Application Note

Application Note

Document No.: AN1084

APM32F4xx_IAP Application Note

Version: V1.0

1 Introduction

This application note provides a guide to how to use USART1 for firmware IAP on APM32F4xx series, including interface block diagram, code implementation and application method.

In Application Programing (IAP) means that the developer implements the bootloader function of MCU. It can provide users with support of application download, verification, incremental update, upgrade or recovery in the product.

APM32F4xx MCU has built-in 1MB Flash space, and a small part of it can be used as the storage space of IAP codes to implement the function of customized IAP.

Contents

2 IAP Introduction

The so-called IAP program, also known as the custom bootloader, is actually a segment of boot program. It is first executed when the chip is started. It can be used for initialization of some hardware or hot update of firmware, and will jump to the corresponding application program after initialization.

The IAP program needs to be written into the chip through the downloader, and the APP can be updated through the IAP using the wired UART, IIC, USB, SPI or other buses, depending on the designed IAP program. In addition, for the wireless hot update of APP, the chip APP program is usually written through UAR pass-through using WiFi and Bluetooth.

2.1 IAP functions

1. Judge whether the APP needs to be updated within a certain period of time. If needed, receive the APP program and burn it into the flash space of the specified address. And obtain the stack top pointer and reset program pointer from the IAP program, and then jump to execute the program.

2. Wait for timeout or after receiving user instruction, directly obtain the previous stack top pointer and reset program pointer at the IAP program address and jump to execute it.

3 Custom IAP Functions

IAP (In Application Programming) means that users can use custom programs to burn a certain area of user Flash of MCU, and update and upgrade App programs through IAP program. In the actual work, it makes it convenient to use the reserved communication interfaces (serial port, USB, network port, Bluetooth, etc.) to complete the upgrade of the program after the product is released, to avoid disassembling the product and use the simulation downloader to update the application programs.

To implement the IAP function, two parts of work need to be done.

1. Bootloader program;

2. App program.

Note:

The examples in the SDK contain the function of updating the APP programs with two different addresses. APP1 is stored in 0x8004000 ~ 0x8008000 addresses, and APP2 is stored in 0x8008000 ~ 0x800C000 addresses. For the convenience of description, only IAP + APP1 situation is introduced here. As for IAP + APP1 + APP2, you can understand it by analogy.

3.1 Overall design architecture

In APM32F407IGxx, the chip flash space is 1MB, so we define the IAP architecture as follows. It includes Ymodem protocol, USART1 receive and transmit and menu, flash operation, IAP space configuration and application jump.

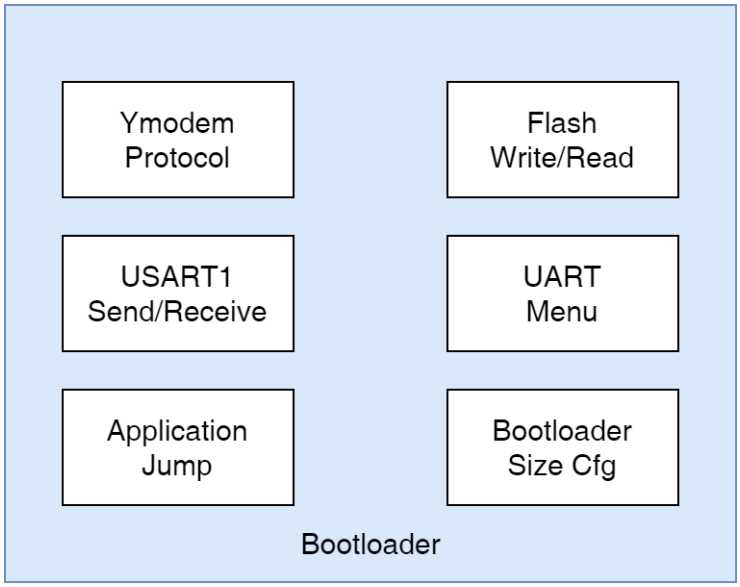

Figure 1 Overall Architecture

3.2 Code address division

The IAP program area is set within the address range of 0x8000000 ~ 0x8004000. The Application program area is set within the address range of 0x8004000 ~ 0x8100000. Here attention should be paid to following two points:

First, pay attention that if the address is divided by a multiple of 1024 bytes, other special addresses will be abnormal.

Second, since F4xx series erases flash in sector, pay special attention when assigning App and bootloader addresses.

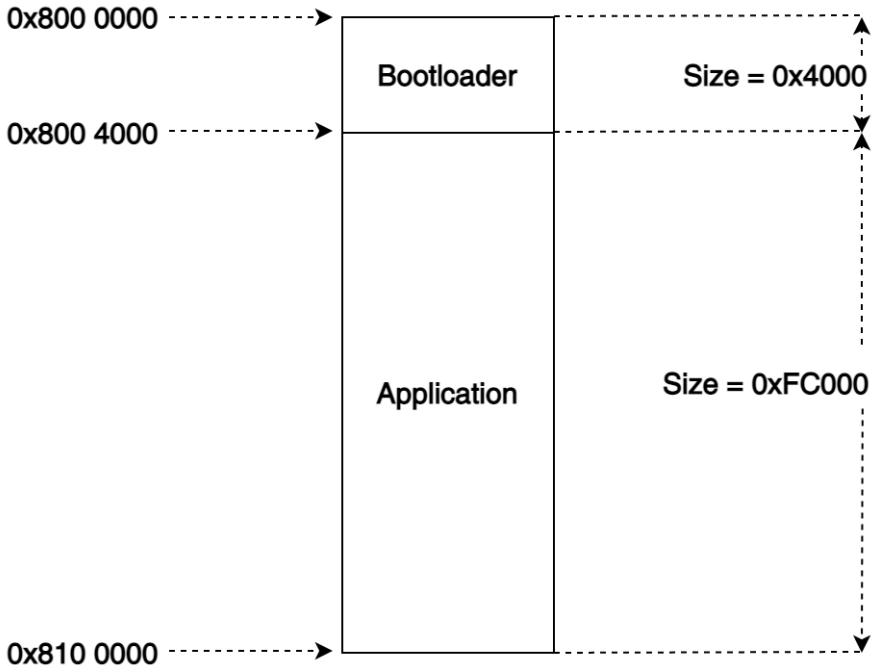

Figure 2 Code Address Division

3.3 Code space division

The following shows a specific code space division diagram. It can be seen from the figure that after IAP and APP1 program codes are added, there are two interrupt vector tables, stack top addresses and interrupt service functions in the whole code space.

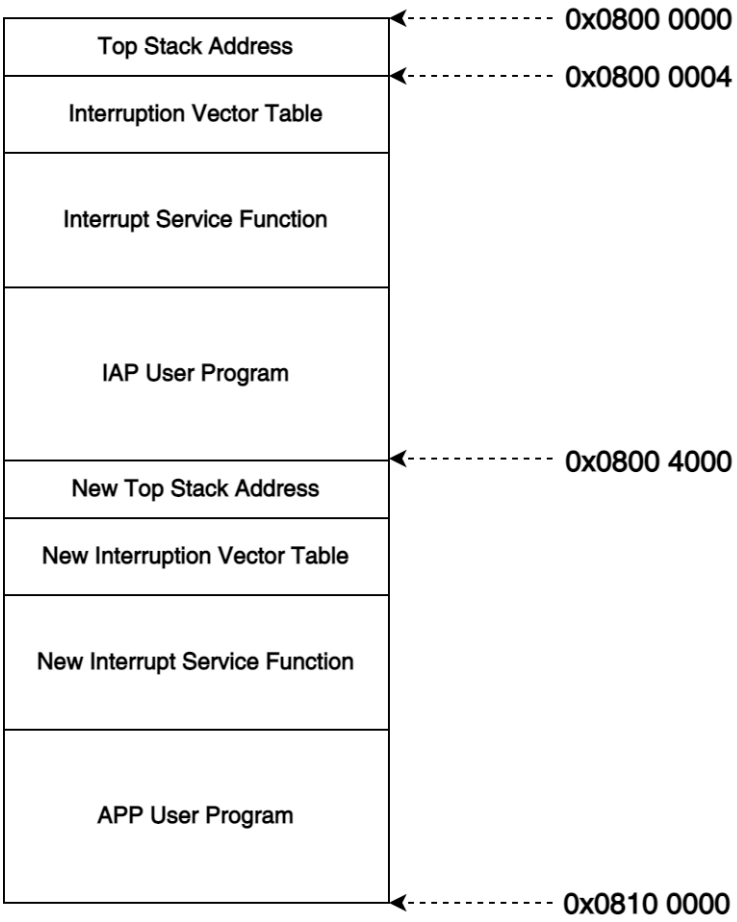

Figure 3 Code Space Division

3.4 IAP function execution process

After the program starts, first take out the reset interrupt vector address from 0x08000004. After the reset interrupt function is executed, jump to the main function of IAP program to execute [①].

When an interrupt request occurs, the program jumps to the interrupt vector table, takes out the interrupt function entry address, and then jumps to the interrupt service function to execute [②]. After the interrupt function is executed, it returns to the main function [③], executes the IAP process, and then jumps to the APP program [④].

It obtains the corresponding interrupt function address from the offset interrupt vector table, and after the corresponding new interrupt service function is executed, it returns to the main function of APP [⑤]. The following [⑥⑦⑧] process is consistent with the above and will not be repeated.

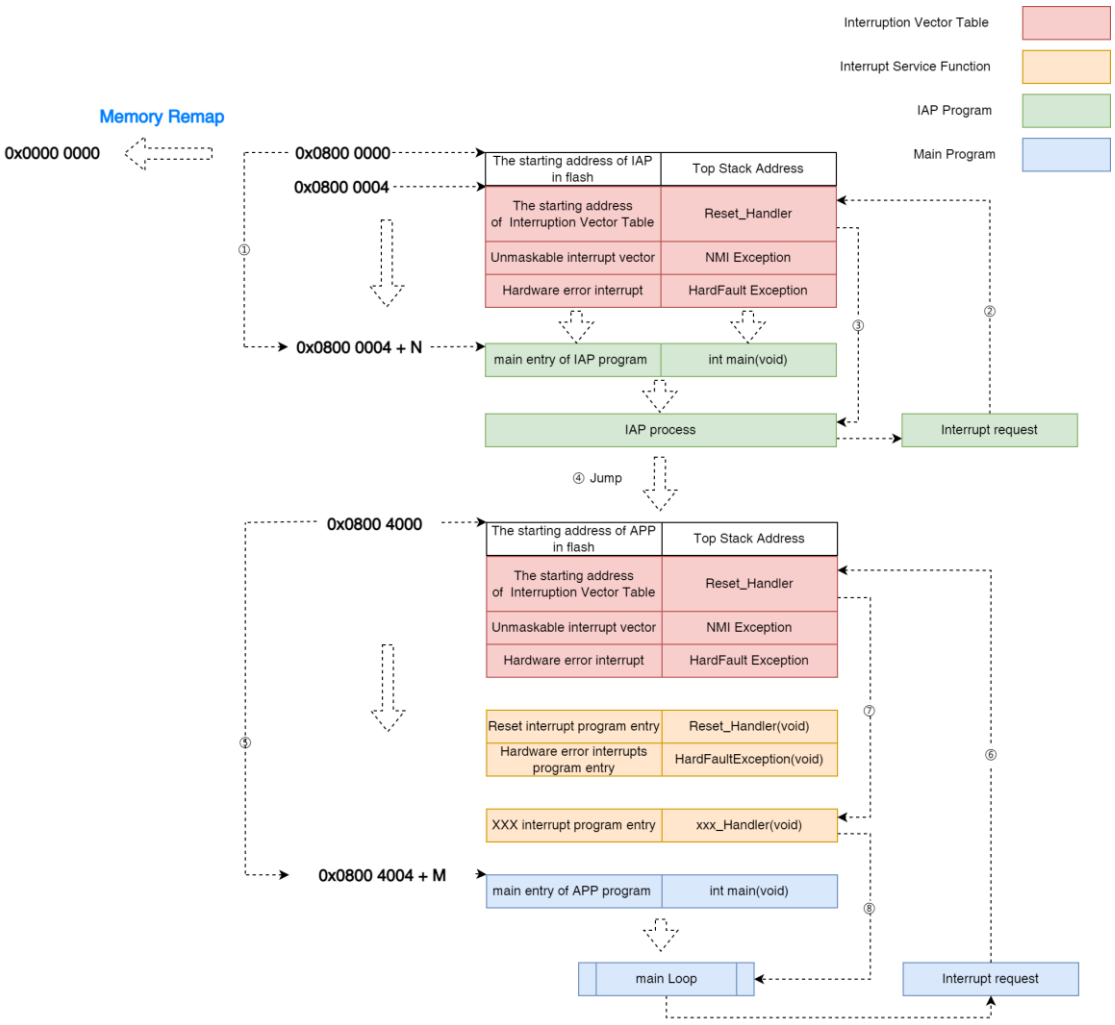

Figure 4 Execution Process

3.5 Interrupt vector table

3.5.1 What is interrupt vector table

As shown in "IAP execution process", the interrupt vector table is an array stored in the Flash area starting from 0x08000004 address (default). The size of the array elements is 4 bytes, and the size of each table item is 4 bytes. These arrays have been initialized in the boot file. The array length of different series is different according to the number of interrupt vectors.

APM32 puts the addresses of interrupt service functions of the core and peripherals into this array according to the priority of interrupts of the core and peripherals. The subscript of the array corresponds to the priority of interrupts, and the number of this interrupt is also called interrupt vector. The smaller the number is, the higher the priority is.

When the boot file is executed, the addresses of the interrupt service functions of the core and peripherals are certain. The addresses are in the interrupt vector table, and the interrupt service functions have been written in the boot file, but these interrupt service functions are empty and have [weak] definition.

If relevant interrupt is used, the corresponding interrupt service function needs to be re-implemented in user program. When rewriting this interrupt service function, the function name must correspond to the interrupt function name defined in the boot file, because the function name corresponds to the address of the interrupt service function.

When an interrupt occurs, because the interrupt vector of each interrupt is different, the CPU will get the vector first. Then query the interrupt vector table by the vector. Finally, find the corresponding interrupt service function according to the corresponding address, to implement the whole interrupt response process.

3.5.2 Setting of interrupt vector table

As mentioned above, the interrupt vector table is stored in the Flash area by default starting from 0x0000 0004 address (0x0800 0004 after the default memory is mapped). Since we changed the start address of the APP program to 0x08004000 when dividing the program area, we need to set the address of the new interrupt vector table in the APP program.

The vector table offset VECT_TAB_OFFSET can be found in the file of system_apm32f4xx.c of APM32F4xx chip SDK. It is macro-defined to reset the address of the interrupt vector table, namely, modify the SCB->VTOR vector table offset register. Here we do not change the library file, but directly set the offset address at the beginning of the main function.

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

    while (1)
    {
    }
}
```

The start address of ROM in the corresponding project file is also modified to 0x08004000.

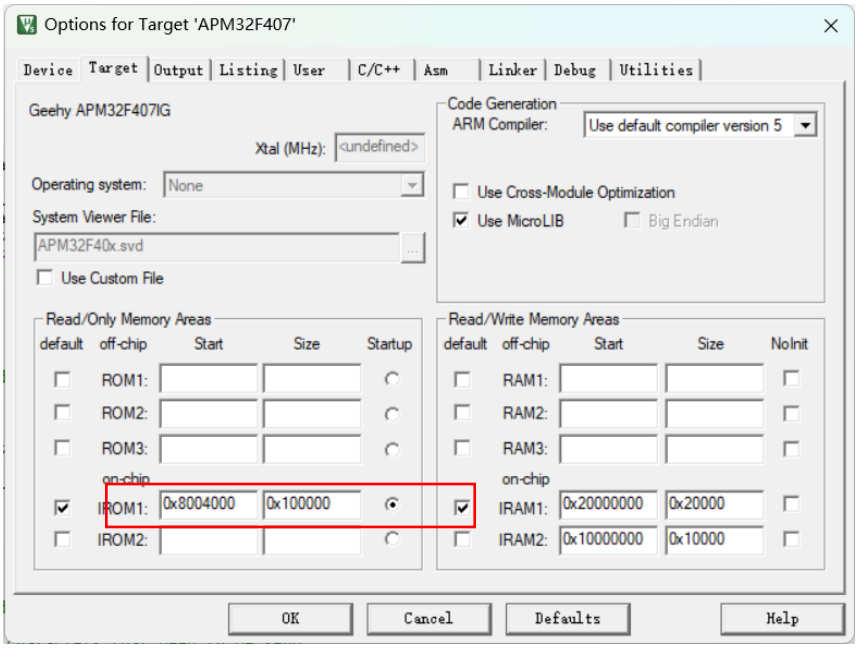

Figure 5 ROM Configuration

4 Design and Application of IAP

4.1 Hardware design

The IAP example corresponding to this application description uses UART to implement the IAP function. The TX and RX pins of USART1, namely PA9 and PA10 pins, are used for external interface.

4.2 Software design

4.2.1 IAP program design

The whole IAP program includes the reading and writing of Flash, the display of USART1 and serial port menu, and the file transmitting and receiving of Ymodem protocol.

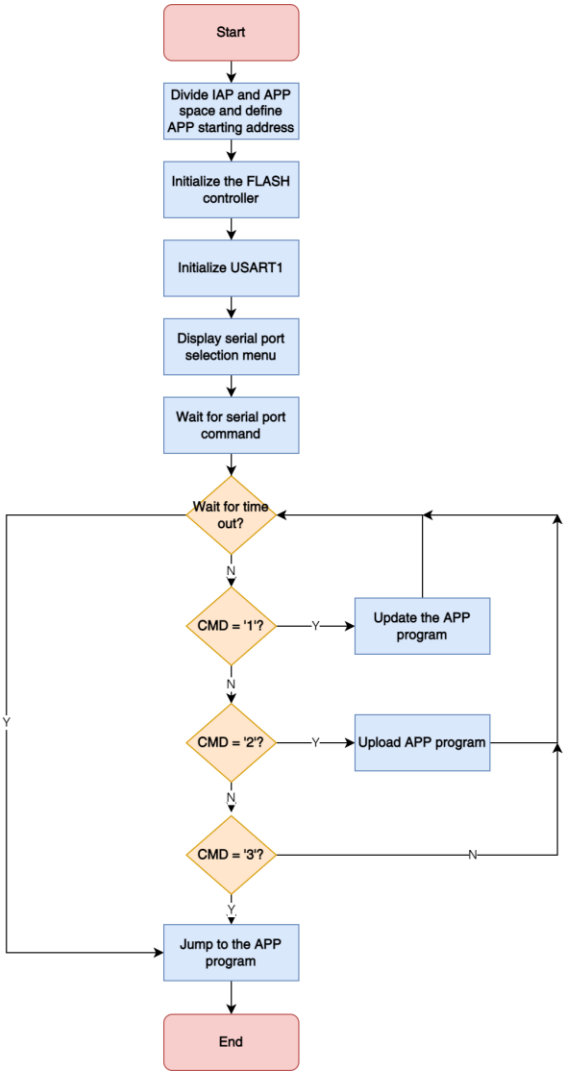

Figure 6 IAP Program Flow

4.2.2 APP program design

The design of APP program is relatively simple, and only one LED light is on and off circularly. However, it should be noted that, for the "interrupt vector table" setting mentioned in the previous chapter, the offset address of the interrupt vector table needs to be set in the main function at the beginning of the APP program.

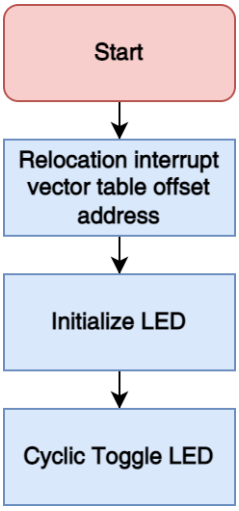

Figure 7 APP Program Flow

4.3 APP firmware upgrade

4.3.1 Generation of APP executable files

APP is the main user program. After the design of IAP program is completed, the update file of APP shall be generated, and then be transmitted to IAP program through a certain protocol to update APP firmware. Generally, the file type of APP update file is .bin file, which can be directly copied to flash to run.

Configure the following command in the User tab of Keil MDK Option configuration to use fromelf.exe to generate bin file, which is generated in the project directory by default.

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

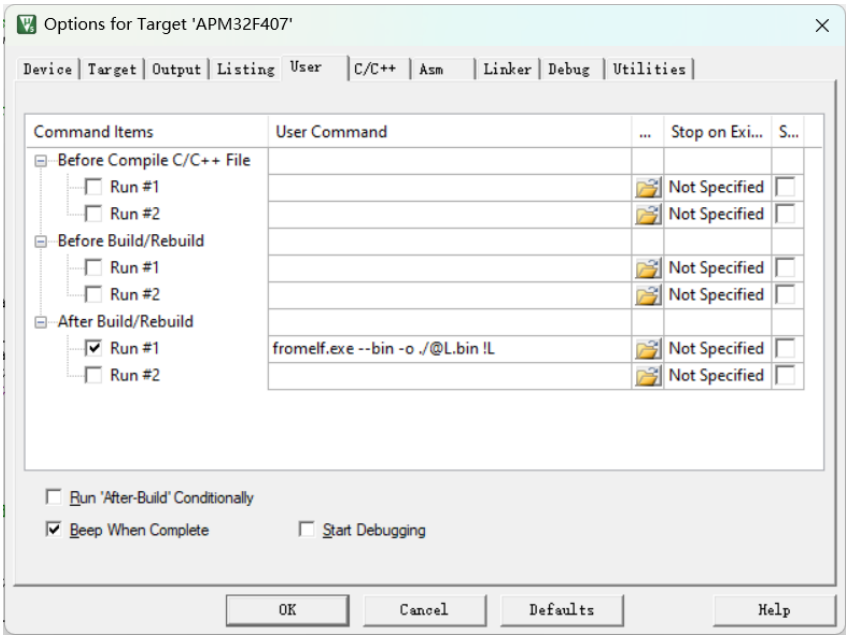

Figure 8 Generate bin File

4.3.2 Firmware upgrade

After the bin file is generated, a transmission method can be selected to transmit the bin file according to the IAP program. Ymodem protocol and hyper terminal software are used here. The following shows the specific operations.

4.3.2.1 Serial menu

Keep the serial port connected to the target board, and after the target board of the downloaded IAP BootLoader program is reset, the hyper terminal software will display the serial port menu in the following figure.

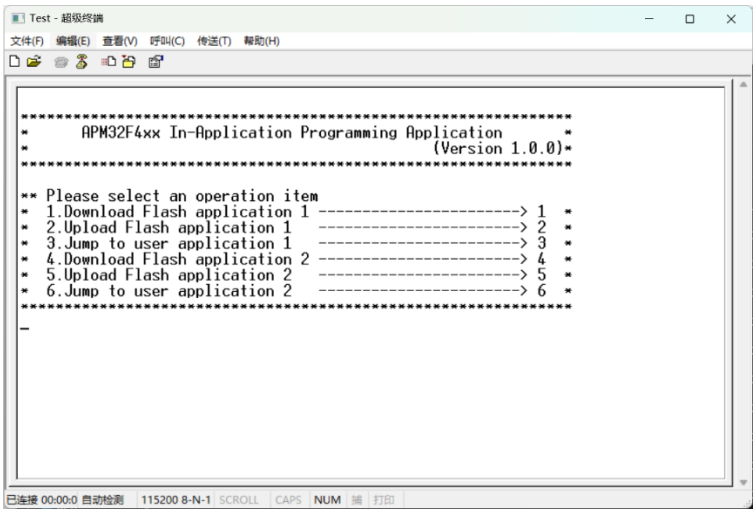

Figure 9 Serial Port Menu

4.3.2.2 Download the bin file of APP1 to the chip

Press the "1" key on the keyboard in the serial port menu to enter the status of waiting for the transmission of the bin file.

Application1 corresponds to IAP_ Application1 project file, and Application2 corresponds to IAP_ Application2 project file.

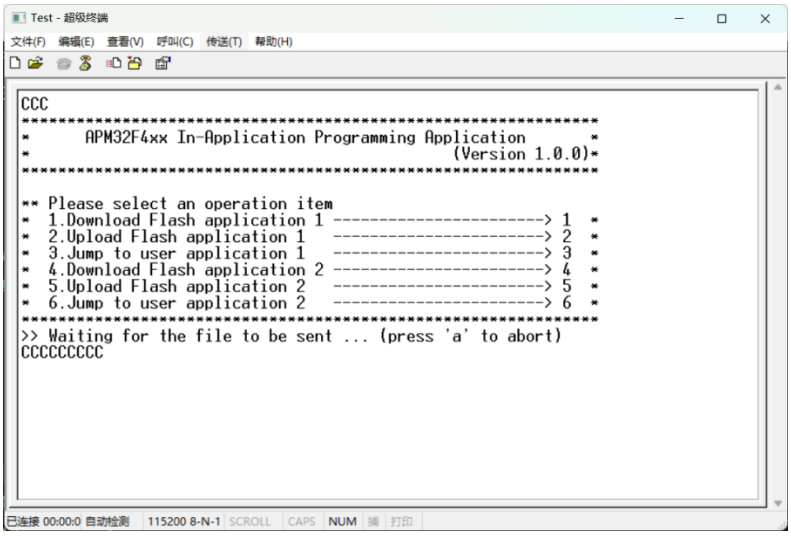

Figure 10 Download bin File

4.3.2.3 Select the bin file to be downloaded

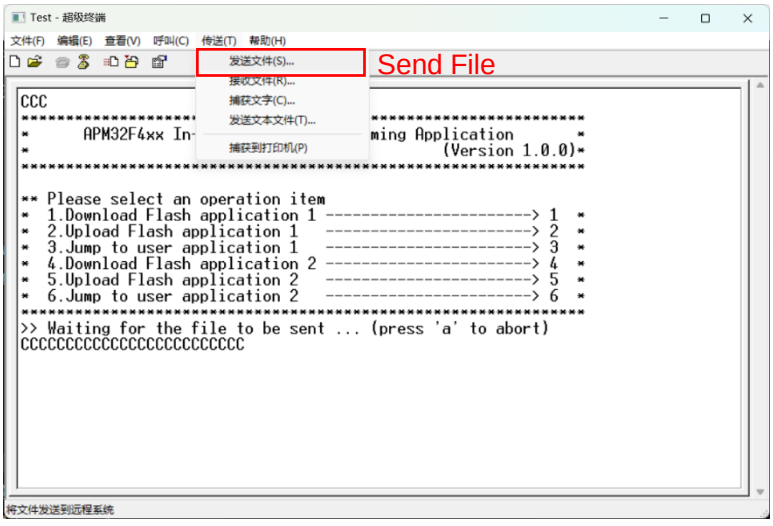

Figure 11 Select bin File

4.3.2.4 Select the Ymodem protocol and send the file

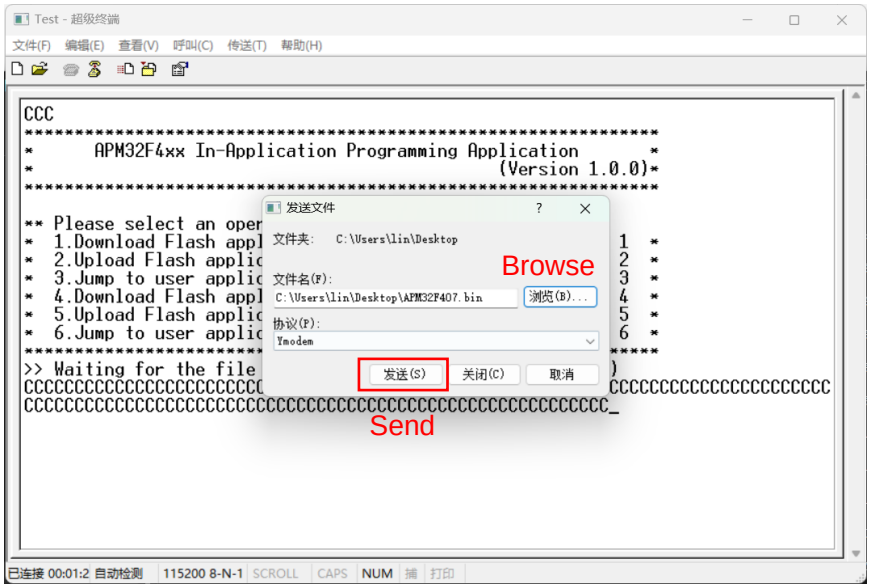

Figure 12 Send a File

4.3.2.5 Downloading is completed

After the file is sent, the software will prompt "Programming Completed Successfully!". Then you can select the required function again.

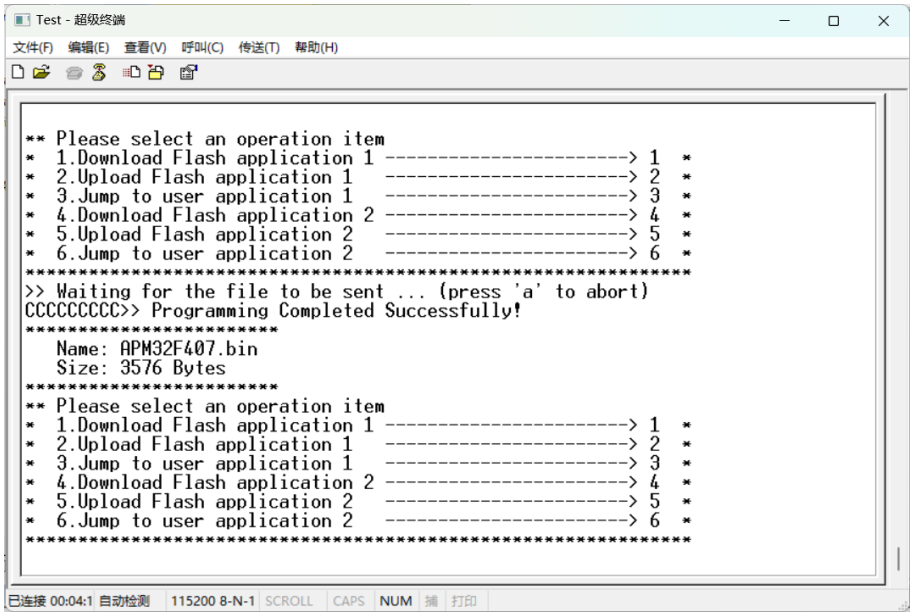

Figure 13 Downloading is Completed

5 Revision History

Table 1 Document Revision History

Date	Version	Change History
May 31, 2022	1.0	New

Statement

This manual is formulated and published by Zhuhai Geehy Semiconductor Co., Ltd. (hereinafter referred to as "Geehy"). The contents in this manual are protected by laws and regulations of trademark, copyright and software copyright. Geehy reserves the right to correct and modify this manual at any time. Please read this manual carefully before using the product. Once you use the product, it means that you (hereinafter referred to as the "users") have known and accepted all the contents of this manual. Users shall use the product in accordance with relevant laws and regulations and the requirements of this manual.

1. Ownership of rights

This manual can only be used in combination with chip products and software products of corresponding models provided by Geehy. Without the prior permission of Geehy, no unit or individual may copy, transcribe, modify, edit or disseminate all or part of the contents of this manual for any reason or in any form.

The "Geehy" or "Geehy" words or graphics with "®" or "TM" in this manual are trademarks of Geehy. Other product or service names displayed on Geehy products are the property of their respective owners.

2. No intellectual property license

Geehy owns all rights, ownership and intellectual property rights involved in this manual.

Geehy shall not be deemed to grant the license or right of any intellectual property to users explicitly or implicitly due to the sale and distribution of Geehy products and this manual.

If any third party’s products, services or intellectual property are involved in this manual, it shall not be deemed that Geehy authorizes users to use the aforesaid third party’s products, services or intellectual property, unless otherwise agreed in sales order or sales contract of Geehy.

3. Version update

Users can obtain the latest manual of the corresponding products when ordering Geehy products.

If the contents in this manual are inconsistent with Geehy products, the agreement in Geehy sales order or sales contract shall prevail.

4. Information reliability

The relevant data in this manual are obtained from batch test by Geehy Laboratory or cooperative third-party testing organization. However, clerical errors in correction or errors caused by differences in testing environment may occur inevitably. Therefore, users should understand that Geehy does not bear any responsibility for such errors that may occur in this manual. The relevant data in this manual are only used to

guide users as performance parameter reference and do not constitute Geehy's guarantee for any product performance.

Users shall select appropriate Geehy products according to their own needs, and effectively verify and test the applicability of Geehy products to confirm that Geehy products meet their own needs, corresponding standards, safety or other reliability requirements. If loses are caused to users due to the user's failure to fully verify and test Geehy products, Geehy will not bear any responsibility.

5. Compliance requirements

Users shall abide by all applicable local laws and regulations when using this manual and the matching Geehy products. Users shall understand that the products may be restricted by the export, re-export or other laws of the countries of the product suppliers, Geehy, Geehy distributors and users. Users (on behalf of itself, subsidiaries and affiliated enterprises) shall agree and promise to abide by all applicable laws and regulations on the export and re-export of Geehy products and/or technologies and direct products.

6. Disclaimer

This manual is provided by Geehy "as is". To the extent permitted by applicable laws, Geehy does not provide any form of express or implied warranty, including without limitation the warranty of product merchantability and applicability of specific purposes.

Geehy will bear no responsibility for any disputes arising from the subsequent design and use of Geehy products by users.

7. Limitation of liability

In any case, unless required by applicable laws or agreed in writing, Geehy and/or any third party providing this manual "as is" shall not be liable for damages, including any general damages, special direct, indirect or collateral damages arising from the use or no use of the information in this manual (including without limitation data loss or inaccuracy, or losses suffered by users or third parties).

8. Scope of application

The information in this manual replaces the information provided in all previous versions of the manual.