AN0005
应用笔记

AT32 OTA using the USART

前言

对于大多数基于闪存的系统，一项重要要求是能够在最终产品中安装固件时进行更新。此OTA（Over-the-Air）是用户自己的程序在运行过程中对User Flash的部分区域进行烧写，目的是为了在产品发布后可以方便地通过预留的通信口，对产品中的固件程序进行更新升级。

本应用笔记的目的是提供在AT32微控制器上创建OTA应用程序的一般准则。

AT32微控制器可以运行用户特定的固件来对微控制器中嵌入的闪存执行OTA。此功能可以使用产品可用和支持的任何通信接口。使用自定义协议协议的USART是本应用笔记中的示例。

支持型号列表：

支持型号	AT32 全系列

目录

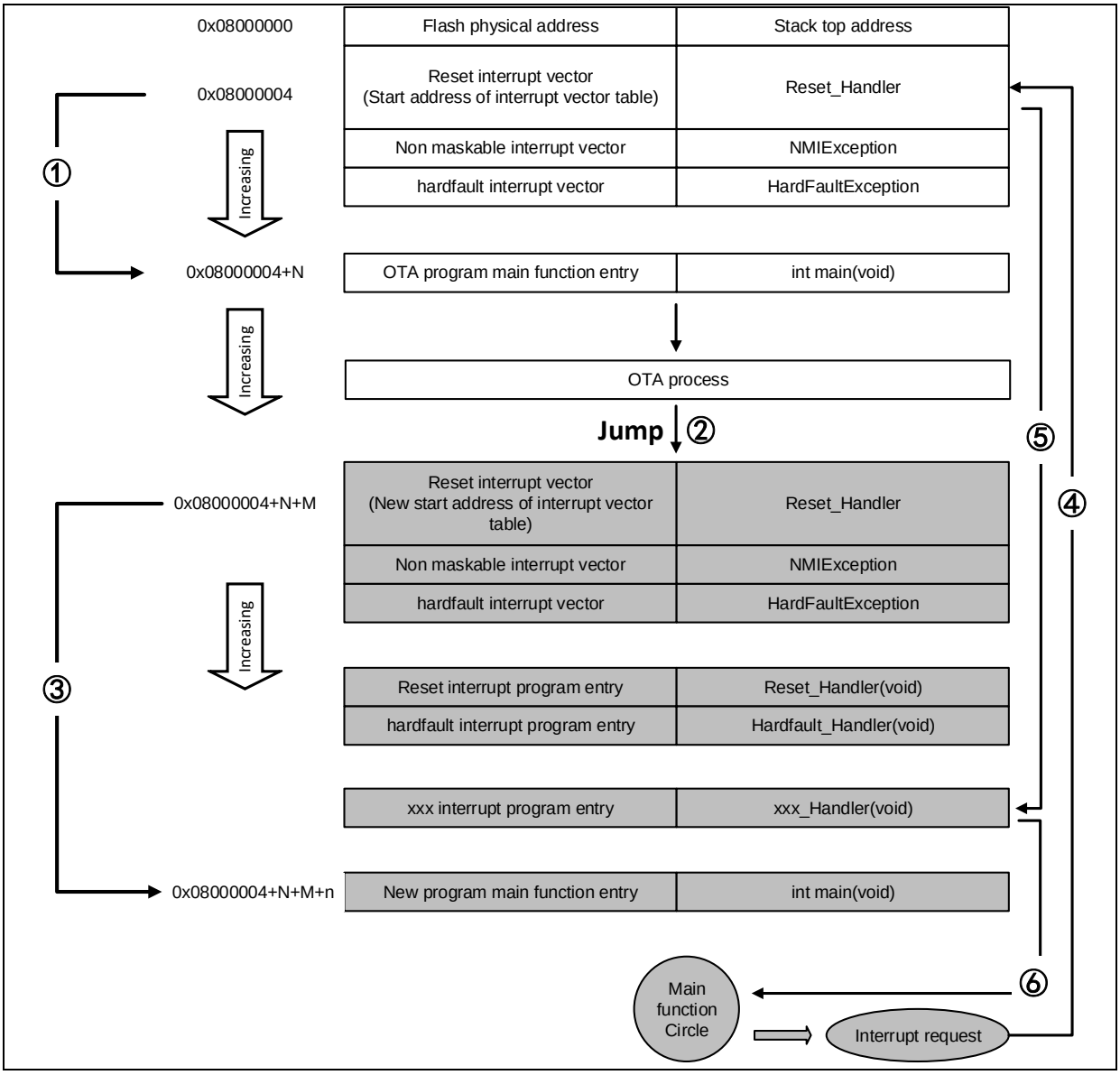
表目录

图目录

1 概述

空中下载技术 OTA（Over-the-Air Technology）是用户自己的程序在运行过程中对 User Flash 的部分区域进行烧写，目的是为了在产品发布后可以方便地通过预留的通信口，对产品中的固件程序进行更新升级。通常实现 OTA 功能时，即用户程序运行中作自身的更新操作，需要在设计固件程序时编写两个项目代码，第一个项目程序为 Bootloader 区域，第二个项目程序 App 代码为真正的功能代码，执行应用和升级。这两部分项目代码同时烧录在 User Flash 中。

图 1. OTA 代码执行流程



在上图所示流程中，MCU 复位后，从 0x08000004 地址取出复位中断向量的地址，并跳转到复位中断服务程序，在运行完复位中断服务程序之后跳转到 Bootloader 的 main 函数，如图标号①所示；在执行完 Bootloader 以后（App 代码为图中 FLASH 灰底部分 App 程序的复位中断向量起始地址为 0x08000004+N+M），跳转至 App 程序的复位向量表，取出 App 程序的复位中断向量的地址，并跳转执行 App 程序的复位中断服务程序，随后跳转至 App 程序的 main 函数，如图标号②和③所示，同样 main 函数为一个死循环，并且注意到此时 AT32 的 FLASH，在不同位置上，共有两个中断向量表。

在 main 函数执行过程中，如果 CPU 得到一个中断请求，PC 指针仍强制跳转到地址 0x08000004 中断向量表处，而不是 App 程序的中断向量表，如图标号④所示；程序再根据我们设置的中断向量表

偏移量，跳转到对应中断源新的中断服务程序中，如图标号⑤所示；在执行完中断服务程序后，程序返回 main 函数继续运行，如图标号⑥所示。

通过以上两个过程的分析，我们知道 OTA 程序必须满足两个要求：

1) App程序必须在Bootloader程序之后的某个偏移量为x的地址开始。
2) 必须将App程序的中断向量表相应的移动，移动的偏移量为x。

1.1 AT32 USART OTA 快速使用方法

1.1.1 硬件资源

文档中是用 AT-START-AT32F403A 实验板的硬件条件为例，OTA demo 源代码还包括 AT32 其他型号，用户只需编译对应型号工程烧录于 AT-START 实验板运行即可。

1) 指示灯LED2/LED3/LED4
2) USART1(PA9/PA10)
3) AT-START实验板

1.1.2 软件资源

1) tool_release
   - IAP_Programmer.exe，PC机tool，用于演示OTA升级流程
2) source_code
   - Bootloader，Bootloader源程序，运行LED2闪烁
   - App_led3_toggle，App1源程序，运行LED3闪烁
   - App_led4_toggle，App2源程序，运行LED4闪烁

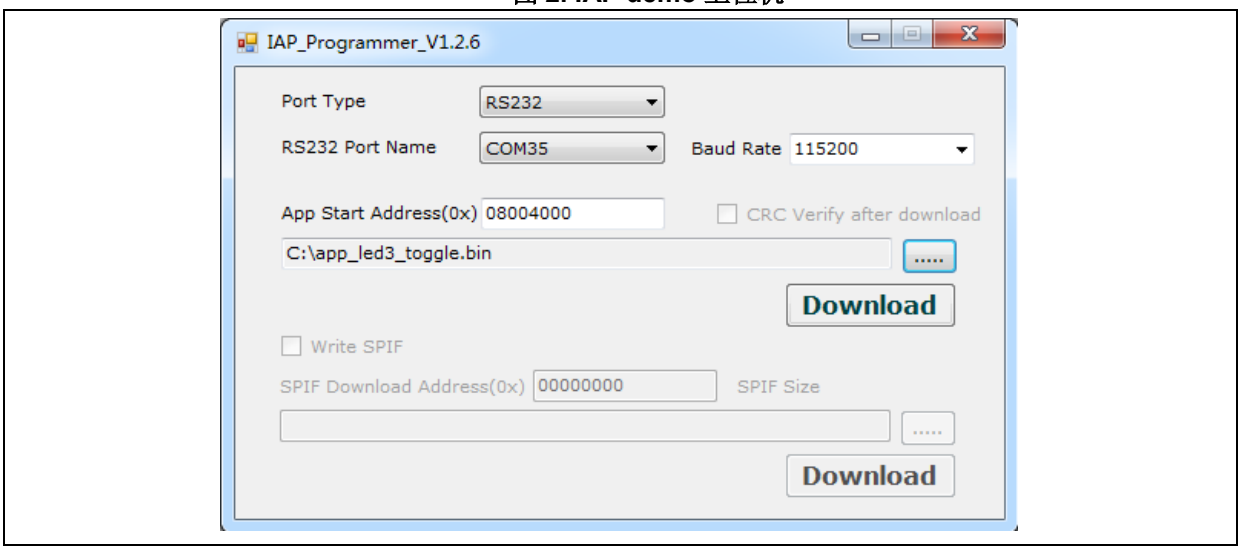
注：工程基于keil v5建立，若用户需要在其他编译环境上使用，请参考对应BSP目录AT32F403A_407_Firmware_Library_V2.x.x\project\at_start_f403a\templates中各种编译环境（例如IAR6/7/8,keil 4/5, eclipse_gcc）进行对应修改即可。

1.2 OTA demo 使用

本文档描述了两种常用的 OTA 应用 demo，template app 和 dual app，后面章节会分别介绍。

1) 打开Bootloader工程源程序，选择对应MCU型号的target编译后下载到实验板
2) 打开IAP_Programmer.exe
3) 选择正确的串口、APP下载地址和bin文档，点击Download下载，如下图
4) 观察LED2/3/4闪烁，LED2闪烁-Bootloader工作，LED3闪烁-App1工作，LED4闪烁-App2工作

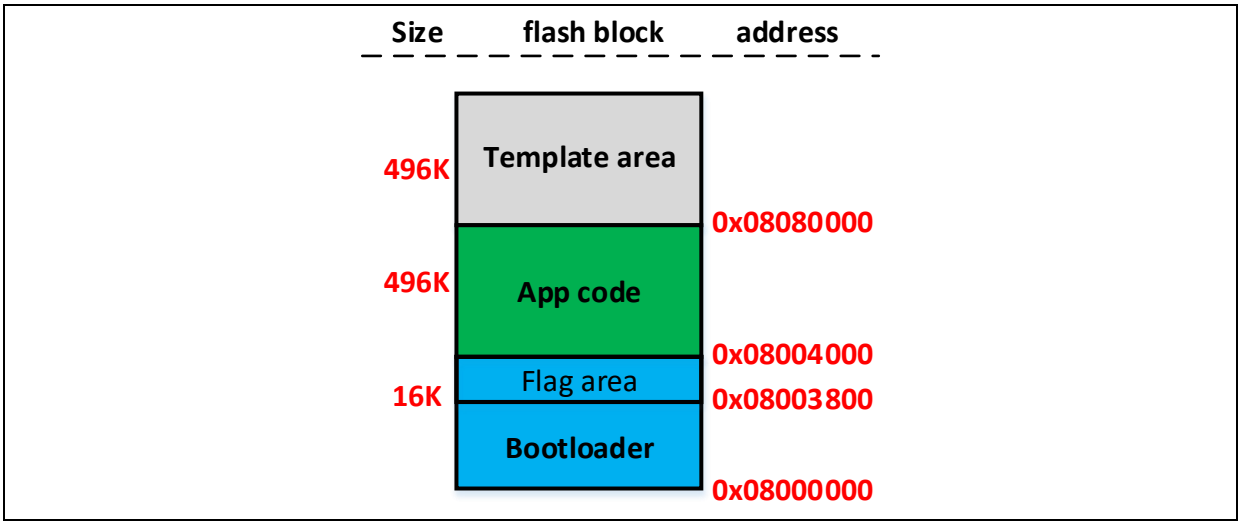
图 2. IAP demo 上位机



2 Template app OTA 程序设置

2.1 地址分布

图 3. Flash 地址分配

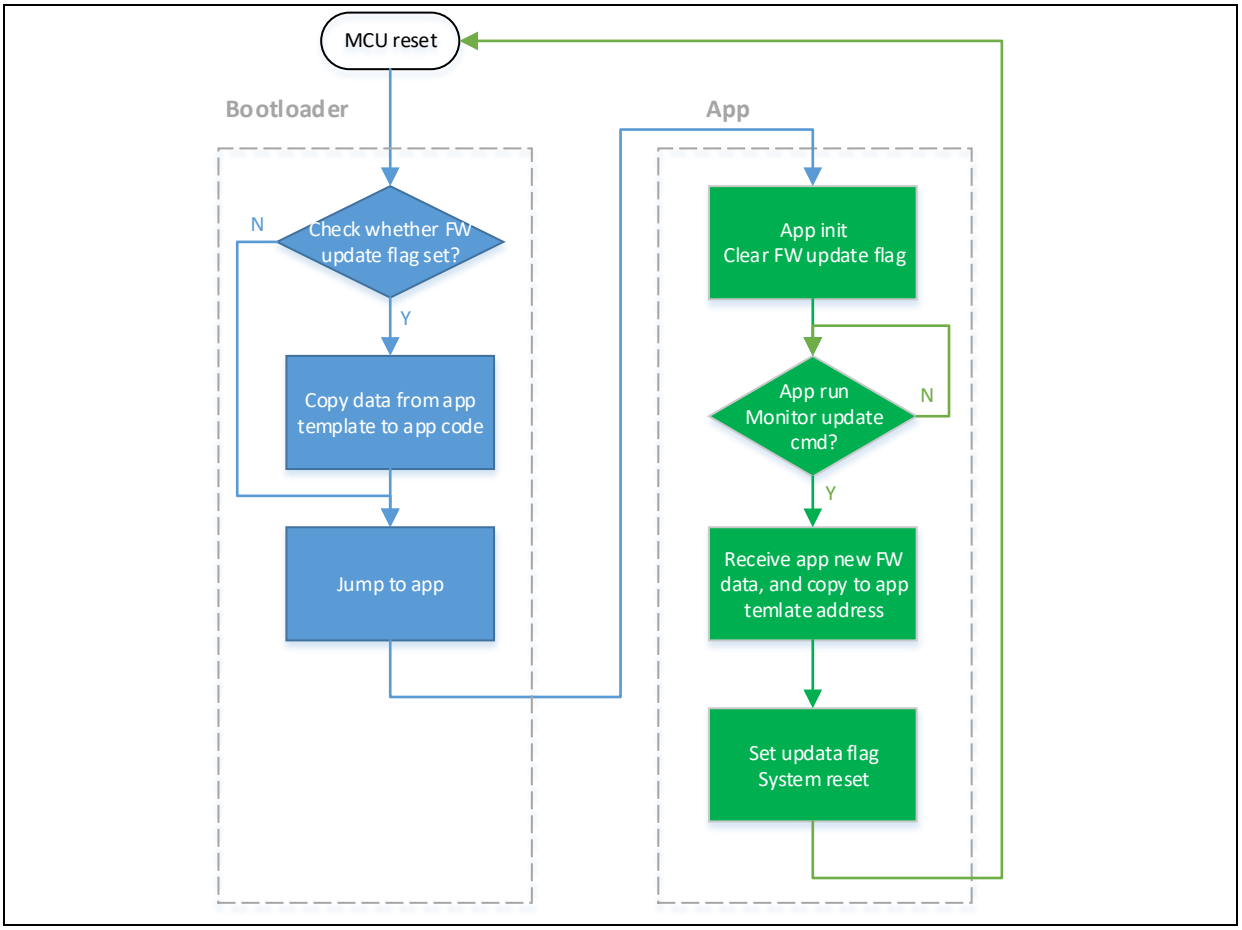


注：Bootloader区域最后一个扇区，用于存放防止升级过程出错（掉电等异常情况）的flag，用户编译修改Bootloader时，要保证不覆盖flag的地址。

2.2 执行流程

OTA 分为 Bootloader、App 和 Template 三部分，应用在 App 中执行，Template 仅作为新 App 固件数据的临时存放空间。程序执行整体流程框图如下：

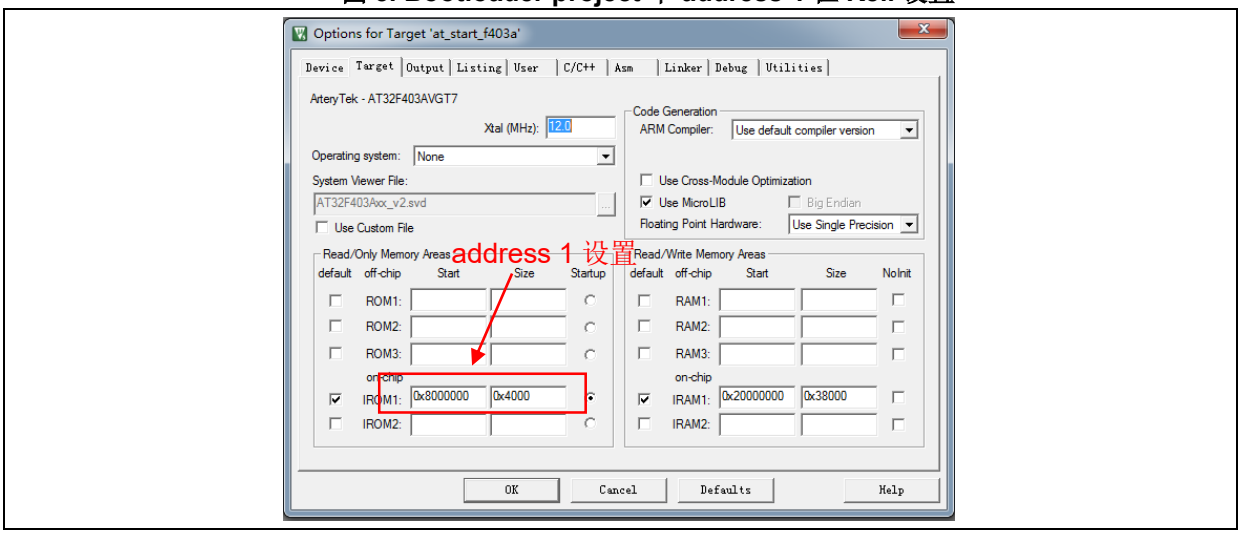
图 4. 程序执行流程



2.3 Bootloader project 设置

1) Keil设置

图 5. Bootloader project 中 address 1 在 Keil 设置



2) Bootloader源程序修改ota.h文件中

图 6. Bootloader project 中 address 2 在程序中设置

```
/* app starting address */
#define APP_START_ADDR            0x08004000

/* the previous sector of app starting address is ota upgrade flag */
#define OTA_UPGRADE_FLAG_ADDR     (APP_START_ADDR - 0x800)
```

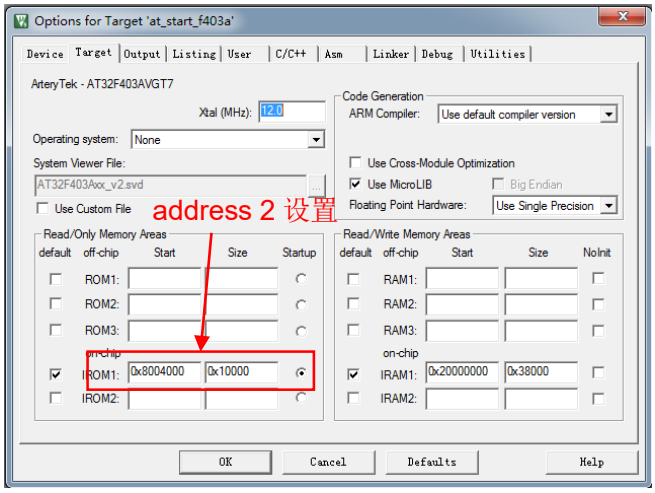
address 2 设置

2.4 App project 设置

OTA demo 提供了 2 个 App 程序供测试用，皆以 address 2（0x800 4000）为起始地址。App1 LED3 闪烁，App2 LED4 闪烁。以 App1 为例，设计步骤如下：

1) Keil工程设置

图 7. App project 中 address 2 在 Keil 设置



2) App1 源程序设置

图 8. App project 向量表偏移在程序中设置

```
/* config vector table offset */
nvic_vector_table_set(NVIC_VECTTAB_FLASH, 0x4000);
```

中断向量表偏移地址修改

3) 编译生成bin文件

通过 User 选项卡，设置编译后调用 fromelf.exe，根据.axf 文件生成.bin 文件，用于 OTA 更新。通过以上 3 个步骤，我们就可以得到一个.bin 的 APP 程序，通过 Bootloader 程序即可实现更新。

4) 开启debug App code功能

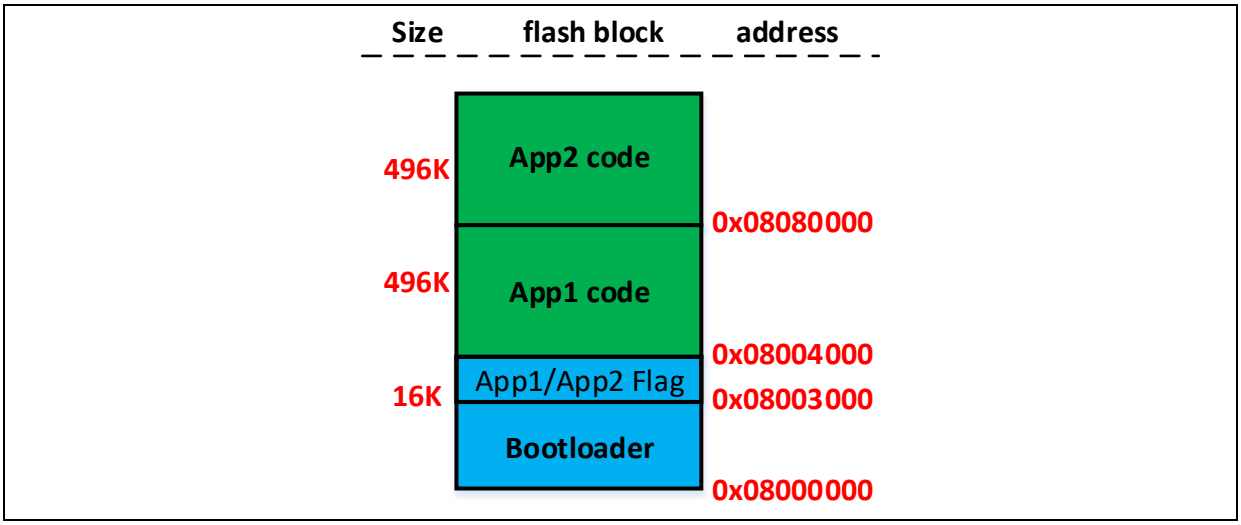
如果在设计 App code 过程中需要对 App project 进行单独调试，请按照以下操作。

- 先下载Bootloader工程
- 再调试App工程

3 Dual app OTA 程序设置

3.1 地址分布

图 9. Flash 地址分配

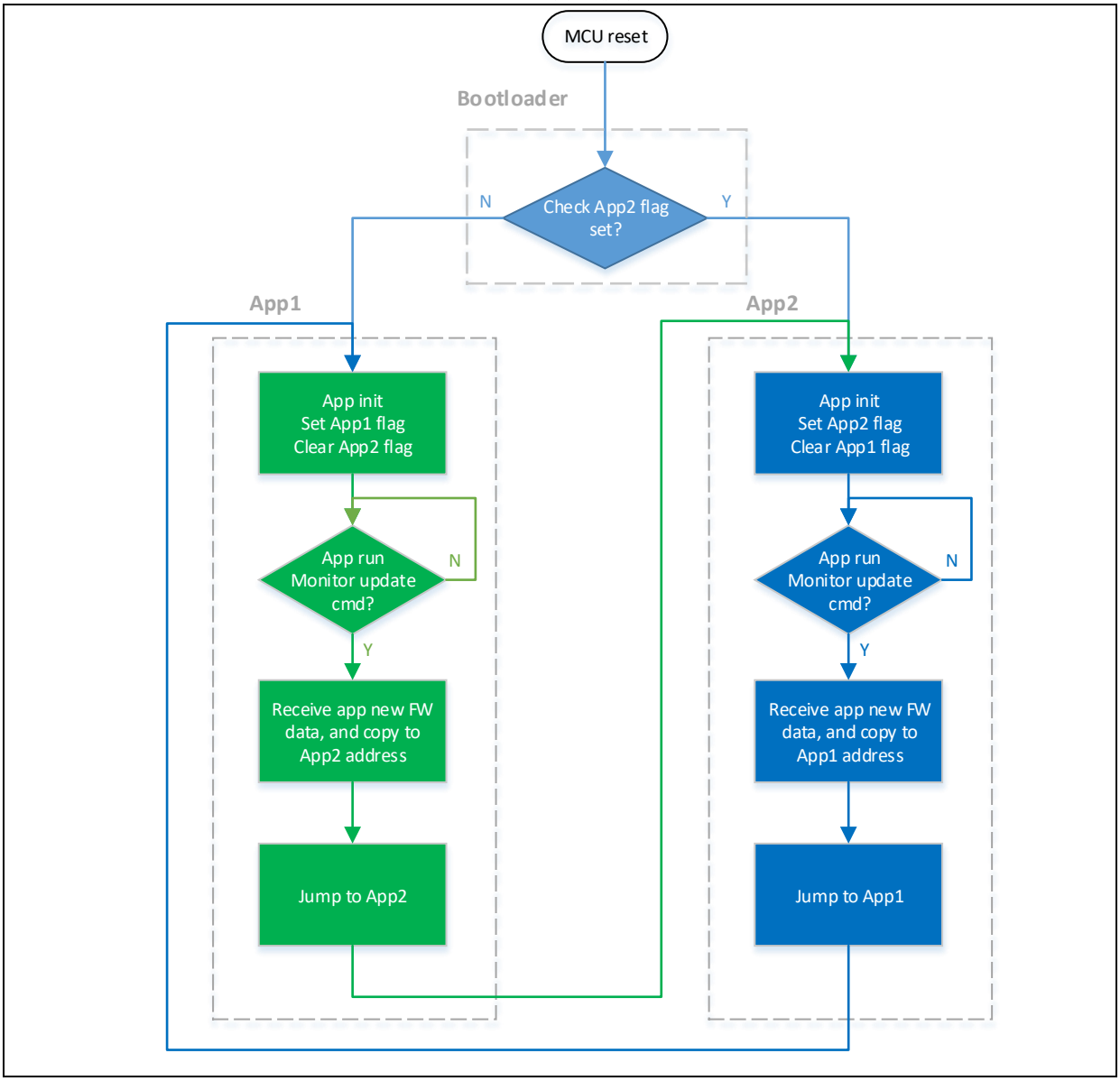


注：Bootloader区域最后2个扇区，用于存放App是否正常的flag，用户编译修改Bootloader时，要保证不覆盖flag的地址。

3.2 执行流程

OTA 分为 Bootloader、App1 和 App2 三部分，应用在 App1 或 App2 中执行。程序执行整体流程框图如下：

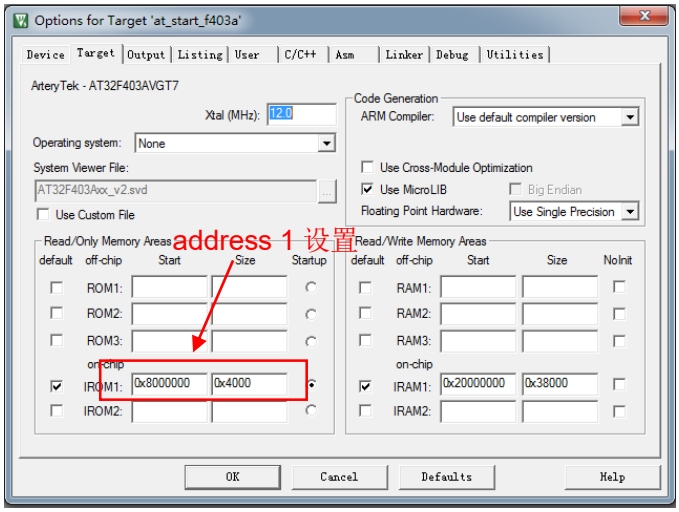
图 10. 程序执行流程



3.3 Bootloader project 设置

3) Keil设置

图 11. Bootloader project 中 address 1 在 Keil 设置



4) Bootloader源程序修改ota.h文件中

图 12. Bootloader project 中 address 2 在程序中设置

```
/* app starting address */
#define APP1_START_ADDR                 0x08004000
#define APP2_START_ADDR                 0x08080000
/* the previous sector of app starting address is ota upgrade flag */
#define OTA_UPGRADE_APP1_FLAG_ADDR      (APP1_START_ADDR - 0x800)
#define OTA_UPGRADE_APP2_FLAG_ADDR      (APP1_START_ADDR - 0x1000)
```

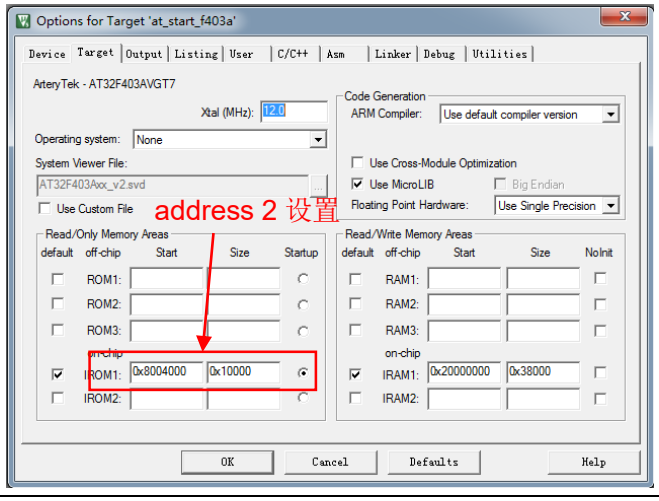
address 设置

3.4 App project 设置

OTA demo 提供了 2 个 App 程序供测试用，app_led3_toggle 以 0x800 4000 为起始地址，app_led4_toggle 以 0x8080000 为起始地址。App1 LED3 闪烁，App2 LED4 闪烁。以 App1 为例，设计步骤如下：

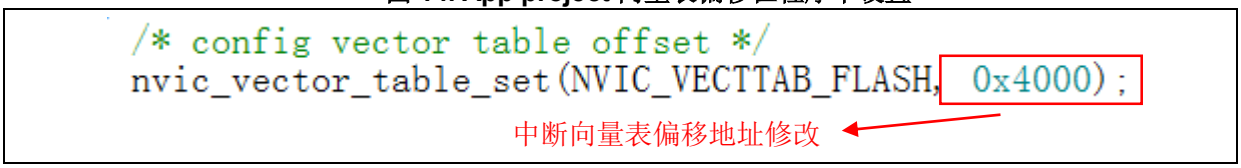
5) Keil工程设置

图 13. App project 中 address 2 在 Keil 设置



6) App1 源程序设置

图 14. App project 向量表偏移在程序中设置



7) 编译生成bin文件

通过 User 选项卡，设置编译后调用 fromelf.exe，根据.axf 文件生成.bin 文件，用于 OTA 更新。通过以上 3 个步骤，我们就可以得到一个.bin 的 APP 程序，通过 Bootloader 程序即可实现更新。

8) 开启debug App code功能

如果在设计 App code 过程中需要对 App project 进行单独调试，请按照以下操作。

a) 先下载Bootloader工程

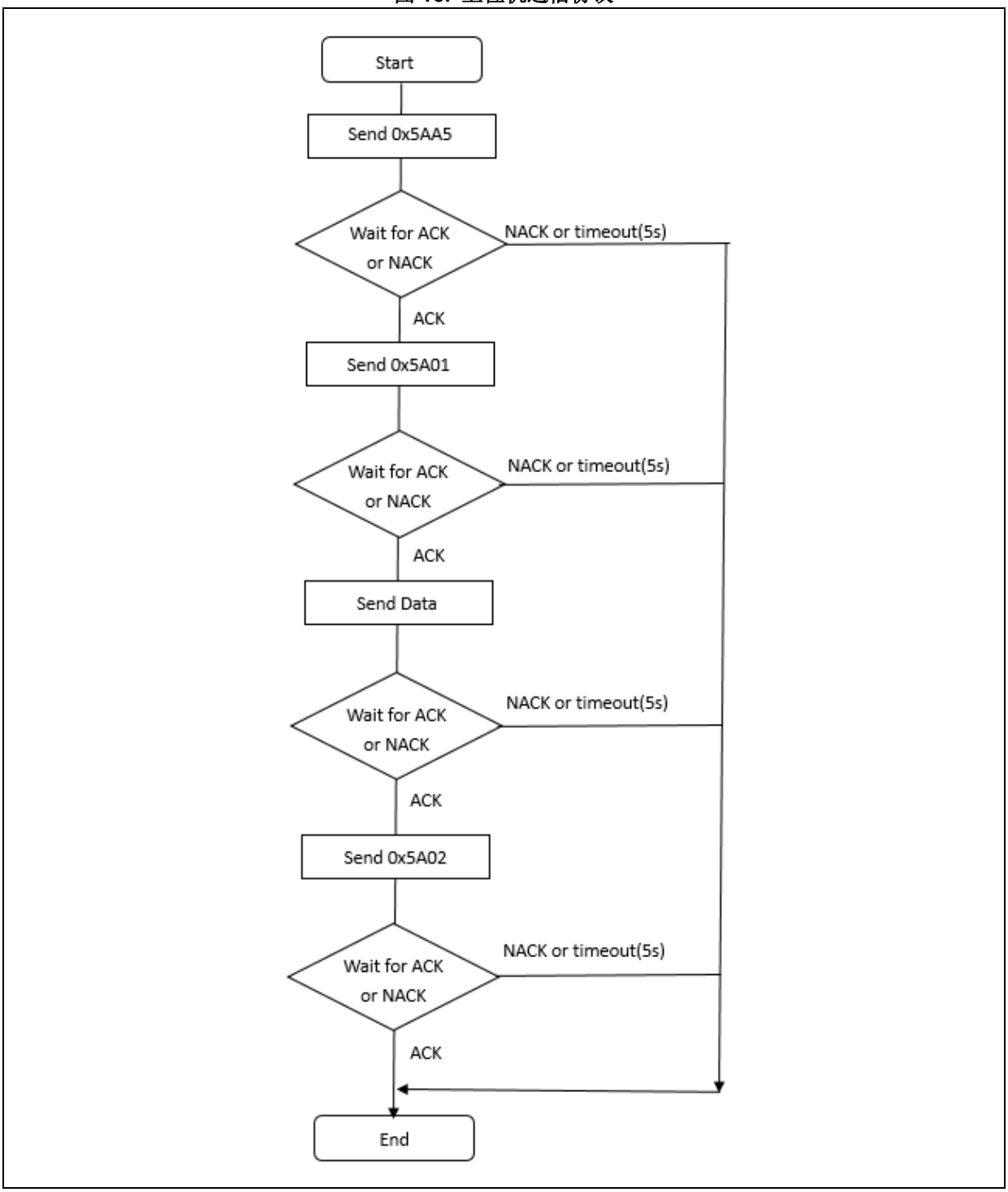
b) 再调试App工程

4 Bootloader/App 与上位机串口通信协议

程序与上位机通信，接收固件升级数据，上位机端和嵌入式端通信协议如下：

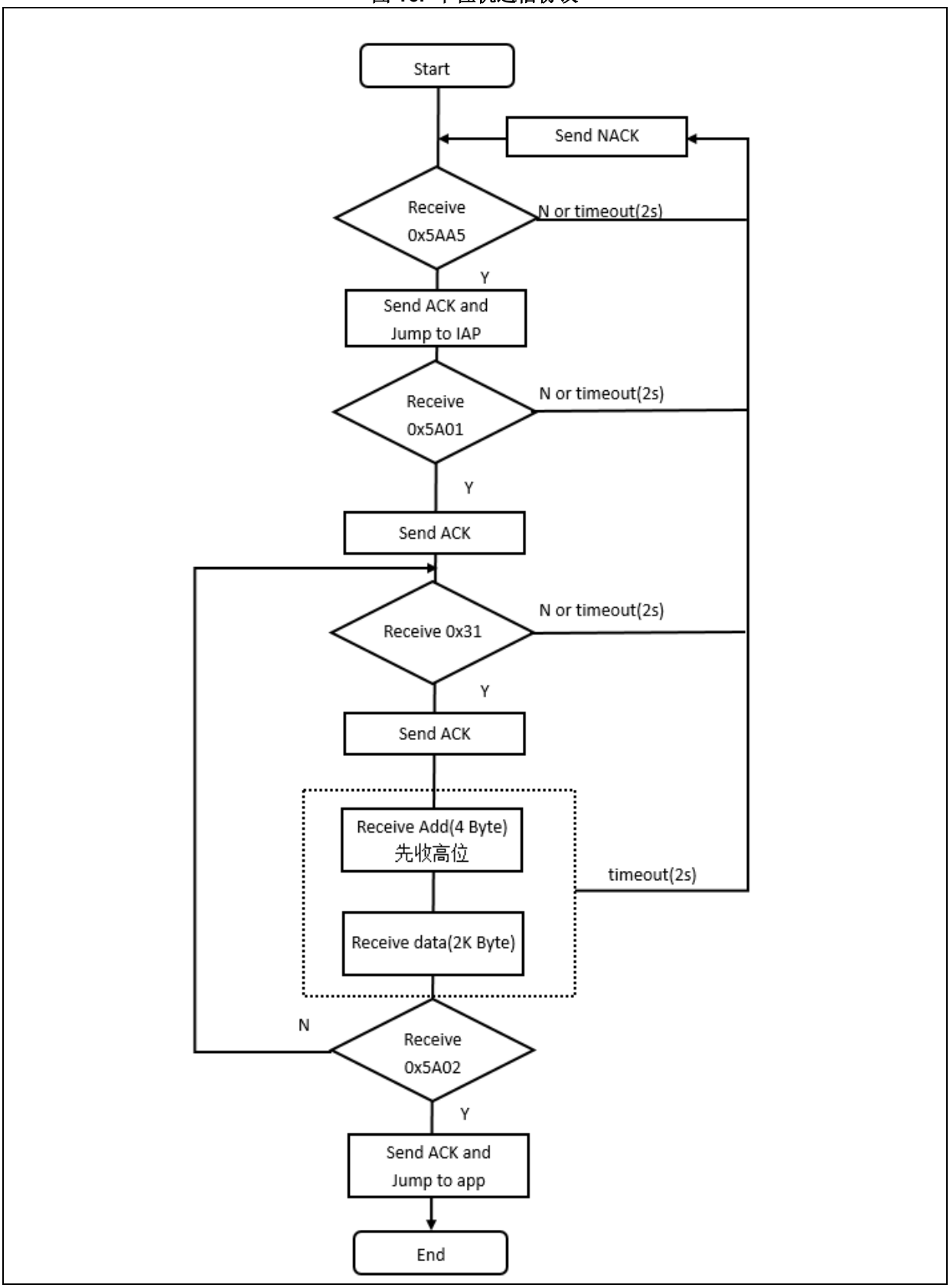
1) 上位机通信协议

图 15. 上位机通信协议



2) 嵌入式端下位机通信协议

图 16. 下位机通信协议



注: *ACK: 0xCCDD*

NACK: 0xEEFF

Data: 0x31+ Addr + 数据 + chenksum（1byte）

Addr：4bytes，高位在前

Kbytes，下载数据，不足2K内容填充0xFF

Checksum：1byte，4bytes的Addr + 2KBytes数据的校验和的低八位

5 版本历史

表 1. 文档版本历史

日期	版本	变更
2021.12.31	2.0.0	最初版本

重要通知 – 请仔细阅读

买方自行负责对本文所述雅特力产品和服务的选择和使用，雅特力概不承担与选择或使用本文所述雅特力产品和服务相关的任何责任。

无论之前是否有过任何形式的表示，本文档不以任何方式对任何知识产权进行任何明示或默示的授权或许可。如果本文档任何部分涉及任何第三方产品或服务，不应被视为雅特力授权使用此类第三方产品或服务，或许可其中的任何知识产权，或者被视为涉及以任何方式使用任何此类第三方产品或服务或其中任何知识产权的保证。

除非在雅特力的销售条款中另有说明，否则，雅特力对雅特力产品的使用和/或销售不做任何明示或默示的保证，包括但不限于有关适销性、适合特定用途（及其依据任何司法管辖区的法律的对应情况），或侵犯任何专利、版权或其他知识产权的默示保证。

雅特力产品并非设计或专门用于下列用途的产品：（A）对安全性有特别要求的应用，例如：生命支持、主动植入设备或对产品功能安全有要求的系统；（B）航空应用；（C）航天应用或航天环境；（D）武器，且/或（E）其他可能导致人身伤害、死亡及财产损害的应用。如果采购商擅自将其用于前述应用，即使采购商向雅特力发出了书面通知，风险及法律责任仍将由采购商单独承担，且采购商应独力负责在前述应用中满足所有法律和法规要求。

经销的雅特力产品如有不同于本文档中提出的声明和/或技术特点的规定，将立即导致雅特力针对本文所述雅特力产品或服务授予的任何保证失效，并且不应以任何形式造成或扩大雅特力的任何责任。