应用笔记

Application Note

文档编号：**AN1084**

APM32F4xx_IAP 应用笔记

版本：**V1.0**

1 引言

本应用笔记提供如何在 APM32F4xx 系列上使用 USART1 进行固件 IAP 的指南，包括接口框图、代码实现和应用方法。

In Application Programing(IAP)，在应用编程是指由开发者实现 MCU 的 bootloader 功能。可以在产品中为用户提供应用程序的下载、校验、增量更新、升级或恢复等支持。

APM32F4xx 微控制器内置 1MB 的 Flash 空间，可以使用其中一小部分作为 IAP 代码的存储空间，以实现自定义 IAP 的功能。

目录

2 IAP 简介

通常所说的 IAP 程序，也有说法是叫做自定义 bootloader，其实就是一段启动程序，它在芯片启动的时候最先被执行，可以用来做一些硬件的初始化或者用作固件热更新，当初始化完成之后跳转到对应的应用程序中去。

IAP 程序需要通过下载器烧写到芯片中，而 APP 则可以通过有线方式的 UART、IIC、USB、SPI 等总线来通过 IAP 来更新，视所设计的 IAP 程序而定。另外，对于无线方式热更新 APP，一般是用 WiFi、bluetooth 通 UAR 透传的方式烧写芯片 APP 程序。

2.1 IAP 功能

1.在一定时间内判断是否需要更新 APP，如果需要则接收 APP 程序并将其烧写到指定地址的 flash 空间里。并从该 IAP 程序中获取栈顶指针和复位程序指针，然后跳转执行程序。

2.等待超时或接收到用户指令后，则在该 IAP 程序地址直接获取之前的栈顶指针和复位程序指针并跳转执行。

3 自定义 IAP 功能

IAP（ In Application Programming）在线应用编程，即是用户可以使用自定义程序对单片机用户 Flash 的某一区域进行烧写，通过 IAP 程序来完成对 App 程序的更新升级。实际工作中，是为了产品发布后，可以方便的使用预留的通信接口（串口、USB、网口、蓝牙等）来完成程序的升级，避免需要把产品拆开，再使用仿真下载器来更新应用程序。

而要实现 IAP 功能，要做两部分工作。

1.Bootloader 程序；

2.App 程序。

注意：

SDK 中的例程包含对两个不同地址的 APP 程序进行更新的功能， APP1 存放于 0x8004000 ~ 0x8008000 地址，APP2 存放于 0x8008000 ~ 0x800C000 地址中。这里为了方便讲述，只介绍了 IAP + APP1 的情况。至于 IAP + APP1 + APP2 的情况，大家可以类比来理解。

3.1 总体设计架构

在 APM32F407IGxx 中，芯片 flash 空间为 1MB 大小，那么我们定义 IAP 架构如下。包括 Ymodem 协议，USART1 收发和菜单，flash 操作，IAP 空间配置及应用程序跳转等部分。

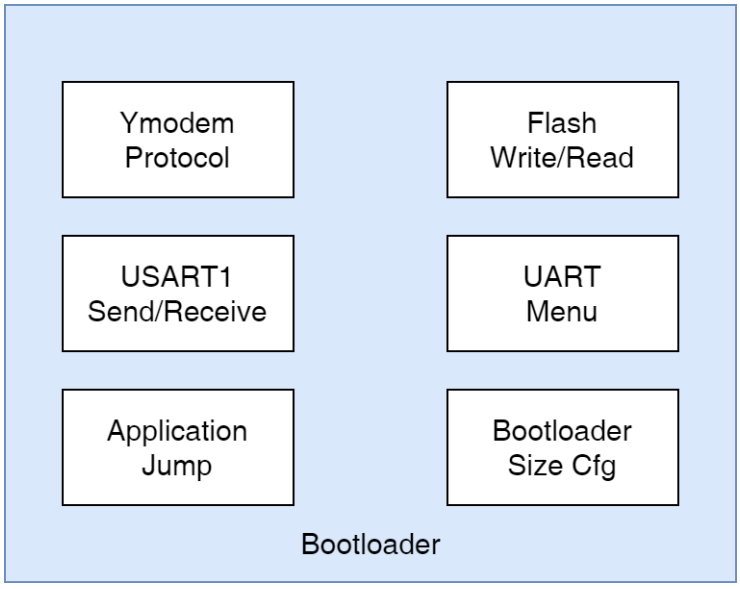

图 1 总体架构

3.2 代码地址划分

IAP 程序区设置于 0x8000000 ~ 0x8004000 地址范围内。Application 程序区设置于 0x8004000 ~ 0x8100000 地址范围。这里要注意两点：

一是要注意以 1024 bytes 的倍数划分地址，其他特殊地址会发生异常。

二是要注意 F4xx 系列是以 sector 扇区为单位来擦除 flash，分配 App 和 bootloader 地址时要留意。

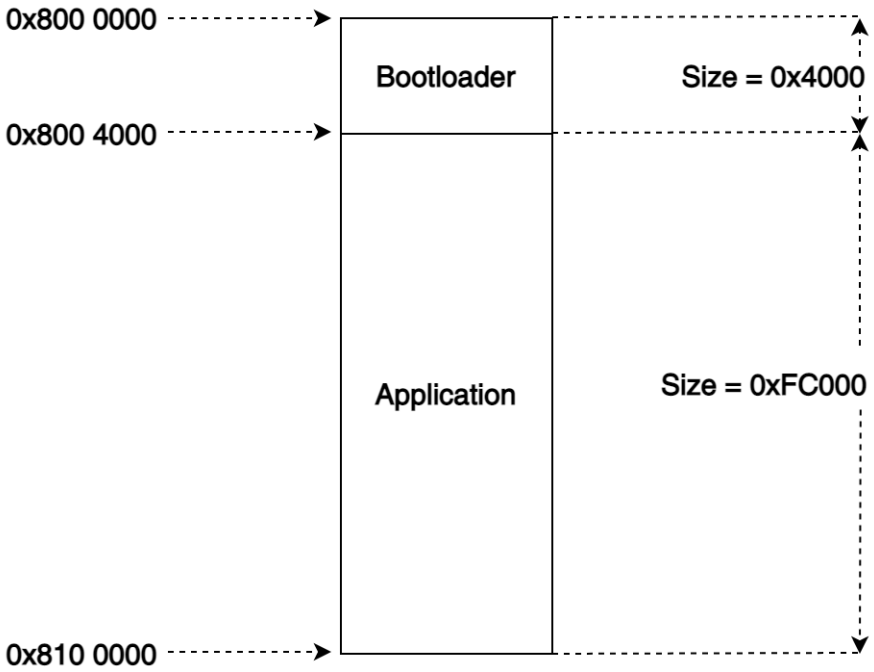

图 2 代码地址划分

3.3 代码空间划分

以下是具体的代码空间划分图示。从图中可以知道，在加入 IAP 和 APP1 程序代码后，整个代码空间中存在两个中断向量表、栈顶地址和中断服务函数。

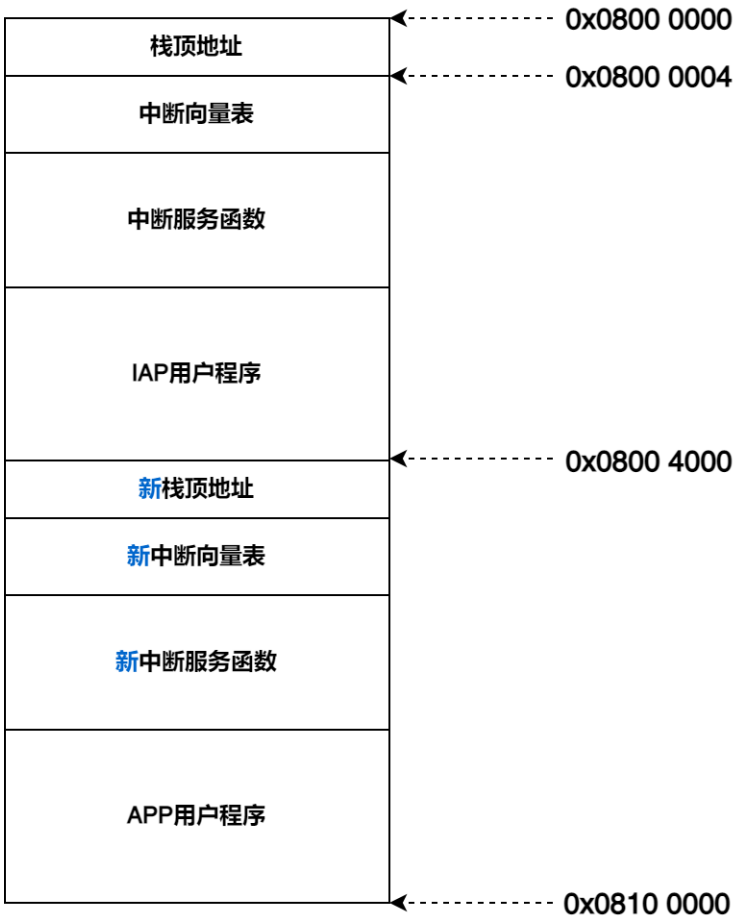

图 3 代码空间划分

3.4 IAP 功能执行流程

程序启动后，先从 0x08000004 处取出复位中断向量地址，执行完复位中断函数后跳转到 IAP 程序 main 函数中执行[①]。

当发生中断请求后，程序跳转到中断向量表中取出中断函数入口地址，再跳转到中断服务函数中执行[②]，执行完中断函数后返回 main 函数中[③]，然后执行 IAP 过程，成功后跳转到 APP 程序[④]。

从偏移后的中断向量表得到相应中断函数地址，执行相应新的中断服务函数后，回到 APP 的 main 函数中[⑤]。后面[⑥⑦⑧]的过程和前述一致，不再赘述。

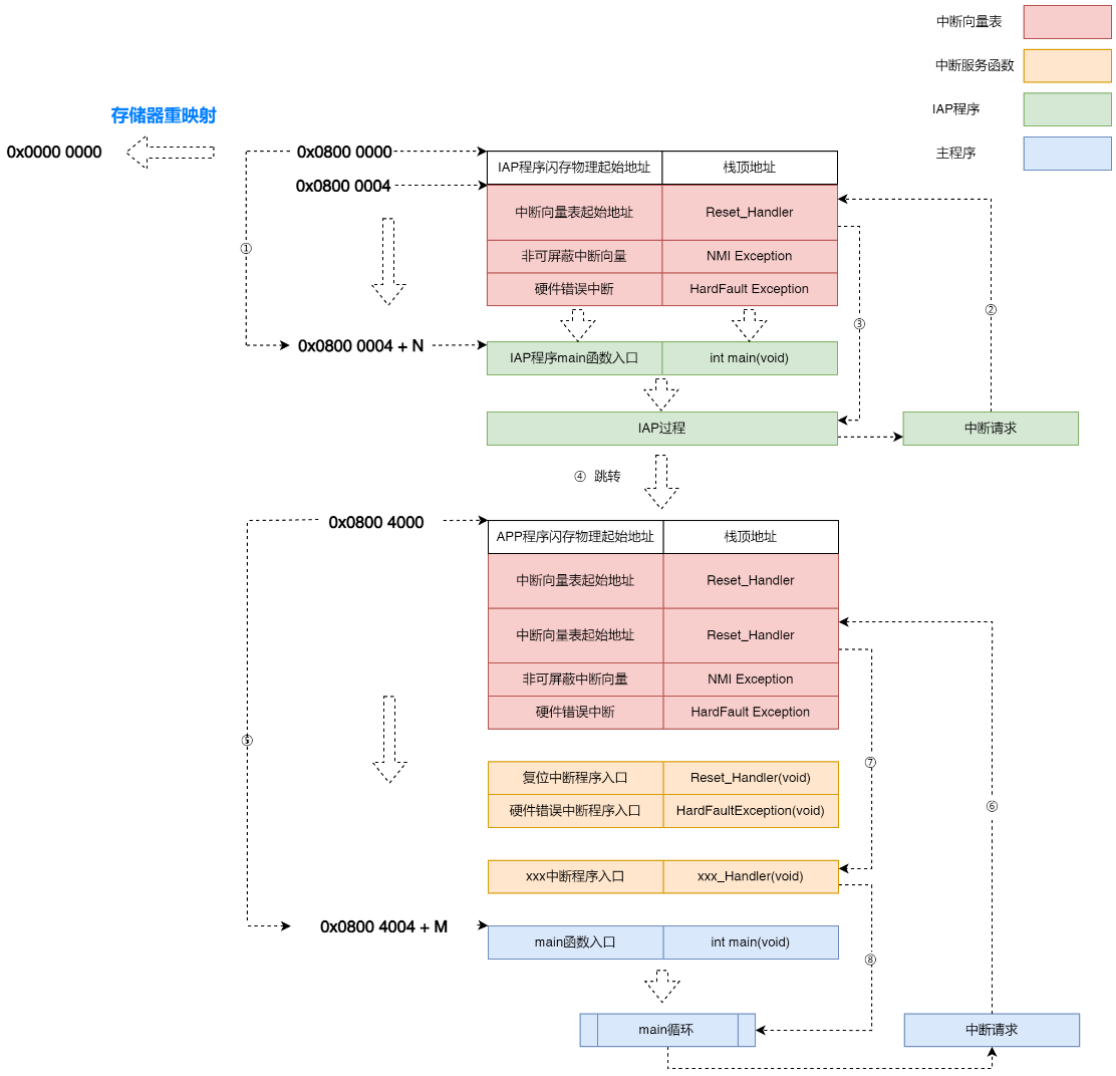

图 4 执行过程

3.5 中断向量表

3.5.1 什么是中断向量表

如“IAP 执行过程”中所示，中断向量表是存放在 Flash 区从 0x08000004 地址(默认)开始的一个数组，数组元素的大小为 4 个字节，即每一表项的大小为 4 个字节，这些数组在启动文件中已经初始化好。不同系列根据中断向量的多少，有不同的数组长度。

APM32 根据内核和外设中断的优先级，把内核和外设的中断服务函数的地址放在这个数组里面，数组的下标跟中断的优先级对应，也把这个中断的编号叫做中断向量，标号越小，优先级越高。

在启动文件执行的时候，内核和外设的中断服务函数地址都是确定的，地址就在中断向量表中，并且在启动文件里面已经写好了中断服务函数，只是这些中断服务函数为空，而且带[weak]弱定义。

如果使用到相关中断则需要在用户程序里重新实现对应的中断服务函数，而且重写这个中断服务函数的时候，函数名必须跟启动文件里定义好的中断函数名对应，这是因为函数名对应的就是中断服务函数的地址。

当中断发生时，因为每个中断的中断向量不一样，CPU 会首先去取向量。然后根据向量来查询中断向量表，最后根据对应的地址找到对应的中断服务函数，从而实现整个中断的响应过程。

3.5.2 中断向量表的设置

如前面介绍所述，中断向量表是默认存放在 Flash 区从 0x0000 0004 地址（默认存储器映射后为 0x0800 0004）的。而我们在划分程序区时改变了 APP 程序的起始地址为 0x08004000，所以在 APP 程序中我们要设置新的中断向量表的地址。

APM32F4xx 芯片 SDK 的 system_apm32f4xx.c 文件中可以找到 VECT_TAB_OFFSET 这个向量表偏移量宏定义来重新设置中断向量表的地址，也即是修改 SCB->VTOR 向量表偏移量寄存器。这里我们不改库文件，直接在 main 函数开头设置偏移地址。

```
int main(void)
{
    SCB->VTOR = FMC_BASE | 0x4000;

    while (1)
    {
    }
}
```

相应的工程文件中 ROM 起始地址也修改为 0x08004000。

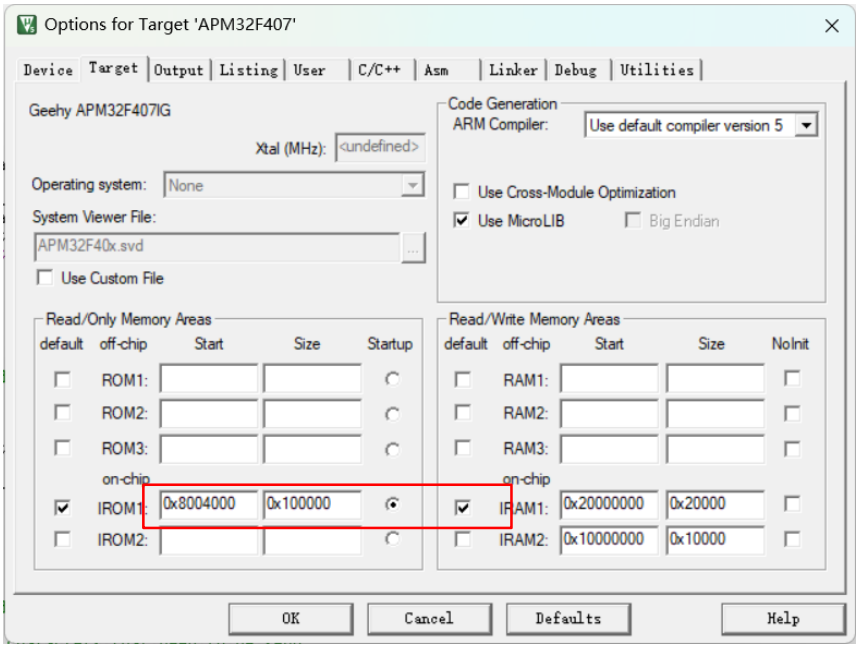

图 5 ROM 配置

4 IAP 的设计和应用

4.1 硬件设计

本应用说明对应的 IAP 例程，是使用了 UART 方式来实现 IAP 功能的。对外接口使用了 USART1 的 TX 和 RX 引脚，即 PA9 和 PA10 引脚。

4.2 软件设计

4.2.1 IAP 程序设计

整个 IAP 程序包括 Flash 的读写，USART1 及串口菜单的显示，以及 Ymodem 协议的文件发送和接收。

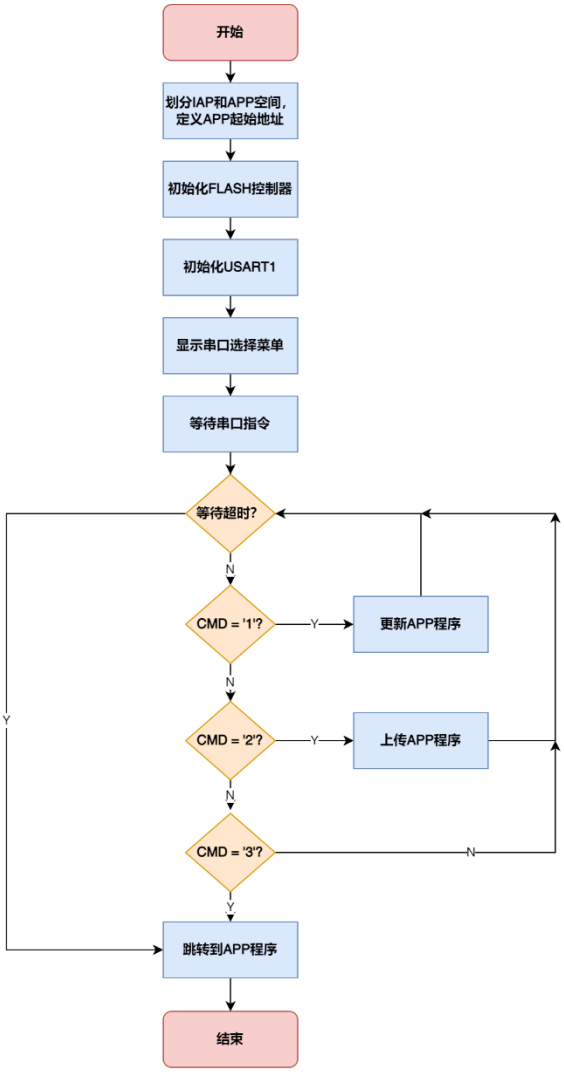

图 6 IAP 程序流程

4.2.2 APP 程序设计

APP 程序的设计就比较简单，只是循环亮灭一个 LED 灯。但要注意的是，如上一章提到的“中断向量表”设置，在 APP 程序一开始的 main 函数就需要先设置好中断向量表的偏移地址。

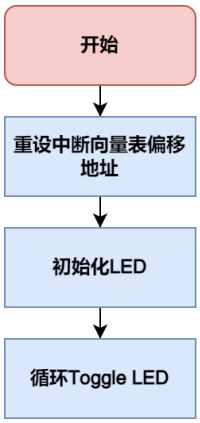

图 7 APP 程序流程

4.3 APP 固件升级

4.3.1 APP 可执行文件的生成

APP 是主用户程序，在完成 IAP 程序的设计后，就要把 APP 的更新文件生成，然后通过一定的协议传输给 IAP 程序来进行 APP 固件的更新。一般来说 APP 更新文件的文件类型为.bin 文件，该文件可以直接拷贝到 flash 中运行。

在 Keil MDK Option 配置的 User 选项卡中配置以下命令，即可使用 fromelf.exe 生成 bin 文件，默认生成在工程目录。

```
/** 命令*/
fromelf.exe --bin -o ./@L.bin !L
```

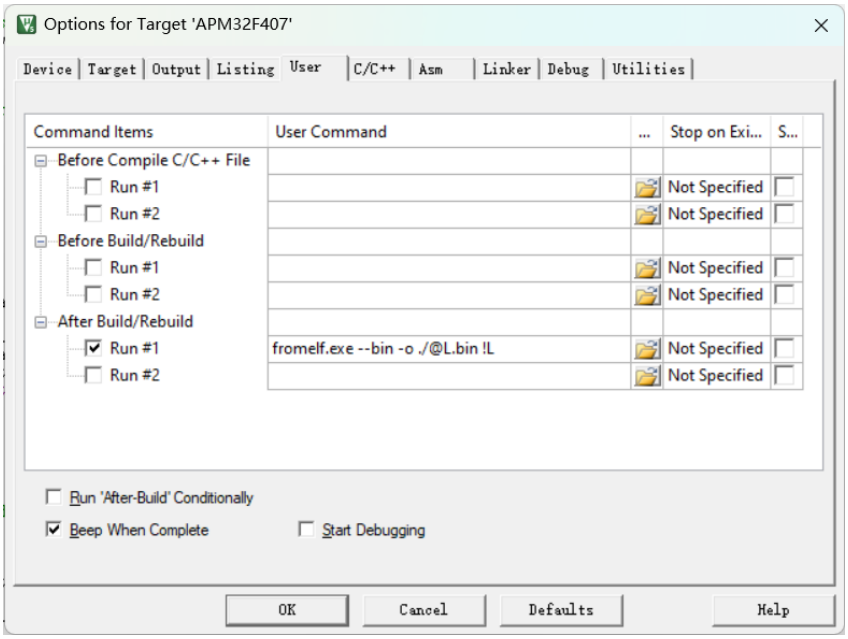

图 8 生成 bin 文件

4.3.2 固件升级

生成 bin 文件后，就可以根据 IAP 程序选择某种传输方式进行 bin 文件的传递了。这里使用了 Ymodem 协议和超级终端软件来实现，以下是具体操作。

4.3.2.1 串口菜单

在保持串口连接到目标板的状态下，在下载好 IAP BootLoader 程序的目标板复位后，超级终端软件会显示下图中的串口菜单。

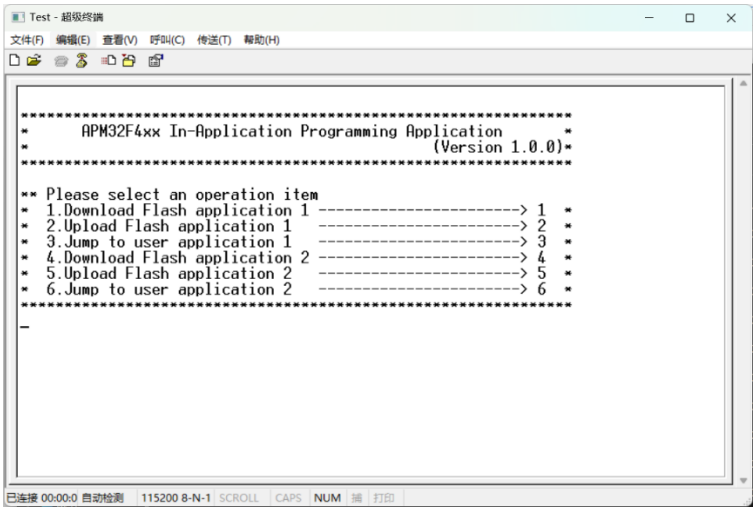

图 9 串口菜单

4.3.2.2 下载 APP1 的 bin 文件到芯片

在串口菜单按下键盘的“1”键，进入等待 bin 文件发送状态。

Application1 对应 IAP_Application1 工程的文件，Application2 对应 IAP_Application2 工程的文件。

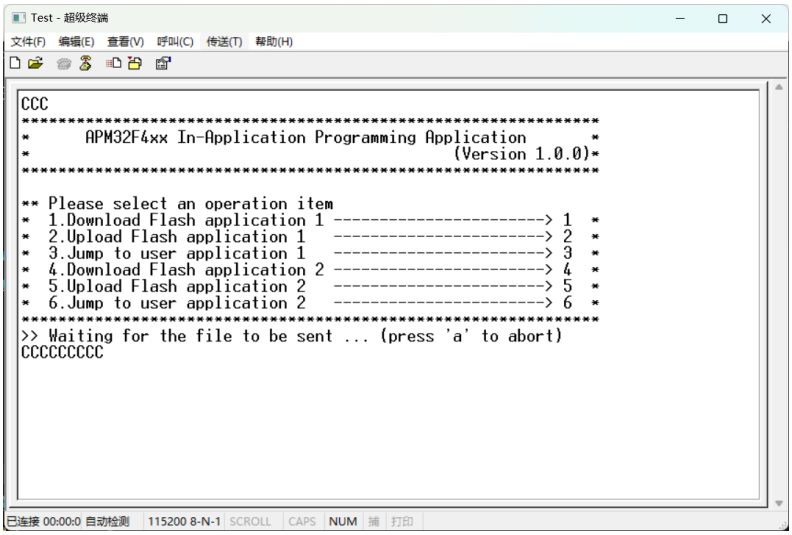

图 10 下载 bin 文件

4.3.2.3 选择要下载的 bin 文件

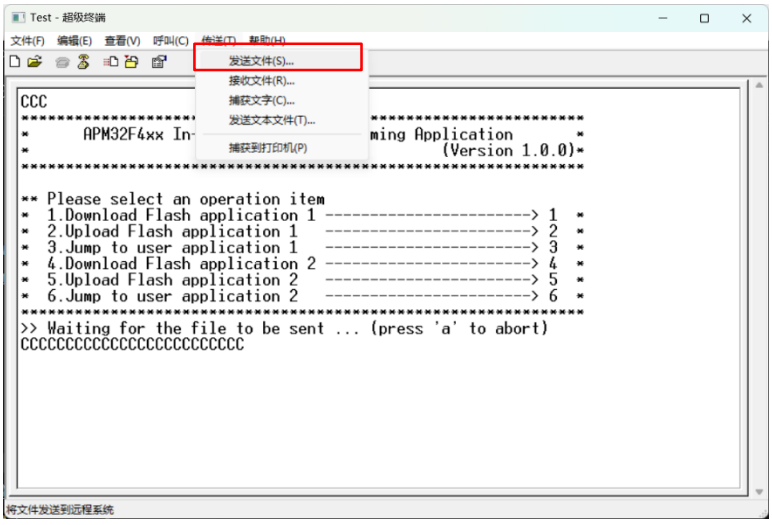

图 11 选择 bin 文件

4.3.2.4 选择 Ymodem 协议并发送文件

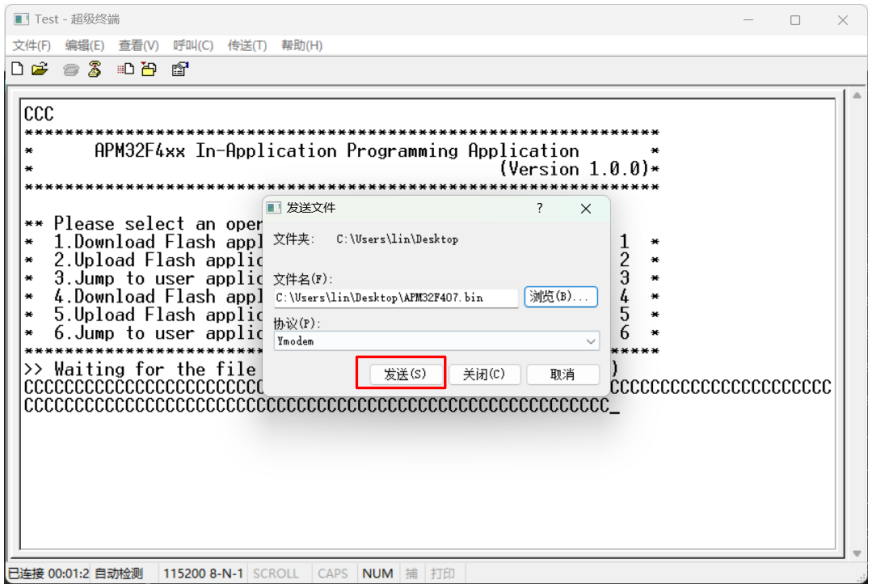

图 12 发送文件

4.3.2.5 下载完成

等待文件发送完成后，软件中会提示“Programming Comleted Successfully！”。然后就可以重新选择所需功能。

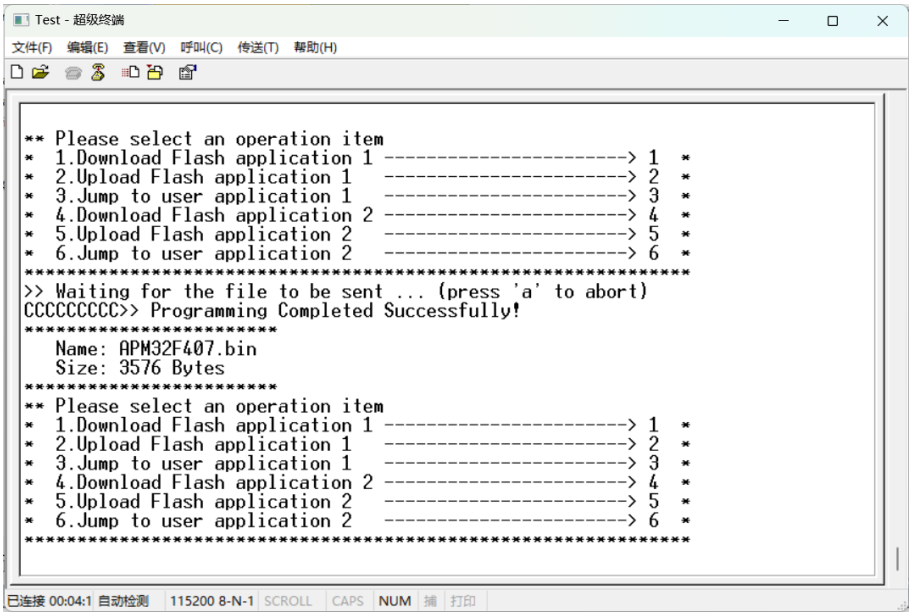

图 13 下载完成

5 版本历史

表格 1 文件版本历史

日期	版本	变更历史
2022.05.31	1.0	新建

声明

本手册由珠海极海半导体有限公司（以下简称“极海”）制订并发布，所列内容均受商标、著作权、软件著作权相关法律法规保护，极海保留随时更正、修改本手册的权利。使用极海产品前请仔细阅读本手册，一旦使用产品则表明您（以下称“用户”）已知悉并接受本手册的所有内容。用户必须按照相关法律法规和本手册的要求使用极海产品。

1、权利所有

本手册仅应当被用于与极海所提供的对应型号的芯片产品、软件产品搭配使用，未经极海许可，任何单位或个人均不得以任何理由或方式对本手册的全部或部分内容进行复制、抄录、修改、编辑或传播。

本手册中所列带有“®”或“TM”的“极海”或“Geehy”字样或图形均为极海的商标，其他在极海产品上显示的产品或服务名称均为其各自所有者的财产。

2、无知识产权许可

极海拥有本手册所涉及的全部权利、所有权及知识产权。

极海不应因销售、分发极海产品及本手册而被视为将任何知识产权的许可或权利明示或默示地授予用户。

如果本手册中涉及任何第三方的产品、服务或知识产权，不应被视为极海授权用户使用前述第三方产品、服务或知识产权，除非在极海销售订单或销售合同中另有约定。

3、版本更新

用户在下单购买极海产品时可获取相应产品的最新版的手册。

如果本手册中所述的内容与极海产品不一致的，应以极海销售订单或销售合同中的约定为准。

4、信息可靠性

本手册相关数据经极海实验室或合作的第三方测试机构批量测试获得，但本手册相关数据难免会出现校正笔误或因测试环境差异所导致的误差，因此用户应当理解，极海对本手册中可能出现的该等错误无需承担任何责任。本手册相关数据仅用于指导用户作为性能参数参照，不构成极海对任何产品性能方面的保证。

用户应根据自身需求选择合适的极海产品，并对极海产品的应用适用性进行有效验证和测试，以确认极海产品满足用户自身的需求、相应标准、安全或其它可靠性要求；若因用户未充分对极海产品进行有效验证和测试而致使用户损失的，极海不承担任何责任。

5、合规要求

用户在使用本手册及所搭配的极海产品时，应遵守当地所适用的所有法律法规。用户应了解产品可能受到产品供应商、极海、极海经销商及用户所在地等各国有关出口、再出口或其它法律的限制，用户（代表其本身、子公司及关联企业）应同意并保证遵守所有关于取得极海产品及／或技术与直接产品的出口和再出口适用法律与法规。

6、免责声明

本手册由极海“按原样”（as is）提供，在适用法律所允许的范围内，极海不提供任何形式的明示或暗示担保，包括但不限于对产品适销性和特定用途适用性的担保。

对于用户后续在针对极海产品进行设计、使用的过程中所引起的任何纠纷，极海概不承担责任。

7、责任限制

在任何情况下，除非适用法律要求或书面同意，否则极海和/或以“按原样”形式提供本手册的任何第三方均不承担损害赔偿责任，包括任何一般、特殊因使用或无法使用本手册相关信息而产生的直接、间接或附带损害（包括但不限于数据丢失或数据不准确，或用户或第三方遭受的损失）。

8、适用范围

本手册的信息用以取代本手册所有早期版本所提供的信息。