AN0005

Application Note

AT32 OTA using the USART

Introduction

For most Flash-based systems, one of the most important requirement is the ability to update firmware installed into the end products. The OTA (over-the-air) means that the user application can write on some of User Flash area during runtime to update the firmware in the products through the reserved communication interfaces after product release in a free manner.

This application note is written to provide general guidelines for creating OTA applications on the AT32 microcontrollers.

The AT32 microcontrollers can run user-specific firmware to implement OTA function on the embedded Flash. This function can use any communication interface available and supported by products. This application note takes the USART using a custom protocol for demonstration.

Applicable products:

Part number	All AT32 series

Contents

List of Tables

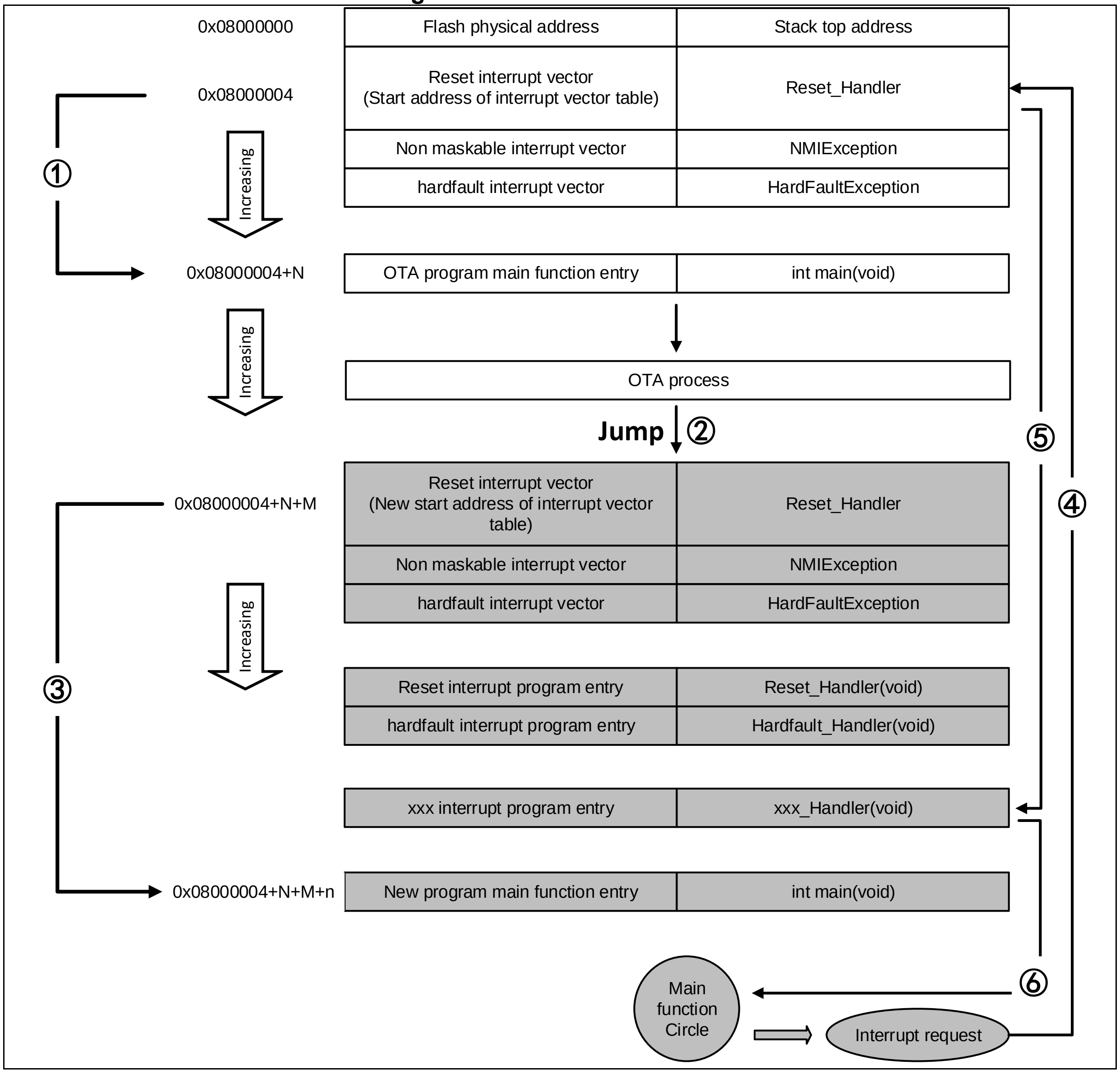
List of Figures

1 Overview

In principle, the OTA (over-the-air) means that the user application can write on some of User Flash area during runtime to update the firmware in the products through the reserved communication interfaces after product release in a free manner.

Generally, to implement OTA function, it is necessary to write two project codes while designing the firmware program, with the first one as the OTA Bootloader and the second as the project program App serving as the actual functional code to execute the application and upgrade. Both project codes are programmed in the User Flash simultaneously.

Figure 1. OTA code execution flow


0x08000000
Flash physical address
Stack top address
0x08000004
Reset interrupt vector (Start address of interrupt vector table)
Reset_Handler
Increasing
①
Non maskable interrupt vector
NMIException
hardfault interrupt vector
HardFaultException
0x08000004+N
OTA program main function entry
int main(void)
Increasing
OTA process
Jump
②
⑤
④
0x08000004+N+M
Reset interrupt vector (New start address of interrupt vector table)
Reset_Handler
Non maskable interrupt vector
NMIException
hardfault interrupt vector
HardFaultException
Increasing
③
Reset interrupt program entry
Reset_Handler(void)
hardfault interrupt program entry
Hardfault_Handler(void)
xxx interrupt program entry
xxx_Handler(void)
0x08000004+N+M+n
New program main function entry
int main(void)
⑥
Main function Circle
Interrupt request

During the execution as shown in the figure, after MCU is reset, it fetches the address of the reset interrupt vector from the address 0x08000004, and jumps to the reset interrupt service program. At the end of running, it jumps to the main function of bootloader, as shown in ①. After executing the bootloader (the App code is the FLASH in grey background, and the start address of App program reset interrupt vector is 0x08000004+N+M), it jumps to the reset vector table of App program, fetches the address of the reset interrupt vector from App program and jumps to execute the reset interrupt service program of App, and then jumps to the main function of App, as shown in ② and ③. The main function is an endless loop, and it should be noted that there are two interrupt vector tables in different positions for the AT32 Flash.

During the execution of the main function, if CPU receives an interrupt request, the PC pointer is still forced to jump to the interrupt vector table of the address 0x08000004, instead of the interrupt vector table of the App program, as shown in ④. Then, the program will jump to the new interrupt service program corresponding to the interrupt source according to the configured interrupt vector table offset value, as shown in ⑤.

After the execution of interrupt service program, the program will return to the main function to continue running, as shown in ⑥.

Based on the above mentioned analysis, it is clear to know that the two conditions must be required for OTA application.

1) App program must start from the address at offset x following the Bootloader;
2) The interrupt vector table of App program must be moved by an offset of x.

1.1 How to execute OTA using AT32 USART

1.1.1 Hardware resources

In this application note, the AT-START-AT32F403A demo board is used as the hardware conditions. OTA demo source codes also include other AT32 series MCUs, and users only need to compile the corresponding project and program it to the AT-START demo board.

1) LED2/LED3/LED4
2) USART1 (PA9/PA10)
3) AT-START demo board

1.1.2 Software resources

1) tool_release
   - IAP_Programmer.exe, PC host tool, to demonstrate OTA upgrade process
2) source_code
   - Bootloader, Bootloader source program, LED2 blinks when working
   - App_led3_toggle, App1 source program, LED3 blinks when working
   - App_led4_toggle, App2 source program, LED4 blinks when working

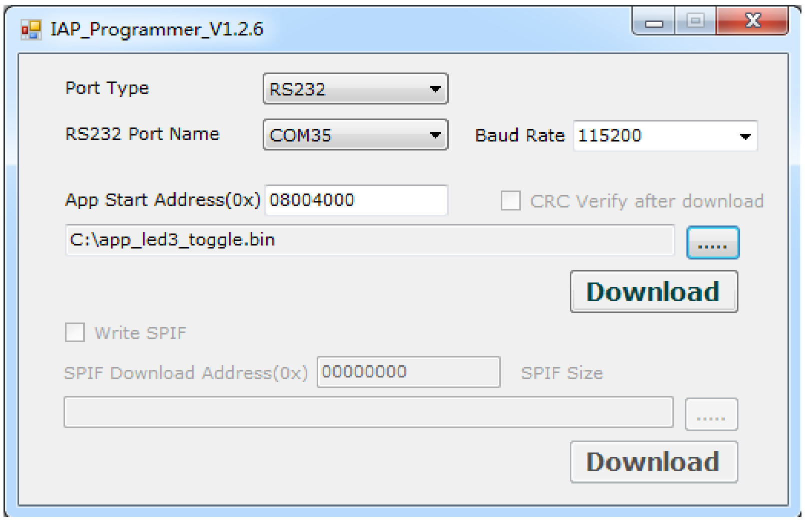
Note: All projects are built around keil v5. If users want to use them in other compiling environments, please refer to AT32F403A_407_Firmware_Library_V2.x.x\project\at_start_f403a\templates (such as IAR6/7/8, keil 4/5, eclipse_gcc) for a simple change.

1.2 How to use OTA demo

This application note mainly introduces two common OTA demos, i.e., template app and dual app.

1) Open the Bootloader project source program, select the target MCU, compile and download to demo board;
2) Open the IAP_Programmer.exe;
3) Select the correct serial interface, APP download address and bin files, then click on "Download", as shown in the figure below;
4) Check the status of LED2/3/4: LED2 blinks-Bootloader working, LED3 blinks-App1 working, LED4 blinks-App2 working.

Figure 2. IAP demo host PC



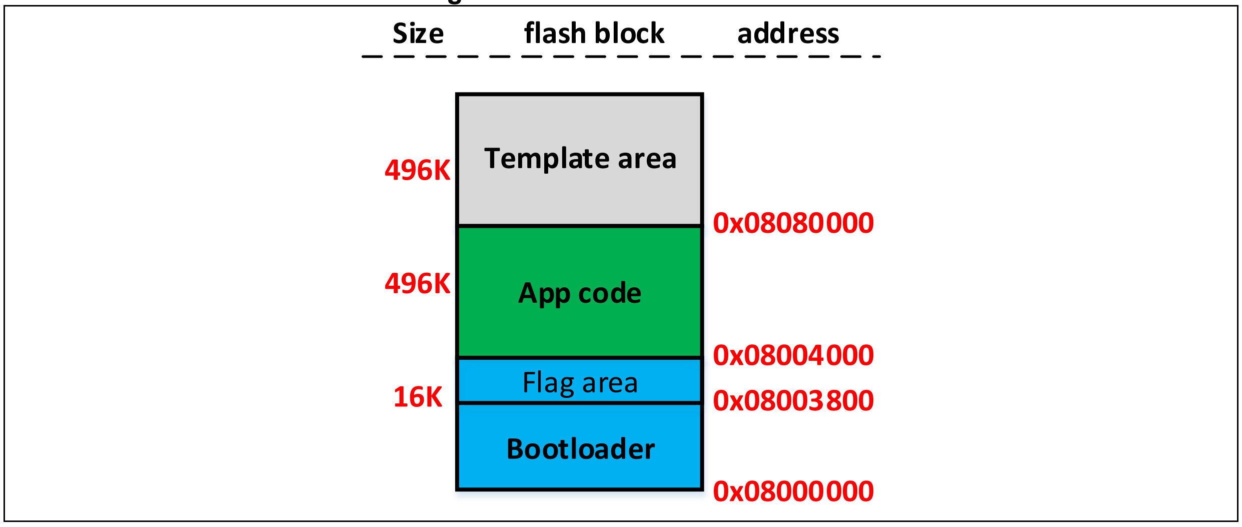
IAP_Programmer_V1.2.6

Port Type: RS232
RS232 Port Name: COM35
Baud Rate: 115200
App Start Address(0x): 08004000
CRC Verify after download: unchecked
C:\app_led3_toggle.bin
Write SPIF: unchecked
SPIF Download Address(0x): 00000000
SPIF Size

2 How to set Template app OTA

2.1 Address distribution

Figure 3. Flash address distribution

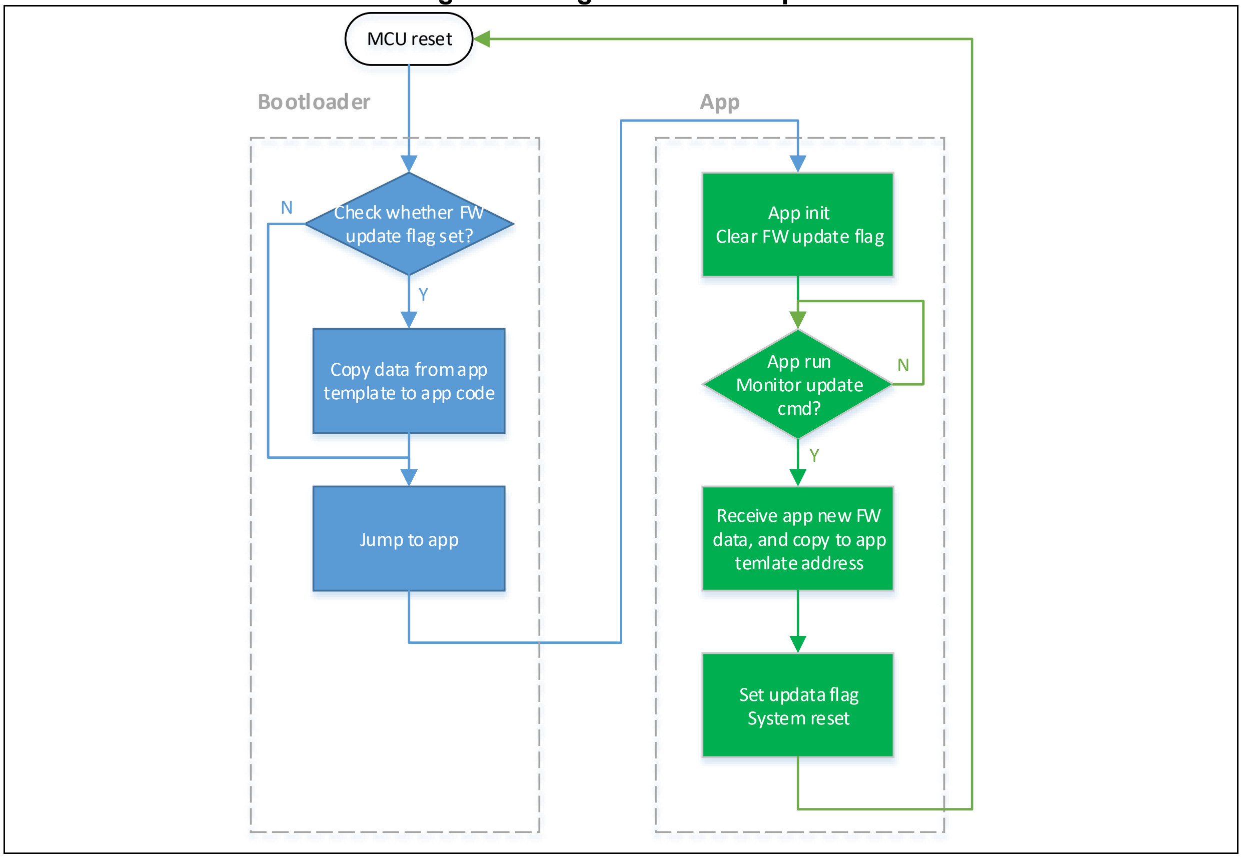


Note: The last page of the Bootloader area is used to place the flag to prevent failures (such as power-off) during upgrading. Please do not overwrite the address of this flag when modifying Bootloader.

2.2 Execution process

The OTA contains Bootloader, App and Template, where the application is executed in App and Template is only used as a temporary storage space for new App firmware data. The program execution process is shown in the figure below.

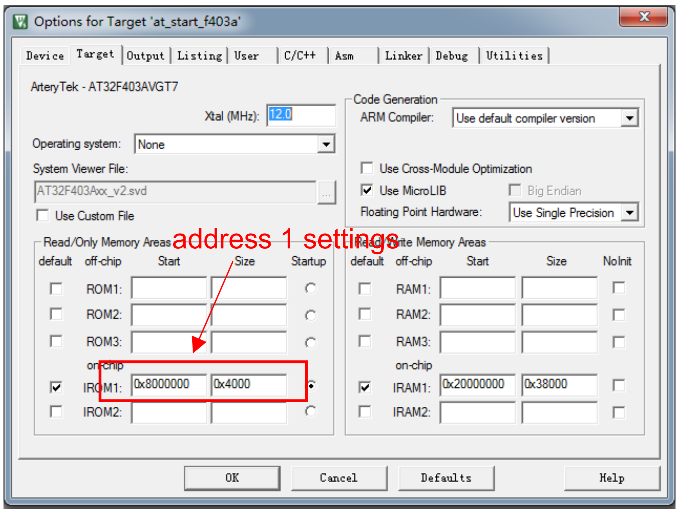
Figure 4. Program execution process


MCU reset
Bootloader
Check whether FW update flag set?
N
Y
Copy data from app template to app code
Jump to app
App
App init
Clear FW update flag
App run
Monitor update cmd?
N
Y
Receive app new FW data, and copy to app temlate address
Set updata flag
System reset

2.3 Bootloader project settings

1) Keil settings

Figure 5. Bootloader project address 1 settings in Keil



ArteryTek - AT32F403AVGT7

Xtal (MHz): 12.0

Operating system: None

System Viewer File: AT32F403Axx_v2.svd

Use Custom File

Code Generation

ARM Compiler: Use default compiler version

Use Cross-Module Optimization

[✓] Use MicroLIB

Big Endian

Floating Point Hardware: Use Single Precision

Read/Only Memory Areas

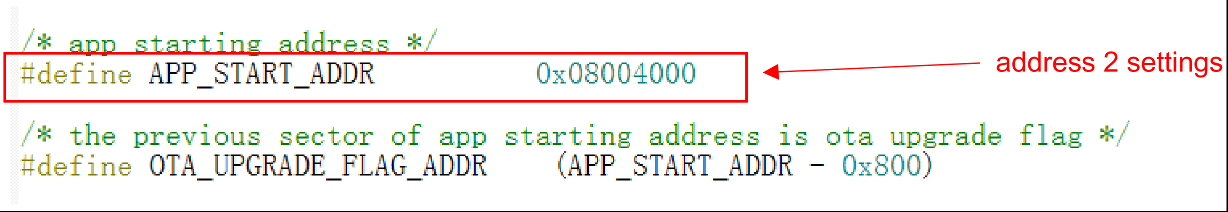
default	off-chip		Start	Size	Startup
		ROM1:			
		ROM2:			
		ROM3:			
	on-chip				
✓		IROM1:	0x8000000	0x4000	●
		IROM2:			

Read/Write Memory Areas

default	off-chip		Start	Size	NoInit
		RAM1:			
		RAM2:			
		RAM3:			
	on-chip				
✓		IRAM1:	0x20000000	0x38000	
		IRAM2:			

2) Bootloader source program modification in ota.h file

Figure 6. Bootloader project address 2 settings in program



```
/* 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)
```

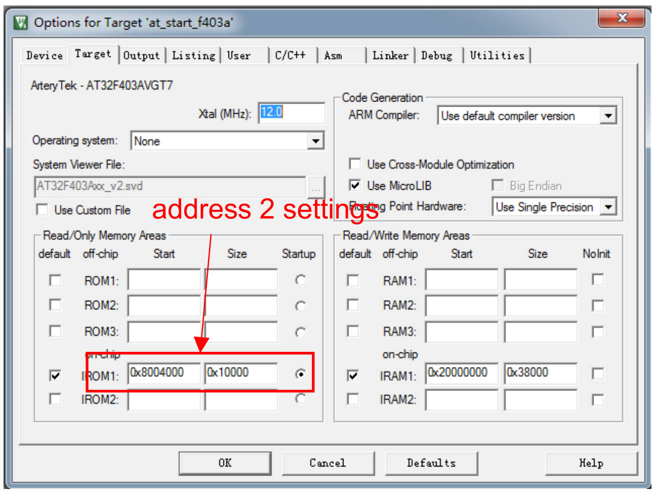
2.4 App project settings

OTA demo provides two Apps for testing, both starting from address 2 (0x800 4000). LED3 blinks for App1 and LED4 blinks for App2.

The App1 is used as an example to demonstrate how to set App project.

1) Keil project settings

Figure 7. App project address 2 settings in Keil



ArteryTek - AT32F403AVGT7

Xtal (MHz): 12.0

Operating system: None

System Viewer File: AT32F403Axx_v2.svd

Use Custom File

Code Generation

ARM Compiler: Use default compiler version

Use Cross-Module Optimization

[✓] Use MicroLIB

Big Endian

Floating Point Hardware: Use Single Precision

Read/Only Memory Areas

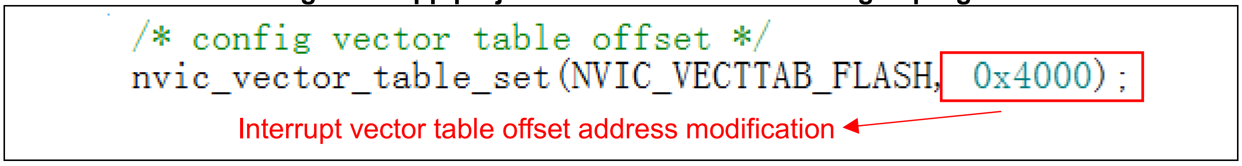
default	off-chip		Start	Size	Startup
		ROM1:			
		ROM2:			
		ROM3:			
	on-chip				
✓		IROM1:	0x8004000	0x10000	●
		IROM2:			

Read/Write Memory Areas

default	off-chip		Start	Size	NoInit
		RAM1:			
		RAM2:			
		RAM3:			
	on-chip				
✓		IRAM1:	0x20000000	0x38000	
		IRAM2:			

2) App1 source program settings

Figure 8. App project vector table offset setting in program



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

3) Compile and generate bin files

   Call the fromelf.exe after compiling through User option, and generate .bin files according to .axf file for OTA update.

   Going through the above three steps, we can get an APP program in .bin format, and perform update through Bootloader.

4) Enable debug App code function

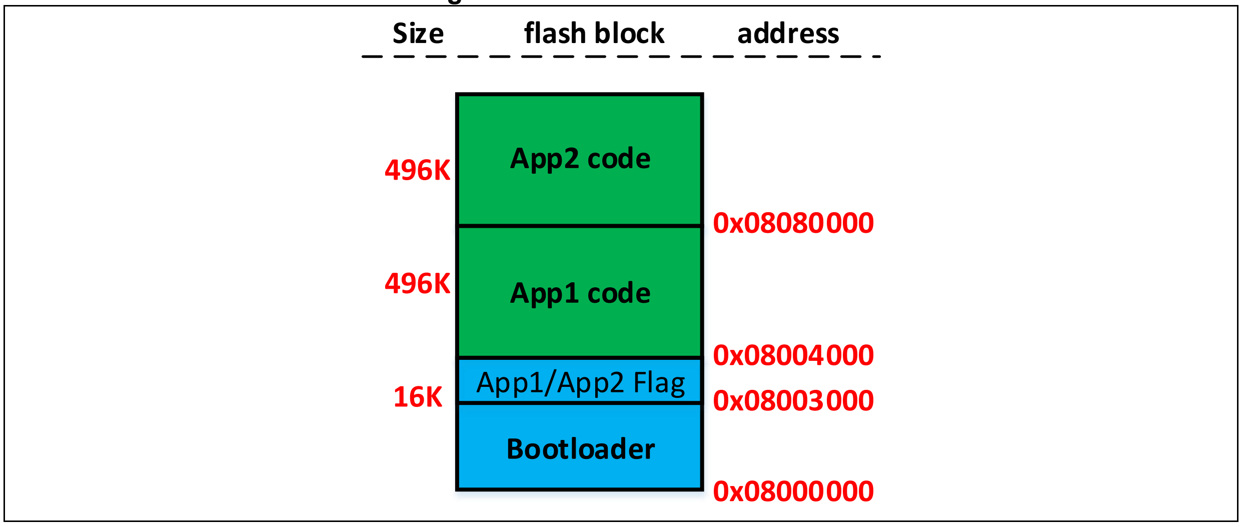
   If it is necessary to debug the App project separately when designing the App code, please proceed as follows:

   - Download Bootloader project
   - Debug App project

3 How to set Dual app OTA

3.1 Address distribution

Figure 9. Flash address distribution

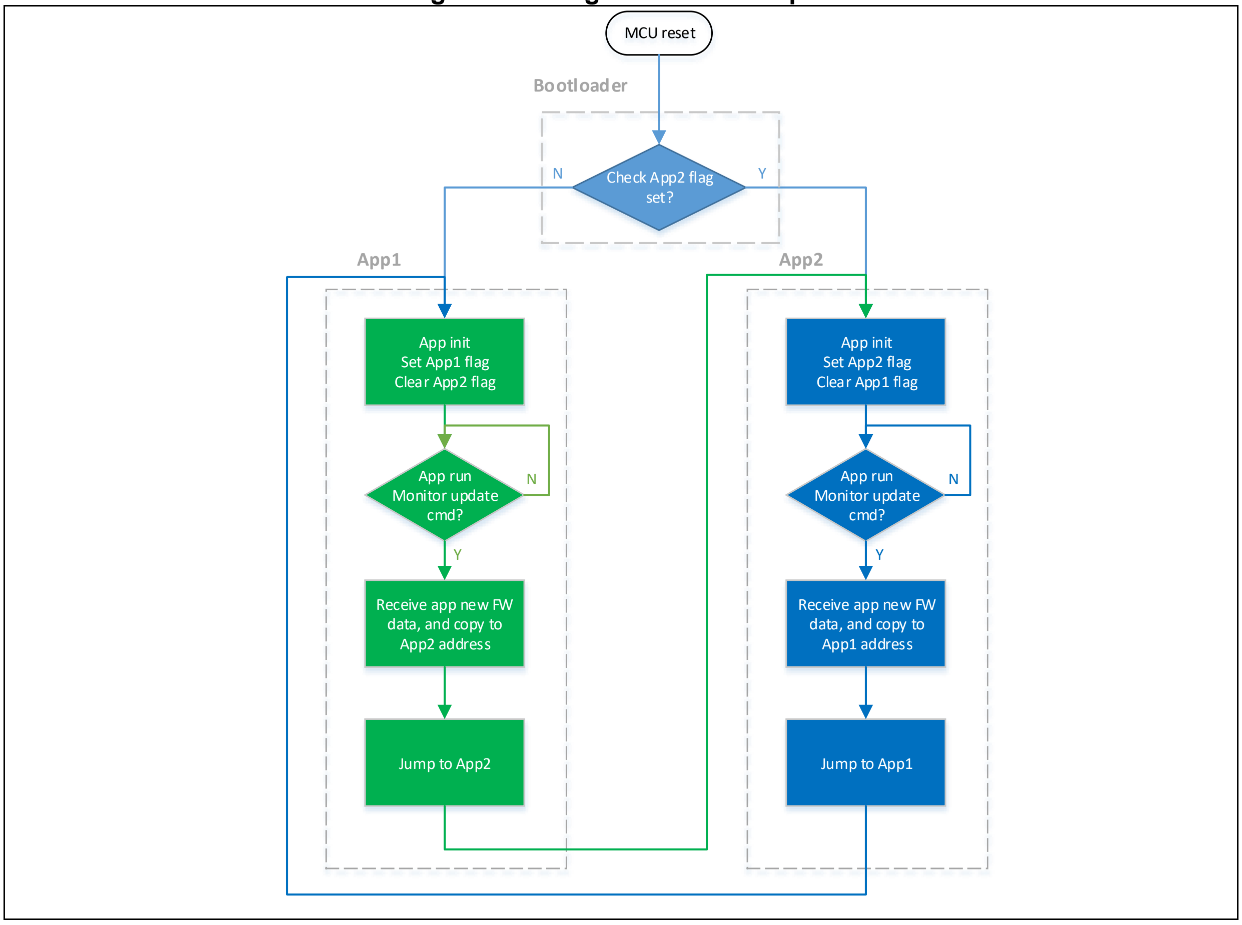


Note: The last two pages of the Bootloader area are used to place the flag that is used to verify the correctness of App. Please do not overwrite the address of this flag when modifying Bootloader.

3.2 Execution process

The OTA contains Bootloader, App1 and App2, where the application is executed in App1 or App2. The program execution process is shown in the figure below.

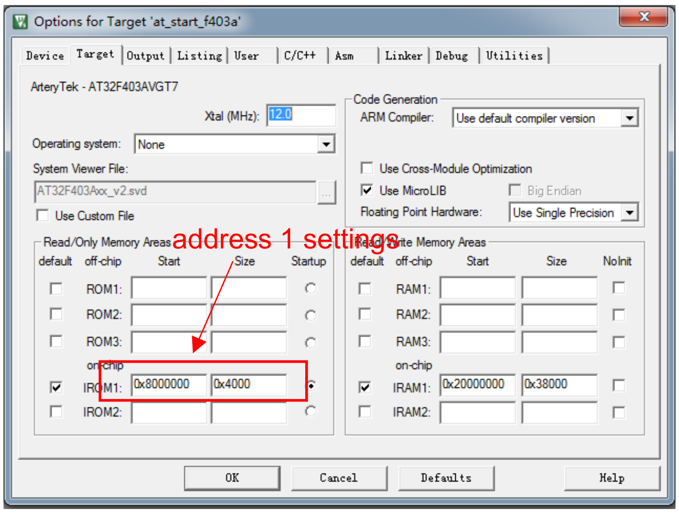
Figure 10. Program execution process



3.3 Bootloader project settings

1) Keil settings

Figure 11. Bootloader project address 1 settings in Keil



Options for Target 'at_start_f403a'

Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities

ArteryTek - AT32F403AVGT7

Xtal (MHz): 12.0

Operating system: None

System Viewer File: AT32F403Axx_v2.svd

Use Custom File: unchecked

Code Generation — ARM Compiler: Use default compiler version

Use Cross-Module Optimization: unchecked; Use MicroLIB: checked; Big Endian: unchecked

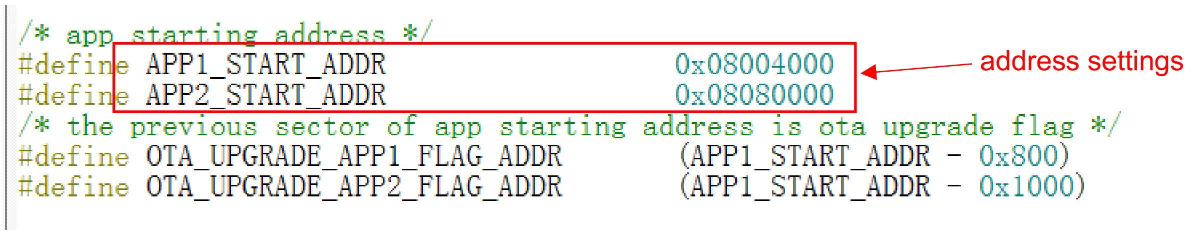
Floating Point Hardware: Use Single Precision

Read/Only Memory Areas — on-chip IROM1: default checked, Start 0x8000000, Size 0x4000, Startup selected

Read/Write Memory Areas — on-chip IRAM1: default checked, Start 0x20000000, Size 0x38000

2) Bootloader source program modification in ota.h file

Figure 12. Bootloader project address 2 settings in program



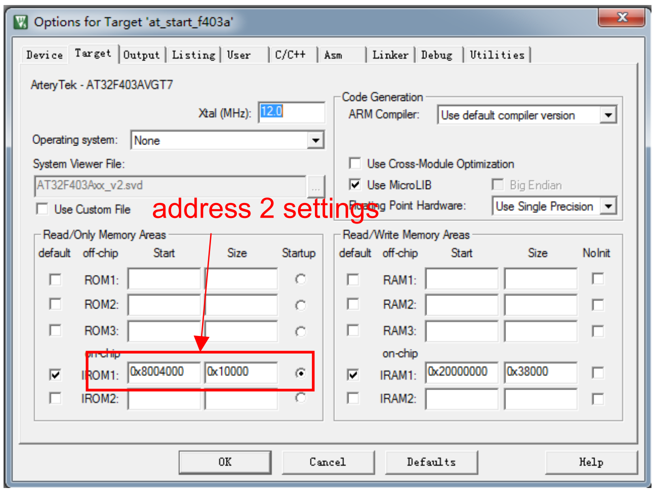
```
/* app starting address */
#define APP1_START_ADDR                     0x08004000
#define APP2_START_ADDR                     0x08080000
/* the previous sector of app starting address is ota upgrade flag */
#define OTA_UPGRADE_APP1_FLAG_ADDR          (APP1_START_ADDR - 0x800)
#define OTA_UPGRADE_APP2_FLAG_ADDR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

3.4 App project settings

OTA demo provides two Apps for testing, where the app_led3_toggle starting from 0x800 4000 and app_led4_toggle starting from 0x8080000. LED3 blinks for App1 and LED4 blinks for App2.
The App1 is used as an example to demonstrate how to set App project.

1) Keil project settings

Figure 13. App project address 2 settings in Keil



Options for Target 'at_start_f403a'

Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities

ArteryTek - AT32F403AVGT7

Xtal (MHz): 12.0

Operating system: None

System Viewer File: AT32F403Axx_v2.svd

Use Custom File: unchecked

Code Generation — ARM Compiler: Use default compiler version

Use Cross-Module Optimization: unchecked; Use MicroLIB: checked; Big Endian: unchecked

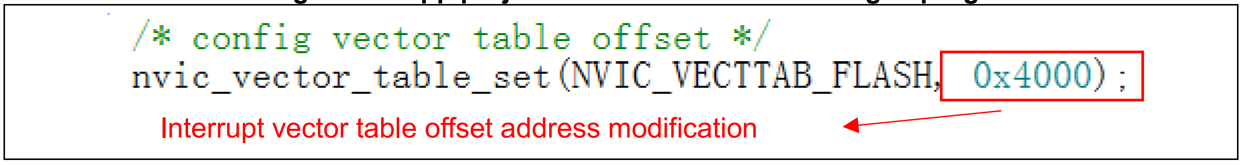
Floating Point Hardware: Use Single Precision

Read/Only Memory Areas — on-chip IROM1: default checked, Start 0x8004000, Size 0x10000, Startup selected

Read/Write Memory Areas — on-chip IRAM1: default checked, Start 0x20000000, Size 0x38000

2) App1 source program settings

Figure 14. App project vector table offset setting in program



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

3) Compile and generate bin files

   Call the fromelf.exe after compiling through User option, and generate .bin files according to .axf file for OTA update.

   Going through the above three steps, we can get an APP program in .bin format, and perform update through Bootloader.

4) Enable debug App code function

   If it is necessary to debug the App project separately when designing the App code, please proceed as follows:

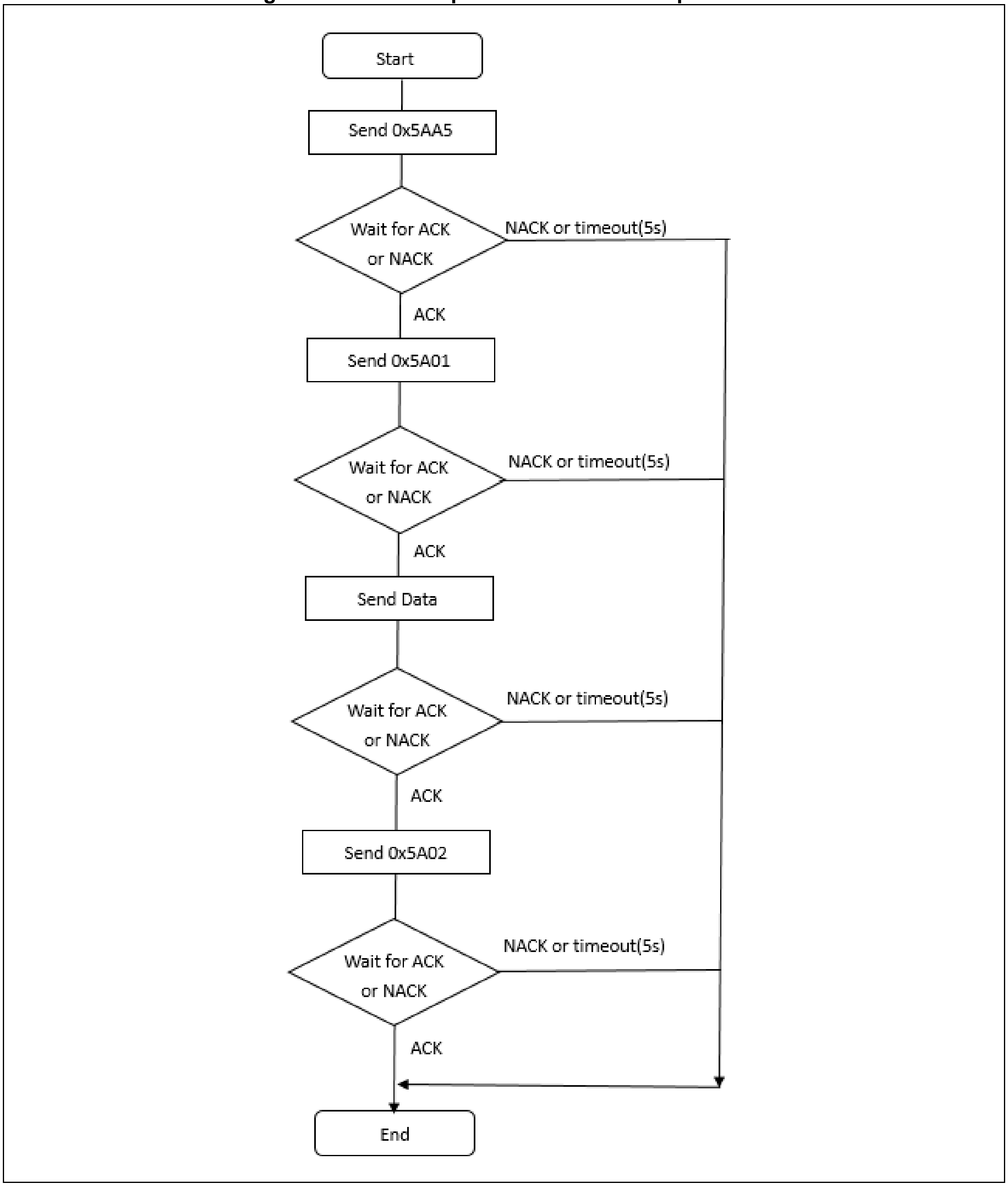
   a) Download Bootloader project
   b) Debug App project

4 Bootloader/App and host communication protocol

The program communicates with the host computer and receives firmware upgrade data. The communication protocol between the host computer and embedded terminal is shown below.

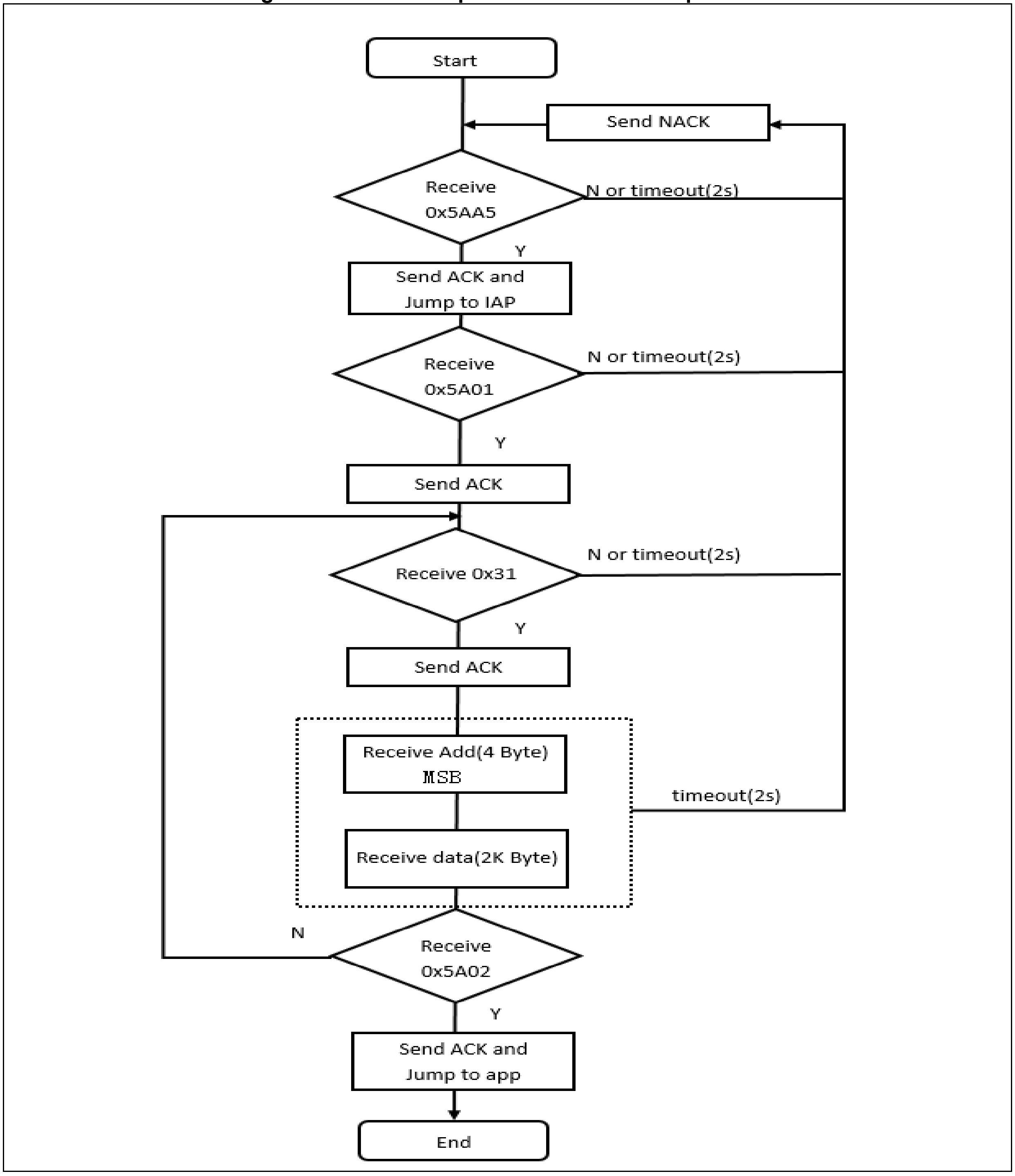
1) Host computer communication protocol

Figure 15. Host computer communication protocol



2) Embedded slave computer communication protocol

Figure 16. Slave computer communication protocol



Note: *ACK: 0xCCDD*

NACK: 0xEEFF

Data: 0x31+ Addr + data + checksum (1byte)

Addr: 4bytes, high bit first

Kbytes, download data, fill in 0xFF for less than 2K content

Checksum: lower eight bits of the checksum of Addr of 1byte and 4bytes + 2Kbytes

5 Revision history

Table 1. Document revision history

Date	Version	Revision note
2021.12.31	2.0.0	Initial release