XR806 OTA 中间件

开发指南

版本号：1.0

发布时间：2020-11-19

版本历史

版本	日期	责任人	版本描述
1.0	2020-11-19	AWA 1029	创建文档。

目录

表格目录

1 前言

1.1 文档简介

本文档主要介绍 OTA 模块的基本原理和使用方法，并说明使用该模块的注意事项，以方便开发者正确使用该模块进行开发。

1.2 目标读者

XR806 SDK 用户、OTA 模块开发使用人员。

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

OTA 模块为系统提供在线升级固件的功能。此文档用以解释说明 OTA 模块相关概念和定义，介绍并指导开发者使用 XR806 SDK 中的 OTA 方案。目前 OTA 模块支持两种升级方式：ping pong 模式和压缩升级模式。本文将介绍这两种升级方式。

OTA 模块在一定程度上依赖 image 模块。Image 相关内容可以参考文档《XR806 Image 中间件 开发指南》。

2.2 规格特性

OTA 中间件模块提供了以下功能特性。

表 2-1 OTA 模块功能特性

规格类型	支持规格	规格描述	备注
升级协议	HTTP	通过网络端 http 协议升级	
	File	通过文件系统升级，可以是 flash 或者 SD 卡	
	自定义协议	通过用户自定义的协议升级	
升级方式	Ping-pong 方式	两个 image 区域都保存完整固件，轮流进行升级	
	压缩方式	仅第一个 image 区域保存完整固件，第二个区域保存压缩固件，升级时将其解压至第一个区域进行使用	
校验方式	NONE	不带校验	
	CRC32	使用 CRC32 算法校验	
	MD5	使用 MD5 值进行校验	
	SHA1	使用 SHA1 算法校验	
	SHA256	使用 SHA256 算法校验	

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 OTA 中间件的文件位置

组件名	文件分类	文件位置
OTA	源码文件	./src/ota
	头文件	./include/ota

关键文件说明如下。

表 2-3 OTA 中间件的文件说明

文件名	文件说明
ota.c	OTA 模块的主函数

说明

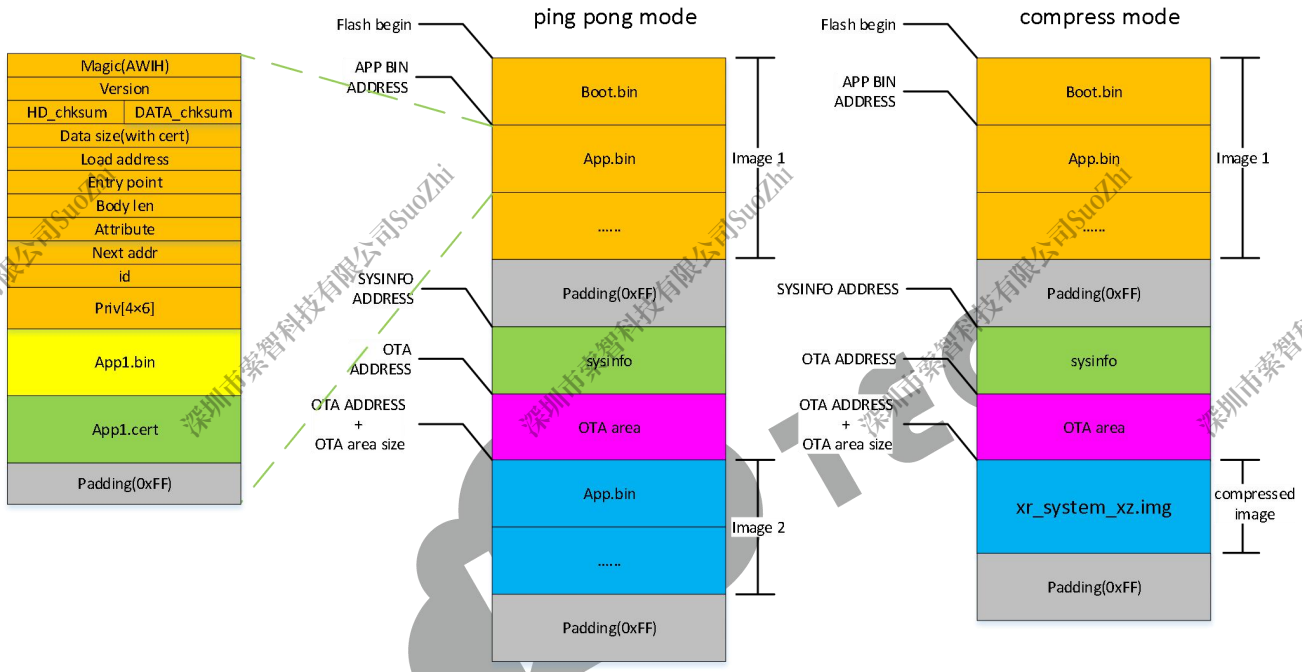
XR806 SDK 可在以下 GitHub 仓库获取：https://github.com/XradioTech/xr806_sdk.git

3 技术说明

3.1 Flash 布局说明

SDK 中的 OTA 方案通过对两个 Image 区域进行操作实现对固件的升级，整个 flash 区域布局如下图所示。

图 3-1 Flash 布局示例



以下对布局的每个区域以及参数进行说明：

3.1.1 Ping-pong 模式

在 ping pong 模式下，image1（不包含 BootLoader）区域和 image2 区域大小相同，均为 Image max size。Image1 和 image2 交替存储 image，当 image1 作为启动项时，image2 作为新 image 的升级区域；当 image2 作为启动项时，image1 作为新 image 的升级区域。

3.1.2 压缩模式

在压缩升级模式下，image1 区域和 image2 区域大小不相同，image1 要大于 image2，image1 的大小为 Image max size（不包含 BootLoader），image2 的大小为 Image xz max size（不包含 BootLoader）。

Image1 区域固定作为启动项，image2 区域用于升级时存储压缩后的 image。这样，image1 区域的空间就可以大大减小，减少 flash 的占用。

3.1.3 Image max size

Image max size 加 BootLoader 后应该大于实际编译打包出来的固件大小，Image xz max size 应该大于实际固件压缩后的大小，这样做的目的是为了给后期开发扩展预留一定的空间，预留出的部分即为 padding 部分。

3.1.4 Image 区域

两个 image 区域的起始应该与 Flash 可擦除块对齐（比如 4K 对齐）。

编译生成的固件包含了 Bootloader 和 image，通过刷机工具，固件会被烧录到 image1 区域，其中 Bootloader 大小需要 4k 对齐。

Image2 区域是不包含 BootLoader 的，因为 BootLoader 只需要一个即可。

3.1.5 OTA 区域

OTA 区域的主要功能是存放 OTA 参数，实际只存放了两个参数：一个是 image 区域启动项序号（即是 image1 为启动项，还是 image2 为启动项），另一个是启动项的 image 是否校验通过的状态标志。

OTA 区域的地址和大小在 image.cfg 文件定义，默认地址定义为 1M 的位置，用户可根据实际使用的 flash 大小以及 image 大小做调整，OTA 区域的大小一般定义为 4k 即可。

3.2 OTA 升级原理

整个 OTA 升级方案一般包括三个部分：正常系统启动，OTA 升级过程，以及系统重启。

3.2.1 正常系统启动

系统启动时，加载 Bootloader，Bootloader 根据 OTA 区域的参数，可以判断出采用的是什么升级模式以及应该运行哪个 image 区域的固件。

3.2.2 OTA 升级过程

OTA 升级固件时，系统根据 OTA 区域的参数，判断本次擦除并更新哪个 Image 区域的固件（压缩模式下，固定更新 image2 区域的固件）。固件更新成功并校验通过后，更新 OTA 区域的参数，并重启系统。

3.2.3 系统重启

系统重启后会根据更新后的 OTA 区域的参数，判断出采用的是什么升级模式。

在 ping pong 模式下，系统会判断并加载最新且校验通过的固件。

在压缩升级模式下，系统会擦除 image1 区域数据，将 image2 数据解压到 image1 区域，解压完后进行校验。如果以上所有流程都正常，则更新 OTA 区域参数；如果固件更新或校验失败，则不更新 OTA 区域，系统下次启动会再次尝试擦除 image1 区域和解压校验 image2 的操作。

3.3 OTA 参数

OTA 升级会使用到一些参数，包括启动 image 的编号，image 是否已校验等信息。

3.3.1 image_raw_cfg_t

OTA 区域存储了 OTA 升级的记录。系统启动时，Bootloader 获取 OTA 区域的信息，就能判断该加载哪个 Image 区域固件的信息。OTA 区域中主要存储结构体类型 image_raw_cfg_t 的数据。定义如下：

```
typedef struct image_raw_cfg {
    uint16_t raw_seq;       /* magic number for image_seq_t */
    uint16_t raw_state;     /* magic number for image_state_t */
```

```
} image_raw_cfg_t;
```

raw_seq 存储了 image 区域的序号。raw_state 表示该 image 区域的校验结果。

注意

- 在压缩升级模式中，`raw_seq` 固定为 `1`。

3.3.2 OTA protocol

OTA protocol 表示 OTA 升级时下载固件的协议。在 ota.h 中定义如下：

```
/**
 * @brief OTA protocol definition
 */
typedef enum ota_protocol {
    OTA_PROTOCOL_FILE    = 0,
    OTA_PROTOCOL_HTTP    = 1,
    OTA_PROTOCOL_FLASH = 2,
    OTA_PROTOCOL_MAX,
} ota_protocol_t;
```

当前 SDK 中支持通过 Http、File 以及 Flash 这三种协议下载升级固件。若用户需要增加自定义的升级协议，需要在此增加枚举类型。

3.3.3 OTA verify

OTA verify 表示对升级固件采用的校验算法。为保证固件在 OTA 下载和烧写 Flash 过程中不出错，可采用 CRC32、MD5、SHA1 或 SHA256 算法对固件进行校验。

目前 SDK 支持的校验算法有：

```
/**
 * @brief OTA image verification algorithm definition
 */
typedef enum ota_verify {
    OTA_VERIFY_NONE      = 0,
    OTA_VERIFY_CRC32     = 1,
    OTA_VERIFY_MD5       = 2,
    OTA_VERIFY_SHA1      = 3,
    OTA_VERIFY_SHA256    = 4,
} ota_verify_t;
```

当启动 OTA 校验功能时，编译打包生成的固件末尾有额外的 verify data。应用代码可根据 verify data 对固件进行校验，其数据结构定义如下：

```
/**
 * @brief OTA image verification data structure definition
 */
#define OTA_VERIFY_MAGIC            0x0055AAFF
#define OTA_VERIFY_DATA_SIZE   32
typedef struct ota_verify_data {
    uint32_t ov_magic;                  /* OTA Verify Header Magic Number */
    uint16_t ov_length;                 /* OTA Verify Data Length          */
    uint16_t ov_version;                /* OTA Verify Version: 0.0         */
    uint16_t ov_type;                   /* OTA Verify Type                       */
    uint16_t ov_reserve;
    uint8_t ov_data[OTA_VERIFY_DATA_SIZE];
} ota_verify_data_t;
```

4 应用说明

4.1 应用简述

OTA 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）。
2. 在用户应用代码中添加 OTA 模块头文件（参见“2.3 文件位置”章节），调用 OTA 接口进行使用（接口使用参见“4.3 接口说明”章节）。

4.2 配置说明

OTA 模块的启用需要使能 SDK 的 OTA 相关功能，具体配置项如下表所示。

表 4-1 XR806 OTA 模块工程配置说明

配置项	配置说明
OTA 功能使能	设置说明： 此项配置用于在 SDK 中启用 OTA 功能。 设置方式： 1、在控制台执行命令“make menuconfig”，将“OTA”功能项打开； 2、在“OTA Policy Select”选项中选择“ping pong mode”或者“image compression mode”。
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。

注意

- “max_size”和“image max size”含义不一样，“max_size”作用于 image.cfg 文件中，用来表示整个 image 的最大值，是包含 BootLoader 的；“image max size”作用于代码中，用来表示 OTA 时需要擦除空间的最大值，是不包含 BootLoader 的，因为 OTA 升级时，BootLoader 不能被擦除。

- 设置“max_size”和“xz_max_size”这两个参数非常重要，必须设置正确，否则会导致擦除过量或者 image 覆盖而无法正常启动问题。
- 在 ping pong 模式时，只能存在“max_size”项；在压缩升级模式时，“max_size”和“xz_max_size”必须都存在。因为 BootLoader 为了兼容两种模式，通过“xz_max_size”的值是否有效来判断采用什么升级模式，有效为压缩模式，无效则为 ping pong 模式。

4.3 接口说明

OTA 模块总共提供 3 套接口，主要功能点包括 OTA 模块初始化、获取升级固件以及固件校验等。

表 4-2 OTA 模块接口简介

接口名	简要介绍
模块初始化接口	用于开关 OTA 模块，以及升级成功后重启系统
获取升级固件接口	用于获取需要升级的固件数据
固件校验接口	用于校验获取到的固件数据是否正确

4.3.1 模块初始化接口

4.3.1.1 ota_init

表 4-3 ota_init 接口函数说明

信息项	说明
原型	ota_status_t ota_init(void);
功能	初始化 OTA 模块
参数	无
返回值	OTA_STATUS_OK：成功 OTA_STATUS_ERROR：失败

4.3.1.2 ota_deinit

表 4-4 ota_deinit 接口函数说明

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

4.3.1.3 ota_reboot

表 4-5 ota_reboot 接口函数说明

信息项	说明
原型	void ota_reboot(void);
功能	重启系统
参数	无
返回值	无

4.3.2 获取升级固件接口

4.3.2.1 ota_get_image

表 4-6 ota_get_image 接口函数说明

信息项	说明
原型	ota_status_t ota_get_image(ota_protocol_t protocol, void *url);
功能	获取需要升级的固件
参数	ota_protocol_t protocol 含义解释：升级固件的协议，支持 http、file 以及 flash 等协议 void *url 含义解释：升级固件的链接地址
返回值	OTA_STATUS_OK：成功 OTA_STATUS_ERROR：失败

4.3.3 固件校验接口

4.3.3.1 ota_get_verify_data

表 4-7 ota_reboot 接口函数说明

信息项	说明
原型	ota_status_t ota_get_verify_data(ota_verify_data_t *data);
功能	获取升级固件的校验数据
参数	ota_verify_data_t *data 含义解释：校验数据的指针
返回值	OTA_STATUS_OK：成功 OTA_STATUS_ERROR：失败

4.3.3.2 ota_verify_image

表 4-8 ota_verify_image 接口函数说明

信息项	说明
原型	ota_status_t ota_verify_image(ota_verify_t verify, uint32_t *value);
功能	通过指定算法校验需要升级的固件，并根据校验结果更新 OTA 区域信息
参数	ota_verify_t verify 含义解释：校验算法 uint32_t *value 含义解释：正确固件通过指定校验算法得到的校验值
返回值	OTA_STATUS_OK：成功 OTA_STATUS_ERROR：失败

5 示例说明

本章节分别以升级固件、协议扩展和压缩模式打包为例，简要说明 OTA 模块接口的使用。

5.1 升级固件示例

5.1.1 示例简介

本例以 Http 协议升级固件的流程，简单介绍 OTA 模块对应接口的使用方法。

5.1.1.1 获取方式

OTA 升级固件示例在 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

5.1.1.2 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》。

5.1.1.3 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，OTA 模块自动初始化，需要 OTA 升级的命令，才会启动固件升级的功能。

表 5-1 OTA 固件升级的命令列表

命令	说明
启动 OTA 固件升级	命令格式： ota <protocol> <url> 命令解释： 使用指定的升级协议向指定的链接地址下载升级固件。 参数说明： protocol：指定的升级协议，可以是 http 或者 file 等。 url：指定的升级固件链接地址。

5.1.2 关键实现

先通过调用 ota_get_image()获取需要被升级的固件，然后调用 ota_get_verify_data()获取校验数据，再调用 ota_verify_image()对校验数据进行校验，如果校验通过，重启系统即可。

具体操作见下面的示例代码。

```
char url[] = "http://192.168.1.100/OTA/xr_system.img";
uint32_t *verify_value;
ota_verify_t verify_type;
ota_verify_data_t verify_data;

//通过 Http 协议下载固件
if (ota_get_image(OTA_PROTOCOL_HTTP, url) != OTA_STATUS_OK) {
    ......                          //出错处理
}

//获取校验数据和校验算法
if (ota_get_verify_data(&verify_data) != OTA_STATUS_OK) {
    verify_type = OTA_VERIFY_NONE;
    verify_value = NULL;
} else {
    verify_type = verify_data.ov_type;
    verify_value = (uint32_t*)(verify_data.ov_data);
}

//校验固件，该接口也会更新 OTA 区域信息
if (ota_verify_image(verify_type, verify_value) != OTA_STATUS_OK) {
    ......                          //出错处理
}

//重启系统
ota_reboot();
```

5.1.3 效果展示

在完成 OTA 固件升级成功后，可以看到系统重启，通过启动打印可以确认是新下载的固件。

5.2 协议扩展示例

5.2.1 示例简介

本例以用户扩展自定义 OTA 升级协议的修改过程，简单介绍 OTA 模块对应接口的使用方法。

5.2.2 关键实现

以下详细列举协议扩展需要修改的代码情况。

1. 在文件 ota.h 中将扩展的协议补充到枚举类型 ota_protocol_t 中。

```
/**
 * @brief OTA protocol definition
 */
typedef enum ota_protocol {
    OTA_PROTOCOL_FILE   = 0,
    OTA_PROTOCOL_HTTP   = 1,
    OTA_PROTOCOL_FLASH = 2,
    OTA_PROTOCOL_XXXX   = 3,
    OTA_PROTOCOL_MAX,
} ota_protocol_t;
```

2. 实现扩展协议下载固件的两个回调函数，回调函数类型在文件 ota_i.h 中定义如下。可参考 Http 协议和 File 协议对两个回调函数的实现（ota_http.h、ota_http.c、ota_file.h、ota_file.c）。

```
typedef ota_status_t (*ota_update_init_t)(void *url);
typedef ota_status_t (*ota_update_get_t)(uint8_t *buf, uint32_t buf_size,
                                         uint32_t *recv_size, uint8_t *eof_flag);
```

3. 将扩展协议的两个回调函数注册到文件 ota.c 的函数 ota_get_image()中。

```
case OTA_PROTOCOL_FILE:
    return ota_update_image(url, ota_update_file_init, ota_update_file_get);
case OTA_PROTOCOL_HTTP:
    return ota_update_image(url, ota_update_http_init, ota_update_http_get);
case OTA_PROTOCOL_XXXX:
    return ota_update_image(url, ota_update_xxxx_init, ota_update_xxxx_get);
```

5.3 压缩模式打包示例

5.3.1 示例简介

本例以压缩模式打包生成压缩固件的流程，简单介绍 OTA 模块对应接口的使用方法。

在压缩升级模式中，需要用命令来将 image 压缩和打包。

5.3.1.1 操作步骤

在 Hello_demo 示例工程配置为压缩升级模式并完成编译后，输入以下打包命令，即可生成对应压缩升级固件。

表 5-2 压缩固件打包的命令列表

命令	说明
启动 OTA 固件升级	命令格式： make image_xz 命令解释： 在对应工程的 gcc 目录中执行命令，会将该工程已经编译好的 image 文件打包生成新的压缩固件，默认命名为“xr_system_img_xz.img”。

5.3.2 效果展示

打包成功时，控制台会输入如下信息：

```
$ make image_xz
cd ../image/xr806 && \
        dd if=xr_system.img of=xr_system.img.temp skip=32768 bs=1c && \
        xz -f -k --no-sparse --armthumb --check=none --lzma2=preset=6,dict=8KiB,lc=3,lp=1,pb=1
xr_system.img.temp && \
        mv xr_system.img.temp.xz image.xz && \
        rm xr_system.img.temp && \
        ../../../../../tools/mkimage -O -c ../../../../../project/image_cfg/image_img_xz.cfg -o
xr_system_img_xz.img
记录了 1007448+0 的读入
记录了 1007448+0 的写出
1007448 字节(1.0 MB)已复制，1.02673 秒，981 kB/秒
cfg string:
{
    "magic"     :"AWIH",
    "version" :"0.4",
    "OTA"        :{"addr": "1024K", "size": "32K"},
    "image"     :{"max_size": "1020K", "xz_max_size": "600K"},
    "extern"    :{"chk_ota_area": "true", "add_ota_area":"flase","verify":"md5"},
    "count"     :1,
    "section" :[
        {"id": "0xa5fe5a01", "bin": "image.xz",     "cert": "null", "flash_offs": "0K",     "sram_offs":
"0xffffffff", "ep": "0xffffffff", "attr": "0x11"}
    ]
}
```

```
********** verify data **********
magic code: 0055AAFF
length: 16
version: 1.0
type: 2
data:
3E 0F 74 64 B4 47 04 63 3C B5 92 FC E3 02 66 6F
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

**********

generate image: xr_system_img_xz.img
```

著作权声明

商标声明

免责声明