

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

1	Ove	rview	5
	1.1	How to execute OTA using AT32 USART	7
		1.1.1 Hardware resources	7
		1.1.2 Software resources	7
	1.2	How to use OTA demo	7
2	How	/ to set Template app OTA	8
	2.1	Address distribution	8
	2.2	Execution process	8
	2.3	Bootloader project settings	9
	2.4	App project settings	9
3	How	<i>ı</i> to set Dual app OTA1	1
	3.1	Address distribution1	1
	3.2	Execution process1	1
	3.3	Bootloader project settings1	2
	3.4	App project settings1	2
4	Воо	tloader/App and host communication protocol1	4
5	Rev	ision history1	6



List of Tables



List of Figures

Figure 1. OTA code execution flow	5
Figure 2. IAP demo host PC	7
Figure 3. Flash address distribution	8
Figure 4. Program execution process	8
Figure 5. Bootloader project address 1 settings in Keil	9
Figure 6. Bootloader project address 2 settings in program	9
Figure 7. App project address 2 settings in Keil	9
Figure 8. App project vector table offset setting in program	. 10
Figure 9. Flash address distribution	. 11
Figure 10. Program execution process	. 11
Figure 11. Bootloader project address 1 settings in Keil	. 12
Figure 12. Bootloader project address 2 settings in program	.12
Figure 13. App project address 2 settings in Keil	. 12
Figure 14. App project vector table offset setting in program	. 13
Figure 15. Host computer communication protocol	. 14
Figure 16. Slave computer communication protocol	.15





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.

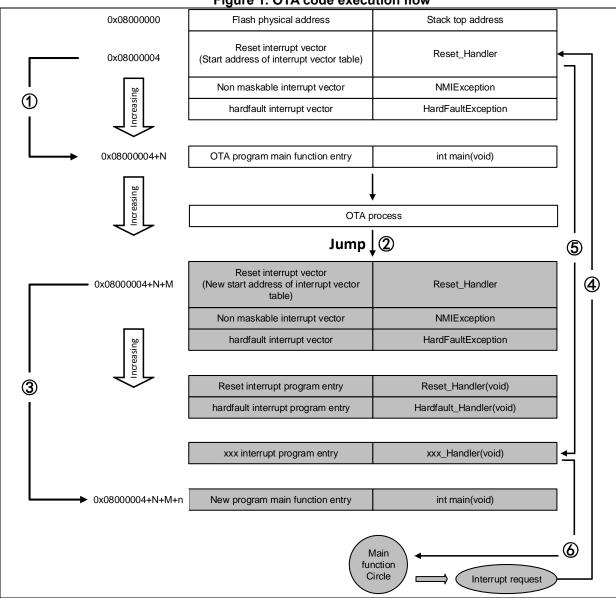


Figure 1. OTA code execution flow



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

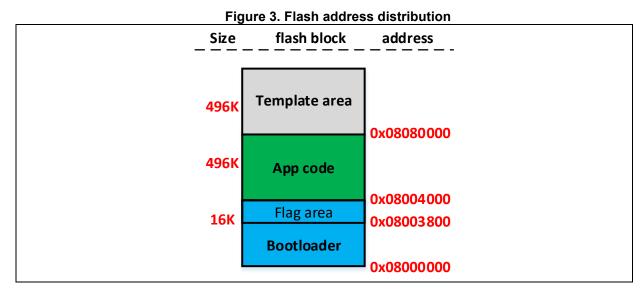
This application note mainly introduces two common OTA demos, i.e., template app and dual app.

- 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.

Port Type	RS232 •		
RS232 Port Name	COM35 -	Baud Rate 115200	•
App Start Address(0x)	08004000	CRC Verify aft	er download
C:\app_led3_toggle.b	in		
		Dowr	nload
Write SPIF			
SPIF Download Addres	ss(0x) 0000000	SPIF Size	
			nload

Figure 2. IAP demo host PC

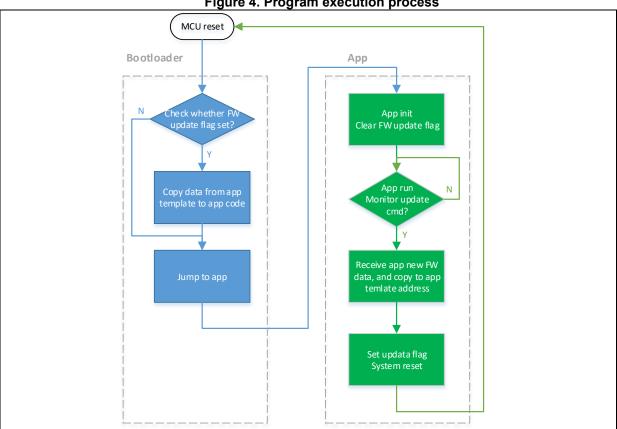
- 2 How to set Template app OTA
- **Address distribution** 2.1

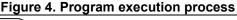


Note: The last page of the Bootloader area is used to place the flag to prevent failures (such as power-off) during upgrading. Please do not overwrite the address of this flag when modifying Bootloader.

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.







2.3 Bootloader project settings

1) Keil settings

Options for Target 'at_start_f403a'		
Device Target Output Listing User	C/C++ Asm Linker Debug U	Jtilities
ArteryTek - AT32F403AVGT7 Xtal (MHz): Operating system: None	Code Generation ARM Compiler: Use de	fault compiler version
System Viewer File: AT32F403Aox_v2.svd	Use MicroLIB Roating Point Hardware:	Big Endian
Read/Only Memory Areas address default off-chip Start /Size	1 SettinggSrite Memory Areas Startup default off-chip Start	
□ ROM1:	O 🗆 RAM1:	
ROM2: ROM3:	O □ RAM2: O □ RAM3:	
on <mark>chip</mark>	on-chip	000 [0x38000 [
OK	Cancel Defaults	Help

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	▲ address 2 settings
/* the previous sector of app #define OTA_UPGRADE_FLAG_ADDR		

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

Options for Target 'at_start_f403a'
Device Target Output Listing User C/C++ Asm Linker Debug Utilities
AtteryTek - AT32F403AVGT7
Xtal (MHz): 12.0 ARM Compiler: Use default compiler version 💌
Operating system: None System Viewer File: Use Cross-Module Optimization
AT32F403Axx_v2.svd 🔽 Use MicroLIB 🗖 Big Endian
Use Custom File address 2 settingers Point Hardware: Use Single Precision -
Read/Only Memory Areas default off-chip Start Size Startup default off-chip Start Size NoInit
□ ROM2: C □ RAM2: □
C RAM3:
on-chip IROM1: [0x8004000 [0x10000] ○ IROM1: [0x20000000 [0x38000]
OK Cancel Defaults Help

2) App1 source program settings

,**17151**

Figure 8. App project vector table offset setting in program

/* config vector table offset */
nvic_vector_table_set(NVIC_VECTTAB_FLASH, 0x4000);
Interrupt vector table offset address modification

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

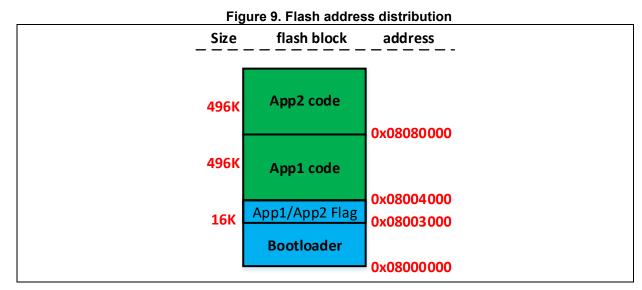
If it is necessary to debug the App project separately when designing the App code, please proceed as follows:

- Download Bootloader project
- Debug App project



3 How to set Dual app OTA

3.1 Address distribution



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.

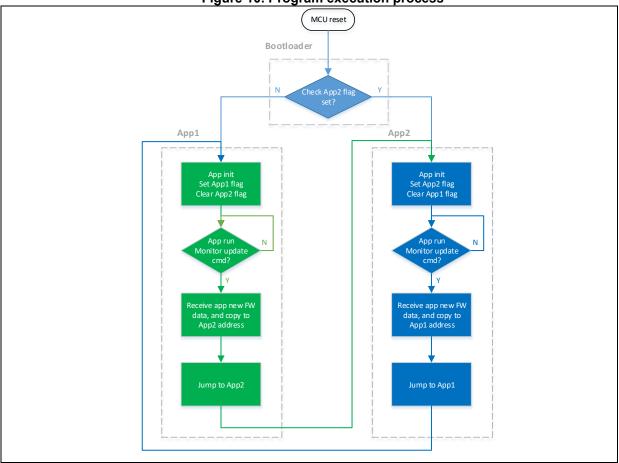


Figure 10. Program execution process



3.3 Bootloader project settings

1) Keil settings

🕅 Options for Target 'at_sta	rt_f403a'					 2
Device Target Output Li	sting User	C/C++ A	sm]]	Linker Debug U	tilities	
ArteryTek - AT32F403AVGT7			-Code (Seneration		
	Xtal (MHz):	2.0			fault compiler versi	on 💌
Operating system: None		-				
System Viewer File:				se Cross-Module Opt		
AT32F403Axx_v2.svd				se MicroLIB	E Big Endian	
Use Custom File		4 4		ng Point Hardware:	Use Single Pre	
Read/Only Memory Areas				Srite Memory Areas off-chip Start	Size	Nolnit
	Size	Startup	derault	·	Size	
ROM1:	_/			RAM1:		
ROM2:				RAM2:	_	
ROM3:		0		RAM3:		
on-chip IROM1: 0x8000000	0x4000	- <u>-</u>	V	on-chip IRAM1: 0x200000	00 0x38000	
IROM2:				IRAM2:		
	1			1.0.0012.		
[OK	Cano	rel	Defaults		Help

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 address settings	3
	#define APP2_START_ADDR	0x08080000	
	/* the previous sector of app starting		
	<pre>#define OTA_UPGRADE_APP1_FLAG_ADDR</pre>	$(APP1_START_ADDR - 0x800)$	
	<pre>#define OTA_UPGRADE_APP2_FLAG_ADDR</pre>	(APP1_START_ADDR - 0x1000)	
- 8			

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	
ArteryTek - AT32F403AVGT7 Xtal (M Operating system: None	MHz): III ARM Compiler: Use default compiler version
System Viewer File: AT32F403Avx_v2.svd	Use Cross-Module Optimization Use MicroLIB Big Endian SS 2 Settings Point Hardware: Use Single Precision
Read/Only Memory Areas default off-chip Start	Read/Write Memory Areas Size Startup default off-chip Start C RAM1:
RoM1: ROM2: ROM3: ROM3:	
	0000 C IRAM1: 0x2000000 0x38000 C IRAM2:



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);
Interrupt vector table offset address modification	

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

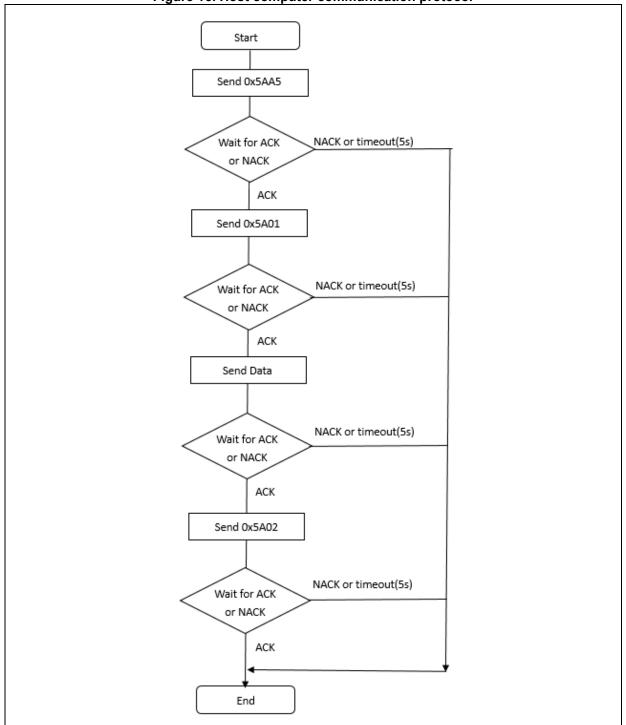
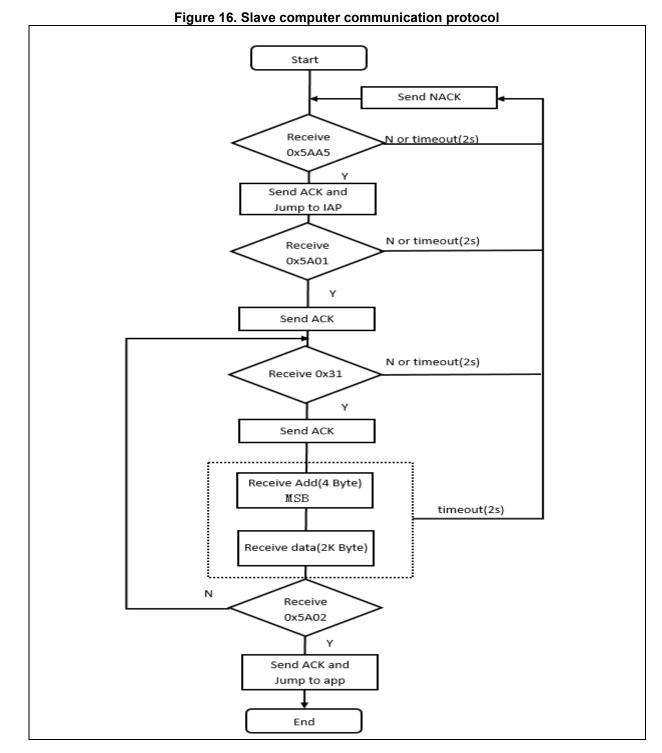


Figure 15. Host computer communication protocol



2) Embedded slave computer communication protocol



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



Purchasers are solely responsible for the selection and use of ARTERY's products and services; ARTERY assumes no liability for purchasers' selection or use of the products and the relevant services.

No license, express or implied, to any intellectual property right is granted by ARTERY herein regardless of the existence of any previous representation in any forms. If any part of this document involves third party's products or services, it does NOT imply that ARTERY authorizes the use of the third party's products or services, or permits any of the intellectual property, or guarantees any uses of the third party's products or services or intellectual property in any way.

Except as provided in ARTERY's terms and conditions of sale for such products, ARTERY disclaims any express or implied warranty, relating to use and/or sale of the products, including but not restricted to liability or warranties relating to merchantability, fitness for a particular purpose (based on the corresponding legal situation in any unjudicial districts), or infringement of any patent, copyright, or other intellectual property right.

ARTERY's products are not designed for the following purposes, and thus not intended for the following uses: (A) Applications that have specific requirements on safety, for example: life-support applications, active implant devices, or systems that have specific requirements on product function safety; (B) Aviation applications; (C) Aerospace applications or environment; (D) Weapons, and/or (E) Other applications that may cause injuries, deaths or property damages. Since ARTERY products are not intended for the above-mentioned purposes, if purchasers apply ARTERY products to these purposes, purchasers are solely responsible for any consequences or risks caused, even if any written notice is sent to ARTERY by purchasers; in addition, purchasers are solely responsible for the compliance with all statutory and regulatory requirements regarding these uses.

Any inconsistency of the sold ARTERY products with the statement and/or technical features specification described in this document will immediately cause the invalidity of any warranty granted by ARTERY products or services stated in this document by ARTERY, and ARTERY disclaims any responsibility in any form.

© 2021 ARTERY Technology – All Rights Reserved