

Ascend 310
V100R001

DSMI(Device System Manage Interface) API 参考

文档版本 01
发布日期 2019-03-12



版权所有 © 华为技术有限公司 2019。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HUAWEI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。

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：<http://www.huawei.com>

客户服务邮箱：support@huawei.com

客户服务电话：4008302118

目 录

1 设备管理接口	1
1.1 dsmi_get_device_count 接口原型	2
1.2 dsmi_list_device 接口原型	2
1.3 dsmi_get_device_health 接口原型	3
1.4 dsmi_get_device_errorcode 接口原型	4
1.5 dsmi_query_errorstring 接口原型	5
1.6 dsmi_get_device_errorinfo 接口原型	7
1.7 dsmi_get_device_temperature 接口原型	8
1.8 dsmi_get_device_power_info 接口原型	9
1.9 dsmi_get_pcie_info 接口原型	10
1.10 dsmi_get_device_voltage 接口原型	11
1.11 dsmi_get_device_utilization_rate 接口原型	12
1.12 dsmi_get_device_frequency 接口原型	13
1.13 dsmi_get_device_flash_count 接口原型	14
1.14 dsmi_get_device_flash_info 接口原型	15
1.15 dsmi_get_memory_info 接口原型	16
1.16 dsmi_get_ecc_info 接口原型	17
1.17 dsmi_get_device_die 接口原型	19
1.18 dsmi_passthru_mcu 接口原型	20
1.19 dsmi_get_board_info 接口原型	21
1.20 dsmi_get_soc_sensor_info 接口原型	22
1.21 dsmi_get_mini2mcu_heartbeat_status 接口原型	24
1.22 dsmi_get_pmu_voltage 接口原型	25
2 MAC 地址管理	27
2.1 dsmi_get_mac_count 接口原型	27
2.2 dsmi_get_mac_addr 接口原型	28
2.3 dsmi_set_mac_addr 接口原型	29
3 风扇管理	31
3.1 dsmi_get_fan_count 接口原型	31
3.2 dsmi_get_fan_speed 接口原型	32
3.3 dsmi_set_fan_speed 接口原型	33
4 配置管理	35

4.1 dsmi_config_ecc_enable 接口原型.....	35
4.2 dsmi_get_ecc_enable 接口原型.....	37
4.3 dsmi_config_p2p_enable 接口原型.....	38
4.4 dsmi_get_p2p_enable 接口原型.....	39
4.5 dsmi_get_system_time 接口原型.....	40
4.6 dsmi_set_device_ip_address 接口原型.....	41
4.7 dsmi_get_device_ip_address 接口原型.....	42
4.8 dsmi_set_user_config 接口原型.....	43
4.9 dsmi_get_user_config 接口原型.....	44
4.10 dsmi_clear_user_config 接口原型.....	45
5 软件升级.....	47
5.1 dsmi_upgrade_start 接口原型.....	47
5.2 dsmi_upgrade_get_state 接口原型.....	49
5.3 dsmi_upgrade_get_component_static_version 接口原型.....	50
5.4 dsmi_get_version 接口原型.....	51
5.5 dsmi_get_component_list 接口原型.....	52
6 芯片复位启动.....	55
6.1 dsmi_pre_reset_soc 接口原型.....	55
6.2 dsmi_rescan_soc 接口原型.....	56
6.3 dsmi_get_device_boot_status 接口原型.....	57

1 设备管理接口

关于本章

- [1.1 dsmi_get_device_count接口原型](#)
- [1.2 dsmi_list_device接口原型](#)
- [1.3 dsmi_get_device_health接口原型](#)
- [1.4 dsmi_get_device_errorcode接口原型](#)
- [1.5 dsmi_query_errorstring接口原型](#)
- [1.6 dsmi_get_device_errorinfo接口原型](#)
- [1.7 dsmi_get_device_temperature接口原型](#)
- [1.8 dsmi_get_device_power_info接口原型](#)
- [1.9 dsmi_get_pcie_info接口原型](#)
- [1.10 dsmi_get_device_voltage接口原型](#)
- [1.11 dsmi_get_device_utilization_rate接口原型](#)
- [1.12 dsmi_get_device_frequency接口原型](#)
- [1.13 dsmi_get_device_flash_count接口原型](#)
- [1.14 dsmi_get_device_flash_info接口原型](#)
- [1.15 dsmi_get_memory_info接口原型](#)
- [1.16 dsmi_get_ecc_info接口原型](#)
- [1.17 dsmi_get_device_die接口原型](#)
- [1.18 dsmi_passthru_mcu接口原型](#)
- [1.19 dsmi_get_board_info接口原型](#)
- [1.20 dsmi_get_soc_sensor_info接口原型](#)
- [1.21 dsmi_get_mini2mcu_heartbeat_status接口原型](#)

1.22 dsmi_get_pmu_voltage接口原型

1.1 dsmi_get_device_count 接口原型

函数原型

```
int dsmi_get_device_count(int *device_count)
```

功能说明

查询系统所有MINI芯片数，支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_count	输出	int	MINI设备个数，取值范围：0~64。

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

```
...  
int ret;  
int device_count = 0;  
ret = dsmi_get_device_count(&device_count);  
...
```

1.2 dsmi_list_device 接口原型

函数原型

```
int dsmi_list_device(int device_id_list[], int count)
```

功能说明

列举所有MINI设备的设备号，支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id_list[]	输出	int	输出所有mini设备的设备号列表，取值范围：0~63。
count	输入	int	MINI设备个数；count由dsmi_get_device_count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。

异常处理

无。

约束说明

无。

调用示例

```
...
int ret = 0;
int device_count = 0;
int device_list[64] = {0};
ret=dsmi_get_device_count(&device_count);
if((ret != 0) || (0 == device_count)){
    //todo:记录日志
    return ERROR;
}
ret = dsmi_list_device(&device_list[0], device_count);
...
```

1.3 dsmi_get_device_health 接口原型

函数原型

```
int dsmi_get_device_health(int device_id, unsigned int *phealth)
```

功能说明

查询设备总体健康状态，支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过 dsmi_list_device接口获取。
phealth	输出	unsigned int *	设备总体健康状态，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： <ul style="list-style-type: none"> ● 0：正常 ● 1：一般告警 ● 2：重要告警 ● 3：紧急告警

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

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;
unsigned int health;
ret = dsmi_get_device_health(0, &health);
...
```

1.4 dsmi_get_device_errorcode 接口原型

函数原型

```
int dsmi_get_device_errorcode(int device_id, int* errorcount,unsigned int *perrorcode)
```

功能说明

查询设备故障码；支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
errorcount	输出	int *	错误码数量，取值范围：0~128。
perrorcode	输出	unsigned int *	错误码。

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

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

1.5 dsmi_query_errorstring 接口原型

函数原型

int dsmi_query_errorstring(int device_id,int errorcode,unsigned char *perrorinfo, int buffsize)

功能说明

查询设备故障描述；支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，有效值范围：0~63，当前实际支持的设备号，通过 dsmi_list_device接口获取。
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。

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

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                    (256)
...
int ret = 0;
int errorcount = 0;
unsigned char perrorinfo[BUFF_SIZE] = {0};
unsigned char perrorcode [ERROR_CODE_MAX_NUM] = {0};
ret = dsmi_get_device_errorcode(0, &errorcount, perrorcode);
if((ret != 0) || (errorcount == 0)) {
    //todo:记录日志
    return ret;
}
ret = dsmi_query_errorstring(0, perrorcode[0], perrorinfo, BUFF_SIZE);
if(ret != 0) {
    //todo:记录日志
    return ret;
}
...
```

1.6 dsmi_get_device_errorinfo 接口原型

函数原型

```
int dsmi_get_device_errorinfo(int device_id,int* errorcount, unsigned char
*perrorinfo,int buffsize)
```

功能说明

查询设备故障信息；支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，有效值范围：0~63，当前实际支持的设备号，通过 dsmi_list_device接口获取。
errorcount	输出	int *	错误数量。
perrorinfo	输出	unsigned char *	对应的错误字符描述。
buffersize	输入	int	带入的buff大小。

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

暂不支持。

调用示例

```
#define BUFF_SIZE      (256)
...
int ret = 0;
int errorcount = 0;
unsigned char perrorinfo[BUFF_SIZE] = {0};
ret = dsmi_get_device_errorinfo(0, &errorcount, perrorinfo, BUFF_SIZE);
if(ret != 0) {
//todo: 记录日志
```

```
return ret;
}
...
```

1.7 dsmi_get_device_temperature 接口原型

函数原型

```
int dsmi_get_device_temperature(int device_id, int *ptemperature)
```

功能说明

查询芯片温度，支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，有效值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
ptemperature	输出	int *	芯片温度：单位摄氏度，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。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
int temp = 0;
ret = dsmi_get_device_temperature(0, &temp);
if(ret != 0) {
    //todo: 记录日志
    return ret;
}
...
```

1.8 dsmi_get_device_power_info 接口原型

函数原型

```
int dsmi_get_device_power_info(int device_id, struct dsmi_power_info_stru *
pdevice_power_info)
```

功能说明

查询设备功耗，支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，有效值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
pdevice_power_info	输出	struct dsmi_power_info_stru *	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。BMC侧计算公式： 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。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
struct dsmi_power_info_stru powerinfo = {0};
ret = dsmi_get_device_power_info(0, &powerinfo);
if(ret != 0) {
//todo: 记录日志
return ret;
}
...
```

1.9 dsmi_get_pcie_info 接口原型

函数原型

```
int dsmi_get_pcie_info(int device_id, struct tag_pcie_idinfo *pcie_idinfo)
```

功能说明

查询PCIe设备信息，只支持PCIe标卡。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，有效值范围：0~63，当前实际支持的设备号，通过 dsmi_list_device接口获取。
pcie_idinfo	输出	struct tag_pcie_idinfo *	PCIe设备信息。

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
struct tag_pcie_idinfo pcie_idinfo = {0};
ret = dsmi_get_pcie_info(0, &pcie_idinfo);
if(ret != 0) {
    //todo: 记录日志
    return ret;
}
...
```

1.10 dsmi_get_device_voltage 接口原型

函数原型

`int dsmi_get_device_voltage(int device_id, unsigned int *pvoltage)`

功能说明

查询芯片电压，支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，有效值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
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。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
unsigned int voltage;
ret = dsmi_get_device_voltage(0, &voltage);
if(ret != 0) {
    //todo: 记录日志
    return ret;
}
...
```

1.11 dsmi_get_device_utilization_rate 接口原型

函数原型

```
int dsmi_get_device_utilization_rate(int device_id, int device_type, unsigned int *putilization_rate)
```

功能说明

获取Davinci占用率，支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，有效值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
device_type	输入	int	设备类型，目前支持如下几种，数值和具体设备类型对应如下。 <ul style="list-style-type: none">● 1：内存● 2：AI CORE● 3：AI CPU● 4：控制CPU● 5：内存带宽
putilization_rate	输出	unsigned int *	Davinci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。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
unsigned int putilization_rate;
ret = dsmi_get_device_utilization_rate(0, DSMI_DEVICE_TYPE_SRAM, &putilization_rate);
if(ret != 0) {
    //todo: 记录日志
    return ret;
}
...
```

1.12 dsmi_get_device_frequency 接口原型

函数原型

```
int dsmi_get_device_frequency(int device_id, int device_type, unsigned int *pfrequency)
```

功能说明

获取Davinci频率，支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，有效值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
device_type	输入	int	设备类型，目前支持如下几种，数值和具体设备类型对应如下。 <ul style="list-style-type: none"> ● 1：指内存 ● 2：指AI CORE
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。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
unsigned int frequency;
ret = dsmi_get_device_frequency(0, DSMI_DEVICE_TYPE_SRAM, &frequency);
if(ret != 0) {
    //todo: 记录日志
    return ret;
}
...
```

1.13 dsmi_get_device_flash_count 接口原型

函数原型

```
int dsmi_get_device_flash_count(int device_id, unsigned int *pflash_count)
```

功能说明

获取Flash个数，支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，有效值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
pflash_count	输出	unsigned int *	返回Flash个数，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。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
unsigned int flash_count = 0;
ret = dsmi_get_device_flash_count(0, &flash_count);
if(ret != 0) {
```

```
//todo: 记录日志
return ret;
}
...
```

1.14 dsmi_get_device_flash_info 接口原型

函数原型

```
int dsmi_get_device_flash_info(int device_id, unsigned int flash_index,
dm_flash_info_stru *pflash_info)
```

功能说明

获取flash设备的信息，支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，有效值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
flash_index	输入	unsigned int	Flash索引号。
pflash_info	输出	dm_flash_info_stru *	返回Flash设备信息。 FLASH 信息结构体定义 struct dmanage_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 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。

异常处理

无。

约束说明

无。

调用示例

```
int i;
int ret = 0;
dm_flash_info_stru flash_info = {0};
unsigned int flash_count = 0;
ret = dsmi_get_device_flash_count(0, &flash_count);
...
For (i = 0; i < flash_count; i++){
ret = dsmi_get_device_flash_info(0, i, &flash_info);
if(ret != 0) {
//todo: 记录日志
return ret;
}
...
}
```

1.15 dsmi_get_memory_info 接口原型

函数原型

```
int dsmi_get_memory_info(int device_id, struct dsmi_memory_info_stru *
pdevice_memory_info)
```

功能说明

获取内存信息，支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，有效值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
pdevice_memory_info	输出	struct dsmi_memory_info_stru *	返回内存信息，内存信息结构体： struct dsmi_memory_info_stru{ unsigned long memory_size;//单位 MB }

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
struct dsmi_memory_info_stru device_memory_info = {0};
ret = dsmi_get_memory_info(0, &device_memory_info);
if(ret != 0) {
    //todo: 记录日志
    return ret;
}
...
```

1.16 dsmi_get_ecc_info 接口原型

函数原型

```
int dsmi_get_ecc_info(int device_id, int device_type, struct dsmi_ecc_info_stru *
pdevice_ecc_info)
```

功能说明

获取ECC信息，支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，有效值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。

参数名	输入/输出	类型	描述
device_type	输入	int	设备类型，目前只支持DSMI_DEVICE_TYPE_DDR。 typedef enum { DSMI_DEVICE_TYPE_DDR, DSMI_DEVICE_TYPE_SRAM, DSMI_DEVICE_TYPE_HBM, DSMI_DEVICE_TYPE_GPU, DSMI_DEVICE_TYPE_NONE=0xff } DSMI_DEVICE_TYPE;
pdevice_ecc_info	输出	struct dsmi_ecc_info_stru *	返回ECC结构体信息。 struct dsmi_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，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
struct dsmi_ecc_info_stru device_ecc_info = {0};
ret = dsmi_get_ecc_info(0, DSMI_DEVICE_TYPE_DDR, &device_ecc_info);
if(ret != 0) {
//todo: 记录日志
return ret;
}
```

1.17 dsmi_get_device_die 接口原型

函数原型

```
int dsmi_get_device_die(int device_id, struct dsmi_soc_die_stru * pdevice_die)
```

功能说明

获取指定设备的DIE ID，支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，有效值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
pdevice_die	输出	struct dsmi_soc_die_stru *	返回DIE结构体信息： struct dsmi_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。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
struct dsmi_soc_die_stru pdevice_die = {0};
ret = dsmi_get_device_die (0,&pdevice_die);
if(ret != 0) {
    //todo: 记录日志
    return ret;
}
```

1.18 dsmi_passthru_mcu 接口原型

函数原型

```
int dsmi_passthru_mcu(int device_id, struct passthru_message_stru
*passthru_message)
```

功能说明

消息转发接口，HOST消息通过MINI转发MCU。对于需要通过MINI获取MCU信息的功能，在HOST构建消息，把消息发送到MINI，MINI再转发到MCU，只支持PCIe标卡。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，有效值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
passthru_message	输入	struct passthru_message_stru *	消息内容，消息结构如下： struct passthru_message_stru{ unsigned int src_len; unsigned int rw_flag; /*0 read ,1 write*/ struct dmp_message_stru src_message; struct dmp_message_stru dest_message; };

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
struct passthru_message_stru assthru_message = {0};
```



```
assthru_message.rw_flag = 1;
assthru_message.src_len = sizeof(struct dmp_message_stru);
assthru_message.src_message.data.req.opcode = 0x08;
...
ret = dsmi_passthru_mcu(0, &assthru_message);
if(ret != 0) {
    //todo:记录日志
    return ret;
}
...
```

1.19 dsmi_get_board_info 接口原型

函数原型

```
int dsmi_get_board_info(int device_id, struct dsmi_board_info_stru* pboard_info)
```

功能说明

获取单板信息，包括单板的board_id, pcb_id, bom_id、slot_id版本号；支持PCIe标卡（只支持slot_id）、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
pboard_info	输出	struct dsmi_board_info_stru *	<pre>struct dsmi_board_info_stru { unsigned int board_id; unsigned int pcb_id; unsigned int bom_id; unsinged int slot_id; 如果单板上有多个芯片，查询是哪个芯片。 };</pre> <p>说明 非底板的PCIE插槽 ID。</p>

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

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

1.20 dsmi_get_soc_sensor_info 接口原型

函数原型

int dsmi_get_soc_sensor_info (int device_id, int sensor_id,TAG_SENSOR_INFO *tsensor_info)

功能说明

获取SOC传感器信息；支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。

参数名	输入/输出	类型	描述
sensor_id	输入	int	指定传感器索引，具体定义如下： enum dmanager_tsensor_id { CLUSTER_TEMP_ID = 0, PERI_TEMP_ID = 1, AICORE0_TEMP_ID, AICORE1_TEMP_ID , AICORE_LIMIT_ID, AICORE_TOTAL_PER_ID, AICORE_ELIM_PER_ID, AICORE_BASE_FREQ_ID, NPU_DDR_FREQ_ID, THERMAL_THRESHOLD_ID, NTC_TEMP_ID, SOC_TEMP_ID, INVALID_ID, };
tsensor_info	输出	TAG_SENSOR_INFO*	类型定义如下： typedef union tag_sensor_info { unsigned char uchar; unsigned short ushort; unsigned int uint; signed int iint; signed char temp[2]; signed int ntc_tmp[4]; unsigned int data[SENSOR_DATA_MAX_LEN]; } TAG_SENSOR_INFO;

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
TAG_SENSOR_INFO sensor_info = {0};
ret = dsmi_get_soc_sensor_info(0, CLUSTER_TEMP_ID, &sensor_info);
if(ret != 0) {
//todo:记录日志
return ret;
}
...
```

1.21 dsmi_get_mini2mcu_heartbeat_status 接口原型

函数原型

```
int dsmi_get_mini2mcu_heartbeat_status(int device_id, unsigned char *status, unsigned int *disconn_cnt)
```

功能说明

获取mini对MCU的心跳状态和计数，状态为connect情况下，disconn_cnt值越小代表链路越稳定；

只支持PCIe标卡，且一块PCIe标卡只有两个mini有心跳功能。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
status	输出	unsigned char *	心跳状态： 0: disconnect 1: connect
disconn_cnt	输出	unsigned int *	心跳失联次数： 范围：0~9999

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
unsigned char tmp_status = 0;
unsigned int tmp_disconn_cnt = 0;
ret = dsmi_cmd_get_i2c_heartbeat_status(2, &tmp_status, &tmp_disconn_cnt);
if(ret != 0) {
    //todo:记录日志
    return ret;
}
...
```

1.22 dsmi_get_pmu_voltage 接口原型

函数原型

```
int dsmi_get_pmu_voltage(int device_id, unsigned char pmu_type, unsigned char channel,
unsigned int *volt_mv)
```

功能说明

查询PMU电压，支持开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，有效值范围：0~63，当前实际支持的设备号，通过 dsmi_list_device 接口获取。
pmu_type	输入	unsigned char	PMU电压类型 0: 主PMU VBUCK 1: 主PMU VOUT 2: 副PMU0 VBUCK 3: 副PMU1 VBUCK
channel	输入	unsigned char	PMU电压通道号，当前支持通道号：0~8、15、18
volt_mv	输出	unsigned int *	PMU电压：单位为毫伏。32位无符号int类型，小字节序。

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
unsigned int voltage_mv = 0;
unsigned char pmu_type = 0;
unsigned char channel = 0;

ret = dsmi_get_pmu_voltage(0, pmu_type, channel, &voltage_mv);
if(ret != 0) {
//todo: 记录日志
return ret;
}
...
```

2MAC 地址管理

关于本章

- 2.1 dsmi_get_mac_count接口原型
- 2.2 dsmi_get_mac_addr接口原型
- 2.3 dsmi_set_mac_addr接口原型

2.1 dsmi_get_mac_count 接口原型

函数原型

```
int dsmi_get_mac_count(int device_id, int* count)
```

功能说明

查询MAC地址数量；支持mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
count	输出	int *	查询出MAC数，取值范围：0~4。

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
int count = -1;
ret = dsmi_get_mac_count(0, &count);
if(ret != 0) {
    //todo:记录日志
    return ret;
}
...
```

2.2 dsmi_get_mac_addr 接口原型

函数原型

int dsmi_get_mac_addr (int device_id, int mac_id, char *pmac_addr)

功能说明

获取指定设备的MAC地址；支持mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
mac_id	输入	int	取值范围：0~3。
pmac_addr	输出	char *	输出6个字节的MAC地址。

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
int count = -1;
char mac_addr[6] = {0};
ret = dsmi_get_mac_count(0,&count);
...
for (i = 0; i < count; i++){
    ret = dsmi_get_mac_addr(0, i, mac_addr);
    if(ret != 0) {
        //todo: 记录日志
    }
    return ret;
}
...
}
```

2.3 dsmi_set_mac_addr 接口原型

函数原型

int dsmi_set_mac_addr(int device_id, int mac_id, char *pmac_addr)

功能说明

设置指定设备的MAC地址；支持mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
mac_id	输入	int	取值范围：0~3。
pmac_addr	输入	char *	设置6个字节的MAC地址。

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
int count = 1;
char mac_addr[6] = {0x52, 0x12, 0x36, 0x26, 0x82, 0x66};
ret = dsmi_set_mac_addr(0, count, &mac_addr);
if(ret != 0) {
    //todo: 记录日志
    return ret;
}
...
```

3 风扇管理

关于本章

- 3.1 [dsmi_get_fan_count接口原型](#)
- 3.2 [dsmi_get_fan_speed接口原型](#)
- 3.3 [dsmi_set_fan_speed接口原型](#)

3.1 dsmi_get_fan_count 接口原型

函数原型

```
int dsmi_get_fan_count(int device_id, int* count)
```

功能说明

获取MINI上小风扇数，大部分场景一个MINI芯片只有一个风扇，但可能存在支持多个风扇的情况，如果有多个小风扇，提供查询有几个风扇的接口；支持mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
count	输出	int *	查询MINI小风扇个数，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。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
int count = 0;
ret = dsmi_get_fan_count(0, &count);
if(ret != 0) {
//todo: 记录日志
return ret;
}
...
```

3.2 dsmi_get_fan_speed 接口原型

函数原型

```
int dsmi_get_fan_speed(int device_id, int fan_id, int *speed)
```

功能说明

查询指定风扇的转速，为风扇实际转速，单位RPM；只支持开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
fan_id	输入	int	如果有多个风扇，fan_id从1开始编号，大于等于1为指定fan_id的风扇转速。fan_id为0表示平均速度，如果有多个风扇，fan_id=0为几个风扇的平均速度。
speed	输出	int *	输出风扇转速值数组，由调用者申请。 调用成功后，该空间存储为风扇转速，单位为RPM，即转/分钟。 取值范围：0~(18000±10%)

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
int speed = 0;
int count = 0;
ret = dsmi_get_fan_count(0, &count);
...
for (i = 1; i <= count; i++){
    ret = dsmi_get_fan_speed(0, i, &speed);
    if(ret != 0) {
        //todo:记录日志
    }
    return ret;
}
...
}
```

3.3 dsmi_set_fan_speed 接口原型

函数原型

int dsmi_set_fan_speed(int device_id, int speed)

功能说明

设置指定风扇的转速，设置值为百分比；只支持开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
speed	输入	int	设置的风扇转速值，范围为25~100。 如果有多个小风扇，每个风扇都设置此转速。

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
ret = dsmi_set_fan_speed(0, 50);
if(ret != 0) {
//todo:记录日志
return ret;
}
```

4 配置管理

关于本章

[4.1 dsmi_config_ecc_enable接口原型](#)

[4.2 dsmi_get_ecc_enable接口原型](#)

[4.3 dsmi_config_p2p_enable接口原型](#)

[4.4 dsmi_get_p2p_enable接口原型](#)

[4.5 dsmi_get_system_time接口原型](#)

[4.6 dsmi_set_device_ip_address接口原型](#)

[4.7 dsmi_get_device_ip_address接口原型](#)

[4.8 dsmi_set_user_config接口原型](#)

[4.9 dsmi_get_user_config接口原型](#)

[4.10 dsmi_clear_user_config接口原型](#)

4.1 dsmi_config_ecc_enable 接口原型

函数原型

```
int dsmi_config_ecc_enable(int device_id, DSMI_DEVICE_TYPE device_type, int enable_flag)
```

功能说明

配置存储ECC的标记为使能或禁用；支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
device_type	输入	DSMI_DEVICE_TYPE	设备类型，目前支持DSMI_DEVICE_TYPE_DDR。 typedef enum { DSMI_DEVICE_TYPE_DDR, DSMI_DEVICE_TYPE_SRAM, DSMI_DEVICE_TYPE_HBM, DSMI_DEVICE_TYPE_GPU, DSMI_DEVICE_TYPE_NONE=0xff } DSMI_DEVICE_TYPE;
enable_flag	输入	int	TRUE：使能 FALSE：禁用

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
ret = dsmi_config_ecc_enable(0, DSMI_DEVICE_TYPE_SRAM, TRUE);
if(ret != 0) {
//todo: 记录日志
return ret;
}
ret = dsmi_config_ecc_enable(0, DSMI_DEVICE_TYPE_SRAM, FALSE);
if(ret != 0) {
//todo: 记录日志
return ret;
}
```


4.2 dsmi_get_ecc_enable 接口原型

函数原型

```
int dsmi_get_ecc_enable(int device_id, DSMI_DEVICE_TYPE device_type, int*  
enable_flag)
```

功能说明

查询存储ECC的使能标记；支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
device_type	输入	DSMI_DEVICE_TYPE	设备类型，目前只支持DSMI_DEVICE_TYPE_DDR。 typedef enum { DSMI_DEVICE_TYPE_DDR, DSMI_DEVICE_TYPE_SRAM, DSMI_DEVICE_TYPE_HBM, DSMI_DEVICE_TYPE_GPU, DSMI_DEVICE_TYPE_NONE=0xff } DSMI_DEVICE_TYPE;
enable_flag	输出	int *	TRUE：使能 FALSE：禁用

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
int enable_flag = 0;
ret = dsmi_get_ecc_enable(0, DSMI_DEVICE_TYPE_DDR, &enable_flag);
if(ret != 0) {
    //todo: 记录日志
    return ret;
}
...
```

4.3 dsmi_config_p2p_enable 接口原型

函数原型

int dsmi_config_p2p_enable(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；仅支持MDC单板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
enable_flag	输入	int	TRUE：使能 FALSE：禁用

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

仅MDC单板支持，其他单板直接返回不支持。

调用示例

```
int ret = 0;
ret = dsmi_config_p2p_enable(0, TRUE);
```

```
if(ret != 0) {
//todo: 记录日志
return ret;
}
...
```

4.4 dsmi_get_p2p_enable 接口原型

函数原型

```
int dsmi_get_p2p_enable(int device_id, int* enable_flag)
```

功能说明

查询Flash存储的P2P使能标记；支持PCIe标卡、mini模块、开发者板、MDC单板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
enable_flag	输出	int *	TRUE：使能 FALSE：禁用

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
int enable_flag = -1;
ret = dsmi_get_p2p_enable (0, &enable_flag);
if(ret != 0) {
//todo: 记录日志
return ret;
}
...
```

4.5 dsmi_get_system_time 接口原型

函数原型

`int dsmi_get_system_time(int device_id, unsigned int *ntime_stamp)`

功能说明

获取系统时间；支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
ntime_stamp	输出	unsigned int *	the number of seconds from 00:00:00, January 1,1970.

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
int time = 0;
ret = dsmi_get_system_time (0, &time);
if(ret != 0) {
//todo: 记录日志
return ret;
}
...
```

4.6 dsmi_set_device_ip_address 接口原型

函数原型

```
int dsmi_set_device_ip_address (int device_id, unsigned int ip_addr, unsigned int netmask)
```

功能说明

设置PCIE卡虚拟网口IPV4的IP地址和网络掩码；支持PCIE标卡。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过 dsmi_list_device接口获取。
ip_addr	输入	unsigned int	IPV4类型的虚拟网口的IP地址，为网络序，支持PCIE板卡上设置。
netmask	输入	unsigned int	IPV4类型的虚拟网口的网络掩码，为网络序，支持PCIE板卡上设置。

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
unsigned int ip_addr = 0X0;
unsigned int netmask = 0X0;

ip_addr = 0XBA01A8C0; /*192.168.1.186*/
netmask = 0X00FFFFFF; /*255.255.255.0*/
ret = dsmi_set_device_ip_address(0, ip_addr, netmask);
if(ret != 0){
    //todo:记录日志
```

```
return ret;
}
...
```

4.7 dsmi_get_device_ip_address 接口原型

函数原型

int dsmi_get_device_ip_address(int device_id, unsigned int *ip_addr, unsigned int *netmask)

功能说明

获取PCIE卡虚拟网口IPV4的IP地址和网络掩码；支持PCIE标卡。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
ip_addr	输出	unsigned int *	IPV4类型的虚拟网口的IP地址，为网络序，只支持PCIE板卡上查询。
netmask	输出	unsigned int *	IPV4类型的虚拟网口的网络掩码，为网络序，只支持PCIE板卡上查询。

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
unsigned int ip_addr = 0X0;
unsigned int netmask = 0X0;

ret = dsmi_get_device_ip_address(0, &ip_addr, &netmask);
```

```
if(ret != 0){  
    //todo:记录日志  
    return ret;  
}  
...
```

4.8 dsmi_set_user_config 接口原型

函数原型

int dsmi_set_user_config(int device_id, const char *config_name, unsigned int buf_size, unsigned char *buf)

功能说明

设置用户配置，支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过 dsmi_list_device接口获取。
config_name	输入	const char *	目前支持处理的配置项名称如下： <ul style="list-style-type: none">● "p2p_enable"● "ddr_ecc_enable"
buf_size	输入	unsigned int	buf长度，最大长度为1024 byte
buf	输入	unsigned char *	buf指针，指向配置项内容

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

```
#define BUF_SIZE 1
int ret = 0;
int device_id = 0;
char *config_name = "p2p_enable";
char buf[BUF_SIZE] = {0};
ret=dsmi_set_user_config(device_id, config_name, BUF_SIZE, buf);
if(ret != 0){
    //todo:记录日志
    return ret;
}
...
```

4.9 dsmi_get_user_config 接口原型

函数原型

```
int dsmi_get_user_config(int device_id, const char *config_name, unsigned int buf_size,
unsigned char *buf)
```

功能说明

获取用户配置，支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过 dsmi_list_device接口获取。
config_name	输入	const char *	目前支持处理的配置项名称如下： <ul style="list-style-type: none">● "p2p_enable"● "ddr_ecc_enable"
buf_size	输入	unsigned int	buf长度，最大长度为1024 byte
buf	输出	unsigned char *	buf指针，指向配置项内容

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

```
#define BUF_SIZE 1
int ret = 0;
int device_id = 0;
char *config_name = "p2p_enable";
char buf[BUF_SIZE] = {0};
ret=dsmi_get_user_config(device_id, config_name, BUF_SIZE, buf);
if(ret != 0){
//todo:记录日志
return ret;
}
...
```

4.10 dsmi_clear_user_config 接口原型

函数原型

```
int dsmi_clear_user_config(int device_id, const char *config_name)
```

功能说明

清除用户配置，支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过 dsmi_list_device接口获取。
config_name	输入	const char *	目前支持处理的配置项名称如下： <ul style="list-style-type: none"> ● "p2p_enable" ● "ddr_ecc_enable"

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
int device_id = 0;
char *config_name = "p2p_enable";
ret=dsmi_clear_user_config(device_id, config_name);
if(ret != 0){
//todo:记录日志
return ret;
}
...
```

5 软件升级

关于本章

- 5.1 dsmi_upgrade_start接口原型
- 5.2 dsmi_upgrade_get_state接口原型
- 5.3 dsmi_upgrade_get_component_static_version接口原型
- 5.4 dsmi_get_version接口原型
- 5.5 dsmi_get_component_list接口原型

5.1 dsmi_upgrade_start 接口原型

函数原型

```
int dsmi_upgrade_start(int device_id, DSMI_COMPONENT_TYPE component_type, char* file_name)
```

功能说明

启动指定D芯片的升级，指定升级固件类型、升级的文件名；支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。

参数名	输入/输出	类型	描述
component_type	输入	DSMI_COMPONENT_TYPE	固件类型，目前支持如下几种： typedef enum dsmi_component_type { DSMI_COMPONENT_TYPE_NVE, DSMI_COMPONENT_TYPE_XLOADER, DSMI_COMPONENT_TYPE_M3FW, DSMI_COMPONENT_TYPE_UEFI, DSMI_COMPONENT_TYPE_TEE, DSMI_COMPONENT_TYPE_KERNEL, DSMI_COMPONENT_TYPE_DTB, DSMI_COMPONENT_TYPE_ROOTFS, DSMI_COMPONENT_TYPE_NONE, UPGRADE_ALL_COMPONENT = 0xFFFFFFFF }DSMI_COMPONENT_TYPE;
file_name	输入	char *	固件文件名（包含路径）。

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

非宿主机环境不支持升级功能

调用示例

```
int ret = 0;
ret = dsmi_upgrade_start(0, DSMI_COMPONENT_TYPE_XLOADER , “./xloader.bin”);
if(ret != 0) {
//todo: 记录日志
return ret;
}
...
```

5.2 dsmi_upgrade_get_state 接口原型

函数原型

```
int dsmi_upgrade_get_state(int device_id, unsigned char *schedule, unsigned char *upgrade_status)
```

功能说明

查询固件升级状态及进度接口；支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
schedule	输出	unsigned char *	升级进度，0~100百分比。
upgrade_status	输出	unsigned char *	升级状态，目前支持如下几种： typedef enum { DSMI_UPGRADE_STATE_IDLE, DSMI_UPGRADE_STATE_UPDATING, DSMI_UPGRADE_STATE_NONSUPPORT, DSMI_UPGRADE_STATE_FAIL, DSMI_UPGRADE_STATE_NONE }DSMI_UPGRADE_STATE;

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
unsigned char schedule;
unsigned char upgrade_status;
ret = dsmi_upgrade_get_state(0, &schedule, & upgrade_status);
if(ret != 0) {
    //todo: 记录日志
    return ret;
}
```

5.3 dsmi_upgrade_get_component_static_version 接口原型

函数原型

```
int dsmi_upgrade_get_component_static_version(int device_id,
DSMI_COMPONENT_TYPE component_type, unsigned char* version_str, int *len)
```

功能说明

查询静态组件版本，查询FLASH上固件的版本；支持PCIe标卡、mini模块、开发板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
component_type	输入	DSMI_COMPONENT_TYPE	固件类型，目前支持如下几种： typedef enum dsmi_component_type { DSMI_COMPONENT_TYPE_NVE, DSMI_COMPONENT_TYPE_XLOADER, DSMI_COMPONENT_TYPE_M3FW, DSMI_COMPONENT_TYPE_UEFI, DSMI_COMPONENT_TYPE_TEE, DSMI_COMPONENT_TYPE_KERNEL, DSMI_COMPONENT_TYPE_DTB, DSMI_COMPONENT_TYPE_ROOTFS, DSMI_COMPONENT_TYPE_NONE, UPGRADE_ALL_COMPONENT = 0xFFFFFFFF }DSMI_COMPONENT_TYPE;
version_str	输出	unsigned char*	用户申请的空间，存放返回的固件版本号。

参数名	输入/输出	类型	描述
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。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
unsigned char version_str[64] = {0};
ret = dsmi_upgrade_get_component_static_version(0, DSMI_COMPONENT_TYPE_XLOADER, version_str, 64);
if(ret != 0) {
    //todo: 记录日志
    return ret;
}
...
```

5.4 dsmi_get_version 接口原型

函数原型

int dsmi_get_version (int device_id, char* version_str, int *len)

功能说明

查询系统版本；支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
version_str	输出	char *	返回系统版本。

参数名	输入/输出	类型	描述
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。

异常处理

无。

约束说明

无。

调用示例

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

5.5 dsmi_get_component_list 接口原型

函数原型

```
int dsmi_get_component_list (int device_id, unsigned int *component_num,
DSMI_COMPONENT_TYPE* component_table)
```

功能说明

获取可升级组件列表；支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。

参数名	输入/输出	类型	描述
component_num	输出	unsigned int *	返回组件数量。
component_table	输出	DSMI_COMPONENT_TYPE*	返回可升级组件列表，具体值含义如下： typedef enum dsmi_component_type { DSMI_COMPONENT_TYPE_NVE, DSMI_COMPONENT_TYPE_XLOADER, DSMI_COMPONENT_TYPE_M3FW, DSMI_COMPONENT_TYPE_UEFI, DSMI_COMPONENT_TYPE_TEE, DSMI_COMPONENT_TYPE_KERNEL, DSMI_COMPONENT_TYPE_DTB, DSMI_COMPONENT_TYPE_ROOTFS, DSMI_COMPONENT_TYPE_NONE, UPGRADE_ALL_COMPONENT = 0xFFFFFFFF }DSMI_COMPONENT_TYPE;

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
unsigned int component_num = 0;
DSMI_COMPONENT_TYPE component_table[8] = {0};
ret = dsmi_get_component_list(0, &component_num, &component_table[0]);
if(ret != 0 || component_num > 8) {
//todo: 记录日志
return ret;
}
```

```
}  
...
```

6 芯片复位启动

关于本章

- 6.1 dsmi_pre_reset_soc接口原型
- 6.2 dsmi_rescan_soc接口原型
- 6.3 dsmi_get_device_boot_status接口原型

6.1 dsmi_pre_reset_soc 接口原型

函数原型

```
int dsmi_pre_reset_soc (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：dsmi_pre_reset_soc -> reset -> dsmi_rescan_soc；支持PCIe标卡。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
ret = dsmi_pre_reset_soc (0);
if(ret != 0) {
//todo: 记录日志
return ret;
}
...
```

6.2 dsmi_rescan_soc 接口原型

函数原型

int dsmi_rescan_soc (int device_id)

功能说明

对指定芯片重新扫描；支持PCIe标卡。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
ret = dsmi_rescan_soc(0);
if(ret != 0) {
//todo: 记录日志
return ret;
}
...
```

6.3 dsmi_get_device_boot_status 接口原型

函数原型

int dsmi_get_device_boot_status(int device_id, enum dsmi_boot_status *boot_status)

功能说明

获取芯片系统的启动状态；支持PCIe标卡、mini模块、开发者板。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	int	指定设备号，取值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
boot_status	输出	enum dsmi_boot_status *	芯片启动状态： enum dsmi_boot_status { DSMI_BOOT_STATUS_UNINIT = 0, /*未初始化*/ DSMI_BOOT_STATUS_BIOS, /* BIOS启动中*/ DSMI_BOOT_STATUS_OS, /* OS启动中*/ DSMI_BOOT_STATUS_FINISH /*启动完成*/ };

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
enum dsmi_boot_status boot_status = 0;
ret = dsmi_get_device_boot_status (0, &boot_status);
if(ret != 0) {
//todo: 记录日志
return ret;
}
...
```