BMLIB 开发参考手册

发行版本 0.4.13

SOPHGO

2025 年 10 月 27 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
+86-10-57590723 +86-10-57590724

CHAPTER 2

Release note

版本	发布日期	说明
V0.1.0	2022.07.12	第一次发布，包含 bmlib, bm-smi and tpu runtime。
V0.2.0	2022.07.30	增加 bmvid；补充文档。
V0.3.0	2022.08.30	增加 soc mode 支持; 增加 bmcv 支持; 支持 bm1684
V0.4.0	2022.09.15	完善 bm1684 支持，增加 soc mode 交叉编译指南; 支持 SC7 加速卡
V0.4.1	2022.09.21	完善 bm1684 soc mode 支持; fix some opencv bug
V0.4.2	2022.10.15	支持 arm pcie mode
V0.4.3	2022.11.15	支持 sc7 hp75 加速卡；支持 rpm 包安装
V0.4.4	2022.12.15	fix bug
V0.4.5	2023.2.7	支持动态算子加载
V0.4.6	2023.3.13	支持 mix mode
V0.4.7	2023.4.13	fix bug
V0.4.8	2023.5.16	add 64 bit dev mem manager
V0.4.9	2023.8.1	add virtual ethernet driver
V1.0.0	2023.9.11	add bm1688

CHAPTER 3

快速开始

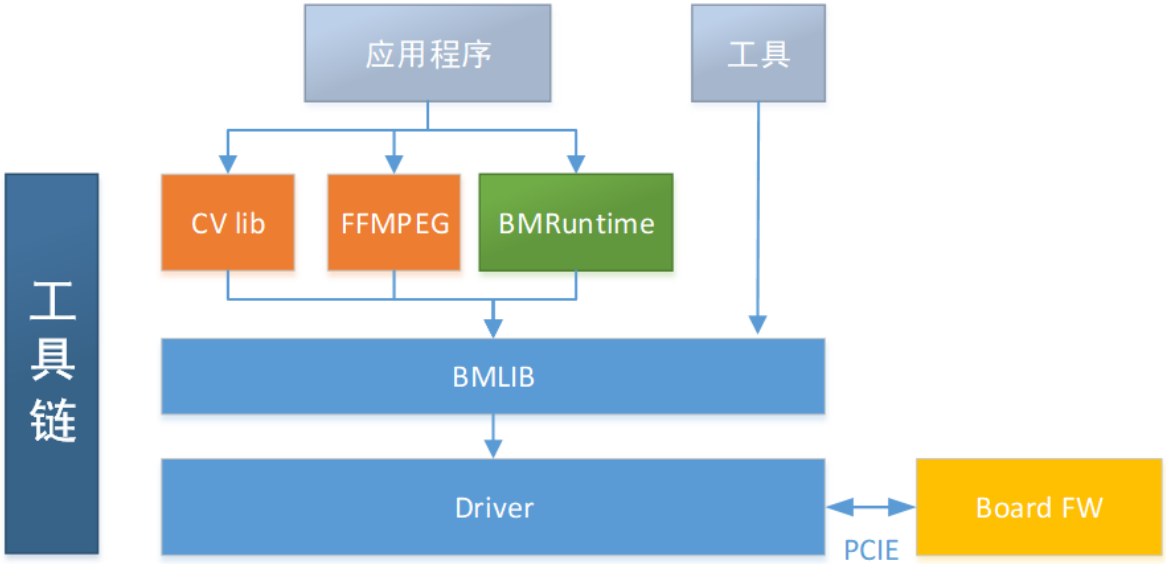
3.1 术语解释

术语	说明
BM1684	算能面向深度学习领域推出的第三代张量处理器
BM1684X	算能面向深度学习领域推出的第四代张量处理器
bm1688	算能面向深度学习领域推出的第五代张量处理器
TPU	BM1684 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。
PCIE Mode	一种产品形态, SDK 运行于 host 平台（可以是 X86/AARCH64 服务器），BM1684 作为 PCIE 接口的深度学习计算加速卡存在
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

- API 的发送和同步
- Global memory 在 host 端的映射和一致性管理
- profile 接口
- A53 的使能和使用
- 杂项管理接口
- Power 控制接口

4.1 Handle 的概念

我们的神经网络加速设备，无论是在 PCIE 模式，还是 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。

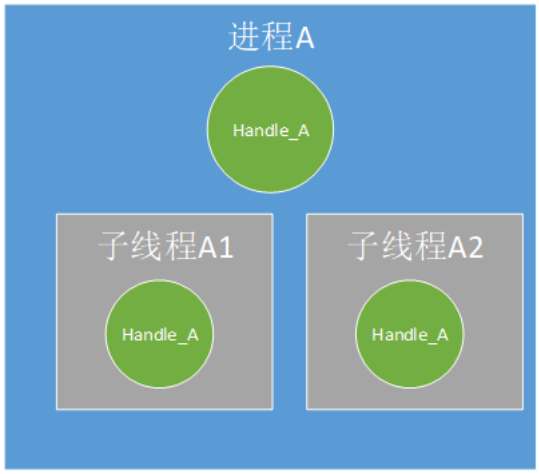
如果一个 api 是通过 handle_A 发送的，则必须通过 handle_A sync；

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

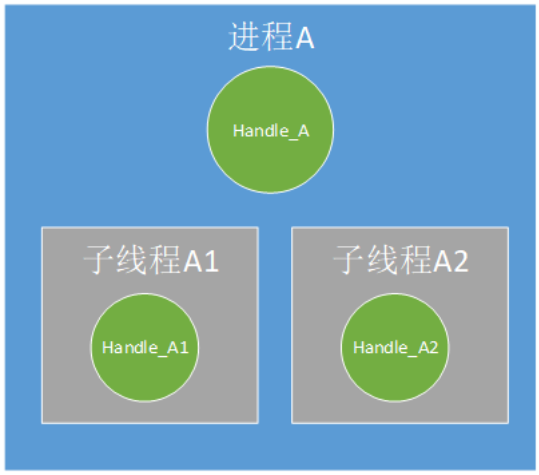
需要注意的是 handle 的创建者和使用者可以不一样，例如进程 A 创建了 handle_A，A 的子线程 A1 也可以使用 handle_A，但是 A1 通过 handle_A 申请的 memory 在统计上算作 A 申请的。

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；



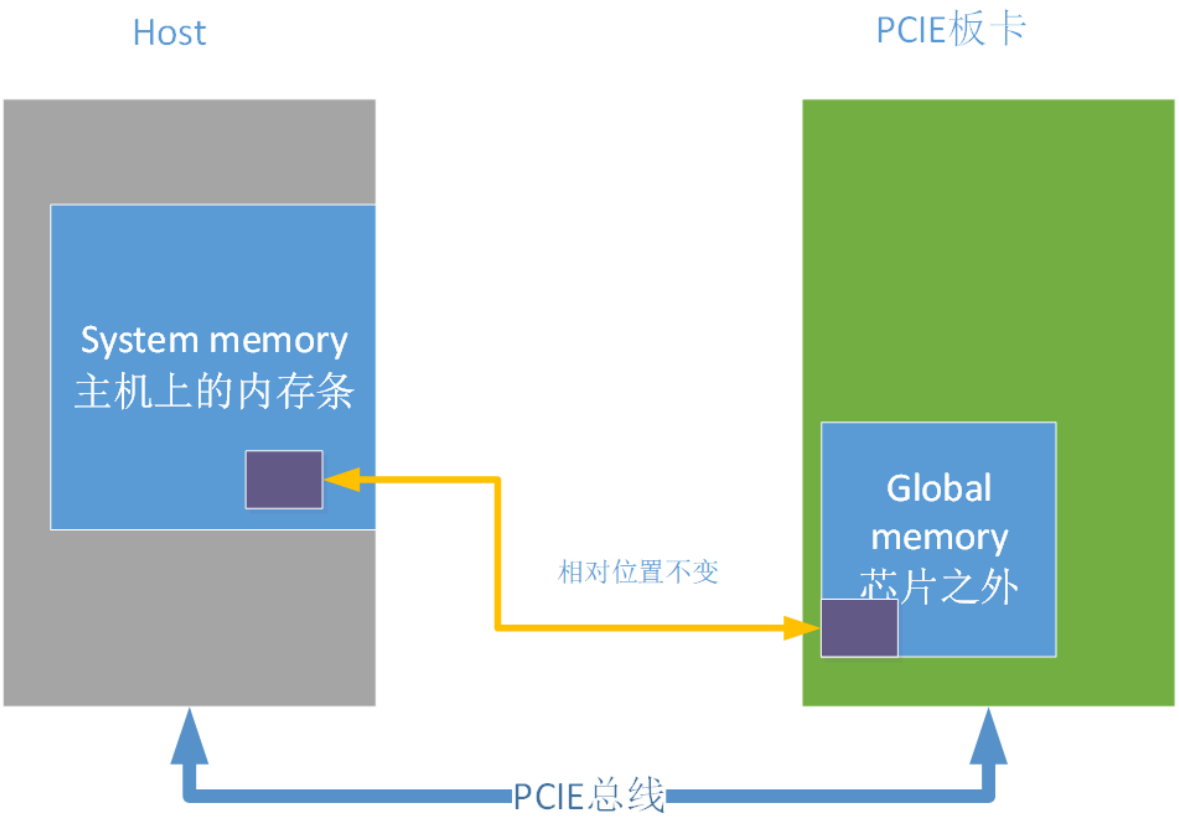
进程 A 中创建 Handle_A，Handle_A 在进程 A 的两个子线程（可以是多个，图中两个只是举例示意）中使用；



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 Memory 的种类



上图以 PCIE 模式介绍 memory 的种类，其中 host 可以是 PC 机/服务器，PCIE 板卡就是 SC5 系列板卡。Host 端的 memory 我们称之 system memory，PCIE 板卡上的 memory 我们称之为 global memory，或者 device memory。BM1684 芯片中有专门的 DMA 硬件单元在 system memory 和 global memory 之间搬运数据。

4.3 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，所以 host 这端的软件还需要调用一个 sync 函数类等到 api 的真正完成。

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。

4.4 Profile 接口

Profile 接口用于获取 tpu 处理 api 花费的时间，这个时间是从 tpu 开始工作后一直累加的（如果有不断的 api 得到处理），如果系统中只有一个进程使用 tpu 设备，我们可以通过计算调用 api 前后 profile 数据的差值来得到 api 的处理时间。

4.5 A53 的使能

在 PCIE 模式下，我们提供了一些接口用来启动 BM1684 中 A53 core，并让他们完成一些加速任务。

4.6 Power 控制

我们提供了接口用于获取和设置 tpu 的工作频率，用户可以自己定义一些自己的功耗控制策略。

4.7 杂项信息接口

用于获取板卡的信息和运行过程中的统计信息。目前包括 memory 总量和使用量，tpu 的利用率

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_type_u64

函数原型：bm_mem_type_t bm_mem_get_type_u64(struct bm_mem_desc_u64 mem);

函数作用：获取一块 u64 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。

5.2.3 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

5.2.4 bm_mem_get_device_addr_u64

函数原型：unsigned long long bm_mem_get_device_addr_u64(struct **bm_mem_desc_** u64 mem);

函数作用：获取 device 类型的 u64 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

5.2.5 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

5.2.6 bm_mem_set_device_addr_u64

函数原型：void bm_mem_set_device_addr_u64(struct bm_mem_desc_u64* pmem, unsigned long long addr);

函数作用：设置一个 device 类型 u64 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

5.2.7 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

5.2.8 bm_mem_get_device_size_u64

函数原型：unsigned long long bm_mem_get_device_size_u64(struct bm_mem_ desc_u64 mem);

函数作用：获取一块 device 类型的 u64 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

5.2.9 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

5.2.10 bm_mem_set_device_size_u64

函数原型：void bm_mem_set_device_size_u64(struct bm_mem_desc_u64* pmem, unsigned long long 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

5.2.11 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

5.2.12 bm_set_device_mem_u64

函数原型：void bm_set_device_mem_u64(bm_device_mem_u64_t* pmem, unsigned long long 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

5.2.13 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

5.2.14 bm_mem_from_device_u64

函数原型：bm_device_mem_u64_t bm_mem_from_device_u64(unsigned long long device_addr, unsigned long long 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

5.2.15 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

5.2.16 bm_mem_get_system_addr_u64

函数原型：void *bm_mem_get_system_addr_u64(struct bm_mem_desc_u64 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

5.2.17 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

5.2.18 bm_mem_set_system_addr_u64

函数原型：void bm_mem_set_system_addr_u64(struct bm_mem_desc_u64* 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

5.2.19 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

5.2.20 bm_mem_from_system_u64

函数原型：bm_system_mem_u64_t bm_mem_from_system_u64(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_neuron_device_u64

函数原型：bm_malloc_neuron_device_u64(bm_handle_t handle, bm_device_mem_u64_t *pmem, unsigned long long n, unsigned long long c, unsigned long long h, unsigned long long 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

5.3.4 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

5.3.5 bm_malloc_device_dword_u64

函数原型：bm_status_t bm_malloc_device_dword_u64(bm_handle_t handle, bm_device_mem_u64_t *pmem, unsigned long long 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

5.3.6 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

5.3.7 bm_malloc_device_byte_u64

函数原型：bm_status_t bm_malloc_device_byte_u64(bm_handle_t handle, bm_device_mem_u64_t *pmem, unsigned long long 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

5.3.8 bm_malloc_device_byte_heap

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

函数作用：在指定的 HEAP 上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
heap_id	输入	所指定分配 GMEM 的 HEAP（0/1/2）
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

5.3.9 bm_malloc_device_byte_heap_u64

函数原型：bm_malloc_device_byte_heap_u64(bm_handle_t handle, bm_device_mem_u64_t *pmem, int heap_id, unsigned long long size);

函数作用：在指定的 HEAP 上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
heap_id	输入	所指定分配 GMEM 的 HEAP（0/1/2）
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

5.3.10 bm_malloc_device_byte_heap_mask

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

函数作用：在指定的一个或多个 HEAP 上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
heap_id_mask	输入	指定分配 GMEM 的 HEAP id 的 mask，每个 bit 代表一个 HEAP，设置为 1 代表可以从这个 HEAP 分配，为 0 代表不能从这个 HEAP 分配，最低位 bit0 代表 heap0，依次增加
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

5.3.11 bm_malloc_device_byte_heap_mask_u64

函数原型：bm_status_t bm_malloc_device_byte_heap_mask_u64(bm_handle_t handle, bm_device_mem_u64_t *pmem, int heap_id_mask, unsigned long long size);

函数作用：在指定的一个或多个 HEAP 上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
heap_id_mask	输入	指定分配 GMEM 的 HEAP id 的 mask，每个 bit 代表一个 HEAP，设置为 1 代表可以从这个 HEAP 分配，为 0 代表不能从这个 HEAP 分配，最低位 bit0 代表 heap0，依次增加
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

5.3.12 bm_malloc_device_mem

函数原型：bm_status_t bm_malloc_device_mem(bm_handle_t handle, unsigned long long *paddr,
int heap_id, unsigned long long size);

函数作用：在指定的 HEAP 上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
paddr	输出	分配出 device memory 的地址
heap_id	输入	所指定分配 GMEM 的 HEAP（0/1/2）
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

5.3.13 bm_malloc_device_mem_mask

函数原型：bm_status_t bm_malloc_device_mem_mask(bm_handle_t handle,
unsigned long long *paddr, int heap_id_mask, unsigned long long size);

函数作用：在指定的 HEAP 上分配指定字节个数大小的 device 类型的 memory

参数介绍：

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
paddr	输出	分配出 device memory 的地址
heap_id_mask	输入	指定分配 GMEM 的 HEAP id 的 mask，每个 bit 代表一个 HEAP，设置为 1 代表可以从这个 HEAP 分配，为 0 代表不能从这个 HEAP 分配，最低位 bit0 代表 heap0，依次增加
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

5.3.14 bm_free_device_mem

函数原型：void bm_free_device_mem(bm_handle_t ctx, unsigned long long paddr);

函数作用：释放一块 device 类型的 memory

参数介绍：

5.3.15 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

5.3.16 bm_free_device_u64

函数原型：void bm_free_device_u64(bm_handle_t handle, bm_device_mem_u64_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

5.3.17 bm_gmem_arm_reserved_request

函数原型：unsigned long long bm_gmem_arm_reserved_request(bm_handle_t handle);

函数作用：获取为 arm926 保留的 gmem 的起始地址

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄

返回值：为 arm926 保留的 gmem 的起始地址（一个绝对地址）

5.3.18 bm_gmem_arm_reserved_release

函数原型：void bm_gmem_arm_reserved_release(bm_handle_t handle);

函数作用：释放为 arm926 保留的 gmem

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄

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_u64

函数原型：bm_status_t bm_memcpy_s2d_u64(bm_handle_t handle,
bm_device_mem_u64_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

5.4.3 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

5.4.4 bm_memcpy_s2d_partial_offset_u64

函数原型：bm_status_t bm_memcpy_s2d_partial_offset_u64(bm_handle_t handle, bm_device_mem_u64_t dst, void *src, unsigned long long size, unsigned long long 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

5.4.5 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

5.4.6 bm_memcpy_s2d_partial_u64

函数原型：bm_status_t bm_memcpy_s2d_partial_u64(bm_handle_t handle,
bm_device_mem_u64_t dst, void *src, unsigned long long 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：

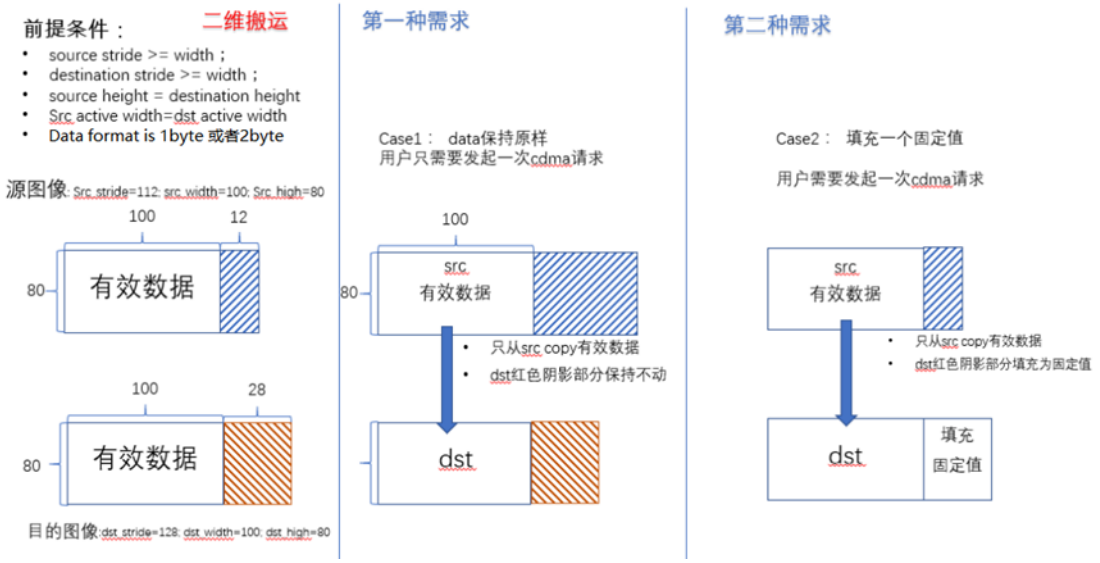
参数名	输入/输出	说明
handle	输入	设备句柄
dst	输入	目标 device memory 的结构体
src	输入	指向 system 内存的指针
size	输入	拷贝的长度

返回值：BM_SUCCESS 代表传输成功；其他错误码代表传输失败

5.4.7 bm_memcpy_s2d_stride

函数原型：bm_status_t bm_memcpy_s2d_stride(bm_handle_t handle,
bm_device_mem_t dst, void *src, struct stride_cfg *stride);

函数作用：拷贝 system 内存到 device 类型内存；效果是从 src 拷贝 stride->width*stride->height 长度的有效数据到 dst 起始地址这个位置上,shape 由 stride->src_width 变为 stride->dst_height，高度保持不变，可参考下图：



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
stride	输入	stride 配置结构体

返回值：BM_SUCCESS 代表传输成功；其他错误码代表传输失败

5.4.8 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

5.4.9 bm_memcpy_d2s_u64

函数原型：bm_status_t bm_memcpy_d2s_u64(bm_handle_t handle, void *dst, bm_device_mem_u64_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

5.4.10 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

5.4.11 bm_memcpy_d2s_partial_offset_u64

函数原型：bm_status_t bm_memcpy_d2s_partial_offset_u64(bm_handle_t handle, void *dst, bm_device_mem_u64_t src, unsigned long long size, unsigned long long 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

5.4.12 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

5.4.13 bm_memcpy_d2s_partial_u64

函数原型：bm_status_t bm_memcpy_d2s_partial_u64(bm_handle_t handle,
void *dst, bm_device_mem_u64_t src, unsigned long long 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：

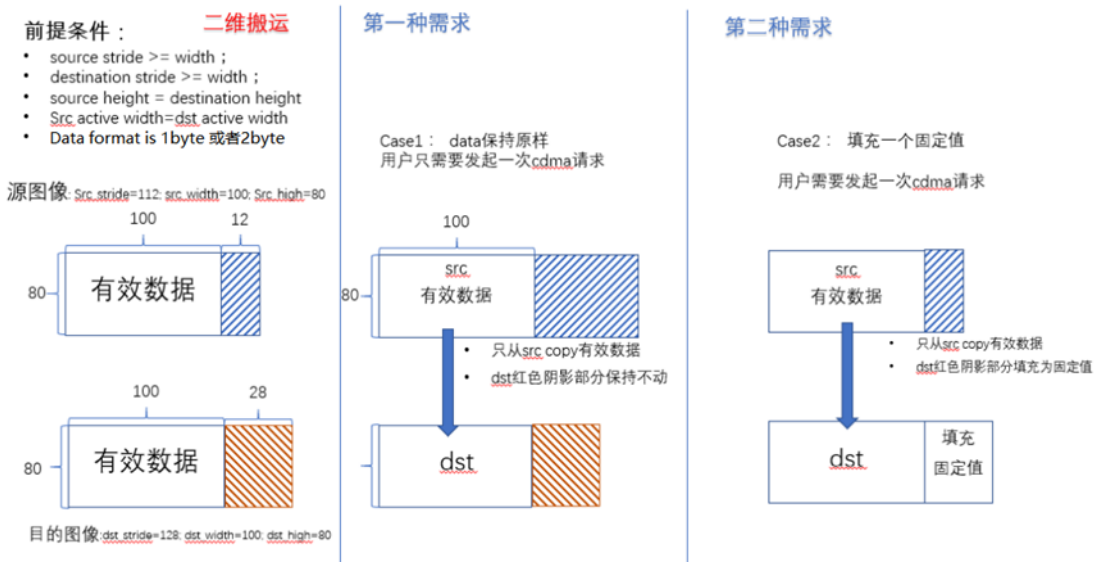
参数名	输入/输出	说明
handle	输入	设备句柄
dst	输入	指向 system 内存的指针
src	输入	源 device memory 的结构体
size	输入	拷贝的长度（单位为 byte）

返回值：BM_SUCCESS 代表传输成功；其他错误码代表传输失败

5.4.14 bm_memcpy_d2s_stride

函数原型：bm_status_t bm_memcpy_d2s_stride(bm_handle_t handle, void *dst,
bm_device_mem_t src, struct stride_cfg *stride);

函数作用：拷贝 system 内存到 device 类型内存；效果是从 src 拷贝 stride->width*stride->height 长度的有效数据到 dst 起始地址这个位置上,shape 由 stride->src_width 变为 stride->dst_height，高度保持不变，可参考下图：



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 system memory 的结构体
src	输入	指向 device 内存的指针
stride	输入	stride 配置结构体

返回值：BM_SUCCESS 代表传输成功；其他错误码代表传输失败

5.4.15 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

5.4.16 bm_mem_convert_system_to_device_neuron_u64

函数原型：bm_status_t
bm_mem_convert_system_to_device_neuron_u64(bm_handle_t handle,
struct bm_mem_desc_u64 *dev_mem, struct bm_mem_desc_u64 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

5.4.17 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

5.4.18 bm_mem_convert_system_to_device_neuron_byte_u64

函数原型：bm_status_t bm_mem_convert_system_to_device_neuron_byte_u64 (bm_handle_t handle, struct bm_mem_desc_u64 *dev_mem,

struct bm_mem_desc_u64 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

5.4.19 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

5.4.20 bm_mem_convert_system_to_device_coeff_u64

函数原型：bm_status_t bm_mem_convert_system_to_device_coeff_u64(bm_handle_t handle,
struct bm_mem_desc_u64 *dev_mem, struct bm_mem_desc_u64 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

5.4.21 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

5.4.22 bm_mem_convert_system_to_device_coeff_byte_u64

函数原型：bm_status_t bm_mem_convert_system_to_device_coeff_byte_u64 (bm_handle_t handle, struct bm_mem_desc_u64 *dev_mem,

struct bm_mem_desc_u64 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_memcpy_d2d_u64

函数原型：bm_memcpy_d2d_u64(bm_handle_t handle, bm_device_mem_u64_t dst, unsigned long long dst_offset, bm_device_mem_u64_t src, unsigned long long src_offset, unsigned long long 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

5.5.8 bm_memcpy_d2d_byte_u64

函数原型：bm_status_t bm_memcpy_d2d_byte_u64(bm_handle_t handle,
bm_device_mem_u64_t dst, unsigned long long dst_offset, bm_device_mem_u64_t src, unsigned long long src_offset, unsigned long long 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

5.5.9 bm_memcpy_d2d_stride_u64

函数原型：bm_status_t bm_memcpy_d2d_stride_u64(bm_handle_t handle,
bm_device_mem_u64_t dst, unsigned long long dst_stride, bm_device_mem_u64_t src, unsigned long long src_stride, unsigned long long 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

5.5.10 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。

5.5.11 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);

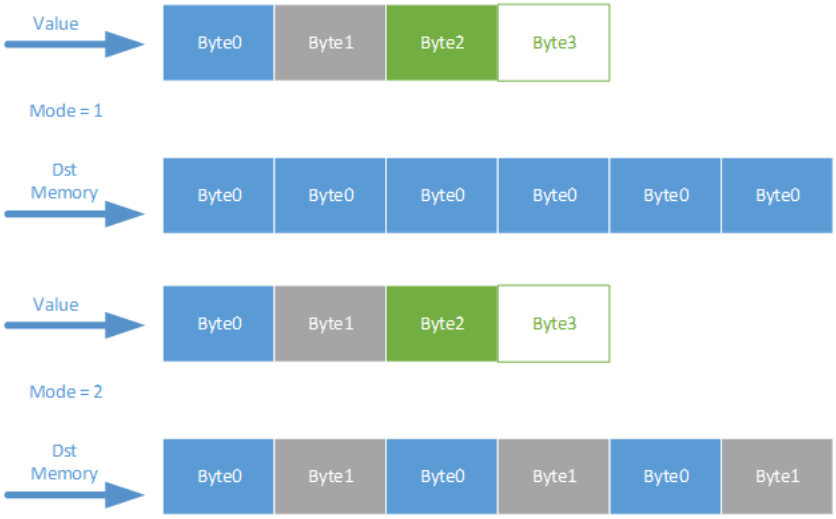
函数作用：用 value 指向的内容和指定的模式填充一块 device memory

参数介绍：

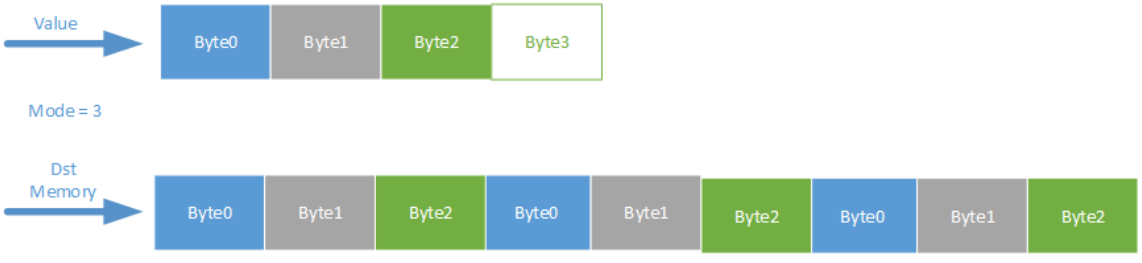
参数名	输入/输出	说明
handle	输入	设备句柄
value	输入	指向需要填充的值
mode	输入	填充模式，详见下图
mem	输入	目标 device memory 结构体

返回值：BM_SUCCESS 代表填充成功；其他错误码代表填充失败

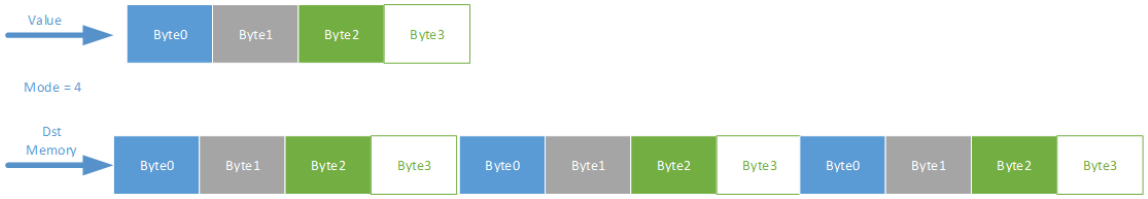
此函数的功能示意图如下：



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

5.5.12 bm_memset_device_ext_to_core

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

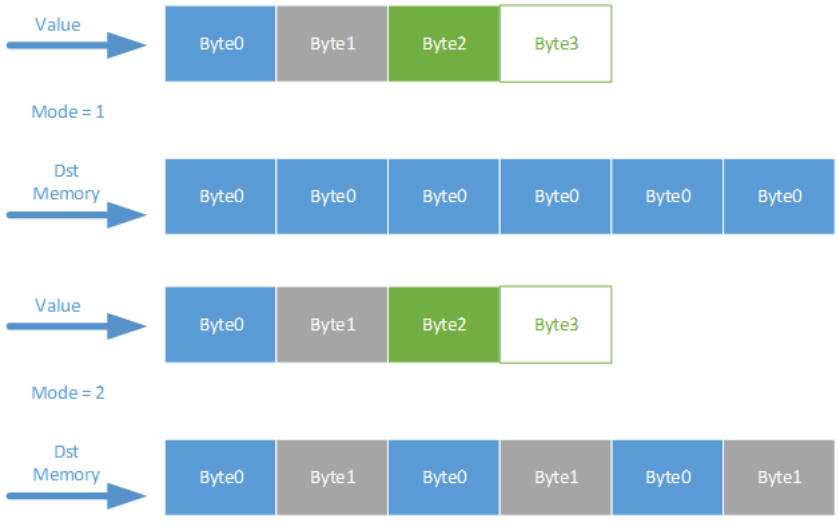
函数作用：用 value 指向的内容和指定的模式填充一块 device memory

参数介绍：

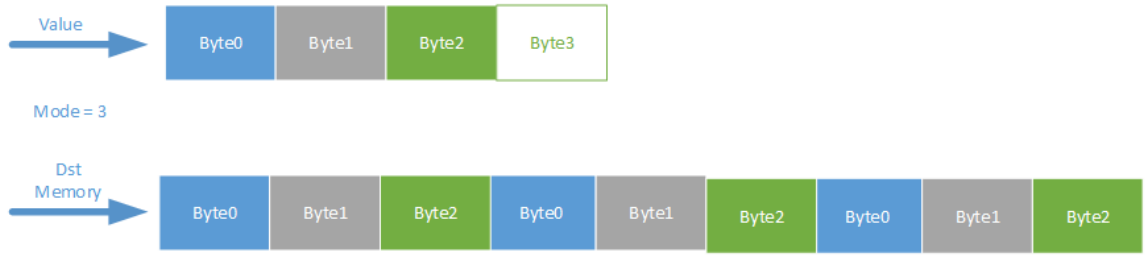
参数名	输入/输出	说明
handle	输入	设备句柄
value	输入	指向需要填充的值
mode	输入	填充模式，详见下图
mem	输入	目标 device memory 结构体
core_id	输入	搬运的 GDMA 设备 id

返回值：BM_SUCCESS 代表填充成功；其他错误码代表填充失败

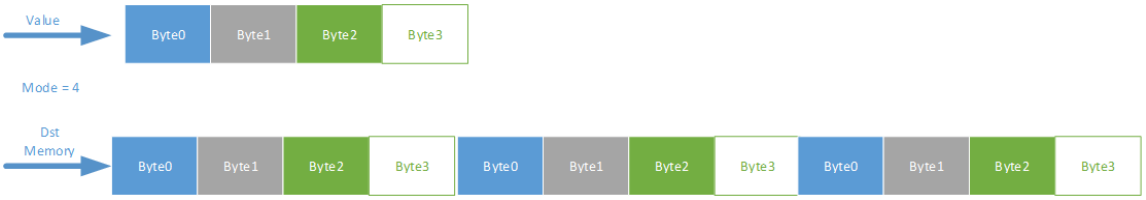
此函数的功能示意图如下：



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 在不同设备间搬运

5.6.1 bm_memcpy_c2c

函数原型：bm_status_t bm_memcpy_c2c(bm_handle_t src_handle, bm_handle_t dst_handle, bm_device_mem_t src, bm_device_mem_t dst, bool force_dst_cdma);

函数作用：将 global memory 从一块设备搬运到另一个设备（目前仅支持同一张卡上的设备）

参数介绍：

参数名	输入/输出	说明
src_handle	输入	源地址的设备句柄
dst_handle	输入	目的地址的设备句柄
src	输入	源目标 device memory 结构体
dst	输入	目的目标 device memory 结构体
force_dst_cdma	输入	强制使用目的 device 的 cdma 进行搬运，默认使用源 device 的 cdma 搬运

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.6.2 bm_memcpy_c2c_stride

函数原型：bm_status_t bm_memcpy_c2c_stride(bm_handle_t src_handle, bm_handle_t dst_handle, bm_device_mem_t src, bm_device_mem_t dst, struct stride_cfg *stride, bool force_use_dst_cdma);

函数作用：将 global memory 从一块设备搬运到另一个设备（目前仅支持同一张卡上的设备）

参数介绍：

参数名	输 入/输出	说明
src_handle	输入	源地址的设备句柄
dst_handle	输入	目的地址的设备句柄
src	输入	源目标 device memory 结构体
dst	输入	目的目标 device memory 结构体
stride	输入	stride 配置结构体
fo rce_dst_cdma	输入	强制使用目的 device 的 cdma 进行搬运，默认使用源 device 的 cdma 搬运

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.7 Global memory 在 host 端的映射和一致性管理

5.7.1 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，并开启 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

5.7.2 bm_mem_mmap_device_mem_u64

函数原型：bm_status_t bm_mem_mmap_device_mem_u64(bm_handle_t handle,
bm_device_mem_u64_t *dmem, unsigned long long *vmem);

函数作用：将一块 global memory 映射到 host 的 user 空间，并开启 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

5.7.3 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

5.7.4 bm_mem_mmap_device_mem_no_cache_u64

函数原型：bm_status_t bm_mem_mmap_device_mem_no_cache_u64(bm_handle_t handle , bm_device_mem_u64_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

5.7.5 bm_mem_invalidate_device_mem

函数原型：bm_status_t bm_mem_invalidate_device_mem(bm_handle_t handle, bm_device_mem_t *dmem);

函数作用：invalidate 一段被映射过的 device memory（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	执行被使无效的 global memory 的结构体指针

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.7.6 bm_mem_invalidate_device_mem_u64

函数原型：bm_status_t bm_mem_invalidate_device_mem_u64(bm_handle_t handle,
bm_device_mem_u64_t *dmem);

函数作用：invalidate 一段被映射过的 device memory（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	执行被使无效的 global memory 的结构体指针

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.7.7 bm_mem_invalidate_partial_device_mem

函数原型：bm_status_t bm_mem_invalidate_partial_device_mem(bm_handle_t handle, bm_device_mem_t *dmem, u32 offset, u32 len)

函数作用：invalidate 一段被映射过的 device memory 的一部分（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	执行被使无效的 global memory 的结构体指针
offset	输入	地址偏移量
len	输入	invalidate 的长度

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.7.8 bm_mem_invalidate_partial_device_mem_u64

函数原型：bm_status_t sg_mem_invalidate_partial_device_mem(bm_handle_t handle, sg_device_mem_t *dmem, unsigned long long offset, unsigned long long len);

函数作用：invalidate 一段被映射过的 device memory 的一部分（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	执行被使无效的 global memory 的结构体指针
offset	输入	地址偏移量
len	输入	invalidate 的长度

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.7.9 bm_mem_flush_device_mem

函数原型：bm_status_t bm_mem_flush_device_mem(bm_handle_t handle, bm_device_mem_t *dmem);

函数作用：flush 一段被映射过的 device global memory（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 flush 的 global memory 的结构体

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.7.10 bm_mem_flush_device_mem_u64

函数原型：bm_status_t bm_mem_flush_device_mem_u64(bm_handle_t handle, bm_device_mem_u64_t *dmem);

函数作用：flush 一段被映射过的 device global memory（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 flush 的 global memory 的结构体

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.7.11 bm_mem_flush_partial_device_mem

函数原型：bm_status_t bm_mem_flush_partial_device_mem(bm_handle_t handle,

bm_device_mem_t *dmem, u32 offset, u32 len)

函数作用：flush 一段被映射过的 device global memory 的一部分（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 flush 的 global memory 的结构体
offset	输入	地址偏移量
len	输入	flush 的长度

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.7.12 bm_mem_flush_partial_device_mem_u64

函数原型：bm_status_t bm_mem_flush_partial_device_mem_u64(bm_handle_t handle,
bm_device_mem_u64_t *dmem, unsigned long long offset, unsigned long long len);

函数作用：flush 一段被映射过的 device global memory 的一部分（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 flush 的 global memory 的结构体
offset	输入	地址偏移量
len	输入	flush 的长度

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.7.13 bm_mem_unmap_device_mem

函数原型：bm_status_t bm_mem_unmap_device_mem(bm_handle_t handle, void *vmem, int size);

函数作用：SOC 模式下，解除 device 内存的映射。（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	unmap 的虚拟地址
size	输入	unmap 的大小

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.7.14 bm_mem_unmap_device_mem_u64

函数原型：bm_status_t bm_mem_unmap_device_mem_u64(bm_handle_t handle, void *vmem , unsigned long long size);

函数作用：SOC 模式下，解除 device 内存的映射。（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	unmap 的虚拟地址
size	输入	unmap 的大小

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.7.15 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

5.8 API 的同步

5.8.1 bm_flush

函数原型：void bm_flush(bm_handle_t handle);

函数作用：此函数的功能等同于 bm_handle_sync，此函数是为了保持对老的代码兼容存在的，不建议再继续使用。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄

返回值：无返回值

5.8.2 bm_device_sync

函数原型：bm_status_t bm_device_sync(bm_handle_t handle);

函数作用：这个函数的含义是：创建 handle 的进程调 A 用这个函数时，在 handle 指向的设备上已经有了 N 个 api 在处理，函数返回后，这 N 个 api 都完成了。

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 代表同步成功；其他错误码代表同步失败

5.8.3 bm_thread_sync

函数原型：bm_status_t bm_thread_sync(bm_handle_t handle);

函数作用：这个函数的确切含义是：等待本 caller thread 在 handle 上之前提交过的所有 api 完成，如果本 caller thread 没有在此 handle 上提交过 api，则直接返回成功；本函数返回不能保证本 caller thread 在其他 handle 上提交过的 api 已经完成。

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 代表同步成功；其他错误码代表同步失败

5.8.4 bm_thread_sync_from_core

函数原型：bm_status_t bm_thread_sync_from_core(bm_handle_t handle, int core_id);

函数作用：这个函数的确切含义是：等待本 caller thread 在 handle 上第 core_id 个核上之前提交过的所有 api 完成，如果本 caller thread 没有在此 handle 上第 core_id 个核上提交过 api，则直接返回成功；本函数返回不能保证本 caller thread 在其他 handle 上第 core_id 个核上提交过的 api 已经完成。

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 代表同步成功；其他错误码代表同步失败

5.8.5 bm_handle_sync

函数原型：bm_status_t bm_handle_sync(bm_handle_t handle);

函数作用：同步提交到当前 handle 上所有的 API 操作，这个函数的含义是：调用这个函数时，通过此 handle 发送的 API 有 N 个，函数返回后，这 N 个 api 都完成了。

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 代表同步成功；其他错误码代表同步失败

5.8.6 bm_handle_sync_from_core

函数原型：bm_status_t bm_handle_sync_from_core(bm_handle_t handle, int core_id);

函数作用：同步提交到当前 handle 上第 core_id 个核上的所有 API 操作，这个函数的含义是：调用这个函数时，通过此 handle 向第 core_id 个核发送的 API 有 N 个，函数返回后，这 N 个 api 都完成了。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
core_id	输入	要同步的核 id

返回值：BM_SUCCESS 代表同步成功；其他错误码代表同步失败

5.8.7 bm_set_sync_timeout

函数原型：bm_status_t bm_set_sync_timeout(bm_handle_t handle, int timeout);

函数作用：设置最大等待 tpu 返回消息的时间，单位 ms

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
timeout	输入	超时时间

返回值：BM_SUCCESS 代表同步成功；其他错误码代表同步失败

5.9 profile 接口

5.9.1 bm_get_profile

函数原型：bm_status_t bm_get_profile(bm_handle_t handle, bm_profile_t *profile);

函数作用：获取当前时间点的 proflile 数据

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
profile	输出	指向一个存放 profling 数据的结构体

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.9.2 bm_get_last_api_process_time_us

函数原型：bm_status_t bm_get_last_api_process_time_us(bm_handle_t handle, unsigned long *time_us);

函数作用：此函数已经废弃

参数介绍：无

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.9.3 bm_enable_perf_monitor

函数原型：bm_status_t bm_enable_perf_monitor(bm_handle_t handle,
bm_perf_monitor_t *perf_monitor);

函数作用：使能 profile monitor

参数介绍：

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.9.4 bm_disable_perf_monitor

函数原型：bm_status_t bm_disable_perf_monitor(bm_handle_t handle,
bm_perf_monitor_t *perf_monitor);

函数作用：关闭 profile monitor

参数介绍：

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.10 power 管理接口

5.10.1 bm_set_clk_tpu_freq

函数原型：bm_status_t bm_set_clk_tpu_freq(bm_handle_t handle, int freq);

函数作用：设置当前 tpu 的工作频率，只在 PCIE 模式有效

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
freq	输入	tpu 的目标工作频率

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.10.2 bm_get_clk_tpu_freq

函数原型：bm_status_t bm_get_clk_tpu_freq(bm_handle_t handle, int *freq);

函数作用：获取当前 tpu 的工作频率

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
freq	输出	保存 tpu 当前工作频率的指针

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.11 设备管理接口

5.11.1 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

5.11.2 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

5.11.3 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

5.11.4 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

5.11.5 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

5.11.6 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

5.11.7 bm_get_gmem_heap_id

函数原型: bm_status_t bm_get_gmem_heap_id(bm_handle_t handle,

bm_device_mem_t *pmem,

unsigned int *heapid);

函数作用：获取 pmem 指向的设备内存的 heap id

参数介绍：

参数名	输入/输出	说明
Handle	输入	设备句柄
Pmem	输入	设备内存指针
Heaped	输出	存放设备内存所在 heap id 的指针

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.11.8 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

5.11.9 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

5.11.10 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

5.11.11 bm_set_debug_mode

函数原型：void bm_set_debug_mode(bm_handle_t handle, int mode);

函数作用：为 tpu fw log 设置 debug 模式备注：此函数 SC3 在使用

参数介绍：

参数名	输入/输出	说明
Handle	输入	设备句柄
Mode	输入	fw log debug 模式，0/1 表示 disable/enable

返回值：void

5.11.12 bmlib_set_api_dbg_callback

函数原型：void bmlib_set_api_dbg_callback(bmlib_api_dbg_callback callback);

函数作用：设置 debug callback 获取 fw log 备注：此函数 SC3 在使用

参数介绍：

参数名	输入/输出	说明
Handle	输入	设备句柄
Callback	输入	要设置的获取 fw log 回调函数的函数指针

返回值：void

5.11.13 bm_get_tpu_current

函数原型：bm_status_t bm_get_tpu_current(bm_handle_t handle, int *tpuc);

函数作用：获取句柄对应设备的电流值，默认单位毫安（mA）。

参数介绍：

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

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.11.14 bm_get_board_max_power

函数原型：bm_status_t bm_get_board_max_power(bm_handle_t handle, int *maxp);

函数作用：获取设备所在板卡支持的最大功耗值，默认单位瓦（W）。

参数介绍：

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

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.11.15 bm_get_board_power

函数原型：bm_status_t bm_get_board_power(bm_handle_t handle, int *boardp);

函数作用：获取设备所在板卡的当前功耗值，默认单位瓦（W）。

参数介绍：

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

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.11.16 bm_get_fan_speed

函数原型：bm_status_t bm_get_fan_speed(bm_handle_t handle, int *fan);

函数作用：获取设备所在板卡的风扇占空比。

参数介绍：

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

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.11.17 bm_get_ecc_correct_num

函数原型：bm_status_t bm_get_ecc_correct_num(bm_handle_t handle, unsigned long *ecc_correct_num);

函数作用：获取设备在 DDR 使能时，纠正错误的次数。

参数介绍：

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

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.11.18 bm_get_12v_atx

函数原型：bm_status_t bm_get_12v_atx(bm_handle_t handle, int *atx_12v);

函数作用：获取设备板级 12V 供电电流，默认单位毫安（mA）。

参数介绍：

参数名	输入/输出	说明
Handle	输入	设备句柄
atx_12v	输出	要获取 atx_12v 的函数指针

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.11.19 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

5.11.20 bm_get_status

函数原型：bm_status_t bm_get_status(bm_handle_t handle, int *status);

函数作用：获取句柄对应的设备状态，0 为活动状态，1 为故障状态。

参数介绍：

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

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.11.21 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

5.11.22 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

5.11.23 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

5.11.24 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

5.11.25 bm_get_board_temp

函数原型：bm_status_t bm_get_board_temp(bm_handle_t handle, unsigned int *board_temp);

函数作用：获取句柄对应设备所在板卡的板级温度，默认单位摄氏度（℃）。

参数介绍：

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

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.11.26 bm_get_chip_temp

函数原型：bm_status_t bm_get_chip_temp(bm_handle_t handle, unsigned int *chip_temp);

函数作用：获取句柄对应设备的温度，默认单位摄氏度（℃）。

参数介绍：

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.11.27 bm_get_tpu_power

函数原型：bm_status_t bm_get_tpu_power(bm_handle_t handle, float *tpu_power);

函数作用：获取句柄对应设备的功耗，默认单位瓦（W）。

参数介绍：

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.11.28 bm_get_tpu_volt

函数原型：bm_status_t bm_get_tpu_volt(bm_handle_t handle, float *tpu_volt);

函数作用：获取句柄对应设备的电压，默认单位毫伏（mV）。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
tpu_volt	输出	存储 tpu_volt 的指针

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.11.29 bm_get_dynfreq_status

函数原型：bm_status_t bm_get_dynfreq_status(bm_handle_t handle, int *dynfreq_status);

函数作用：获取句柄对应设备的动态频率状态，默认为开。

参数介绍：

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.11.30 bm_change_dynfreq_status

函数原型：bm_status_t bm_change_dynfreq_status(bm_handle_t handle, int new_status);

函数作用：设置动态频率状态，默认为开。

参数介绍：

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.11.31 bmdev_get_idle_coreid

函数原型：bm_status_t bmdev_get_idle_coreid(bm_handle_t handle, int *core_id);

函数作用：获取当前为负载最低的 tpu core id。

参数介绍：

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.11.32 bm_get_tpu_scalar_num

函数原型：bm_status_t bm_get_tpu_scalar_num(bm_handle_t handle, unsigned int *core_num);

函数作用：得到 tpu 核（tpu scalar）的数量

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
core_num	输出	tpu scalar 的数量

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.11.33 bm_pwr_ctrl

函数原型：bm_status_t bm_pwr_ctrl(bm_handle_t handle, void *bm_api_cfg_pwr_ctrl);

函数作用：设置 pwr ctrl。

参数介绍：

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.11.34 bm_get_boot_loader_version

函数原型：bm_status_t bm_get_boot_loader_version(bm_handle_t handle,
boot_loader_version *version);

函数作用：获取 boot loader 版本信息。

参数介绍：

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.12 A53 使能

5.12.1 bmcpu_start_cpu

函数原型：bm_status_t bmcpu_start_cpu(bm_handle_t handle, char *boot_file, char *core_file);

函数作用：启动设备上的 ARM 处理器 A53。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
boot_file	输入	ARM 处理器启动的 boot 文件
core_file	输入	ARM 处理器启动的 kernel 文件

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.12.2 bmcpu_open_process

函数原型：int bmcpu_open_process(bm_handle_t handle, unsigned int flags, int timeout);

函数作用：创建运行在 A53 上的进程。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
flags	输入	创建 a53 进程的标志位
timeout	输入	创建 a53 进程的超时时间

返回值：A53 上进程句柄

5.12.3 bmcpu_load_library

函数原型：bm_status_t bmcpu_load_library(bm_handle_t handle, int process_handle, char *library_file, int timeout);

函数作用：加载 A53 上进程所需要的动态库。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
process_handle	输入	A53 上进程句柄
library_file	输入	需要加载的动态库文件
timeout	输入	加载动态库的超时时间

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.12.4 bmcpu_unload_library

函数原型：bm_status_t bmcpu_unload_library(bm_handle_t handle, int process_handle, char *library_file, int timeout);

函数作用：卸载 A53 上进程所需要的动态库。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
process_handle	输入	A53 上进程句柄
library_file	输入	需要卸载的动态库文件
timeout	输入	卸载动态库的超时时间

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.12.5 bmcpu_exec_function

函数原型：int bmcpu_exec_function(bm_handle_t handle, int process_handle, char *function_name, void *function_param, unsigned int param_size, int timeout);

函数作用：在 A53 进程执行指定函数。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
process_handle	输入	A53 上进程句柄
function_name	输入	需要执行的函数名称
function_param	输入	需要执行的函数入参地址
param_size	输入	需要执行的函数入参大小
timeout	输入	A53 执行函数的超时时间

返回值：0 代表成功；大于 0 代表 bmlib 失败，小于 0 代表 function 执行失败

5.12.6 bmcpu_exec_function_ext

函数原型：int bmcpu_exec_function_ext(bm_handle_t handle, int process_handle, char *function_name, void *function_param, unsigned int param_size, unsigned int opt, int timeout);

函数作用：在 A53 进程执行指定函数，设置是否刷新 cache。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
process_handle	输入	A53 上进程句柄
function_name	输入	需要执行的函数名称
function_param	输入	需要执行的函数入参地址
param_size	输入	需要执行的函数入参大小
opt	输入	是否需要刷新 cache
timeout	输入	A53 执行函数的超时时间

返回值：0 代表成功；大于 0 代表 bmlib 失败，小于 0 代表 function 执行失败

5.12.7 bmcpu_exec_function_async

函数原型：bm_status_t bmcpu_exec_function_async(bm_handle_t handle, int process_handle, char *function_name, void *function_param, unsigned int param_size, unsigned long long *api_handle);

函数作用：在 A53 进程执行指定函数。

参数介绍：

返回值：0 代表成功；大于 0 代表 bmlib 失败，小于 0 代表 function 执行失败

5.12.8 bmcpu_exec_function_async_ext

函数原型：bm_status_t bmcpu_exec_function_async_ext(bm_handle_t handle, int process_handle, char *function_name, void *function_param, unsigned int param_size, unsigned int opt, unsigned long long *api_handle);

函数作用：在 A53 进程执行指定函数，设置是否刷新 cache。

参数介绍：

返回值：0 代表成功；大于 0 代表 bmlib 失败，小于 0 代表 function 执行失败

5.12.9 bmcpu_query_exec_function_result

函数原型：int bmcpu_query_exec_function_result(bm_handle_t handle, unsigned long long api_handle, int timeout);

函数作用：查询 A53 进程执行函数的返回结果。

参数介绍：

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.12.10 bmcpu_map_phys_addr

函数原型：void *bmcpu_map_phys_addr(bm_handle_t handle, int process_handle, void *phys_addr, unsigned int size, int timeout);

函数作用：将设备物理地址映射成 A53 能访问的虚拟地址。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
process_handle	输入	A53 上进程句柄
phys_addr	输入	host 侧申请的设备内存对应的虚拟地址
size	输入	申请的内存大小
timeout	输入	A53 映射地址的超时时间

返回值：设备物理地址映射成的 A53 能访问的虚拟地址

5.12.11 bmcpu_unmap_phys_addr

函数原型：bm_status_t bmcpu_unmap_phys_addr(bm_handle_t handle, int process_handle, void *phys_addr, int timeout);

函数作用：释放被 A53 映射的物理地址。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
process_handle	输入	A53 上进程句柄
phys_addr	输入	host 侧申请的设备内存对应的 A53 虚拟地址
timeout	输入	A53 映射地址的超时时间

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.12.12 bmcpu_close_process

函数原型：int bmcpu_close_process(bm_handle_t handle, int process_handle, int timeout);

函数作用：关闭运行在 A53 上的进程。

参数介绍：

参数名	输入/输出	说明
handle	输入	设备句柄
process_handle	输入	A53 上进程句柄
timeout	输入	关闭 a53 进程的超时时间

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.12.13 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

5.12.14 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

5.12.15 bmcpu_set_log

函数原型：bm_status_t bmcpu_set_log(bm_handle_t handle, unsigned int log_level, unsigned int log_to_console, int timeout);

函数作用：设置 cpu log 参数。

参数介绍：

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.12.16 bmcpu_get_log

函数原型：bm_status_t bmcpu_get_log(bm_handle_t handle, int process_handle, char *log_file, int timeout);

函数作用：获取 cpu log 文件。

参数介绍：

返回值：BM_SUCCESS 代表成功；其他错误码代表失败

5.12.17 bmcpu_sync_time

函数原型：bm_status_t bmcpu_sync_time(bm_handle_t handle);

函数作用：同步 device 和 host 时间

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

5.12.18 bm_load_firmware

函数原型：bm_status_t bm_load_firmware(bm_handle_t handle,
const char *firmware_tcm, const char *firmware_ddr);

函数作用：加载 firmware 固件

参数介绍：

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
```

