buffer_property_eo

2.4 高阶接口说明

2.4.1 网络相关接口说明

2.4.1.1 vip_set_network()

- **函数原型：** vip_status_e vip_set_network(IN vip_network network, IN vip_enum property, IN void *value);
- **功能描述：** 配置网络。例如修改优先级。
- **参数说明：** network, 网络句柄; property, 配置类型; value, 配置的具体值
- **返回值：** VIP_SUCCESS, 配置网络成功; 其他, 失败

2.4.1.2 vip_trigger_network()

- **函数原型：** vip_status_e vip_trigger_network(IN vip_network network);
- **功能描述：** 提交运行网络的任务, 开始运行网络, 但不等待 NPU 硬件执行完成就立即返回状态。
- **参数说明：** network, 网络句柄。

- **返回值：**VIP_SUCCESS，提交任务成功；其他，失败
- **其他说明：**

1. NPU 硬件执行在已提交的任务中最高优先级的任务。
2. 调用vip_wait_network()请求状态以同步。
3. 若希望等待执行完成才返回，请使用vip_run_network()。
4. 要设置网络优先级，请使用vip_set_network()。
5. 确保在此 API 调用之前已经绑定有效的输入/输出 buffer，否则报错 VIP_ERROR_MISSING_INPUT_OUTPUT。

2.4.1.3 vip_wait_network()

- **函数原型：** vip_status_e vip_wait_network(IN vip_network network);
- **功能描述：** 等待 VIP 硬件完成指定网络的推理。
- **参数说明：** network，网络句柄。
- **返回值：** VIP_SUCCESS，提交任务成功；其他，失败

2.4.1.4 vip_dup_network()

- **函数原型：** vip_status_e vip_dup_network(IN const void data, IN vip_uint32_t size_of_data, IN vip_dup_network_type_e type, IN vip_network network, OUT vip_network dup_network);
- **功能描述：** 复制现有网络的命令缓冲区以创建网络。
- **参数说明：** data，NBG 文件路径或 NBG 的缓冲区；size_of_data，NBG 的大小；type，NBG 文件的获取方式；network，用于复制的原始网络；dup_network，返回的网络句柄，失败时返回 VIP_NULL。
- **返回值：** VIP_SUCCESS，创建网络成功；其他，失败
- **其他说明：**

1. 网络系数（权重）是共享的，但不是重复的。
2. 如果复制的网络仍在使用的，请不要销毁原始的网络。
3. 参考《NPU_模型部署_开发指南》的《共享权重》一节介绍进行使用。

2.4.2 buffer 处理接口说明

2.4.2.1 vip_create_buffer_from_handle()

- **函数原型：**`vip_status_e vip_create_buffer_from_handle(IN const vip_buffer_create_params_t create_param, IN const vip_ptr handle_logical, IN vip_uint32_t handle_size, OUT vip_buffer buffer);`
- **功能描述：**根据句柄创建 NPU 的 buffer，实为把句柄关联的物理地址映射到新建 buffer。
- **参数说明：**`create_param`，参数结构体；`handle_logical`，提供给新建 buffer 的地址；`handle_size`，大小，要求 64bytes 对齐；`buffer`，返回的 buffer 句柄，失败时返回 `VIP_NULL`。
- **返回值：**`VIP_SUCCESS`，创建 buffer 成功；其他，失败
- **其他说明：**

1. 此 API，需使用 NPU MMU。否则，API 返回 `VIP_ERROR_FAILURE`。
2. buffer 大小，要求 64bytes 对齐。

2.4.2.2 vip_create_buffer_from_physical()

- **函数原型：**`vip_status_e vip_create_buffer_from_physical(IN const vip_buffer_create_params_t create_param, IN const vip_address_t physical_table, IN const vip_uint32_t size_table, IN vip_uint32_t physical_num, OUT vip_buffer buffer);`
- **功能描述：**根据指定的物理内存地址创建 NPU 的 buffer。
- **参数说明：**`create_param`，参数结构体；`physical_table`，包含物理内存地址的表；`size_table`，包含内存块大小的表；`physical_num`，离散内存块的数量，若连续内存则为 1；`buffer`，返回的 buffer 句柄，失败时返回 `VIP_NULL`。
- **返回值：**`VIP_SUCCESS`，创建 buffer 成功；其他，失败
- **其他说明：**

1. 若离散内存块，第一个内存块需 64bytes 对齐。

2.4.2.3 vip_create_buffer_from_fd()

- **函数原型：**`vip_status_e vip_create_buffer_from_fd(IN const vip_buffer_create_params_t create_param, IN vip_uint32_t fd, IN vip_uint32_t memory_size, OUT vip_buffer buffer);`
- **功能描述：**根据 DMA buffer 创建 NPU 的 buffer，实为把物理地址映射到新建 buffer。
- **参数说明：**`create_param`，参数结构体；`fd`，DMA buffer 的句柄；`memory_size`，内存大小，要求 64bytes 对齐；`buffer`，返回的 buffer 句柄，失败时返回 `VIP_NULL`。
- **返回值：**`VIP_SUCCESS`，创建 buffer 成功；其他，失败
- **其他说明：**

1. 此 API 仅适用于 Linux 和 Android。
2. 参考《NPU_模型部署_开发指南》的《零拷贝调用》一节介绍进行使用。

2.4.2.4 vip_create_buffer_cache()

- **函数原型：** vip_status_e vip_create_buffer_cache(IN vip_buffer_create_params_t create_param, IN vip_uint32_t size_of_param, OUT vip_buffer buffer);
- **功能描述：** 根据 buffer 参数使用 malloc 接口申请空间（带 cache，以及 64bytes 对齐），并使用 vip_create_buffer_from_handle() 新创建 buffer 供应用使用。
- **参数说明：** create_param，参数结构体；size_of_param，参数的大小；buffer，返回的 buffer 句柄，失败时返回 VIP_NULL。
- **返回值：** VIP_SUCCESS，创建 buffer 成功；其他，失败
- **其他说明：**

1. 此 API 在 1.13.0 版本及以上才新增。

2.5 返回值错误码说明

返回值错误码定义如下表所示：

错误码	错误详情
VIP_SUCCESS(0)	执行成功
VIP_ERROR_TIMEOUT(-1)	NPU 超时，可能 NPU 卡在某处
VIP_ERROR_IO(-2)	IO 相关错误
VIP_ERROR_INVALID_ARGUMENTS(-3)	参数无效
VIP_ERROR_NOT_SUPPORTED(-4)	未支持功能
VIP_ERROR_OUT_OF_RESOURCE(-5)	资源不足
VIP_ERROR_OUT_OF_MEMORY(-6)	内存不足
VIP_ERROR_INVALID_NETWORK(-7)	NBG 文件无效
VIP_ERROR_MISSING_INPUT_OUTPUT(-8)	未绑定有效的输入输出 buffer
VIP_ERROR_NETWORK_NOT_PREPARED(-9)	网络未准备好
VIP_ERROR_NETWORK_INCOMPATIBLE(-10)	NBG 文件与当前 NPU 硬件版本不匹配
VIP_ERROR_FAILURE(-11)	故障
VIP_ERROR_POWER_OFF(-12)	NPU 硬件断电
VIP_ERROR_POWER_STOP(-13)	NPU 已损坏
VIP_ERROR_RECOVERY(-14)	NPU 在挂起后需恢复
VIP_ERROR_CANCELED(-15)	网络被取消

3 libawnn 说明

基于 VIPLite API 封装的更简洁的 API，便于快速开发，详见[ai-sdk/examples/libawnn_viplite](#)。API 说明如下：

awnn_init()

- **函数原型：**void awnn_init();
- **功能描述：**初始化，每个应用进程只需调用 1 次，输入参数为预留 heap 内存大小
- **参数说明：**无
- **返回值：**无

awnn_uninit()

- **函数原型：**void awnn_uninit();
- **功能描述：**反初始化，应用进程退出时或不使用 NPU 时调用
- **参数说明：**无
- **返回值：**无

awnn_create()

- **函数原型：**Awnn_Context_t awnn_create(const char nbg);
- **功能描述：**通过 nbg 文件创建网络模型，返回 Awnn_Context_t 指针
- **参数说明：**nbg，NBG 文件
- **返回值：**新创建网络的对应信息指针

awnn_destroy()

- **函数原型：**void awnn_destroy(Awnn_Context_t *context);
- **功能描述：**销毁 Awnn_Context_t 指定的网络
- **参数说明：**context，指定网络对应的信息
- **返回值：**无

awnn_set_input_buffers()

- **函数原型：**void awnn_set_input_buffers(Awnn_Context_t *context, void **input_buffers);

- **功能描述：**设置输入 buffer
- **参数说明：**context，指定网络对应的信息；
- **返回值：**无

awnn_get_output_buffers()

- **函数原型：**float **awnn_get_output_buffers(Awnn_Context_t *context);
- **功能描述：**获取输出 buffer
- **参数说明：**context，指定网络对应的信息
- **返回值：**输出 buffer 的指针

awnn_get_output_buffer()

- **函数原型：**void awnn_get_output_buffer(Awnn_Context_t info, int i);
- **功能描述：**获取输出 buffer
- **参数说明：**info，指定网络对应的信息；i，输出 buffer 的索引
- **返回值：**无

awnn_run()

- **函数原型：**void awnn_run(Awnn_Context_t *context);
- **功能描述：**运行 Awnn_Context_t 指定的网络
- **参数说明：**context，指定网络对应的信息
- **返回值：**无

awnn_dump_io()

- **函数原型：**void awnn_dump_io(Awnn_Context_t context, const char path);
- **功能描述：**dump 出网络的 tensor
- **参数说明：**context，指定网络对应的信息；path，保存路径
- **返回值：**无




著作权声明

版权所有 © 2024 珠海全志科技股份有限公司。保留一切权利。

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

商标声明

、 全志科技、（不完全列举）均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标，产品名称，和服务名称，均由其各自所有人拥有。

免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。