XR806 Image 中间件

开发指南

版本号：1.0

发布时间：2020-11-12

版本历史

版本	日期	责任人	版本描述
1.0	2020-11-12	AWA 1029	创建文档。

目录

版本历史 i
目录 ii
表格目录 iv
1 前言 1
1.1 文档简介 1
1.2 目标读者 1
1.3 适用范围 1
1.4 文档约定 1
1.4.1 标志说明 1
1.4.2 地址与数据描述方法约定 1
1.4.3 数值单位约定 2
2 概述 3
2.1 背景说明 3
2.2 规格特性 3
2.3 文件位置 3
3 技术说明 5
3.1 Image 布局说明 5
3.1.1 Image 区域 5
3.1.1.1 不启用 OTA 5
3.1.1.2 启用 OTA 6
3.1.2 Image sequence 6
3.2 Section 说明 6
3.2.1 Section ID 7
3.2.2 Segment 7
3.3 OTA 参数获取 8
4 应用说明 9
4.1 应用简述 9
4.2 配置说明 9
4.3 接口说明 10
4.3.1 初始化接口 10
4.3.1.1 image_init 10
4.3.1.2 image_deinit 10

4.3.2 读写接口 11
4.3.2.1 image_read 11
4.3.2.2 image_write 11
4.3.3 参数获取和设置接口 12
4.3.3.1 image_get_ota_param 12
4.3.3.2 image_set_running_seq 12
4.3.3.3 image_get_running_seq 12
4.3.3.4 image_get_section_addr 12
4.3.3.5 image_get_checksum 13
4.3.3.6 image_check_header 13
4.3.3.7 image_check_section 14
4.3.3.8 image_check_sections 14
4.3.3.9 image_get_cfg 14
4.3.3.10 image_set_cfg 15
5 示例说明 16
5.1 示例简介 16
5.1.1 准备工作 16
5.1.2 操作步骤 16
5.2 关键实现 16
5.3 效果展示 17

表格目录

表 2-1 image 功能特性 3
表 2-2 image 中间件的文件位置 3
表 2-3 image 中间件的文件说明 3
表 4-1 XR806 image 模块工程配置说明 9
表 4-2 image 中间件模块接口简介 10
表 4-3 image_init()接口函数说明 10
表 4-4 image_deinit 接口函数说明 10
表 4-5 image_read 接口函数说明 11
表 4-6 image_write 接口函数说明 11
表 4-7 image_get_ota_param 接口函数说明 12
表 4-8 image_set_running_seq 接口函数说明 12
表 4-9 image_get_running_seq 接口函数说明 12
表 4-10 image_get_section_addr 接口函数说明 12
表 4-11 image_get_checksum 接口函数说明 13
表 4-12 image_check_header 接口函数说明 13
表 4-13 image_check_section 接口函数说明 14
表 4-14 image_check_sections 接口函数说明 14
表 4-15 image_get_cfg 接口函数说明 14
表 4-16 image_set_cfg 接口函数说明 15

1 前言

1.1 文档简介

此文档用以解释 Image 模块相关的概念并介绍接口的使用，帮助开发者了解如何使用 Image 模块管理和操作固件。

1.2 目标读者

XR806 SDK 用户。

1.3 适用范围

此文档适用于 XR806 SDK，支持 XR806 系列芯片产品。

1.4 文档约定

1.4.1 标志说明

本文档采用各种醒目的标志来表示在操作过程中应该特别注意的地方，这些标志的含义如下：

标识	说明
 警告	该标志后的说明应给予格外关注，如果不遵守，可能会导致人员受伤或死亡。
 注意	提醒操作中应注意的事项。不当的操作可能会损坏器件，影响可靠性、降低性能等。
 说明	为准确理解文中指令、正确实施操作而提供的补充或强调信息。
 窍门	一些容易忽视的小功能、技巧。了解这些功能或技巧能帮助解决特定问题或者节省操作时间。

1.4.2 地址与数据描述方法约定

本文档在描述地址、数据时遵循如下约定：

符号	例子	说明
0x	0x0200，0x79	地址或数据以 16 进制表示。
0b	0b010，0b00 000 111	数据采用二进制表示(寄存器描述除外)。
X	00X，XX1	数据描述中，X 代表 0 或 1。 例如，00X 代表 000 或 001；XX1 代表 001，011，101 或 111。

1.4.3 数值单位约定

本文档在描述数据容量（如 NAND 容量）时，单位词头代表的是 1024 的倍数；描述频率、数据速率等时则代表的是 1000 的倍数。具体如下：

类型	符号	对应数值
数据容量（如 NAND 容量）	1 K	1024
	1 M	1 048 576
	1 G	1 073 741 824
频率，数据速率等	1 k	1000
	1 M	1 000 000
	1 G	1 000 000 000

2 概述

2.1 背景说明

XR806 SDK 的 image 模块主要用于管理和操作固件。系统启动过程中对固件的加载和检查，以及 OTA 升级过程中对固件的更新和验证，都用到 image 模块提供的功能。

为更好理解 image 模块的相关概念，建议先通过文档《XR806 Flash 布局方案 开发指南》了解固件打包和布局相关内容。此外，由于 OTA 功能在一定程度上依赖 image 模块，因此本文档内容也有助于了解系统的 OTA 功能。

2.2 规格特性

Image 支持以下功能特性。

表 2-1 image 功能特性

规格类型	支持规格	规格描述	备注
image 区域	支持两个 image 区域	支持使用两个 image 区域，可选择其中一个区域运行或进行 OTA 升级	
	支持获取 bin 文件数据	支持获取每个 bin 文件的 header、body 和 tailer	
image 校验	支持 image 完整性校验	支持 image 完整性校验，校验失败则不予运行	

2.3 文件位置

以 SDK 包为根目录，本中间件涉及到的主要文件位置如下。

表 2-2 image 中间件的文件位置

组件名	文件分类	文件位置
image	源码文件	./src/image
	头文件	./include/sys

关键文件说明如下。

表 2-3 image 中间件的文件说明

文件名	文件说明
image.c	image 模块的主函数

说明

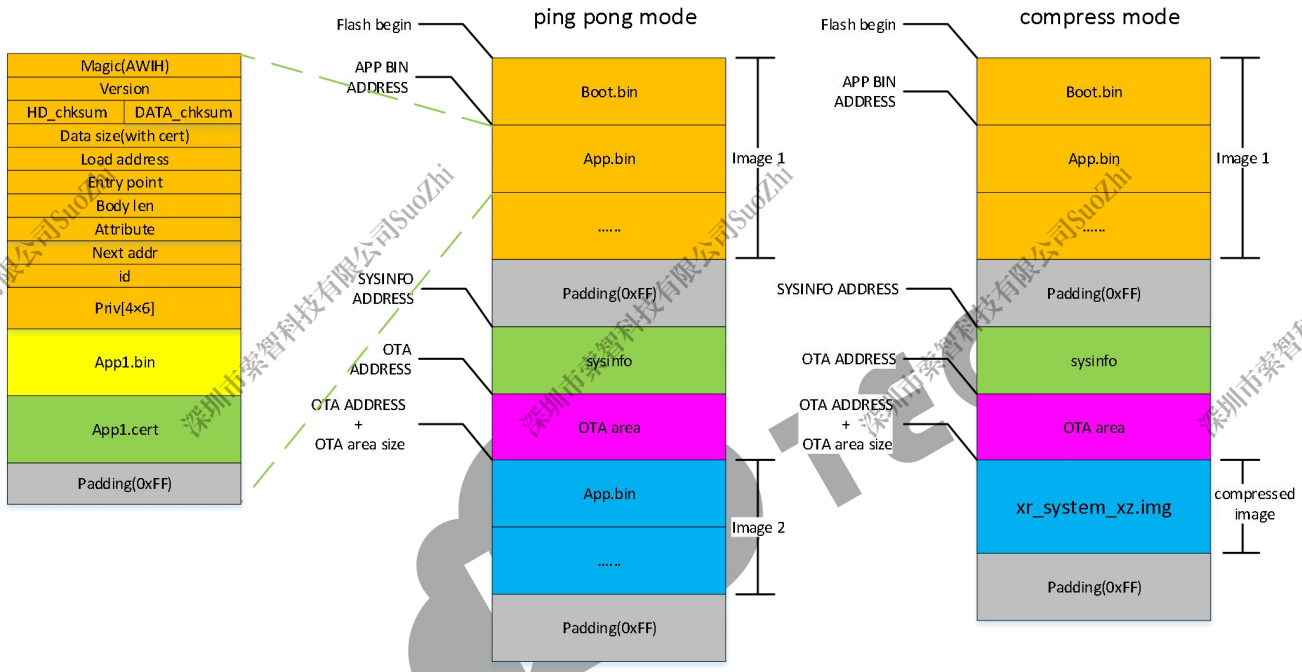
XR806 SDK 可在以下 GitHub 仓库获取：https://github.com/XradioTech/xr806_sdk.git

3 技术说明

3.1 Image 布局说明

XR806 SDK 中的 image 区域区域布局如下图所示。

图 3-1 image 布局示例



3.1.1 Image 区域

Image 区域是指在 Flash 上划分出来用于存放固件的区域。开发者如果通过 FDCM 模块或其他方式改写或擦除 image 区域，可能导致固件损坏系统无法启动。

3.1.1.1 不启用OTA

在不启用 OTA 功能时，只需要在 Flash 上指定一个 image 区域。

工程目录下 prj_config.h 文件中配置 image 区域的位置。

```
/* image flash ID */
#define PRJCONF_IMG_FLASH                   0

/* image start address, including bootloader */
#define PRJCONF_IMG_ADDR                    0x00000000
```

宏 PRJCONF_IMG_FLASH 为 Flash 设备号（具体定义可参考 Flash 驱动），用来指定 image 区域所在的具体 Flash 设备，宏 PRJCONF_IMG_ADDR 表示 image 区域（包含 boot.bin）的起始地址。

注意

Image 区域固定从 0 地址开始，所以宏 PRJCONF_IMG_ADDR 不允许被修改。

Image.cfg 文件中配置 image 区域的大小

```
"image"     : {"max_size": "1020K"},
```

由上面三个配置，就可以确定一个 image 区域。

3.1.1.2 启用OTA

在启用 OTA 功能时，需要在 Flash 上指定两个 image 区域。其中第一个 image 区域仍通过上述方式确定。第二个 image 区域的 Flash 设备号、起始地址和大小通过文件 image.cfg 配置，配置示例如下：

```
"OTA"       : {"flash": "1", "addr": "1024K", "size": "4K"},
#if (__CONFIG_OTA_POLICY == 0x01)
    "image"     : {"max_size": "1020K", "xz_max_size": "600K"},
#else
    "image"     : {"max_size": "1020K"},
#endif
```

如果第二个 image 区域与第一个 image 区域在相同的 Flash 设备，则配置可简化为：

```
"OTA"       : {"addr": "1024K", "size": "4K"},
#if (__CONFIG_OTA_POLICY == 0x01)
    "image"     : {"max_size": "1020K", "xz_max_size": "600K"},
#else
    "image"     : {"max_size": "1020K"},
#endif
```

其中”addr”表示 OTA 区域的起始地址，”size”表示 OTA 区域的大小，那么第二个 image 区域（不包含 boot.bin）的起始地址就是”addr + size”。如果采用 OTA 压缩升级，则第二个 Image 区域的大小由”xz_max_size”决定，否则由”max_size”决定。

3.1.2 Image sequence

支持 OTA 功能时，要求在 Flash 上有两个 image 区域来存放固件。Image sequence 用于指定不同 image 区域的固件。Image sequence 在代码中定义如下：

```
typedef uint8_t image_seq_t;
```

3.2 Section 说明

从 image.cfg 中可以看到，一个 image 由多个 Section 组成。一个 Section 包含了头部和 bin 文件，也可能在尾部包含证书。

3.2.1 Section ID

Section ID 是每个 Section 的唯一标识。Image 模块根据 Section ID 找到对应的 Section 以及 Section 内部的 bin。打包时固件中的 Section ID 来自固件配置文件 image.cfg，代码中的 Section ID 定义在文件 image.h 中（示例如下），应保证这两处定义的一致性。

```
#define IMAGE_BOOT_ID            (0xA5FF5A00)
#define IMAGE_APP_ID             (0xA5FE5A01)
#define IMAGE_APP_XIP_ID         (0xA5FD5A02)
#define IMAGE_NET_ID             (0xA5FC5A03)
#define IMAGE_NET_AP_ID          (0xA5FB5A04)
#define IMAGE_WLAN_BL_ID         (0xA5FA5A05)
#define IMAGE_WLAN_FW_ID         (0xA5F95A06)
#define IMAGE_WLAN_SDD_ID        (0xA5F85A07)
#define IMAGE_APP_EXT_ID         (0xA5F75A08) /* for XR32 only */
#define IMAGE_APP_PSRAM_ID       (0xA5F65A09)
```

3.2.2 Segment

打包生成的固件除了包含 bin 文件本身以外，在每个 bin 文件前还增加了一个长 64 个字节的头部，每个 bin 文件的尾部也可以包含证书等信息。Segment 用来指定 Section 中的头部（HEADER）、bin 文件本身（BODY）或是尾部（TAILER）。Segment 在文件 image.h 中的定义如下：

```
typedef enum image_segment {
    IMAGE_SEG_HEADER    = 0,
    IMAGE_SEG_BODY = 1,
    IMAGE_SEG_TAILER    = 2,
} image_seg_t;
```

头部（HEADER）64 个字节的结构体定义如下：

```
/**
 * @brief Section header definition (64 Bytes)
 */
typedef struct section_header {
    uint32_t magic_number;    /* magic number */
    uint32_t version;         /* version */
    uint16_t header_chksum; /* header checksum */
    uint16_t data_chksum;     /* data checksum */
    uint32_t data_size;       /* data size(with cert) */
    uint32_t load_addr;       /* load address */
    uint32_t entry;           /* entry point */
    uint32_t body_len;        /* body length */
    uint32_t attribute;       /* attribute */
    uint32_t next_addr;       /* next section address */
    uint32_t id;              /* section ID */
    uint32_t priv[6];         /* private data */
```

```
} section_header_t;
```

3.3 OTA 参数获取

OTA 功能的初始化参数（定义如下）可从 image 模块提供的接口获得。参数主要包括两个 image 区域的位置、大小以及 Bootloader 区域大小等信息。

```
typedef struct image_ota_param {
    uint32_t  ota_flash : 8;   /* flash ID of OTA area */
    uint32_t  ota_size    : 24;    /* size of OTA area */
    uint32_t  ota_addr;              /* start addr of OTA area */
    uint16_t  img_max_size;       /* image max size (excluding bootloader, the unit is K) */
    uint16_t  img_xz_max_size; /* compressed image max size (the unit is K)*/
    uint32_t  bl_size;               /* bootloader size */
    image_seq_t running_seq;   /* running image sequence */
    uint8_t    flash[IMAGE_SEQ_NUM];      /* flash ID which the image on */
    uint32_t  addr[IMAGE_SEQ_NUM];      /* image start addr, excluding bootloader */
} image_ota_param_t;
```

4 应用说明

4.1 应用简述

Image 中间件模块已经内嵌到 XR806 SDK，应用步骤如下：

1. 确认/检查所用 Flash 设备是否能正常运行（Flash 模块设备配置可参考《XR806 Flash 驱动开发指南》文档）。

4.2 配置说明

Image 模块的启用需要进行相关工程配置，如果使能 OTA 功能，则需要进行 OTA 相关配置。具体配置项如下表所示。

表 4-1 XR806 image 模块工程配置说明

配置项	配置说明
Image 区域	设置说明： 配置 image 区域的位置。 设置位置： 工程目录下 prj_config.h 文件 设置方式： 1、配置宏 PRJCONF_IMG_FLASH 为 Flash 设备号（具体定义可参考 Flash 驱动）； 2、配置宏 PRJCONF_IMG_ADDR 为 image 区域的起始地址
OTA 地址和大小	设置说明： 此项配置用于定义 OTA 区域的地址和大小。 设置位置： 工程使用的 image.cfg 文件 设置方式： 1、修改 image.cfg 文件中“OTA”的“addr”和“size”字段即可，注意需要 Flash 可擦除块对齐（比如 4K 对齐）。
Image 最大值	设置说明： 此项配置用于定义编译生成的 image 最大值，包括压缩模式的 image 最大值。 设置方式： 1、修改 image.cfg 文件中“image”的“max_size”和“xz_max_size”字段即可。

4.3 接口说明

XR806 image 中间件模块为用户提供 3 套操作接口，位于 SDK/include/image/image.h 文件中，各接口的简要说明如下表所示。

表 4-2 image 中间件模块接口简介

接口名	简要说明
初始化接口	用于 image 模块的初始化和反初始化
读写接口	用于读写 image 和 section 的数据内容
参数获取和设置接口	用于获取和设置需要的参数

上述接口均于 SDK/src/image/image.c 文件中实现，XR806 image 中间件模块接口的详细说明如下。

4.3.1 初始化接口

4.3.1.1 image_init

表 4-3 image_init()接口函数说明

信息项	说明
原型	int image_init(uint32_t flash, uint32_t addr, uint32_t max_size);
功能	初始化 image 模块
参数	uint32_t flash 含义解释：flash 设备号 uint32_t addr 含义解释：flash 区域的起始地址 uint32_t max_size 含义解释：flash 区域的大小
返回值	0：成功 -1：失败

4.3.1.2 image_deinit

表 4-4 image_deinit 接口函数说明

信息项	说明
原型	void image_deinit(void);
功能	反初始化 image 模块
参数	无
返回值	无

4.3.2 读写接口

4.3.2.1 image_read

表 4-5 image_read 接口函数说明

信息项	说明
原型	uint32_t image_read(uint32_t id, image_seg_t seg, uint32_t offset, void *buf, uint32_t size);
功能	读取已加载生效固件中指定区域的数据
参数	uint32_t id 含义解释：待读取的 section 的 id image_seg_t seg 含义解释：segment 类型 uint32_t offset 含义解释：表示所读区域起始地址相对于指定 Segment 起始地址的偏移 void *buf 含义解释：读取数据存放 Buffer 的指针 uint32_t size 含义解释：所读数据的长度
返回值	所读数据的长度

4.3.2.2 image_write

表 4-6 image_write 接口函数说明

信息项	说明
原型	uint32_t image_write(uint32_t id, image_seg_t seg, uint32_t offset, void *buf, uint32_t size);
功能	向已加载生效固件中指定区域写入数据
参数	uint32_t id 含义解释：待写入的 section 的 id image_seg_t seg 含义解释：segment 类型 uint32_t offset 含义解释：表示所写区域起始地址相对于指定 Segment 起始地址的偏移 void *buf 含义解释：写入数据存放 Buffer 的指针 uint32_t size 含义解释：所写数据的长度
返回值	写入数据的长度

4.3.3 参数获取和设置接口

4.3.3.1 image_get_ota_param

表 4-7 image_get_ota_param 接口函数说明

信息项	说明
原型	const image_ota_param_t *image_get_ota_param(void);
功能	获取 OTA 功能初始化的参数
参数	无
返回值	OTA 参数结构体的指针

4.3.3.2 image_set_running_seq

表 4-8 image_set_running_seq 接口函数说明

信息项	说明
原型	void image_set_running_seq(image_seq_t seq);
功能	设置此次系统能够启动将加载的 Image sequence
参数	image_seq_t seq 含义解释：加载固件的 Image sequence
返回值	无

4.3.3.3 image_get_running_seq

表 4-9 image_get_running_seq 接口函数说明

信息项	说明
原型	image_seq_t image_get_running_seq(void);
功能	获取加载固件的 image sequence
参数	无
返回值	加载固件的 image sequence

4.3.3.4 image_get_section_addr

表 4-10 image_get_section_addr 接口函数说明

信息项	说明
原型	uint32_t image_get_section_addr(uint32_t id);
功能	获取已加载生效固件中指定 Section 的 HEADER 在 Flash 中的起始地址
参数	uint32_t id 含义解释：指定的 Section ID

信息项	说明
返回值	返回指定 section 的起始地址

4.3.3.5 image_get_checksum

表 4-11 image_get_checksum 接口函数说明

信息项	说明
原型	uint16_t image_get_checksum(void *buf, uint32_t len);
功能	计算 Buffer 中指定长度数据的 16-bit 累加和
参数	void *buf 含义解释：待计算的 Buffer 的指针 uint32_t len 含义解释：计算的长度
返回值	返回 16-bit 累加和计算结果

4.3.3.6 image_check_header

表 4-12 image_check_header 接口函数说明

信息项	说明
原型	image_val_t image_check_data(section_header_t *sh, void *body, uint32_t body_len, void *tailer, uint32_t tailer_len);
功能	校验 BODY 加 TAILER 部分的 16-bit 累加和
参数	section_header_t *sh 含义解释：HEADER 结构体的指针 void *body 含义解释：BODY 部分数据的 Buffer 指针 uint32_t body_len 含义解释：BODY 部分数据的长度 void *tailer 含义解释：TAILER 部分数据的 Buffer 指针 uint32_t tailer_len 含义解释：TAILER 部分数据的长度
返回值	返回校验结果

4.3.3.7 image_check_section

表 4-13 image_check_section 接口函数说明

信息项	说明
原型	image_val_t image_check_section(image_seq_t seq, uint32_t id);
功能	校验 Flash 上指定 image 区域固件的指定 Section 的 HEADER、BODY 和 TAILER 部分的 16-bit 累加和是否正确
参数	image_seq_t seq 含义解释：指定校验的 image 区域的 image sequence uint32_t id 含义解释：section id
返回值	返回校验结果

4.3.3.8 image_check_sections

表 4-14 image_check_sections 接口函数说明

信息项	说明
原型	image_val_t image_check_sections(image_seq_t seq);
功能	校验 Flash 上指定 image 区域固件的所有 Section 的 HEADER、BODY 和 TAILER 部分的 16-bit 累加和是否正确
参数	image_seq_t seq 含义解释：指定校验的 image 区域的 image sequence
返回值	返回校验结果

4.3.3.9 image_get_cfg

表 4-15 image_get_cfg 接口函数说明

信息项	说明
原型	int image_get_cfg(image_cfg_t *cfg);
功能	获取 image 配置
参数	image_cfg_t *cfg 含义解释：存储 image 配置的 buffer 指针
返回值	0：成功 -1：失败

4.3.3.10 image_set_cfg

表 4-16 image_set_cfg 接口函数说明

信息项	说明
原型	int image_set_cfg(image_cfg_t *cfg);
功能	设置 image 配置
参数	image_cfg_t *cfg 含义解释：待写入的 image 配置
返回值	0：成功 -1：失败

5 示例说明

Image 接口用于获取 image 相关参数以及加载 bin 文件运行，下面以 bootloader 中调用 image 模块接口为例，简要说明 image 模块接口的使用。

5.1 示例简介

Image 示例在 hello_demo 示例工程中有实现，位于 XR806 SDK 的/project/demo/hello_demo 目录，以下此示例工程简称为 hello_demo 示例工程。

说明

XR806 SDK 可在以下 GitHub 仓库获取：https://github.com/XradioTech/xr806_sdk.git

5.1.1 准备工作

hello_demo 示例工程的硬件准备如下：

1. 评估板：运行示例工程代码。
2. 串口线：连接评估板的 Uart0 插针，用于 console 控制台的输入输出。
3. PC 机：用于镜像烧录和 console 控制的输入输出。

hello_demo 示例工程的软件准备，包括烧写工具、代码编译和烧写操作，请参见《XR806_SDK_快速入门指南》。

5.1.2 操作步骤

Hello_demo 示例工程完成烧写后，复位即可，示例代码自动运行，image 模块自动初始化，自动加载 bin 文件运行。

5.2 关键实现

系统复位启动时，bootloader 自动初始化 image 模块，加载 bin 文件到 ram 中运行，流程简单介绍如下：

1.初始化 image 模块

```
image_init(PRJCONF_IMG_FLASH, PRJCONF_IMG_ADDR, 0);
```

2.获取 OTA 功能初始化的参数

```
const image_ota_param_t *iop = image_get_ota_param();
```

3.根据 OTA 功能初始化参数判断是否支持 OTA 功能，并确定加载哪个 image 区域的固件后，设置 image sequence

```
image_set_running_seq(*image_seq);
```

4.从 Flash 中读取 IMAGE_APP_ID 对应 Section 的 HEADER，并校验 HEADER 的 16-bit 累加和

```
image_read(IMAGE_APP_ID, IMAGE_SEG_HEADER, 0, &sh, IMAGE_HEADER_SIZE);
image_check_header(&sh);
```

5.从 Flash 中读取 IMAGE_APP_ID 对应 Section 的 BODY（即 bin 文件数据），并校验 BODY 的 16-bit 累加和

```
image_read(IMAGE_APP_ID, IMAGE_SEG_BODY, 0, (void *)sh.load_addr, sh.data_size);
image_check_data(&sh, (void *)sh.load_addr, sh.data_size, NULL, 0)
```

6. 反初始化 image 模块

```
image_deinit();
```

5.3 效果展示

在系统启动后，正常运行应用代码，即可认为 image 模块正常使用。

著作权声明

商标声明

免责声明