XR871 Image Developer Guide

Revision 1.0

Nov 3, 2017

Copyright @2017 Xradio Technology Co., Ltd. All Rights Reserved

Declaration

THIS DOCUMENTATION IS THE ORIGINAL WORK AND COPYRIGHTED PROPERTY OF XRADIO TECHNOLOGY ("XRADIO"). REPRODUCTION IN WHOLE OR IN PART MUST OBTAIN THE WRITTEN APPROVAL OF XRADIO AND GIVE CLEAR ACKNOWLEDGEMENT TO THE COPYRIGHT OWNER.

THE INFORMATION FURNISHED BY XRADIO IS BELIEVED TO BE ACCURATE AND RELIABLE. XRADIO RESERVES THE RIGHT TO MAKE CHANGES IN CIRCUIT DESIGN AND/OR SPECIFICATIONS AT ANY TIME WITHOUT NOTICE. XRADIO DOES NOT ASSUME ANY RESPONSIBILITY AND LIABILITY FOR ITS USE. NOR FOR ANY INFRINGEMENTS OF PATENTS OR OTHER RIGHTS OF THE THIRD PARTIES WHICH MAY RESULT FROM ITS USE. NO LICENSE IS GRANTED BY IMPLICATION OR OTHERWISE UNDER ANY PATENT OR PATENT RIGHTS OF XRADIO. THIS DATASHEET NEITHER STATES NOR IMPLIES WARRANTY OF ANY KIND, INCLUDING FITNESS FOR ANY PARTICULAR APPLICATION.

THIRD PARTY LICENCES MAY BE REQUIRED TO IMPLEMENT THE SOLUTION/PRODUCT. CUSTOMERS SHALL BE SOLELY RESPONSIBLE TO OBTAIN ALL APPROPRIATELY REQUIRED THIRD PARTY LICENCES. XRADIO SHALL NOT BE LIABLE FOR ANY LICENCE FEE OR ROYALTY DUE IN RESPECT OF ANY REQUIRED THIRD PARTY LICENCE. XRADIO SHALL HAVE NO WARRANTY, INDEMNITY OR OTHER OBLIGATIONS WITH RESPECT TO MATTERS COVERED UNDER ANY REQUIRED THIRD PARTY LICENCE.

Revision History

Version	Data	Summary of Changes
1.0	2017-11-3	Initial Version

Table 1-1 Revision History

 Copyright @2017 Xradio Technology Co., Ltd. All Rights Reserved

Contents

 Copyright @2017 Xradio Technology Co., Ltd. All Rights Reserved

Tables

 Copyright @2017 Xradio Technology Co., Ltd. All Rights Reserved

1 概述

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。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。

为更好理解 Image 模块的相关概念，建议先通过文档《XR871 Memory Layout Developer Guide》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。

1.1 相关概念和定义

1.1.1 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。

在不考虑 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 文件中可配置 Image 区域的位置和大小。

```
#define PRJCONF_IMG_FLASH               (0)
#define PRJCONF_IMG_ADDR                (0x00000000)
#define PRJCONF_IMG_SIZE                ((1 << 20) - (4 << 10))
```

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 区域的起始地址，宏 PRJCONF_IMG_SIZE 表示 Image 区域的大小。由以上三个宏就可以确定一个 Image 区域。

在考虑 OTA 功能时，需要在 Flash 上指定两个 Image 区域。其中第一个 Image 区域仍通过上述三个宏确定。第二个 Image 区域大小与第一个 Image 区域大小相同，同样是由宏 PRJCONF_IMG_SIZE 确定。第二个 Image 区域的 Flash 设备号和起始地址通过文件 image.cfg 配置，配置示例如下：

```
"OTA"     :{"flash": "1", "addr": "0x00100000"},
```

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": "0x00100000"},
```

1.1.2 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 enum image_sequence {
    IMAGE_SEQ_1ST    = 0,
    IMAGE_SEQ_2ND    = 1,
```

 Copyright @2017 Xradio Technology Co., Ltd. All Rights Reserved

```
    IMAGE_SEQ_NUM    = 2,
} image_seq_t;
```

1.1.3 Section ID

从文档《XR871 Memory Layout Developer Guide》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。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)
```

1.1.4 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;
```

1.1.5 Image validity

在 Section HEADER 中包含了 HEADER 部分的累加和，以及 BODY 加 TAILER 部分的累加和。固件累加和校验的结果用 Image validity 表示，在文件 image.h 中定义如下：

```
typedef enum image_validity {
    IMAGE_INVALID    = 0,
    IMAGE_VALID      = 1,
} image_val_t;
```

1.2 Image 与 OTA

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  flash[IMAGE_SEQ_NUM];
    uint32_t  addr[IMAGE_SEQ_NUM];
```

 Copyright @2017 Xradio Technology Co., Ltd. All Rights Reserved

```
    uint32_t  image_size;
    uint32_t  boot_size;
} image_ota_param_t;
```

 Copyright @2017 Xradio Technology Co., Ltd. All Rights Reserved

2 使用说明

2.1 代码位置

相关代码请参考：

sdk-code/include/sys/image.h

sdk-code/src/image/image.c

2.2 接口说明

下面对 Image 模块提供的接口进行简要说明。

1. 初始化 Image 模块，在系统启动时调用。输入参数 flash、addr 和 size 表示第一个 Image 区域的 Flash 设备号、在 Flash 上的起始地址和区域大小。

```
void image_init(uint32_t flash, uint32_t addr, uint32_t size);
```

2. 反初始化 Image 模块。

```
void image_deinit(void);
```

3. 获取 OTA 功能初始化的参数，该接口在 Image 模块初始化之后才能调用。输入参数 param 是 OTA 功能初始化参数结构体的指针。

```
void image_get_ota_param(image_ota_param_t *param);
```

4. 设置此次系统启动将加载固件的 Image sequence。用于 OTA 功能指定此次启动用哪个 Image 区域上的固件。输入参数 seq 表示所选择固件的 Image sequence。

```
void image_set_running_seq(image_seq_t seq);
```

5. 获取此次系统启动已加载生效固件的 Image sequence。输入参数 seq 是用于获取固件 Image sequence 的指针。

```
void image_get_running_seq(image_seq_t *seq);
```

6. 获取已加载生效固件中指定 Section 的 HEADER 在 Flash 中的起始地址。输入参数 id 为指定的 Section ID。执行成功，返回合法地址；执行失败，返回无效地址 IMAGE_INVALID_ADDR。

```
uint32_t image_get_section_addr(uint32_t id);
```

7. 读取已加载生效固件中指定区域的数据。输入参数 id 和 seg 用于指定具体 Section 和 Segment，输入参数 offset 表示所读区域起始地址相对于指定 Segment 起始地址的偏移，输入参数 buf 和 size 分别表示读取数据存放 Buffer 的指针和所读数据的长度。返回所读数据的长度。

```
uint32_t image_read(uint32_t id, image_seg_t seg, uint32_t offset, void *buf, uint32_t size);
```

8. 向已加载生效固件中指定区域写入数据。输入参数 id 和 seg 用于指定具体 Section 和 Segment，输入参数 offset 表示所写区域起始地址相对于指定 Segment 起始地址的偏移，输入参数 buf 和 size 分别表示写入数据存放 Buffer 的指针和所写数据的长度。返回写入数据的长度。

```
uint32_t image_write(uint32_t id, image_seg_t seg, uint32_t offset, void *buf, uint32_t size);
```

 Copyright @2017 Xradio Technology Co., Ltd. All Rights Reserved

9. 计算 Buffer 中指定长度数据的 16-bit 累加和。输入参数 buf 为 Buffer 的指针，输入参数 len 表示 Buffer 中计算 16-bit 累加和的数据长度。返回 16-bit 累加和计算结果。

```
uint16_t image_get_checksum(void *buf, uint32_t len);
```

10. 校验 HEADER 部分的 16-bit 累加和。输入参数是 HEADER 结构体的指针。16-bit 累加和校验通过，返回 IMAGE_VALID，否则返回 IMAGE_INVALID。

```
image_val_t image_check_header(section_header_t *sh);
```

11. 校验 BODY 加 TAILER 部分的 16-bit 累加和。输入参数是 HEADER 结构体的指针（HEADER 包含进行比对的 16-bit 累加和）。输入参数 body 和 body_len 分别表示 BODY 部分数据的 Buffer 指针和长度。输入参数 tailer 和 tailer_len 分别表示 TAILER 部分数据的 Buffer 指针和长度。16-bit 累加和校验通过，返回 IMAGE_VALID，否则返回 IMAGE_INVALID。

```
image_val_t image_check_data(section_header_t *sh, void *body, uint32_t body_len,
                             void *tailer, uint32_t tailer_len);
```

12. 校验 Flash 上指定 Image 区域（OTA 功能有两个放固件的 Image 区域）固件的指定 Section 的 HEADER、BODY 和 TAILER 部分的 16-bit 累加和。输入参数 seq 和 id 指定 Image 区域和 Section。16-bit 累加和校验通过，返回 IMAGE_VALID，否则返回 IMAGE_INVALID。

```
image_val_t image_check_section(image_seq_t seq, uint32_t id);
```

13. 校验 Flash 上指定 Image 区域（OTA 功能有两个放固件的 Image 区域）固件的所有 Section 的 HEADER、BODY 和 TAILER 部分的 16-bit 累加和。输入参数 seq 指定 Image 区域。16-bit 累加和校验通过，返回 IMAGE_VALID，否则返回 IMAGE_INVALID。

```
image_val_t image_check_sections(image_seq_t seq);
```

2.3 使用示例

下面是 bootloader 中调用 Image 模块接口的例子：

1. 初始化 Image 模块。

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

2. 获取 OTA 功能初始化的参数

```
image_get_ota_param(&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.body_len);
image_check_data(&sh, (void *)sh.load_addr, sh.body_len, NULL, 0)
```

6. 反初始化 Image 模块。

```
image_deinit();
```

 Copyright @2017 Xradio Technology Co., Ltd. All Rights Reserved