Atlas 300I 推理卡
1.0.11 以下
DCMI API 参考（型号 3000, 3010）

文档版本	13
发布日期	2022-06-01

华为技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司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，华为公司对本文档内容不做任何明示或暗示的声明或保证。

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。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： https://e.huawei.com

前言

概述

本文档详细的描述了Atlas 300I 推理卡（型号 3000）和Atlas 300I 推理卡（型号 3010）DCMI（DaVinci Card Management Interface）接口，用户可使用这些接口进行设备管理、配置管理、软件升级、芯片复位启动等操作。

本文档适用于Atlas 300I 推理卡（型号3000）和Atlas 300I 推理卡（型号3010）。如两个型号内容一致，则不作区分，产品名称统称为Atlas 300I 推理卡。

读者对象

本文档主要适用于以下工程师：

- 企业管理员
- 企业终端用户

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
危险	表示如不避免则将会导致死亡或严重伤害的具有高等级风险的危害。
警告	表示如不避免则可能导致死亡或严重伤害的具有中等级风险的危害。
注意	表示如不避免则可能导致轻微或中度伤害的具有低等级风险的危害。
须知	用于传递设备或环境安全警示信息。如不避免则可能会导致设备损坏、数据丢失、设备性能降低或其它不可预知的结果。 “须知”不涉及人身伤害。

符号	说明
说明	对正文中重点信息的补充说明。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。

修改记录

文档版本	发布日期	修改说明
13	2022-06-01	第十三次正式发布。 修改**1 简介**章节。
12	2021-07-31	第十二次正式发布。
11	2021-04-29	第十一次正式发布。
10	2021-04-14	第十次正式发布。 **7 调用示例**增加NPU 1.X.X系列版本的调用示例。
09	2021-01-25	第九次正式发布。 新增： **5.5 dcmi_reset_device_inband接口原型**。
08	2020-10-26	第八次正式发布。 新增： **2.22 dcmi_get_pcie_error_cnt接口原型**。 **3.4 dcmi_clear_pcie_error_cnt接口原型**。
07	2020-09-14	第七次正式发布。 • **4.4 dcmi_mcu_get_version接口原型**增加固件版本为1.73.5.5及以上的接口信息。 • **4.5 dcmi_get_version接口原型**增加固件版本为1.73.5.5及以上的接口信息。 • 修改产品名称为“Atlas 300I 推理卡”。
06	2020-07-27	第六次正式发布。

文档版本	发布日期	修改说明
05	2020-07-15	第五次正式发布。 新增： • **2.20 dcmi_mcu_get_board_info接口原型**。 • **2.21 dcmi_mcu_get_power_info接口原型**。
04	2020-03-30	第四次正式发布。 新增**6 证书查询配置接口**。
03	2019-09-30	第三次正式发布。 修改产品名称。原名“Atlas 300”改为“Atlas 300 AI加速卡”。
02	2019-07-10	第二次正式发布。
01	2019-04-30	第一次正式发布。

目 录

1 简介

DCMI接口包含设备管理接口、配置管理接口、MCU升级控制接口、芯片复位接口。用户可调用接口完成二次开发。不支持Windows操作系统。

须知

DSMI接口从1.0.10版本开始废弃，计划于2023年版本不再提供，请使用对应的DCMI接口进行替代。

2 设备管理接口

2.1 dcmi_init 接口原型

函数原型

int dcmi_init(void)

功能说明

初始化，完成服务器上所有Atlas 300I 推理卡的初始化扫描等操作。

须知

在调用其他接口之前先调用初始化接口。

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

无。

调用示例

```
...
int ret;
ret = dcmi_init();
...
```

2.2 dcmi_get_card_num_list 接口原型

函数原型

int dcmi_get_card_num_list(int *card_num, int *card_list, int list_length)

功能说明

查询系统所有Atlas 300I 推理卡的个数和ID。

参数说明

参数名	输入/输出	类型	描述
card_num	输出	int	Atlas 300I 推理卡个数，取值范围：1~16。
card_list	输出	int	Atlas 300I 推理卡的ID数组。
list_length	输入	int	输出参数card_list的数组长度。

返回值

类型	描述
int	处理结果： ● 返回0成功 ● 失败返回错误码，错误码小于0

异常处理

无。

约束说明

无。

调用示例

```
...
int ret;
int device_count = 0;
int card_id_list[16];
ret = dcmi_get_card_num_list(&device_count, card_id_list, 16);
...
```

2.3 dcmi_get_card_elabel 接口原型

函数原型

int dcmi_get_card_elabel (int card_id, struct dcmi_elabel_info_stru * pelabel_info)

功能说明

获取芯片的电子标签信息。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。

pelabel_info	输出	struct dcmi_elabel_info_stru *	#define MAX_LENTH 256
			struct dcmi_ elabel _info_stru
			{
			char product_name[MAX_LENTH]; //单板名称
			char model[MAX_LENTH]; //预留，目前未使用
			char manufacturer[MAX_LENTH]; //厂商
			char serial_number[MAX_LENTH]; //SN号
			};

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

无。

调用示例

```
...
DCMI_ELABEL_INFO_STRU elableInfo;
int ret = 0;
int card_id = 0;
memset(&elableInfo, 0, sizeof(elableInfo));
ret = dcmi_get_card_elabel(card_id, &elableInfo);
if (ret != 0)
{  //todo:记录日志 return ERROR;
}
...
```

2.4 dcmi_get_device_num_in_card 接口原型

函数原型

int dcmi_get_device_num_in_card(int card_id, int *device_num)

功能说明

列举编号为card_id的Atlas 300I 推理卡上的设备数量。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
device_num	输出	int	芯片个数。

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
int device_num = 0;
int card_id = 0x56;
ret = dcmi_get_device_num_in_card(card_id, &device_num);
if((ret != 0) || (0 == device_num))
{  //todo:记录日志 return ERROR;
}
...
```

2.5 dcmi_get_device_health 接口原型

函数原型

int dcmi_get_device_health(int card_id, int device_id, unsigned int *phealth)

功能说明

查询芯片的总体健康状态。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
device_id	输入	int	指定芯片编号，取值范围0~3，当前实际支持的范围通过**2.4 dcmi_get_device_num_in_card接口原型**获取。
phealth	输出	unsigned int *	芯片总体健康状态，只代表本部件，不包括与本部件存在逻辑关系的其它部件，内容定义为： ● 0：正常 ● 1：一般告警 ● 2：重要告警 ● 3：紧急告警 ● 0xFFFFFFFF：该设备不存在

返回值

类型	描述
int	处理结果： ● 返回0成功 ● 失败返回错误码，错误码小于0

异常处理

无。

约束说明

如果有多个告警，以最严重的告警作为设备的告警。

调用示例

```
...
int ret = 0;
int card_id = 0x56;
int device_id = 1;
unsigned int health;
ret = dcmi_get_device_health(card_id, device_id, &health);
...
```

2.6 dcmi_get_device_errorcode 接口原型

函数原型

int dcmi_get_device_errorcode(int card_id, int device_id, int* errorcount, unsigned int *perrorcode, int *errorwidth)

功能说明

查询设备故障码。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
device_id	输入	int	指定芯片编号，取值范围0~3，当前实际支持的范围通过**2.4 dcmi_get_device_num_in_card接口原型**获取。
errorcount	输出	int *	错误码数量，取值范围：0~128。
perrorcode	输出	unsigned int *	错误码。
errorwidth	输出	int *	每个错误码占用的字节空间。

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

无。

调用示例

```
#define ERROR_CODE_MAX_NUM              (128)
...
int ret = 0;
int card_id = 0x56;
int device_id = 0;
int errorcount = 0;
unsigned int perrorcode[ERROR_CODE_MAX_NUM] = {0};
int error_code_width;
ret = dcmi_get_device_errorcode(card_id, device_id, &errorcount, perrorcode, &error_code_width);
if(ret != 0) {
//todo:记录日志
return ret;
}
...
```

2.7 dcmi_get_device_errorinfo 接口原型

函数原型

int dcmi_get_device_errorinfo(int card_id, int device_id, int errorcode, unsigned char *perrorinfo, int buffsize)

功能说明

查询设备故障描述。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。

参数名	输入/输出	类型	描述
device_id	输入	int	指定芯片编号，取值范围0~3，当前实际支持的范围通过**2.4 dcmi_get_device_num_in_card接口原型**获取。
errorcode	输入	int	要查询的错误码。
perrorinfo	输出	unsigned char *	对应的错误字符描述。
buffsize	输入	int	带入的buff大小，固定为48字节。

返回值

类型	描述
int	处理结果： ● 返回0成功 ● 失败返回错误码，错误码小于0

异常处理

无。

约束说明

暂不支持。

调用示例

```
#define ERROR_CODE_MAX_NUM  (128)
#define BUFF_SIZE           (48)
...
int ret = 0;
int card_id = 0x56;
int device_id = 0;
int errorcount = 0;
int error_code_width = 0;
unsigned char perrorinfo[BUFF_SIZE] = {0};
unsigned char perrorcode [ERROR_CODE_MAX_NUM] = {0};
ret = dcmi_get_device_errorcode(card_id, device_id, &errorcount, perrorcode, &error_code_width);
if((ret != 0) || (errorcount == 0)) {
//todo:记录日志
return ret;
}
ret = dcmi_get_device_errorinfo(card_id, device_id, perrorcode[0], perrorinfo, BUFF_SIZE);
if(ret != 0) {
//todo:记录日志
return ret;
}
...
```

2.8 dcmi_get_device_temperature 接口原型

函数原型

int dcmi_get_device_temperature (int card_id, int device_id, int *tmprature)

功能说明

查询芯片温度。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
device_id	输入	int	指定芯片编号，取值范围0~3，当前实际支持的范围通过**2.4 dcmi_get_device_num_in_card接口原型**获取。
tmprature	输出	int *	芯片温度：单位为摄氏度，精度为1摄氏度，有小数点则四舍五入。16位带符号类型，小字节序。设备侧返回的值就是实际温度。

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
```

```
int temp = 0;
int card_id = 0x56;
int device_id = 0;
ret = dcmi_get_device_temperature(card_id, device_id, &temp);
if(ret != 0) {
//todo：记录日志
return ret;
}
…
```

2.9 dcmi_get_device_power_info 接口原型

函数原型

int dcmi_get_device_power_info(int card_id, int device_id, int *power)

功能说明

查询芯片功耗。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
device_id	输入	int	指定芯片编号，取值范围0~3，当前实际支持的范围通过**2.4 dcmi_get_device_num_in_card接口原型**获取。
power	输出	int	设备功耗：单位为W，精度为0.1W。16位无符号short类型，小字节序。计算公式：value=reading*0.1。

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
int card_id = 0x56;
int device_id = 0;
int power = 0;
ret = dcmi_get_device_power_info(card_id, device_id, &power);
if(ret != 0) {
//todo：记录日志
return ret;
}
...
```

2.10 dcmi_get_pcie_info 接口原型

函数原型

int dcmi_get_pcie_info (int card_id, int device_id, struct dcmi_tag_pcie_idinfo *pcie_idinfo)

功能说明

查询芯片的PCIe（Peripheral Component Interconnect Express）信息。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
device_id	输入	int	指定芯片编号，取值范围0~3，当前实际支持的范围通过**2.4 dcmi_get_device_num_in_card接口原型**获取。
pcie_idinfo	输出	struct dcmi_tag_pcie_idinfo *	PCIe信息。 typedef struct dcmi_tag_pcie_idinfo{ unsigned int deviceid; unsigned int venderid; unsigned int subvenderid; unsigned int subdeviceid; unsigned int bdf_deviceid; unsigned int bdf_busid; unsigned int bdf_funcid; }DCMI_TAG_PCIE_IDINFO;

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
int card_id = 0x56;
int device_id = 0;
struct dcmi_tag_pcie_idinfo pcie_idinfo = {0};
ret = dcmi_get_pcie_info(card_id, device_id, &pcie_idinfo);
if(ret != 0) {
//todo：记录日志
return ret;
}
...
```

2.11 dcmi_get_device_voltage 接口原型

函数原型

int dcmi_get_device_voltage (int card_id, int device_id, unsigned int *pvoltage)

功能说明

查询芯片电压。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。

参数名	输入/输出	类型	描述
device_id	输入	int	指定芯片编号，取值范围0~3，当前实际支持的范围通过**2.4 dcmi_get_device_num_in_card接口原型**获取。
pvoltage	输出	unsigned int *	芯片电压：单位为V，精度为0.01V。16位无符号short类型，小字节序。 BMC（Baseboard Management Controller）侧计算公式：value=reading*0.01。

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
unsigned int  voltage;
int card_id = 0x56;
int device_id = 0;
ret = dcmi_get_device_voltage(card_id, device_id, &voltage);
if(ret != 0) {
//todo：记录日志
return ret;
}
…
```

2.12 dcmi_get_device_utilization_rate 接口原型

函数原型

int dcmi_get_device_utilization_rate(int card_id, int device_id,int subdevice_type, unsigned int *putilization_rate)

功能说明

获取芯片占用率。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
device_id	输入	int	指定芯片编号，取值范围0~3，当前实际支持的范围通过**2.4 dcmi_get_device_num_in_card接口原型**获取。
subdevice_type	输入	int	组件类型，目前支持如下几种： 1：内存 2：AI CORE 3：AI CPU 4：控制CPU 5：内存带宽
putilization_rate	输出	unsigned int *	昇腾AI处理器利用率，单位：%。

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
unsigned int putilization_rate;
int card_id = 0x56;
```

```
int device_id = 0;
ret = dcmi_get_device_utilization_rate(card_id, device_id, 1, &putilization_rate);
if(ret != 0) {
//todo：记录日志
return ret;
}
...
```

2.13 dcmi_get_device_frequency 接口原型

函数原型

int dcmi_get_device_frequency (int card_id, int device_id, int device_type, unsigned int *pfrequency)

功能说明

获取芯片的组件频率。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
device_id	输入	int	指定芯片编号，取值范围0~3，当前实际支持的范围通过**2.4 dcmi_get_device_num_in_card接口原型**获取。
device_type	输入	int	组件类型，目前支持如下几种： 1：内存 2：控制CPU
pfrequency	输出	unsigned int *	频率，单位MHz。

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

无。

调用示例

```
...
int ret = 0;
unsigned int frequency;
int card_id = 0x56;
int device_id = 0;
ret = dcmi_get_device_frequency(card_id, device_id, 2, &frequency);
if(ret != 0) {
//todo：记录日志
return ret;
}
...
```

2.14 dcmi_get_device_flash_count 接口原型

函数原型

int dcmi_get_device_flash_count(int card_id, int device_id, unsigned int *pflash_count)

功能说明

获取芯片中Flash个数。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
device_id	输入	int	指定芯片编号，取值范围0~3，当前实际支持的范围通过**2.4 dcmi_get_device_num_in_card接口原型**获取。
pflash_count	输出	unsigned int *	返回Flash个数，取值范围：目前固定为1。

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

无。

调用示例

```
...
int ret = 0;
int card_id = 0x56;
int device_id = 0;
unsigned int flash_count = 0;
ret = dcmi_get_device_flash_count(card_id, device_id, &flash_count);
if(ret != 0) {
//todo：记录日志
return ret;
}
...
```

2.15 dcmi_get_device_flash_info 接口原型

函数原型

int dcmi_get_device_flash_info (int card_id, int device_id, unsigned int flash_index, struct dcmi_flash_info_stru *pflash_info);

功能说明

获取芯片内Flash的信息。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。

参数名	输入/输出	类型	描述
device_id	输入	int	指定芯片编号，取值范围0~3，当前实际支持的范围通过2.4 dcmi_get_device_num_in_card接口原型获取。
flash_index	输入	unsigned int	Flash索引号。

pflash_info	输出	dcmi_flash_info_stru *	返回Flash信息。
			Flash信息结构体定义：
			struct dcmi_flash_info_stru {
			unsigned long flash_id; /* combined device & manufacturer code */
			unsigned short device_id; /* device id */
			unsigned short vendor; /* the primary vendor id */
			unsigned int state; /*flash health*/
			unsigned long size; /* total size in bytes */
			unsigned int sector_count; /* number of erase units */
			unsigned short manufacturer_id; /* manufacturer id */
			}

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

无。

调用示例

```
int i;
int ret = 0;
```

```
int card_id = 0x56;
int device_id = 0;
struct dcmi_flash_info_stru flash_info = {0};
unsigned int flash_count = 0;
ret = dcmi_get_device_flash_count(card_id, device_id,, &flash_count);
…
For (i = 0; i < flash_count; i++){
ret = dcmi_get_device_flash_info(card_id, device_id, i, &flash_info);
if(ret != 0) {
//todo：记录日志
return ret;
}
…
}
…
```

2.16 dcmi_get_memory_info 接口原型

函数原型

int dcmi_get_memory_info(int card_id, int device_id, struct dcmi_memory_info_stru * pdevice_memory_info)

功能说明

获取芯片的内存信息。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
device_id	输入	int	指定芯片编号，取值范围0~3，当前实际支持的范围通过**2.4 dcmi_get_device_num_in_card接口原型**获取。

pdevice_memory_info	输出	struct dcmi_memory_info_stru *	返回内存信息，内存信息结构体：
			struct dcmi_memory_info_stru{
			unsigned long memory_size;/*单位MB*/
			unsigned int freq; /*频率*，不支持，如需查看内存频率，请使用**2.13 dcmi_get_device_frequency接口原型**。/
			unsigned int utiliza; /*占用率*，不支持，如需查看内存利用率，请使用**2.12 dcmi_get_device_utilization_rate接口原型**。/
			}

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
int card_id = 0x56;
int device_id = 0;
struct dcmi_memory_info_stru device_memory_info = {0};
ret = dcmi_get_memory_info(card_id, device_id, &device_memory_info);
if(ret != 0) {
//todo：记录日志
return ret;
}
...
```

2.17 dcmi_get_ecc_info 接口原型

函数原型

int dcmi_get_ecc_info(int card_id, int device_id, int device_type, struct dcmi_ecc_info_stru * pdevice_ecc_info)

功能说明

获取ECC信息，目前仅支持查询的device_type为：DCMI_DEVICE_TYPE_DDR，其他类型会返回不支持。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。

参数名	输入/输出	类型	描述
device_id	输入	int	指定芯片编号，取值范围0~3，当前实际支持的范围通过**2.4 dcmi_get_device_num_in_card接口原型**获取。
device_type	输入	int	组件类型，目前支持如下几种： typedef enum { DCMI_DEVICE_TYPE_DDR, DCMI_DEVICE_TYPE_SRAM, DCMI_DEVICE_TYPE_HBM, DCMI_DEVICE_TYPE_AICORE, DCMI_DEVICE_TYPE_NONE=0xff } DCMI_DEVICE_TYPE;

pdevice_ecc_info	输出	struct dcmi_ecc_info_stru *	返回ECC结构体信息：
			struct dcmi_ecc_info_stru{
			int enable_flag;
			unsigned int single_bit_error_count;
			unsigned int double_bit_error_count;
			}

返回值

类型	描述
int	处理结果： • 返回0成功 • 返回0x01，表示不支持 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

无。

调用示例

```
...
int ret = 0;
int card_id = 0x56;
int device_id = 0;
```

```
struct dcmi_ecc_info_stru device_ecc_info = {0};
ret = dcmi_get_ecc_info(card_id, device_id, DCMI_DEVICE_TYPE_DDR, &device_ecc_info);
if(ret != 0) {
//todo：记录日志
return ret;
}
...
```

2.18 dcmi_get_device_die 接口原型

函数原型

int dcmi_get_device_die (int card_id, int device_id, struct dcmi_soc_die_stru * pdevice_die)

功能说明

获取指定设备的DIE ID。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
device_id	输入	int	指定芯片编号，取值范围0~3，当前实际支持的范围通过**2.4 dcmi_get_device_num_in_card接口原型**获取。

pdevice_die	输出	struct dcmi_soc_die_stru *	返回DIE结构体信息：
			unsigned int *pdevice_die
			struct dcmi_soc_die_stru{
			unsigned int soc_die[5];
			}

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

无。

调用示例

```
...
int ret = 0;
int card_id = 0x56;
int device_id = 0;
struct dcmi_soc_die_stru pdevice_die = {0};
ret = dcmi_get_device_die(card_id, device_id, &pdevice_die);
if(ret != 0) {
//todo：记录日志
return ret;
}
...
```

2.19 dcmi_get_board_info 接口原型

函数原型

int dcmi_get_board_info(int card_id, int device_id, struct dcmi_board_info_stru* pboard_info)

功能说明

获取Atlas 300I 推理卡信息：包括board id，pcb id，bom id等信息。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
device_id	输入	int	指定芯片编号，取值范围0~3，当前实际支持的范围通过**2.4 dcmi_get_device_num_in_card接口原型**获取。

参数名	输入/输出	类型	描述

pboard_info	输出	dcmi_board_info_stru *	struct dcmi_board_info_stru {
			unsigned int board_id;
			unsigned int pcb_id;
			unsigned int bom_id;
			unsigned int position_id;
			};
			说明
			position_id表示芯片在标卡上的位置编号。

返回值

类型	描述
int	处理结果： ● 返回0成功 ● 失败返回错误码，错误码小于0

异常处理

无。

约束说明

无。

调用示例

```
...
int ret = 0;
int card_id = 0x56;
int device_id = 0;
struct dcmi_board_info_stru board_info = {0};
ret = dcmi_get_board_info(card_id, device_id, &board_info);
if(ret != 0) {
//todo:记录日志
return ret;
}
...
```

2.20 dcmi_mcu_get_board_info 接口原型

函数原型

- NPU 20.0.0版本：

  int dcmi_mcu_get_board_info(int card_id, struct dsmi_board_info_stru * pboard_info)

- NPU 20.0.0以上及NPU 20.1.X及以上系列版本：

  int dcmi_mcu_get_board_info(int card_id, struct dcmi_board_info_stru * pboard_info)

功能说明

获取Atlas 300I 推理卡信息：包括board id，pcb id，bom id等信息。

参数说明

- NPU 20.0.0版本：

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。

pboard_info	输出	dsmi_board_info_stru *	struct dsmi_board_info_stru {
			unsigned int board_id;
			unsigned int pcb_id;
			unsigned int bom_id;
			unsigned int slot_id;
			};

- NPU 20.0.0以上及NPU 20.1.X及以上系列版本：

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。

pboard_info	输出	dcmi_board_info_stru *	struct dcmi_board_info_stru {
			unsigned int board_id;
			unsigned int pcb_id;
			unsigned int bom_id;
			unsigned int position_id;
			};

返回值

类型	描述
int	处理结果： ● 返回0成功 ● 失败返回错误码，错误码小于0

异常处理

无。

约束说明

该接口仅适用NPU 20.X.X及以上系列版本。

调用示例

- NPU 20.0.0版本：

```
...
int ret = 0;
int card_id = 0x56;
struct dsmi_board_info_stru board_info = {0};
ret = dcmi_mcu_get_board_info(card_id, &board_info);
if(ret != 0) {
//todo:记录日志
return ret;
}
...
```

- NPU 20.0.0以上及NPU 20.1.X及以上系列版本：

```
...
int ret = 0;
int card_id = 0x56;
struct dcmi_board_info_stru board_info = {0};
ret = dcmi_mcu_get_board_info(card_id, &board_info);
if(ret != 0) {
//todo:记录日志
return ret;
}
...
```

2.21 dcmi_mcu_get_power_info 接口原型

函数原型

int dcmi_mcu_get_power_info(int card_id, int *power)

功能说明

获取Atlas 300I 推理卡功耗信息。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
power	输出	int*	功耗。单位为W，精度为0.1W。16位无符号short类型，小字节序。计算公式：value=reading*0.1。

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

该接口仅适用NPU 20.X.X及以上系列版本。

调用示例

```
...
int ret = 0;
int card_id = 0x56;
int power = 0;
ret = dcmi_mcu_get_power_info (card_id, &power);
if(ret != 0) {
//todo:记录日志
return ret;
}
...
```

2.22 dcmi_get_pcie_error_cnt 接口原型

函数原型

int dcmi_get_pcie_error_cnt(int card_id, int device_id, struct dcmi_chip_pcie_err_rate_stru *pcie_err_code_info)

功能说明

查询芯片的PCIe（Peripheral Component Interconnect Express）误码计数信息。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
device_id	输入	int	指定芯片编号，取值范围0~3，当前实际支持的范围通过**2.4 dcmi_get_device_num_in_card接口原型**获取。
pcie_err_code_info	输出	struct dcmi_chip_pcie_err_rate_stru *	typedef struct dcmi_chip_pcie_err_rate_stru { unsigned int reg_deskew_fifo_overflow_intr_status; unsigned int reg_symbol_unlock_intr_status; unsigned int reg_deskew_unlock_intr_status; unsigned int reg_phystatus_timeout_intr_status; unsigned int symbol_unlock_counter; unsigned int pcs_rx_err_cnt; unsigned int phy_lane_err_counter; unsigned int pcs_rcv_err_status; unsigned int symbol_unlock_err_status; unsigned int phy_lane_err_status; unsigned int dl_lcrc_err_num; unsigned int dl_dcrc_err_num; } PCIE_ERR_RATE_INFO_STU;

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

该接口仅适用NPU 20.1.0及以上版本。

调用示例

```
...
int ret = 0;
int card_id = 0x56;
int device_id = 0;
struct dcmi_chip_pcie_err_rate_stru pcie_err_code_info = {0};
ret = dcmi_get_pcie_error_cnt(card_id, device_id, &pcie_err_code_info);
if(ret != 0) {
//todo:记录日志
return ret;
}
...
```


3 配置管理接口

3.1 dcmi_config_ecc_enable 接口原型

函数原型

int dcmi_config_ecc_enable (int card_id, int device_id, int enable_flag)

功能说明

配置存储ECC的标记为使能或禁用，调用该接口配置ECC标记后，需要复位标卡，配置才能生效。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
device_id	输入	int	指定芯片编号，取值范围0~3，当前实际支持的范围通过**2.4 dcmi_get_device_num_in_card接口原型**获取。
enable_flag	输入	int	• 0：禁用。 • 1：使能。

返回值

类型	描述
int	处理结果： ● 返回0成功 ● 返回0x01，表示不支持 ● 失败返回错误码，错误码小于0

异常处理

无。

约束说明

调用该接口的程序必须在物理机的root用户下运行，不支持在物理机的非root用户，或在容器、虚拟机下运行。

调用示例

```
int ret = 0;
int card_id = 0x56;
int device_id = 0;
int enable_flag = 1;
ret = dcmi_config_ecc_enable(card_id, device_id, enable_flag);
if(ret != 0) {
//todo：记录日志
return ret;
}
ret = dcmi_config_ecc_enable(card_id, device_id, enable_flag);
if(ret != 0) {
//todo：记录日志
return ret;
}
```

3.2 dcmi_config_p2p_enable 接口原型

函数原型

int dcmi_config_p2p_enable (int card_id, int device_id, int enable_flag)

功能说明

配置Flash存储的P2P标记为使能或禁用。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
device_id	输入	int	指定芯片编号，取值范围0~3，当前实际支持的范围通过**2.4 dcmi_get_device_num_in_card接口原型**获取。
enable_flag	输入	int	• 0：禁用。 • 1：使能。

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

NPU 20.X.X系列版本没有此接口。

调用示例

```
int ret = 0;
int card_id = 0x56;
int device_id = 0;
int enable_flag = 1;
ret = dcmi_config_p2p_enable(card_id, device_id, enable_flag);
if(ret != 0) {
//todo：记录日志
return ret;
}
…
```

3.3 dcmi_get_p2p_enable 接口原型

函数原型

int dcmi_get_p2p_enable(int card_id, int device_id, int* enable_flag)

功能说明

查询Flash存储的P2P使能标记。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
device_id	输入	int	指定芯片编号，取值范围0~3，当前实际支持的范围通过**2.4 dcmi_get_device_num_in_card接口原型**获取。
enable_flag	输出	int *	• 0：禁用。 • 1：使能。

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
int enable_flag = 1;
int card_id = 0x56;
int device_id = 0;
ret = dcmi_get_p2p_enable(card_id, device_id, &enable_flag);
if(ret != 0) {
//todo：记录日志
return ret;
}
...
```

3.4 dcmi_clear_pcie_error_cnt 接口原型

函数原型

int dcmi_clear_pcie_error_cnt(int card_id, int device_id)

功能说明

清除芯片的PCIe（Peripheral Component Interconnect Express）误码计数信息。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
device_id	输入	int	指定芯片编号，取值范围0~3，当前实际支持的范围通过**2.4 dcmi_get_device_num_in_card接口原型**获取。

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

该接口仅适用NPU 20.1.0及以上版本。

调用示例

```
int ret = 0;
int card_id = 0x56;
int device_id = 0;
ret = dcmi_clear_pcie_error_cnt(card_id, device_id);
if(ret != 0) {
//todo：记录日志
return ret;
```

```
}
...
```

4 MCU 升级控制接口

4.1 dcmi_mcu_upgrade_control 接口原型

函数原型

int dcmi_mcu_upgrade_control(int card_id, int upgrade_type)

功能说明

控制MCU（Multipoint Control Unit）升级。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
upgrade_type	输入	int	取值范围： • 1：启动升级 • 2：停止升级 • 3：生效

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

1. 异步调用。
2. 调用该接口的程序必须在物理机的root用户下运行，不支持在物理机的非root用户，或在容器、虚拟机下运行。

调用示例

```
int ret = 0;
int card_id = 0x56;
int device_id = 0;
ret = dcmi_mcu_upgrade_control(card_id,1);
if(ret != 0) {
//todo：记录日志
return ret;
}
...
```

4.2 dcmi_mcu_upgrade_transfile 接口原型

函数原型

int dcmi_mcu_upgrade_transfile(int card_id, char *file)

功能说明

将MCU的升级包传至MCU。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过2.2 dcmi_get_card_num_list接口原型获取。
file	输入	char *	MCU升级包。

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

1. 异步调用。
2. 调用该接口的程序必须在物理机的root用户下运行，不支持在物理机的非root用户，或在容器、虚拟机下运行。

调用示例

```
int ret = 0;
int card_id = 0x56;
ret = dcmi_mcu_upgrade_transfile(card_id,"./mcu.bin");
if(ret != 0) {
//todo：记录日志
return ret;
}
...
```

4.3 dcmi_mcu_get_upgrade_statues 接口原型

函数原型

int dcmi_mcu_get_upgrade_statues(int card_id, int *status, int *progress)

功能说明

查询MCU升级状态及进度。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过2.2 dcmi_get_card_num_list接口原型获取。
status	输出	int *	升级进度，0~100百分比。

参数名	输入/输出	类型	描述
progress	输出	int *	升级状态，目前支持如下几种： 0：升级成功 1：升级中 2：不支持升级 3：升级失败 4：获取状态失败

返回值

类型	描述
int	处理结果： ● 返回0成功 ● 失败返回错误码，错误码小于0

异常处理

无。

约束说明

该接口不支持在容器和虚拟机下运行。

调用示例

```
...
int ret = 0;
int card_id = 0x56;
int status;
int progress;
ret = dcmi_mcu_get_upgrade_statues(card_id,  &status, &progress);
if(ret != 0) {
//todo：记录日志
return ret;
}
...
```

4.4 dcmi_mcu_get_version 接口原型

函数原型

- 固件版本为1.73.5.5以下：

  int dcmi_mcu_get_version(int card_id, char *data_info, int *len)

- 固件版本为1.73.5.5及以上：

  int dcmi_mcu_get_version(int card_id, char *version_str, int max_version_len, int *len)

功能说明

查询MCU版本号。

参数说明

- 固件版本为1.73.5.5以下：

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
data_info	输出	char*	用户申请的空间，存放返回的固件版本号。 x.y[.z]格式，x表示Major版本，y表示Minor版本，z表示Revision版本。
len	输出	int *	版本号长度。

- 固件版本为1.73.5.5及以上：

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
version_str	输出	char*	用户申请的空间，存放返回的固件版本号。 x.y[.z]格式，x表示Major版本，y表示Minor版本，z表示Revision版本。
max_version_len	输入	int	version_str空间的最大长度。
len	输出	int *	版本号长度。

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

无。

调用示例

- 固件版本为1.73.5.5以下：

```
int ret = 0;
char version_str[16] = {0};
int card_id = 0x56;
int len = 0;
ret = dcmi_mcu_get_version(card_id, version_str, &len);
if(ret != 0) {
//todo：记录日志
return ret;
}
…
```

- 固件版本为1.73.5.5及以上：

```
int ret = 0;
char version_str[16] = {0};
int card_id = 0x56;
int len = 0;
ret = dcmi_mcu_get_version(card_id, version_str, sizeof(version_str), &len)
if(ret != 0) {
//todo：记录日志
return ret;
}
…
```

4.5 dcmi_get_version 接口原型

函数原型

- 固件版本为1.73.5.5以下：

  int dcmi_get_version(int card_id, int device_id, char* verison_str, int *len)

- 固件版本为1.73.5.5及以上：

  int dcmi_get_version(int card_id, int device_id, char *version_str, unsigned int version_len, int *len)

功能说明

查询芯片驱动版本。

参数说明

- 固件版本为1.73.5.5以下：

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
device_id	输入	int	指定芯片编号，取值范围0~3，当前实际支持的范围通过**2.4 dcmi_get_device_num_in_card接口原型**获取。
verison_str	输出	char *	返回驱动版本。 该空间由用户申请，大小为64Byte。
len	输出	int *	返回版本号长度。

- 固件版本为1.73.5.5及以上：

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
device_id	输入	int	指定芯片编号，取值范围0~3，当前实际支持的范围通过**2.4 dcmi_get_device_num_in_card接口原型**获取。
version_str	输出	char *	返回驱动版本。 该空间由用户申请。
version_len	输入	unsigned int	version_str空间的大小。
len	输出	int *	返回版本号长度。

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

无。

调用示例

- 固件版本为1.73.5.5以下：

```
int ret = 0;
int len = -1;
int card_id = 0x56;
int device_id = 0;
unsigned char version_str[64] = {0};
ret = dcmi_get_version(card_id, device_id, version_str, &len);
if(ret != 0) {
//todo：记录日志
return ret;
}
…
```

- 固件版本为1.73.5.5及以上：

```
int ret = 0;
int len = -1;
int card_id = 0x56;
int device_id = 0;
unsigned char version_str[64] = {0};
ret = dcmi_get_version(card_id, device_id, version_str, sizeof(version_str), &len);
if(ret != 0) {
//todo：记录日志
return ret;
}
…
```

5 芯片复位接口

5.1 dcmi_pre_reset_soc 接口原型

函数原型

int dcmi_pre_reset_soc (int card_id, int device_id)

功能说明

芯片预复位，发起芯片预复位接口，预复位目的是解除上层驱动及软件对此芯片的依赖。预复位完成后可以对此芯片进行隔离或实际复位操作。

一般用于实际复位完成前触发操作流程：dcmi_pre_reset_soc -> reset –> dcmi_rescan_soc

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
device_id	输入	int	指定芯片编号，取值范围0~3，当前实际支持的范围通过**2.4 dcmi_get_device_num_in_card接口原型**获取。

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

调用该接口的程序必须在物理机的root用户下运行，不支持在物理机的非root用户，或在容器、虚拟机下运行。

调用示例

```
int ret = 0;
int card_id = 0x56;
int device_id = 0;
ret = dcmi_pre_reset_soc(card_id, device_id);
if(ret != 0) {
//todo：记录日志
return ret;
}
...
```

5.2 dcmi_reset_device 接口原型

函数原型

int dcmi_reset_device (int card_id, int device_id)

功能说明

复位芯片。

在调用该接口前，需要调用dcmi_pre_reset_soc接口预复位芯片。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过2.2 dcmi_get_card_num_list接口原型获取。

参数名	输入/输出	类型	描述
device_id	输入	int	指定芯片编号，取值范围0~3，当前实际支持的范围通过**2.4 dcmi_get_device_num_in_card接口原型**获取。

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

1. 在调用该接口前，需要调用dcmi_pre_reset_soc接口预复位芯片。
2. 调用该接口的程序必须在物理机的root用户下运行，不支持在物理机的非root用户，或在容器、虚拟机下运行。

调用示例

```
int ret = 0;
int card_id = 0x56;
int device_id = 0;
ret = dcmi_reset_device(card_id, device_id);
if(ret != 0) {
//todo：记录日志
return ret;
}
```

5.3 dcmi_rescan_soc 接口原型

函数原型

int dcmi_rescan_soc (int card_id, int device_id)

功能说明

对指定芯片重新扫描。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
device_id	输入	int	指定芯片编号，取值范围0~3，当前实际支持的范围通过**2.4 dcmi_get_device_num_in_card接口原型**获取。

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

调用该接口的程序必须在物理机的root用户下运行，不支持在物理机的非root用户，或在容器、虚拟机下运行。

调用示例

```
int ret = 0;
int card_id = 0x56;
int device_id = 0;
ret = dcmi_rescan_soc(card_id, device_id);
if(ret != 0) {
//todo：记录日志
return ret;
}
...
```

5.4 dcmi_get_device_boot_status 接口原型

函数原型

int dcmi_get_device_boot_status(int card_id, int device_id, enum dcmi_boot_status *boot_status);

功能说明

获取芯片系统的启动状态。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
device_id	输入	int	指定芯片编号，取值范围0~3，当前实际支持的范围通过**2.4 dcmi_get_device_num_in_card接口原型**获取。
boot_status	输出	enum dcmi_boot_status *	enum dcmi_boot_status { DCMI_BOOT_STATUS_UNINIT = 0, /*未初始化*/ DCMI_BOOT_STATUS_BIOS, * BIOS启动中*/ DCMI_BOOT_STATUS_OS, /* OS启动中*/ DCMI_BOOT_STATUS_FINISH /*启动完成*/ }DSMI_BOOT_STATUS;

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
int card_id = 0x56;
int device_id = 0;
enum dcmi_boot_status boot_status = 0;
ret = dcmi_get_device_boot_status(card_id, device_id, &boot_status);
if(ret != 0) {
//todo：记录日志
return ret;
}
…
```

5.5 dcmi_reset_device_inband 接口原型

函数原型

int dcmi_reset_device_inband (int card_id, int device_id)

功能说明

带内复位芯片。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号，当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
device_id	输入	int	指定芯片编号，取值范围0~3，当前实际支持的范围通过**2.4 dcmi_get_device_num_in_card接口原型**获取。

返回值

类型	描述
int	处理结果： • 返回0成功 • 失败返回错误码，错误码小于0

异常处理

无。

约束说明

该接口仅适用NPU 20.2.X及以上系列版本。

调用该接口的程序必须在物理机的root用户下运行，不支持在物理机的非root用户，或在容器、虚拟机下运行。

调用示例

```
int ret = 0;
int card_id = 0x56;
int device_id = 0;
ret = dcmi_reset_device_inband(card_id, device_id);
if(ret != 0) {
//todo：记录日志
return ret;
}
...
```


6 证书查询配置接口

证书查询配置接口只能在Atlas 300I 推理卡npu-smi及MCU版本为以下情况下使用。

- 1.4.X系列版本：1.4.0及以上版本
- 20.X.X系列版本：所有版本均支持。

6.1 dcmi_mcu_get_license_info 接口原型

函数原型

int dcmi_mcu_get_license_info(int card_id, char *license, int *len)

功能说明

查询存放在MCU的用户License证书。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号。 当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
license	输出	char *	返回证书的内容。
len	输出	int *	返回证书的长度。

返回值

类型	描述
int	处理结果： • 返回0成功。 • 失败返回错误码，错误码小于0。

异常处理

无。

约束说明

该接口不支持在容器和虚拟机下运行。

调用示例

```
int ret = 0;
int card_id = 0x56;
char license[256] = {0};
int len = 0;
ret = dcmi_mcu_get_license_info(card_id, license, &len);
if(ret != 0) {
//todo：记录日志
return ret;
}
...
```

6.2 dcmi_mcu_set_license_info 接口原型

函数原型

int dcmi_mcu_set_license_info(int card_id, char *license, int len)

功能说明

向MCU写入用户License证书。

参数说明

参数名	输入/输出	类型	描述
card_id	输入	int	指定Atlas 300I 推理卡的编号。 当前实际支持的编号通过**2.2 dcmi_get_card_num_list接口原型**获取。
license	输入	char *	写入证书的内容。
len	输入	int	写入证书的长度。

返回值

类型	描述
int	处理结果： ● 返回0成功。 ● 失败返回错误码，错误码小于0。

异常处理

无。

约束说明

1. 异步接口。
2. 调用该接口的程序必须在物理机的root用户下运行，若在物理机的非root用户，或在容器、虚拟机下运行，则会返回权限错误。

调用示例

```
int ret = 0;
int card_id = 0x56;
char license[4] = {0};
int len = 4;
ret = dcmi_mcu_set_license_info(card_id, license, len);
if(ret != 0) {
//todo：记录日志
return ret;
}
```

7 调用示例

各接口说明处给的是接口的调用片段，此处以调用dcmi_get_card_num_list接口为例，给出调用示例，说明需要include的头文件（dcmi_interface_api.h）、调用接口的逻辑等。

您可以从已安装驱动和npu-smi工具环境的“/usr/local/dcmi/”目录下获取dcmi_interface_api.h文件。

示例代码文件 get_card_num_list.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include "dcmi_interface_api.h"

#define MAX_CARD_NUM (16)
#define NPU_OK (0)

int main(int argc, char ** argv)
{
    int ret;
    int card_count;
    int card_id_list[MAX_CARD_NUM] = {0};

    ret = dcmi_init();
    if (ret != NPU_OK) {
        printf("Failed to init dcmi.\n");
        return ret;
    }

    ret = dcmi_get_card_num_list(&card_count, card_id_list, MAX_CARD_NUM);
    if (ret != NPU_OK) {
        printf("Failed to get card number.\n");
        return ret;
    }
    printf("card count is %d\n", card_count);

    return ret;
}
```

NPU 20.X.X 及以上系列版本使用调用示例

步骤1 以HwHiAiUser用户将get_card_num_list.c、dcmi_interface_api.h传到Host侧服务器的同一个目录下。

步骤2 以HwHiAiUser用户登录到Host侧服务器。

步骤3 执行如下命令，编译get_card_num_list.c中的代码，生成可执行文件card_list。编译时需要使用驱动中的dsmi相关的驱动，具体如下：

```
gcc get_card_num_list.c /usr/local/dcmi/libdcmi.so /usr/local/Ascend/driver/lib64/*.so -o card_list -lm
```

说明

以上命令中，“*/usr/local/Ascend* ”为驱动的默认安装路径，请根据实际情况进行替换。

步骤4 执行可执行文件card_list。

```
./card_list
```

----结束

NPU 1.X.X 系列版本使用调用示例

步骤1 以root用户将get_card_num_list.c、dcmi_interface_api.h传到Host侧服务器的同一个目录下。

步骤2 以root用户登录到Host侧服务器。

步骤3 执行如下命令，编译get_card_num_list.c中的代码，生成可执行文件card_list。编译时需要使用驱动中的dsmi相关的驱动，具体如下：

```
gcc get_card_num_list.c /usr/local/dcmi/libdcmi.so /usr/local/HiAI/driver/lib64/*.so -o card_list
```

说明

以上命令中，“*/usr/local/HiAI* ”为驱动的默认安装路径，请根据实际情况进行替换。

步骤4 执行可执行文件card_list。

```
./card_list
```

----结束

A 附录

A.1 缩略语

B		
BMC	Baseboard Management Controller	主板管理控制单元
D		
DCMI	DaVinci Card Management Interface	达芬奇卡管理接口
M		
MCU	Microcontroller Unit	微控制单元
MDC	MetaData Controller	元数据控制设备
P		
PCIe	Peripheral Component Interconnect Express	快捷外围部件互连标准

A.2 免责声明

- 本文档可能包含第三方信息、产品、服务、软件、组件、数据或内容（统称“第三方内容”）。华为不控制且不对第三方内容承担任何责任，包括但不限于准确性、兼容性、可靠性、可用性、合法性、适当性、性能、不侵权、更新状态等，除非本文档另有明确说明。在本文档中提及或引用任何第三方内容不代表华为对第三方内容的认可或保证。
- 用户若需要第三方许可，须通过合法途径获取第三方许可，除非本文档另有明确说明。

A.3 如何获取帮助

日常维护或故障处理过程中遇到难以解决或者重大问题时，请寻求华为技术有限公司的技术支持。

A.3.1 收集必要的故障信息

在进行故障处理前，需要收集必要的故障信息。

收集的信息主要包括：

- 客户的详细名称、地址
- 联系人姓名、电话号码
- 故障发生的具体时间
- 故障现象的详细描述
- 设备类型及软件版本
- 故障后已采取的措施和结果
- 问题的级别及希望解决的时间

A.3.2 做好必要的调试准备

在寻求华为技术支持时，华为技术支持工程师可能会协助您做一些操作，以进一步收集故障信息或者直接排除故障。

在寻求技术支持前请准备好单板和端口模块的备件、螺丝刀、螺丝、串口线、网线等可能使用到的物品。

A.3.3 如何使用文档

华为技术有限公司提供全面的随设备发货的指导文档。指导文档能解决您在日常维护或故障处理过程中遇到的常见问题。

为了更好的解决故障，在寻求华为技术支持前，建议充分使用指导文档。

A.3.4 获取技术支持

华为技术有限公司通过办事处、公司二级技术支持体系、电话技术指导、远程支持及现场技术支持等方式向用户提供及时有效的技术支持。

技术支持网址

查阅技术资料合集：**https://e.huawei.com/cn/** > 技术支持 > 产品和解决方案支持 > 服务器-智能计算 > 昇腾计算

查阅技术资料的使用流程：**https://www.hiascend.com** > 文档

自助平台与论坛

如果您想进一步学习和交流：

- 访问**华为服务器信息服务平台**，获取相关服务器产品资料。

- 访问**华为企业业务智能问答系统**，快速查询产品问题。
- 访问**华为企业互动社区（服务器）**，进行硬件产品学习交流。
- 访问**开发者论坛**，进行AI应用开发学习交流。

公告

有关产品生命周期、预警和整改公告请访问**技术支持 > 公告 > 产品公告**。

案例库

参阅已有案例进行学习：**计算产品案例查询助手**。

说明

计算产品案例查询助手目前仅面向华为合作伙伴及华为工程师开放。

获取华为技术支持

如果在设备维护或故障处理过程中，遇到难以确定或难以解决的问题，通过文档的指导仍然不能解决，请通过如下方式获取技术支持：

- 联系华为技术有限公司客户服务中心。

  中国区企业用户请通过以下方式联系我们：

  - 客户服务电话：400-822-9999
  - 客户服务邮箱：**support_e@huawei.com**

    企业网全球各地区客户服务热线可以通过以下网站查找：**企业用户全球服务热线**

  中国区运营商用户请通过以下方式联系我们：

  - 客户服务电话：400-830-2118
  - 客户服务邮箱：**support@huawei.com**

    运营商全球各地区客户服务热线可以通过以下网站查找：**运营商用户全球服务热线**
- 联系华为技术有限公司驻当地办事处的技术支持人员。