BMLIB 开发参考手册

发行版本 0.4.9

SOPHGO

2025 年 01 月 23 日

目录

CHAPTER 1

声明

法律声明

注意

您购买的产品、服务或特性等应受算能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的购买或使用范围之内。除非合同另有约定, 算能对本文档内容不做任何明示或默示的声明或保证。由于产品版本升级或其他原因, 本文档内容会不定期进行更新。除非另有约定, 本文档仅作为使用指导, 本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

技术支持

地址
北京市海淀区丰豪东路 9 号院中关村集成电路设计园 (ICPARK)1 号楼

邮编
100094

网址
https://www.sophgo.com/

邮箱
sales@sophgo.com

电话
010-57590723

CHAPTER 2

Release note

版本	发布日期	说明
V0.1.0	2025.01.20	初版发布, 包含 bmlib 接口使用指南。

CHAPTER 3

快速开始

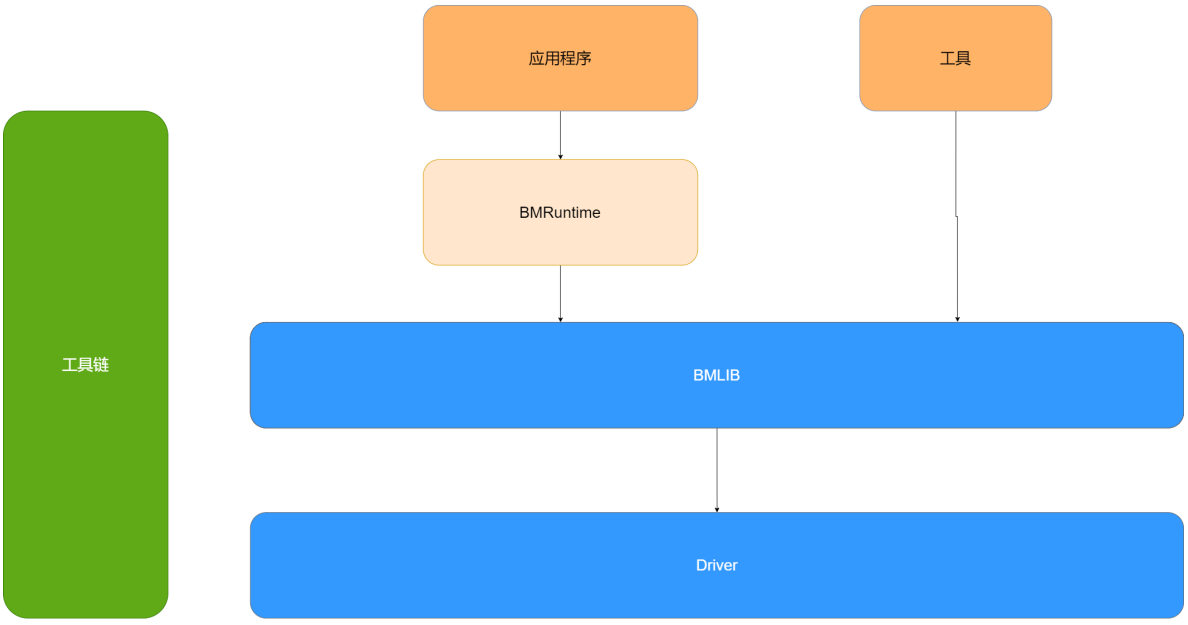
3.1 术语解释

术语	说明
TPU	SGTPUV8 内部神经网络处理单元
SOC Mode	一种产品形态，SDK 运行于 A53 AARCH64 平台，TPU 作为平台总线设备。
Driver	Driver 是 API 接口访问硬件的通道
Gmem	卡上用于 NPU 加速的 DDR 内存
Handle	一个用户进程 (线程) 使用设备的句柄，一切操作都要通过 handle

CHAPTER 4

bmlib 的基本概念和功能

基于算能神经网络加速芯片设计的 SDK 的简单功能框图如下：



bmlib 是在内核驱动之上封装的一层底层软件库，完成的主要功能有：

- 设备 handle 的创建和销毁
- Memory help 函数接口
- Global memory 的申请和释放
- 数据在 host 和 global memory 之间的搬运

- 数据在 global memory 内部的搬运
- Global memory 在 host 端的映射和一致性管理
- 杂项管理接口

4.1 Handle 的概念

我们的神经网络加速设备，在 SOC 模式，安装完 tpu 的驱动后，会成为一个标准的字符设备。上层用户进程如果要使用这个设备，需要在这个设备上创建一个 handle 句柄。

Handle 是管理 api，申请 memory，释放 memory 的 handle，如果一个进程创建了两个 handle，名字为 handle_A1，handle_A2，这是两个独立的 handle。

如果线程 B 是进程 A 的子线程，进程 A 创建 handle_A，线程 B 创建 handle_B，那么 handle_A 和 handle_B 也是两个独立的 handle。

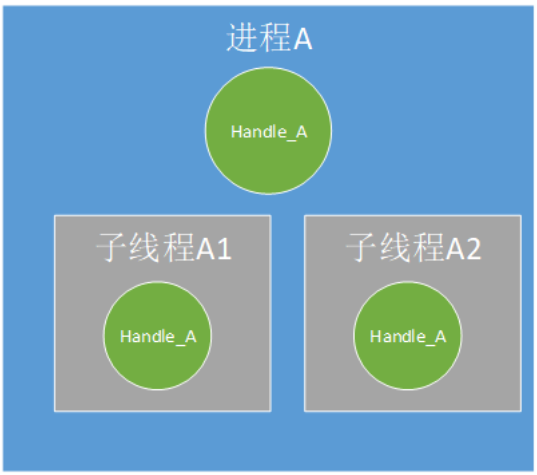
如果一个 api 是通过 handle_A 发送的，则必须通过 handle_A sync；

如果一块 memory 是通过 handle_A 申请的，则必须通过 handle_A 释放；

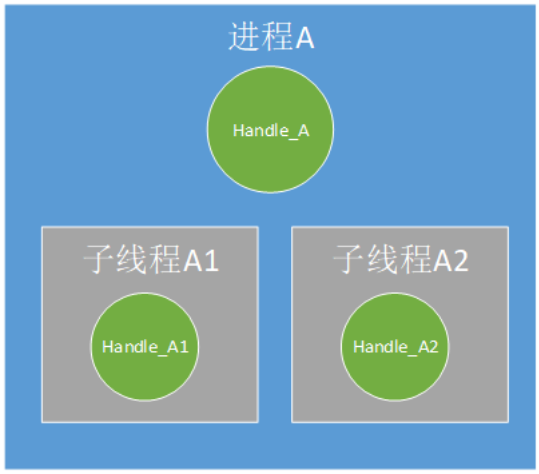
需要注意的是 handle 的创建者和使用者可以不一样，例如进程 A 创建了 handle_A，A 的子线程 A1 也可以使用 handle_A，但是 A1 通过 handle_A 申请的 memory 在统计上算作 A 申请的。

我们推荐以下四种使用 handle 的方式：

进程 A 中创建 Handle_A，Handle_A 只在进程 A 中使用；



进程 A 中创建 Handle_A，Handle_A 在进程 A 的两个子线程（可以是多个，图中两个只是举例示意）中使用；



进程 A 及其子线程（可以是多个，图中两个只是举例示意）各自创建并使用自己创建的 Handle；

进程 A 创建多个 Handle，每个子线程分别使用这些 Handle。

4.2 Api 的概念和同步

Host 这端的软件如果想让 tpu 完成一个任务，需要向 tpu 发送一个 “api”，类似于一个命令。请注意发送 api 的函数和 api 的执行完成是异步的，他会在发送完 api 后阻塞等待，直到信号量通知后发送 api 的接口继续处理，发送 api 的接口返回才是真正完成。

目前发送 api 的动作，都已经封在 bmcv/bmrt 功能库中了，客户无法直接发送 api，只能通过调用 bmcv/bmrt 的接口发送 api。

调用完 bmcv/bmrt 的接口发送 api 后，是否需要调用 sync 函数等待 api 的完成，请参考 bmcv/bmrt 文档，bmcv/bmrt 的接口可能已经将 sync 函数也封装在 bmcv/bmrt 的接口函数中了，这样 bmcv/bmrt 的接口函数返回后，api 已经完成了。

4.3 TPU 驱动基本功能介绍

主要功能是为算子库执行提供环境支持，包含 TPU 硬件时钟使能、TPU 寄存器地址范围的映射，TPU 或 GDMA 各自 clk 的开启是通过在 TPU_SYS 寄存器中将 reg_clk_tpu_en 和 reg_clk_gdma_en 各自置高来实现的。在使用 tpu 之前，需要通过 tpu 内部 cfg_en 信号对 tpu 进行使能，否则 tpu 的门控时钟不会被打开；配置 tpu_reg 的 0x100 地址为 1 后开始进行指令配置，举例说明：

```
# 读写tpu寄存器地址，可以通过mmap接口进行地址映射，然后再进行读写操作
ioctl(fd, BMDEV_SET_IOMAP_TPYE, {type});
mmap(NULL, size, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
# 其中type对应TPU_sys各个模块类型，如下：
enum {
  MMAP_GDMA = 0,
  MMAP_SYS,
```

(续下页)

(接上页)

```
  MMAP_REG,
  MMAP_SMEM,
  MMAP_LMEM
};
```

CHAPTER 5

bmlib 详细接口介绍

5.1 设备 handle 的创建和销毁

5.1.1 bm_dev_getcount

函数原型：bm_status_t bm_dev_getcount(int *count)

函数作用：获取当前系统中，存在多少个 sophon 设备，如果获取的设备个数为 N，则 devid 的合法取值为 [0,N-1]。

参数介绍：

参数名	输入/输出	说明
count	输出	用于存放 sophon 设备个数的指针

返回值：BM_SUCCESS 代表获得正确个数；其他错误码代表无法获取个数

5.1.2 bm_dev_query

函数原型：bm_status_t bm_dev_query(int devid)

函数作用：根据设备索引值查询某个设备是否存在

参数介绍：

参数名	输入/输出	说明
devid	输入	被查询设备的索引值

返回值：BM_SUCCESS 代表存在这个设备；其他错误码代表不存在这个设备

5.1.3 bm_dev_request

函数原型：bm_status_t bm_dev_request(bm_handle_t *handle, int devid)

函数作用：在指定的设备上创建 handle

参数介绍：

参数名	输入/输出	说明
handle	输出	保存创建的 handle 的指针
devid	输入	指定具体设备

返回值：BM_SUCCESS 代表创建成功；其他错误码代表创建失败

5.1.4 bm_get_devid

函数原型：int bm_get_devid(bm_handle_t *handle)

函数作用：根据给定 handle 获取设备索引

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄

返回值：handle 指向的 int 型设备索引

5.1.5 bm_dev_free

函数原型：void bm_dev_free(bm_handle_t handle)

函数作用：释放创建的 handle

参数介绍：

参数名	输入/输出	说明
handle	输入	将要被释放的 handle

返回值：无

5.2 memory help 函数接口

5.2.1 bm_mem_get_type

函数原型：bm_mem_type_t bm_mem_get_type(struct bm_mem_desc mem);

函数作用：获取一块 memory 的种类

参数介绍：

参数名	输入/输出	说明
mem	输入	被查询的 memory

返回值：BM_MEM_TYPE_DEVICE，代表 global memory；BM_MEM_TYPE_SYSTEM，代表 linux 系统 user 层 memory。

5.2.2 bm_mem_get_device_addr

函数原型：unsigned long long bm_mem_get_device_addr(struct bm_mem_desc mem);

函数作用：获取 device 类型的 memory 的地址

参数介绍：

参数名	输入/输出	说明
mem	输入	被查询的 memory

返回值：返回 device memory 的地址，64bit 的一个无符号数字

5.2.3 bm_mem_set_device_addr

函数原型：void bm_mem_set_device_addr(struct bm_mem_desc *pmem, unsigned long long addr);

函数作用：设置一个 device 类型 memory 的地址

参数介绍：

参数名	输入/输出	说明
pmem	输入/输出	被设置的 memory 的指针
addr	输入	memory 被设置的地址

返回值：无

5.2.4 bm_mem_get_device_size

函数原型：unsigned int bm_mem_get_device_size(struct bm_mem_desc mem);

函数作用：获取一块 device 类型的 memory 的大小

参数介绍：

参数名	输入/输出	说明
mem	输入	被查询的 memory

返回值：返回 memory 大小，32 位的无符号数

5.2.5 bm_mem_set_device_size

函数原型：void bm_mem_set_device_size(struct bm_mem_desc *pmem, unsigned int size);

函数作用：设置一块 device 类型的 memory 的大小

参数介绍：

参数名	输入/输出	说明
pmem	输入/输出	被设置的 memory 的指针
size	输入	memory 的大小，单位是 byte

返回值：无

5.2.6 bm_set_device_mem

函数原型：void bm_set_device_mem(bm_device_mem_t *pmem, unsigned int size, unsigned long long addr);

函数作用：填充一个 device 类型的 memory 的大小和地址

参数介绍：

参数名	输入/输出	说明
pmem	输入/输出	被设置的 memory 的指针
size	输入	memory 的大小，单位是 byte
addr	输入	memory 的地址

返回值：无

5.2.7 bm_mem_from_device

函数原型：bm_device_mem_t bm_mem_from_device(unsigned long long device_addr, unsigned int len);

函数作用：根据地址和大小构建一个 bm_device_mem_t 类型的结构体

参数介绍：

参数名	输入/输出	说明
device_addr	输入	memory 的地址
len	输入	memory 的大小，单位是 byte

返回值：一个 bm_device_mem_t 类型的结构体

5.2.8 bm_mem_get_system_addr

函数原型：void *bm_mem_get_system_addr(struct bm_mem_desc mem);

函数作用：获取 system 类型 memory 的地址

参数介绍：mem，被查询的 memory

参数名	输入/输出	说明
mem	输入	被查询的 memory

返回值：返回一个 memory 的地址

5.2.9 bm_mem_set_system_addr

函数原型：void bm_mem_set_system_addr(struct bm_mem_desc *pmem, void *addr);

函数作用：设置一个 system 类型 memory 的地址

参数介绍：

参数名	输入/输出	说明
pmem	输入/输出	被设置的 memory 的指针
addr	输入	system 地址指针

返回值：无

5.2.10 bm_mem_from_system

函数原型：bm_system_mem_t bm_mem_from_system(void *system_addr);

函数作用：根据一个 system 指针构建一个 bm_system_mem_t 类型的结构体

参数介绍：

参数名	输入/输出	说明
system_addr	输入	system 地址指针

返回值：一个 bm_system_mem_t 类型的结构体

5.3 Global memory 的申请和释放

5.3.1 bm_mem_null

函数原型：bm_device_mem_t bm_mem_null(void);

函数作用：返回一个类型非法的 bm memory 结构体

参数介绍：无

返回值：一个 bm_device_mem_t 类型的结构体

5.3.2 bm_malloc_neuron_device

函数原型：bm_status_t bm_malloc_neuron_device(bm_handle_t handle, bm_device_mem_t *pmem, int n, int c, int h, int w);

函数作用：根据 batch 的形状信息申请一块 device 类型的 memory，每个神经元的大小为一个 FP32(4 bytes)

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
pmem	输出	分配出 device memory 的指针
n/c/h/w	输入	batch 的形状

返回值：BM_SUCCESS 代表分配成功；其他错误码代表分配失败

5.3.3 bm_malloc_device_dword

函数原型：bm_status_t bm_malloc_device_dword(bm_handle_t handle, bm_device_mem_t *pmem, int count);

函数作用：分配 count 个 DWORD（4 bytes）大小的 device 类型的 memory

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
pmem	输出	分配出 device memory 的指针
count	输入	需要分配的 dword 的个数

返回值：BM_SUCCESS 代表分配成功；其他错误码代表分配失败

5.3.4 bm_malloc_device_byte

函数原型：bm_status_t bm_malloc_device_byte(bm_handle_t handle, bm_device_mem_t *pmem, unsigned int size);

函数作用：分配指定字节个数大小的 device 类型的 memory

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
pmem	输出	分配出 device memory 的指针
size	输入	需要分配的 byte 的个数

返回值：BM_SUCCESS 代表分配成功；其他错误码代表分配失败

5.3.5 bm_free_device

函数原型：void bm_free_device(bm_handle_t handle, bm_device_mem_t mem);

函数作用：释放一块 device 类型的 memory

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
mem	输入	要释放的 device memory

返回值：无

5.4 数据在 host 和 global memory 之间的搬运

5.4.1 bm_memcpy_s2d

函数原型：bm_status_t bm_memcpy_s2d(bm_handle_t handle, bm_device_mem_t dst, void *src);

函数作用：拷贝 system 内存到 device 类型的内存中

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
dst	输入	目标 device memory 的结构体
src	输入	指向 system 内存的指针

返回值：BM_SUCCESS 代表传输成功；其他错误码代表传输失败

5.4.2 bm_memcpy_s2d_partial_offset

函数原型：bm_status_t bm_memcpy_s2d_partial_offset(bm_handle_t handle,

bm_device_mem_t dst, void *src,

unsigned int size,

unsigned int offset);

函数作用：拷贝 system 内存到 device 类型内存，指定长度和 device 内存的起始地址 offset，效果是从 src 拷贝 size 长度的数据到 (dst 起始地址 +offset) 这个位置上。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
dst	输入	目标 device memory 的结构体
src	输入	指向 system 内存的指针
size	输入	拷贝的长度
offset	输入	本次拷贝在 device memory 端相对于这块 device memory 起始地址的 offset

返回值：BM_SUCCESS 代表传输成功；其他错误码代表传输失败

5.4.3 bm_memcpy_s2d_partial

函数原型：bm_status_t bm_memcpy_s2d_partial(bm_handle_t handle, bm_device_mem_t dst, void *src, unsigned int size);

函数作用：拷贝 system 内存到 device 类型内存，指定长度；效果是从 src 拷贝 size 长度的数据到 dst 起始地址这个位置上。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
dst	输入	目标 device memory 的结构体
src	输入	指向 system 内存的指针
size	输入	拷贝的长度

返回值：BM_SUCCESS 代表传输成功；其他错误码代表传输失败

5.4.4 bm_memcpy_d2s

函数原型：bm_status_t bm_memcpy_d2s(bm_handle_t handle, void *dst, bm_device_mem_t src);

函数作用：拷贝 device 类型内存到 system 内存

参数介绍：handle, 设备句柄；dst，指向 system 内存的指针结构体；src，device memory；

参数名	输入/输出	说明
handle	输入	设备句柄
dst	输入	指向 system 内存的指针
src	输入	源 device memory 的结构体

返回值：BM_SUCCESS 代表传输成功；其他错误码代表传输失败

5.4.5 bm_memcpy_d2s_partial_offset

函数原型：bm_status_t bm_memcpy_d2s_partial_offset(bm_handle_t handle, void *dst, bm_device_mem_t src, unsigned int size, unsigned int offset);

函数作用：拷贝 device 类型内存到 system 内存，指定大小，和 device memory 端的 offset，效果是从 device memory 起始地址 +offset 拷贝 size 字节数据到 dst 上。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
dst	输入	指向 system 内存的指针
src	输入	源 device memory 的结构体
size	输入	拷贝的长度（单位为 byte）
offset	输入	本次拷贝在 device memory 端相对于这块 device memory 起始地址的 offset

返回值：BM_SUCCESS 代表传输成功；其他错误码代表传输失败

5.4.6 bm_memcpy_d2s_partial

函数原型：bm_status_t bm_memcpy_d2s_partial(bm_handle_t handle, void *dst, bm_device_mem_t src, unsigned int size);

函数作用：拷贝 device 类型内存到 system 内存，指定大小；效果是从 device memory 起始地址拷贝 size 字节数据到 dst 上。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
dst	输入	指向 system 内存的指针
src	输入	源 device memory 的结构体
size	输入	拷贝的长度（单位为 byte）

返回值：BM_SUCCESS 代表传输成功；其他错误码代表传输失败

5.4.7 bm_mem_convert_system_to_device_neuron

函数原型：bm_status_t bm_mem_convert_system_to_device_neuron(bm_handle_t handle, struct bm_mem_desc *dev_mem, struct bm_mem_desc sys_mem, bool need_copy, int n, int c, int h, int w);

函数作用：按照 batch 形状申请一块 device 类型的 memory（一个神经元大小为 FP32(4 bytes)），按需将一段 system memory 内存 copy 到这块 device memory 上。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
dev_mem	输出	指向分配出的 device memory 的指针
sys_mem	输入	system 类型的 memory 结构体
need_copy	输入	是否需要将 system 内存 copy 到新分配的这块 device memory 上
n/c/h/w	输入	batch 的形状

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.4.8 bm_mem_convert_system_to_device_neuron_byte

函数原型：bm_status_t bm_mem_convert_system_to_device_neuron_byte(

bm_handle_t handle, struct bm_mem_desc *dev_mem, struct bm_mem_desc sys_mem, bool need_copy, int n, int c, int h, int w);

函数作用：按照 batch 形状申请一块 device 类型的 memory（一个神经元大小为 1 bytes），按需将一段 system memory 内存 copy 到这块 device memory 上。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
dev_mem	输出	指向分配出的 device memory 的指针
sys_mem	输入	system 类型的 memory 结构体
need_copy	输入	是否需要将 system 内存 copy 到新分配的这块 device memory 上
n/c/h/w	输入	batch 的形状

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.4.9 bm_mem_convert_system_to_device_coeff

函数原型：bm_status_t bm_mem_convert_system_to_device_coeff(bm_handle_t handle, struct bm_mem_desc *dev_mem, struct bm_mem_desc sys_mem, bool need_copy, int coeff_count);

函数作用：按照系数元素个数申请一块 device 类型的 memory（一个系数元素大小为 4 个 bytes），按需将一段 system memory 内存 copy 到这块 device memory 上。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
dev_mem	输出	指向分配出的 device memory 的指针
sys_mem	输入	system 类型的 memory 结构体
need_copy	输入	是否需要将 system 内存 copy 到新分配的这块 device memory 上
coeff_count	输入	系数元素的个数

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.4.10 bm_mem_convert_system_to_device_coeff_byte

函数原型：bm_status_t bm_mem_convert_system_to_device_coeff_byte(

bm_handle_t handle, struct bm_mem_desc *dev_mem, struct bm_mem_desc sys_mem, bool need_copy, int coeff_count);

函数作用：按照系数元素个数申请一块 device 类型的 memory（一个系数元素大小为 1 个 byte），按需将一段 system memory 内存 copy 到这块 device memory 上。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
dev_mem	输出	指向分配出的 device memory 的指针
sys_mem	输入	system 类型的 memory 结构体
need_copy	输入	是否需要将 system 内存 copy 到新分配的这块 device memory 上
coeff_count	输入	系数元素的个数，单位 byte

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.5 数据在 global memory 内部的搬运

5.5.1 bm_memcpy_d2d

函数原型：bm_status_t bm_memcpy_d2d(bm_handle_t handle, bm_device_mem_t dst, int dst_offset, bm_device_mem_t src, int src_offset, int len);

函数作用：将一块 device 类型的 memory 拷贝到另外一块 device 类型的 memory，指定大小和目的、源数据的 offset；效果是从 (src 起始地址 + src_offset) 拷贝 len 个 DWORD（4 字节）的数据到 (dst 起始地址 + dst_offset)

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
dst	输入	目标 device memory 结构体
dst_offset	输入	用于计算数据拷贝的起始位置的 offset
src	输入	源 device memory 结构体
src_offset	输入	用于计算数据拷贝的起始位置的 offset
len	输入	数据 copy 长度，单位是 DWORD（4 字节）

返回值：BM_SUCCESS 代表传输成功；其他错误码代表传输失败

5.5.2 bm_memcpy_d2d_with_core

函数原型：bm_status_t bm_memcpy_d2d_with_core(bm_handle_t handle, bm_device_mem_t dst, int dst_offset, bm_device_mem_t src, int src_offset, int len, int core_id);

函数作用：指定使用第 core_id 个 GDMA，将一块 device 类型的 memory 拷贝到另外一块 device 类型的 memory，指定大小和目的、源数据的 offset；效果是从 (src 起始地址 + src_offset) 拷贝 len 个 DWORD（4 字节）的数据到 (dst 起始地址 + dst_offset)

参数介绍：

返回值：BM_SUCCESS 代表传输成功；其他错误码代表传输失败

5.5.3 bm_memcpy_d2d_byte

函数原型：bm_status_t bm_memcpy_d2d_byte(bm_handle_t handle, bm_device_mem_t dst, size_t dst_offset, bm_device_mem_t src, size_t src_offset, size_t size);

函数作用：将一块 device 类型的 memory 拷贝到另外一块 device 类型的 memory，指定大小和目的、源数据的 offset；效果是从 (src 起始地址 + src_offset) 拷贝 len 个字节的数据到 (dst 起始地址 + dst_offset)

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
dst	输入	目标 device memory 结构体
dst_offset	输入	用于计算数据拷贝的起始位置的 offset
src	输入	源 device memory 结构体
src_offset	输入	用于计算数据拷贝的起始位置的 offset
size	输入	数据 copy 长度，单位是字节

返回值：BM_SUCCESS 代表传输成功；其他错误码代表传输失败

5.5.4 bm_memcpy_d2d_byte_with_core

函数原型：bm_status_t bm_memcpy_d2d_byte_with_core(bm_handle_t handle, bm_device_mem_t dst, size_t dst_offset, bm_device_mem_t src, size_t src_offset, size_t size, int core_id);

函数作用：指定使用第 core_id 个 GDMA，将一块 device 类型的 memory 拷贝到另外一块 device 类型的 memory，指定大小和目的、源数据的 offset；效果是从 (src 起始地址 + src_offset) 拷贝 len 个字节的数据到 (dst 起始地址 + dst_offset)

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
dst	输入	目标 device memory 结构体
dst_offset	输入	用于计算数据拷贝的起始位置的 offset
src	输入	源 device memory 结构体
src_offset	输入	用于计算数据拷贝的起始位置的 offset
size	输入	数据 copy 长度，单位是字节
core_id	输入	搬运的 GDMA 设备 id

返回值：BM_SUCCESS 代表传输成功；其他错误码代表传输失败

5.5.5 bm_memcpy_d2d_stride

函数原型：bm_status_t bm_memcpy_d2d_stride(bm_handle_t handle, bm_device_mem_t dst, int dst_stride, bm_device_mem_t src,

int src_stride, int count, int format_size);

函数作用：将一块 device 类型的 memory 拷贝到另外一块 device 类型的 memory，指定目的、源数据的 stride，数据的个数，以及数据的类型字节大小；效果是从 src 起始地址按 src_stride

为间隔大小拷贝 count 个元素大小为 format_size 字节的数据到 dst 起始地址，以 dst_stride 为间隔大小存储。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
dst	输入	目标 device memory 结构体
dst_stride	输入	目标每个元素的间隔
src	输入	源 device memory 结构体
src_stride	输入	源数据的每个元素的间隔
count	输入	需要拷贝的元素的个数
format_size	输入	每个元素的字节大小，比如 float 类型字节大小是 4，uint8_t 类型字节大小是 1；拷贝个数、stride 都是以 format_size 为单位

限制条件：dst_stride 通常为 1；只有一种情况可以不为 1：dst_stride = 4 且 src_stride = 1 且 format_size = 1。

返回值：BM_SUCCESS 代表传输成功；其他错误码代表传输失败

5.5.6 bm_memcpy_d2d_stride_with_core

函数原型：bm_status_t bm_memcpy_d2d_stride_with_core(bm_handle_t handle, bm_device_mem_t dst, int dst_stride, bm_device_mem_t src, int src_stride, int count, int format_size, int core_id);

函数作用：指定使用第 core_id 个 GDMA，将一块 device 类型的 memory 拷贝到另外一块 device 类型的 memory，指定目的、源数据的 stride，数据的个数，以及数据的类型字节大小；效果是从 src 起始地址按 src_stride 为间隔大小拷贝 count 个元素大小为 format_size 字节的数据到 dst 起始地址，以 dst_stride 为间隔大小存储。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
dst	输入	目标 device memory 结构体
dst_stride	输入	目标每个元素的间隔
src	输入	源 device memory 结构体
src_stride	输入	源数据的每个元素的间隔
count	输入	需要拷贝的元素的个数
format_size	输入	每个元素的字节大小，比如 float 类型字节大小是 4，uint8_t 类型字节大小是 1；拷贝个数、stride 都是以 format_size 为单位
core_id	输入	搬运的 GDMA 设备 id

限制条件：dst_stride 通常为 1；只有一种情况可以不为 1：dst_stride = 4 且 src_stride = 1 且 format_size = 1。

返回值：BM_SUCCESS 代表传输成功；其他错误码代表传输失败

5.5.7 bm_memset_device

函数原型：bm_status_t bm_memset_device(bm_handle_t handle, const int value, bm_device_mem_t mem);

函数作用：用 value 填充一块 device memory

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
value	输入	需要填充的值
mem	输入	目标 device memory 结构体，此函数只能填充大小为 4 字节整数倍的 global memory 空间

返回值：BM_SUCCESS 代表填充成功；其他错误码代表填充失败

本函数的作用和 bm_memset_device_ext 函数 mode 为 4 时的作用一样。

5.5.8 bm_memset_device_ext

函数原型：bm_status_t bm_memset_device_ext(bm_handle_t handle, void* value, int mode, bm_device_mem_t mem);

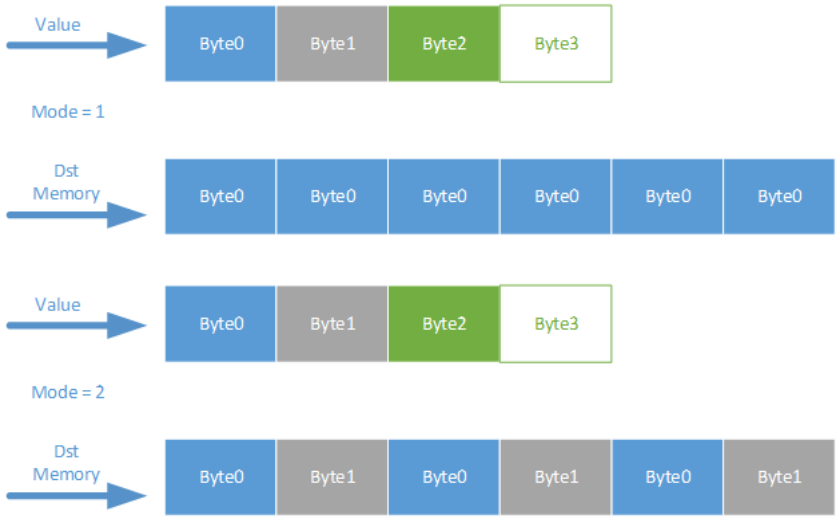
函数作用：用 value 指向的内容和指定的模式填充一块 device memory

参数介绍：

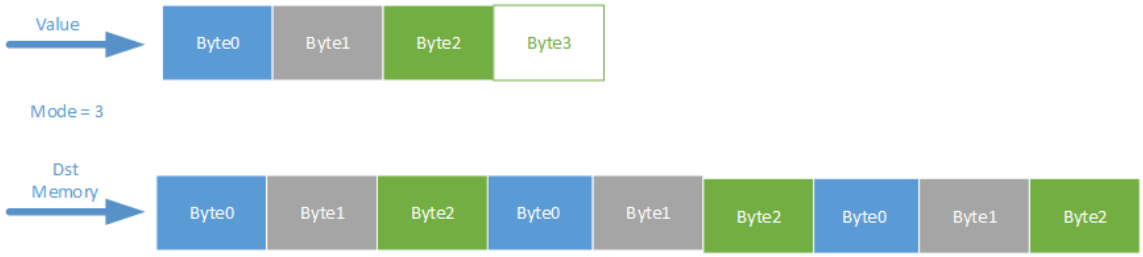
参数名	输入/输出	说明
handle	输入	设备句柄
value	输入	指向需要填充的值
mode	输入	填充模式，详见下图
mem	输入	目标 device memory 结构体

返回值：BM_SUCCESS 代表填充成功；其他错误码代表填充失败

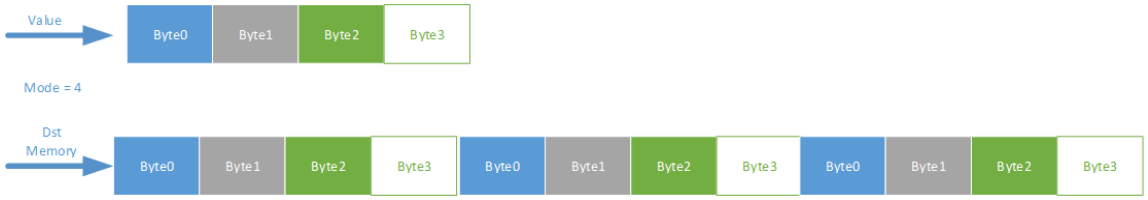
此函数的功能示意图如下：



Mode 为 2 时，dst memory 的 size 必须是 2 字节的整数倍



Mode 为 3 时，dst memory 的 size 必须是 3 字节的整数倍



Mode 为 4 时，dst memory 的 size 必须是 4 字节的整数倍

5.6 Global memory 在 host 端的映射和一致性管理

5.6.1 bm_mem_mmap_device_mem

函数原型：bm_status_t bm_mem_mmap_device_mem(bm_handle_t handle, bm_device_mem_t *dmem, unsigned long long *vmem);

函数作用：将一块 global memory 映射到 host 的 user 空间。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
dmem	输入	执行被映射的 global memory 的结构体
vmem	输出	存储映射出来的虚拟地址的指针

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.6.2 bm_mem_mmap_device_mem_no_cache

函数原型：bm_status_t bm_mem_mmap_device_mem_no_cache(bm_handle_t handle, bm_device_mem_t *dmem, unsigned long long *vmem);

函数作用：将一块 global memory 映射到 host 的 user 空间，并关闭 cache（只在 soc 模式下面有效，pcie 模式下不支持）

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
dmem	输入	执行被映射的 global memory 的结构体
vmem	输出	存储映射出来的虚拟地址的指针

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.6.3 bm_mem_vir_to_phy

函数原型：bm_status_t bm_mem_vir_to_phy(bm_handle_t handle, unsigned long long vmem，unsigned long long *device_mem);

函数作用：SOC 模式下，可以将 bm_mem_mmap_device_mem 函数得到的虚拟地址转换成 device 内存的物理地址。（只在 soc 模式下面有效，pcie 模式下不支持）

参数名	输入/输出	说明
handle	输入	设备句柄
vmem	输入	虚拟地址
device_mem	输出	设备上的物理地址

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.7 API 的同步

5.8 设备管理接口

5.8.1 bm_get_misc_info

函数原型：bm_status_t bm_get_misc_info(bm_handle_t handle, struct bm_misc_info *pmisc_info);

函数作用：获取设备相关的 misc 信息

参数介绍：

参数名	输入/输出	说明
Handle	输入	设备句柄
pmisc_info	输出	存放 misc 数据的指针

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.8.2 bm_get_card_num

函数原型：bm_status_t bm_status_t bm_get_card_num(unsigned int *card_num);

函数作用：获取设备上卡的数量

参数介绍：

参数名	输入/输出	说明
card_num	输出	存放卡数量的指针

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.8.3 bm_get_card_id

函数原型：bm_status_t bm_get_card_id(bm_handle_t handle, unsigned int *card_id);

函数作用：获取设备对应卡的编号

参数介绍：

参数名	输入/输出	说明
Handle	输入	设备句柄
card_id	输出	存放卡 id 的指针

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.8.4 bm_get_chip_num_from_card

函数原型：bm_get_chip_num_from_card(unsigned int card_id, unsigned int *chip_num, unsigned int *dev_start_index);

函数作用：获取卡上的设备编号

参数介绍：

参数名	输入/输出	说明
card_id	输入	卡编号
chip_num	输出	卡上设备数量
dev_start_index	输出	卡上设备起始编号

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.8.5 bm_get_chipid

函数原型：bm_status_t bm_get_chipid(bm_handle_t handle, unsigned int *p_chipid);

函数作用：获取设备对应的芯片 ID(0x1684 和 0x1686)

参数介绍：

参数名	输入/输出	说明
Handle	输入	设备句柄
p_chipid	输出	存放芯片 ID 的指针

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.8.6 bm_get_stat

函数原型：bm_status_t bm_get_stat(bm_handle_t handle, bm_dev_stat_t *stat);

函数作用：获取 handle 对应的设备的运行时统计信息

参数介绍：

参数名	输入/输出	说明
Handle	输入	设备句柄
Stat	输出	存放统计信息的指针

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.8.7 bmlib_log_get_level

函数原型：int bmlib_log_get_level(void);

函数作用：获取 bmlib log 等级

参数介绍：void

返回值：bmlib log 等级

5.8.8 bmlib_log_set_level

函数原型：void bmlib_log_set_level(int level);

函数作用：设置 bmlib log 等级

参数介绍：

参数名	输入/输出	说明
Level	输入	要设置的 bmlib log 的等级

返回值：void

5.8.9 bmlib_log_set_callback

函数原型：void bmlib_log_set_callback((callback)(const char* , int , const char, va_list));

函数作用：设置 callback 获取 bmlib log

参数介绍：

参数名	输入/输出	说明
Callback	输入	设置获取 bmlib log 的回调函数的函数指针

返回值：void

5.8.10 bm_get_sn

函数原型：bm_status_t bm_get_sn(bm_handle_t handle, char *sn);

函数作用：获取板卡序列号（共 17 位）。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
sn	输出	要获取 sn 的函数指针

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.8.11 bm_get_tpu_minclk

函数原型：bm_status_t bm_get_tpu_minclk(bm_handle_t handle, unsigned int *tpu_minclk);

函数作用：获取句柄对应设备的最小工作频率，默认单位兆赫兹（MHz）。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
tpu_minclk	输出	要获取 tpu_minclk 的函数指针

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.8.12 bm_get_tpu_maxclk

函数原型：bm_status_t bm_get_tpu_maxclk(bm_handle_t handle, unsigned int *tpu_maxclk);

函数作用：获取句柄对应设备的最大工作频率，默认单位兆赫兹（MHz）。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
tpu_maxclk	输出	要获取 tpu_maxclk 的函数指针

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.8.13 bm_get_driver_version

函数原型：bm_status_t bm_get_driver_version(bm_handle_t handle, int *driver_version);

函数作用：获取板卡安装的驱动版本。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
dr iver_version	输出	要获取 driver_version 的函数指针

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.8.14 bm_get_board_name

函数原型：bm_status_t bm_get_board_name(bm_handle_t handle, char *name);

函数作用：获取当前板卡的名称，名称: 芯片 id-板卡类型（如：1684-SC5+）。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
name	输出	要获取 name 的函数指针

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.8.15 bmcpu_reset_cpu

函数原型：bm_status_t bmcpu_reset_cpu(bm_handle_t handle);

函数作用：使 A53 进入关机状态。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.8.16 bm_reset_tpu

函数原型：bm_status_t bm_reset_tpu(bm_handle_t handle);

函数作用：安全的重启 TPU 系统（仅 bm1688 使用）

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

CHAPTER 6

相关数据结构定义

6.1 bm_status_t

typedef enum {

BM_SUCCESS = 0,

BM_ERR_DEVNOTREADY = 1, /* Device not ready yet */

BM_ERR_FAILURE = 2, /* General failure */

BM_ERR_TIMEOUT = 3, /* Timeout */

BM_ERR_PARAM = 4, /* Parameters invalid */

BM_ERR_NOMEM = 5, /* Not enough memory */

BM_ERR_DATA = 6, /* Data error */

BM_ERR_BUSY = 7, /* Busy */

BM_ERR_NOFEATURE = 8, /* Not supported yet */

BM_NOT_SUPPORTED = 9

} bm_status_t;

6.2 bm_mem_type_t

```
typedef enum {
BM_MEM_TYPE_DEVICE = 0,
BM_MEM_TYPE_HOST = 1,
BM_MEM_TYPE_SYSTEM = 2,
BM_MEM_TYPE_INT8_DEVICE = 3,
BM_MEM_TYPE_INVALID = 4
} bm_mem_type_t;
```

6.3 bm_mem_flags_t

```
typedef union {
struct {
bm_mem_type_t mem_type : 3;
unsigned int reserved : 29;
} u;
unsigned int rawflags;
} bm_mem_flags_t;
```

6.4 bm_mem_desc_t

```
typedef struct bm_mem_desc {
union {
struct {
unsigned long device_addr;
unsigned int reserved;
int dmabuf_fd;
} device;
struct {
void *system_addr;
unsigned int reserved0;
```

```
int reserved1;
} system;
} u;
bm_mem_flags_t flags;
unsigned int size;
} bm_mem_desc_t;
```

6.5 bm_misc_info

```
struct bm_misc_info {
int pcie_soc_mode; /0—pcie; 1—soc/
int ddr_ecc_enable; /0—disable; 1—enable/
unsigned int chipid;
#define BM1682_CHIPID_BIT_MASK (0X1 F 0)
#define BM1684_CHIPID_BIT_MASK (0X1 F 1)
unsigned long chipid_bit_mask;
unsigned int driver_version;
int domain_bdf;
};
```

6.6 bm_profile_t

```
typedef struct bm_profile {
unsigned long cdma_in_time;
unsigned long cdma_in_counter;
unsigned long cdma_out_time;
unsigned long cdma_out_counter;
unsigned long tpu_process_time;
unsigned long sent_api_counter;
unsigned long completed_api_counter;
} bm_profile_t;
```


6.7 bm_heap_stat

```
struct bm_heap_stat {
unsigned int mem_total;
unsigned int mem_avail;
unsigned int mem_used;
}
```

6.8 bm_dev_stat_t

```
typedef struct bm_dev_stat {
int mem_total;
int mem_used;
int tpu_util;
int heap_num;
struct bm_heap_stat heap_stat[4];
} bm_dev_stat_t;
```

6.9 bm_log_level

```
#define BMLIB_LOG_QUIET -8
#define BMLIB_LOG_PANIC 0
#define BMLIB_LOG_FATAL 8
#define BMLIB_LOG_ERROR 16
#define BMLIB_LOG_WARNING 24
#define BMLIB_LOG_INFO 32
#define BMLIB_LOG_VERBOSE 40
#define BMLIB_LOG_DEBUG 48
#define BMLIB_LOG_TRACE 56
```