

RT-Thread OTA 用户手册

1、介绍

OTA (Over-the-Air Technology) 即空中下载技术，用于在线下载升级设备固件。

RT-Thread 提供的 OTA 技术支持以下特性：

- **固件防篡改**：自动检测固件签名，保证固件安全可靠
- **固件加密**：支持 AES-128 加密算法，提高固件下载、存储安全性
- **固件压缩**：降低固件大小，减少 Flash 空间占用，节省传输流量，降低下载时间
- **差分升级**：根据版本差异生成差分包，进一步节省 Flash 空间及传输流量
- **断点续传**：下载出错时自动恢复，节省下载流量
- **智能还原**：固件损坏时，自动还原至出厂固件，提升可靠性

2、原理

OTA 升级其实就是 IAP 在线编程。在嵌入式设备 OTA 中，通常通过串口或者网络等方式，将升级数据包下载到 Flash，然后将下载得到的数据包搬运到 MCU 的代码执行区域进行覆盖，以完成设备固件升级更新的功能。因此，OTA 升级一个非常重要的工作就是对 Flash 的操作。

嵌入式设备的 OTA 升级一般是不基于文件系统的，而是通过对 Flash 划分为不同的功能区域来完成 OTA 功能。

2.1 Flash 分区

嵌入式设备的 Flash 资源都比较有限，一般会划分为 **bootloader**、**application**、**download** 三个分区。

Flash 分区表如下所示：

1	Flash 分区表
2	+-----+ boot_start_addr
3	
4	bootloader
5	
6	+-----+ boot_end_addr
7	+-----+ app_start_addr
8	
9	application
10	
11	+-----+ app_end_addr
12	+-----+ down_start_addr
13	
14	download
15	
16	+-----+ down_end_addr
17	+-----+
18	
19	others
20	

- **bootloader** 分区
bootloader 主要负责代码的搬运工作。通常存放在复位时 MCU 执行的起始地址处，然后通过 Bootloader 来引导执行 application 程序。
- **application** 分区
application 区域存储用户自己的应用代码。
- **download** 分区
download 分区存储用户的 **升级文件**，亦是待升级的 application 程序。

flash 分区表定义在 bootloader 固件中，如果需要修改分区表的定义，则要修改 bootloader 程序。具体 flash 分区大小如何设置，需要参考具体的 MCU 平台。

2.1.1 beken7231 平台 flash 分区

在 beken7231 平台中，Flash 分区表定义如下表所示：

- 物理分区：（Flash 上固件的真实地址）

分区名称	分区起始地址	分区结束地址	分区大小
bootloader	0x0	0x11000	68K
app	0x11000	0x107800	986k
download	0x108000	0x1FD000	980k

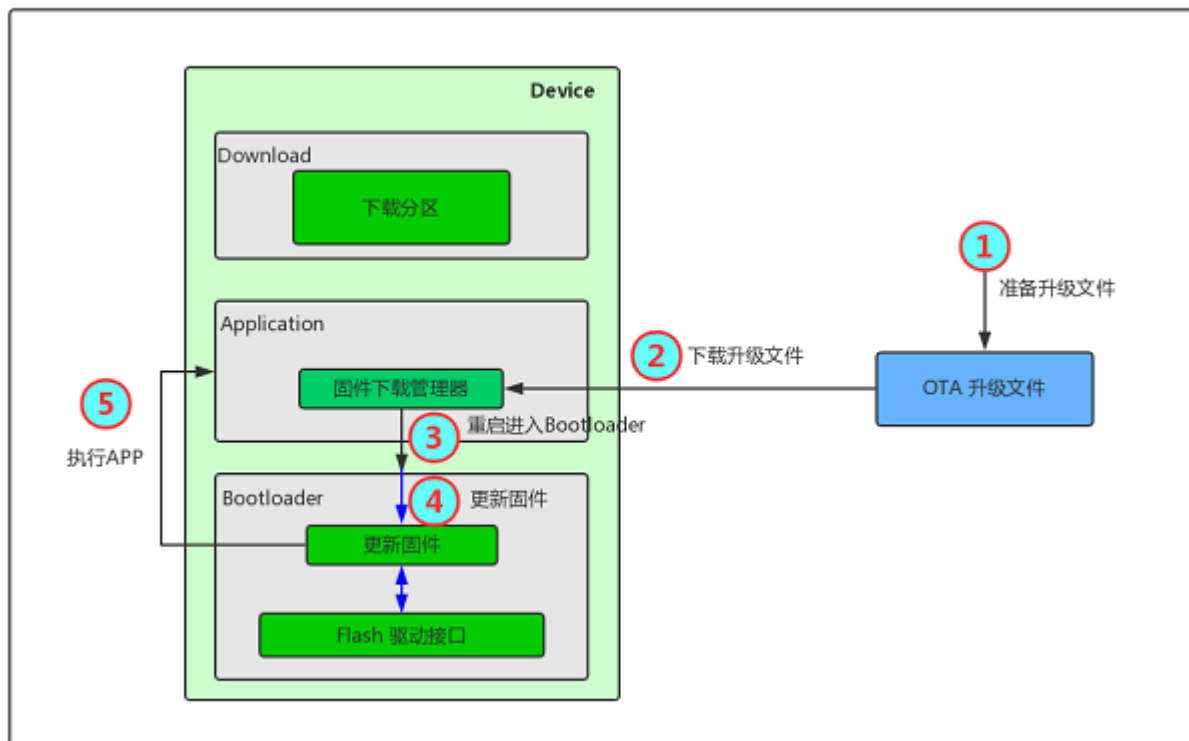
- 逻辑分区：（CPU 真正执行的地址）

分区名称	分区起始地址	分区结束地址	分区大小
bootloader	0x0	0x10000	64K
app	0x10000	0xF8000	928k
download	0x108000	0x1FD000	980k

如上边所示，由于 beken7231 平台自身的特性，将 Flash 分区表表述为 物理分区 和 逻辑分区 两类。其中 逻辑分区 是 CPU 执行 物理分区 上的程序时的 CPU 可见地址。CPU 执行的程序区包括 bootloader 和 app，其他分区为 Flash 控制器的读写区域。

2.2 OTA 升级过程

OTA 升级流程简述为下图所示：



1. 准备升级文件

升级文件为对应 MCU 的可执行程序

2. 下载升级文件

Application 应用程序中对应的下载管理器，会将升级文件下载到 Flash 的 Download 分区

3. 重启进入 Bootloader

Application 应用程序中对应的下载管理器在下载完升级文件后，会重启系统，执行 Bootloader 程序

4. 更新固件

Bootloader 程序检测到 Download 分区有升级文件，执行固件更新动作，并检验固件的完整性和合法性

5. 执行 Application

固件更新并校验成功后，Bootloader 会引导执行 Application 程序

3、Ymodem 方式升级

Ymodem 协议是由 Xmodem 协议演变而来的一个非常高效的文件传输协议，在嵌入式设备中通常用于进行文件传输及 IAP 在线升级。

3.1 烧录 all.bin

在 RT-Thread 中，Ymodem 属于应用层程序，并没有集成到 Bootloader 中，因此在初次使用 Ymodem OTA 升级的时候，首先需要烧录提供的 `all.bin` 固件到你的设备中。

`all.bin` (包含了 bootloader、app 和 download 固件)，默认提供了 Ymodem 功能支持。

3.2 升级固件准备

3.2.1 使能 Ymodem OTA 功能

在 menuconfig 中配置使能 Ymodem OTA 例程，如下所示：

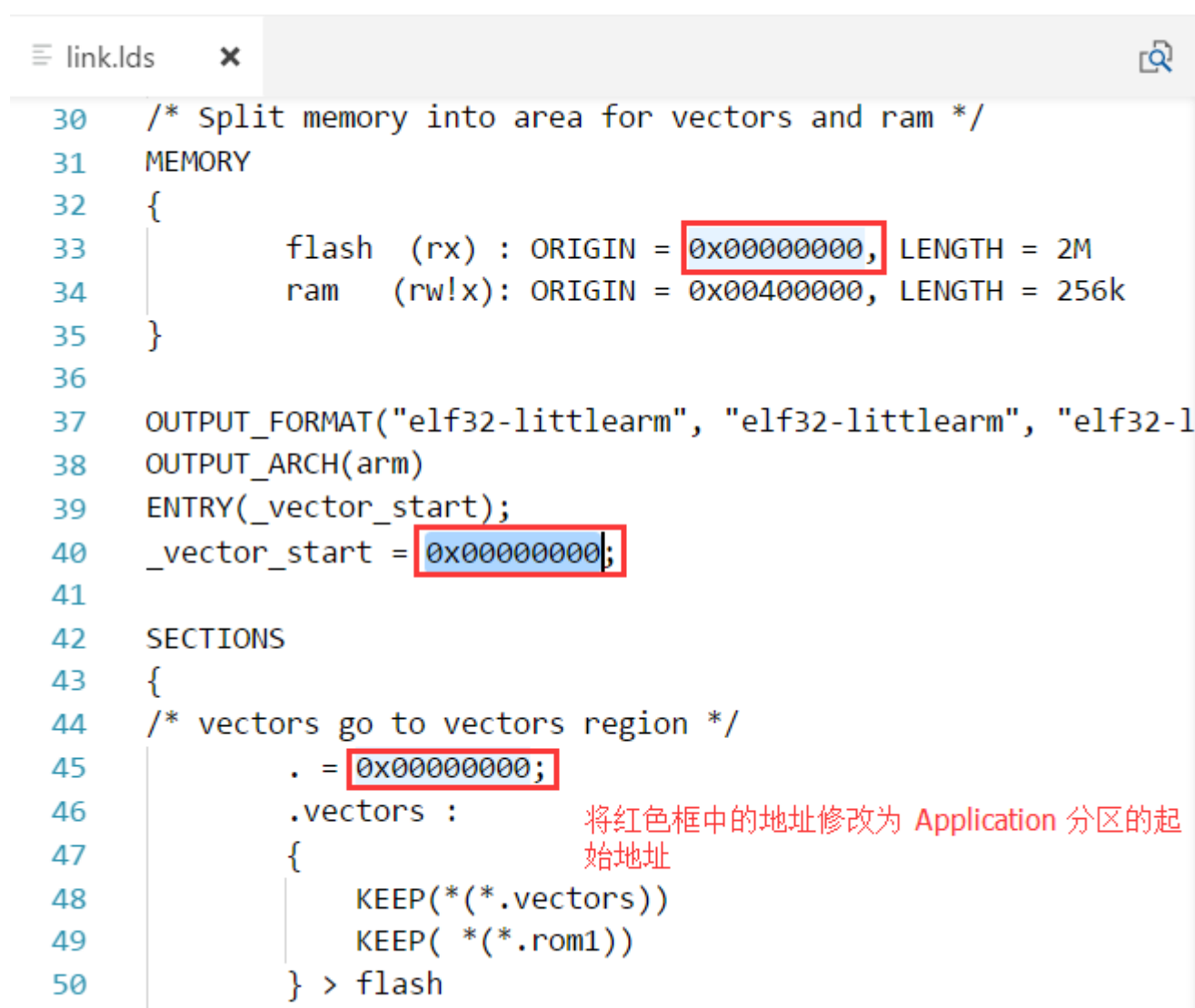
```
1 Application Samples Config --->
2   RT-Thread OTA samples --->
3     [*] Ymodme OTA mode
```

3.2.2 修改链接脚本配置

通常，我们的程序都是从 Flash 代码区的起始地址开始运行的。但是，Flash 代码区的起始地址开始的空间被 bootloader 程序占用了，因此我们需要修改链接脚本，让 application 程序从 Flash 的 application 区域起始地址开始排放。

一般，我们只需要修改链接脚本里中 Flash 和 SECTION 段的起始地址为 application 分区的起始地址即可。application 分区信息必须跟对应 MCU 平台的 Flash 分区表完全一致。

以 GCC 链接脚本为例，介绍如何修改，如下所示：



```
link.ld
30  /* Split memory into area for vectors and ram */
31  MEMORY
32  {
33      flash  (rx) : ORIGIN = 0x00000000, LENGTH = 2M
34      ram    (rw!x): ORIGIN = 0x00400000, LENGTH = 256k
35  }
36
37  OUTPUT_FORMAT("elf32-littlearm", "elf32-littlearm", "elf32-l
38  OUTPUT_ARCH(arm)
39  ENTRY(_vector_start);
40  _vector_start = 0x00000000;
41
42  SECTIONS
43  {
44      /* vectors go to vectors region */
45      . = 0x00000000;
46      .vectors :
47      {
48          KEEP(*(*.vectors))
49          KEEP( *(*.rom1))
50      } > flash
```

将红色框中的地址修改为 Application 分区的起始地址

在 beken7231 平台上，应用程序分区 app 起始地址为 0x00010000，上图红色框中的地址应该修改为 0x00010000。

修改链接脚本后，重新编译生成固件 rtthread.bin。

3.2.3 OTA 固件打包

编译器编译出来的应用程序 `rtthread.bin` 属于原始固件，并不能用于 RT-Thread OTA 的升级固件，需要用户使用 `RT-Thread OTA 固件打包器` 打包生成 `.rbl` 后缀名的固件，然后才能进行 OTA 升级。

`RT-Thread OTA 固件打包器` 如下图所示：



RT-Thread OTA 固件打包器

选择固件

C:\Users\Administrator\Documents\workspace\rtthread.bin

保存路径

C:\Users\Administrator\Desktop\rtthread.rbl

压缩算法

不压缩

加密算法

不加密

加密密钥

加密 IV

固件名称

app

固件版本

2.0

结果：

打包成功

HASH_CODE : 2A4A54D2

RAW_SIZE : 7728

HDR_CRC32 : 7AE1EC04

PKG_SIZE : 7744

BODY_CRC32 : 3D907EF

TIMESTAMP : 1517645310

开始打包

COPYRIGHT (C) 2012-2018, Shanghai Real-Thread Technology Co., Ltd

Ver: 1.0.0.0

用户可以根据需要，选择是否对固件进行加密和压缩，提供多种压缩算法和加密算法支持，基本操作步骤如下：

- 选择待打包的固件
- 选择生成固件的位置
- 选择压缩算法
- 选择加密算法
- 配置加密密钥（不加密则留空）
- 配置加密 IV（不加密则留空）
- 填写固件名称（对应分区名称）
- 填写固件版本
- 开始打包
- OTA 升级

固件打包过程中有 `固件名称` 的填写，这里注意需要填入 Flash 分区表中对应分区的名称，不能有误。

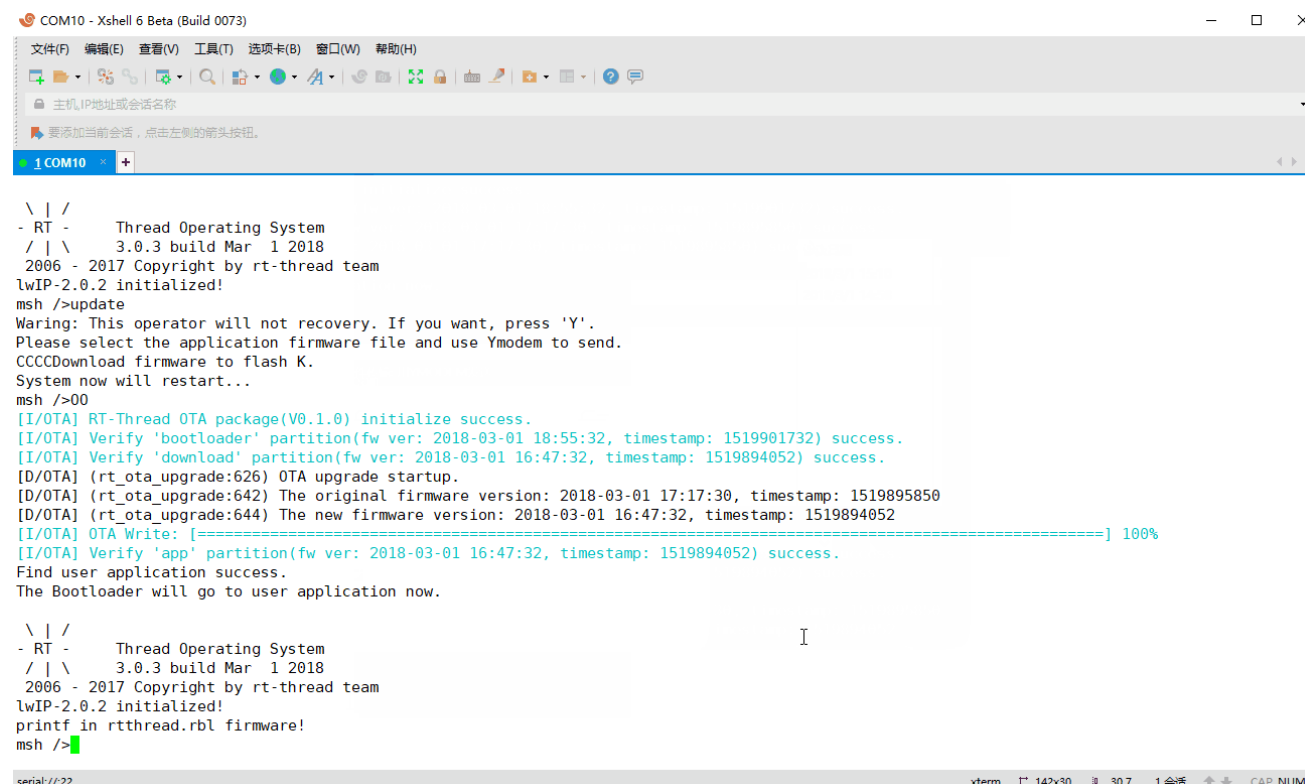
Note:

提供的 `bootloader.bin` 中默认的 **加密密钥** 是 `0123456789ABCDEF0123456789ABCDEF` , **加密 IV** 是 `0123456789ABCDEF` 。用户在打包固件的时候, 必须使用跟 `bootloader` 中相同的 **加密密钥** 和 **加密 IV** 。

3.3 Ymodem 升级

烧录了 `all.bin` 的设备, 可以直接使用 Ymodem 进行在线升级。

- 将设备与电脑连接并确定驱动正常
- 打开 SecureCRT 或者 XShell 这类具有 Ymodem 功能的终端工具, 串口配置 115200 8 1 N
- 输入 `update` 命令
- 提示需要输入 Y 进行确认
- 通过 Ymodem 发送待更新的 `rtthread.rbl` 固件
- 此后, 开发板将会自动完成更新



```
\ | /
- RT -      Thread Operating System
/ | \      3.0.3 build Mar 1 2018
2006 - 2017 Copyright by rt-thread team
lwIP-2.0.2 initialized!
msh />update
Waring: This operator will not recovery. If you want, press 'Y'.
Please select the application firmware file and use Ymodem to send.
CCCCDownload firmware to flash K.
System now will restart...
msh />00
[I/OTA] RT-Thread OTA package(V0.1.0) initialize success.
[I/OTA] Verify 'bootloader' partition(fw ver: 2018-03-01 18:55:32, timestamp: 1519901732) success.
[I/OTA] Verify 'download' partition(fw ver: 2018-03-01 16:47:32, timestamp: 1519894052) success.
[D/OTA] (rt_ota_upgrade:626) OTA upgrade startup.
[D/OTA] (rt_ota_upgrade:642) The original firmware version: 2018-03-01 17:17:30, timestamp: 1519895850
[D/OTA] (rt_ota_upgrade:644) The new firmware version: 2018-03-01 16:47:32, timestamp: 1519894052
[I/OTA] OTA Write: [=====] 100%
[I/OTA] Verify 'app' partition(fw ver: 2018-03-01 16:47:32, timestamp: 1519894052) success.
Find user application success.
The Bootloader will go to user application now.

\ | /
- RT -      Thread Operating System
/ | \      3.0.3 build Mar 1 2018
2006 - 2017 Copyright by rt-thread team
lwIP-2.0.2 initialized!
printf in rtthread.rbl firmware!
msh />
```

3.4 注意事项

- 如果 Ymodem 与 log 共用一个串口, 则在 Ymodem 升级过程中, 不允许有其它 log 输出, 否则会破坏 Ymodem 协议, 导致传输失败。
- 默认提供的 `all.bin` 是通过 `UART1` 来进行升级的, 如果用户使用 `UART2` , 请烧录 `bin/others/shell_using_uart2_all.bin` 固件。

4、Http 方式升级

4.1 烧录 all.bin

OTA 升级需要 Bootloader 的支持, 我们首先需要将提供的 `all.bin` 固件烧录到你的设备中。

`all.bin` (包含了 `bootloader` 、 `app` 和 `download` 固件), 默认提供了 Ymodem 和 HTTP OTA 功能支持。

4.2 升级固件准备

4.2.1 使能 HTTP OTA 功能

在 menuconfig 中配置使能 HTTP OTA 例程，如下所示：

```
1 Application Samples Config --->
2   RT-Thread OTA samples --->
3     [*] HTTP OTA mode
```

4.2.2 修改链接脚本配置

参考 [3.2.2 章节](#)。

4.2.3 OTA 固件打包

参考 [3.2.3 章节](#)。

4.3 HTTP OTA 升级

使用烧录了 all.bin 的设备，演示如何进行 HTTP 方式 OTA 升级。

- 将设备与电脑通过串口连接
- 将升级固件（如 `rtthread.rbl`）上传到 HTTP 服务器
- 在串口中，使用命令 `http_ota [uri]` 进行 OTA 升级
将 `[uri]` 替换为您的 HTTP 服务器上 `rtthread.rbl` 固件的地址；
命令示例：`http_ota http://192.168.10.135:80/rtthread.rbl`。

```
1 msh />
2 msh />http_ota http://192.168.10.135:80/rtthread.rbl
3 [I/HTTP_OTA] Start erase flash download partition!
4 [I/HTTP_OTA] Erase flash (download) partition success!
5 [I/HTTP_OTA] Download:
  [=====] 100%
6 [I/OTA] Verify 'download' partition(fw ver: 24.4, timestamp: 1521791141) success.
7 reboot system
```

4.4 注意事项

- 网络连接超时错误（-7）在网络比较差的环境下使用 HTTP OTA 的时候，会发生网络连接超时的问题，会返回 -7 错误，重试即可，或者切换到良好的网络环境。
- HTTP OTA 功能依赖 `webclient` 包
- 打开 URI 失败
 - URI 链接有误
 - 没有连接网络

5、常见问题

1. 程序无法从 `bootloader` 跳转到 `app`

编译 **app** 固件的时候，没有修改 **链接脚本**。

2. 串口没有任何输出

串口选择错误，**all.bin** 使用的是 UART1 串口，如果使用的是 UART2 串口，请烧录 `bin/all.bin`。