AN0023

Getting started guide for AT32F403A & AT32F407 series

Introduction

This user manual provides information on how to use AT32F403A and AT32F407 MCU for project development in a quickly manner. AT32F407 adds Ethernet (ETH) function versus AT32F403A.

Applicable products：

Part number	AT32F403A
	AT32F407

Contents

List of tables

List of figures

1. Development environment

MCU resources download address:

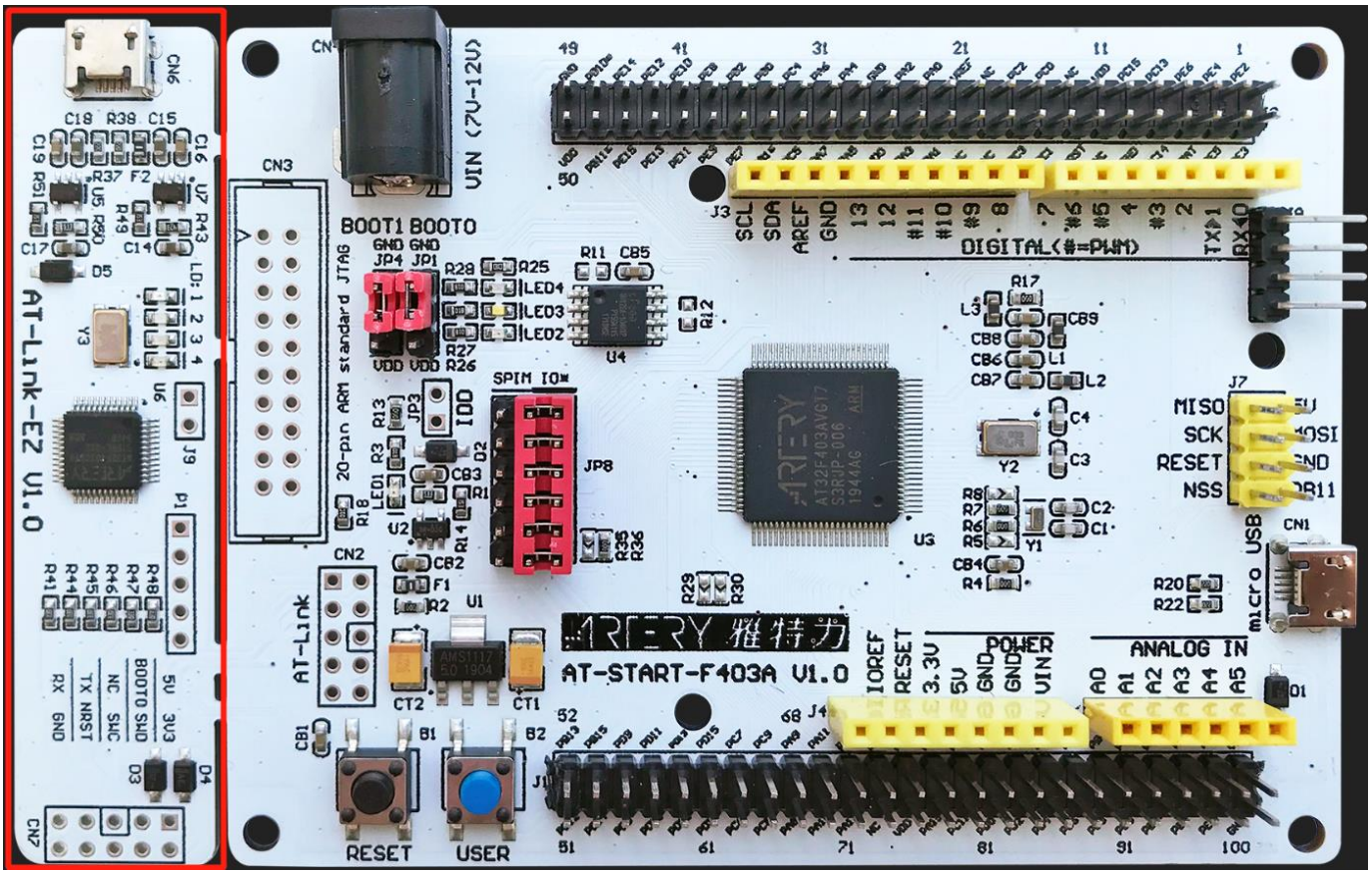
■ Visit Artery website: http://www.arterytek.com

1.1. Set up AT32 development environment

1.1.1 Debug tools

At present, both AT32F403A and AT32F407 evaluation boards are equipped with AT-Link-EZ debug tool, as shown in the red circle on the left side of the figure below. It can also be dissembled and used separately with other circuit boards, supporting IDE online debugging, online programming, USB-to-serial interface and other functions.

Figure 1. Physical photo of AT32F403A evaluation board



1.1.2 Programming tools and software

■ AT programming tools and software: AT-Link /AT-Link-EZ / J-Link/ ICP/ISP.

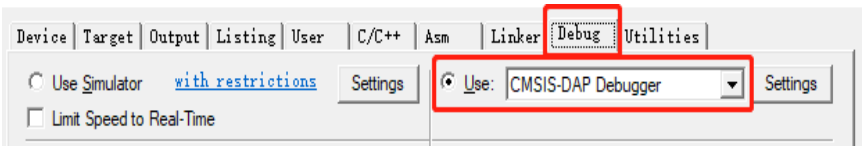
■ Third-party programming tools:

- Xuanwei: https://xuanweikeji.taobao.com
- Maxwiz: www.maxwiz.com.cn
- ZLG: http://tools.zlg.cn/tools
- Amo:http://www.amomcu.cn

1.1.3 AT32 KEIL & IAR development environment

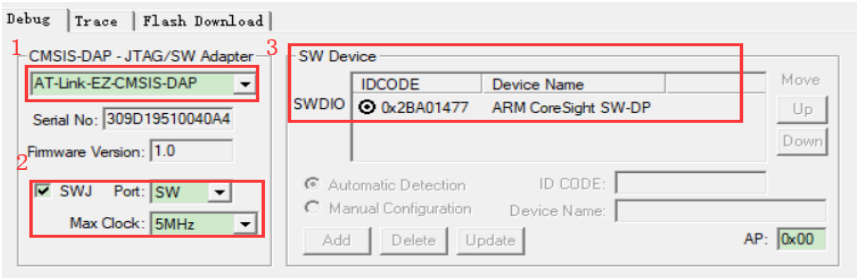
① For Keil compiling system, Keil 4.74, 5.23 or above is recommended.
When using AT-Link-EZ in Keil environment, select "***CMSIS-DAP debugger*** "in "***Debug***".

Figure 2. Keil Debug option



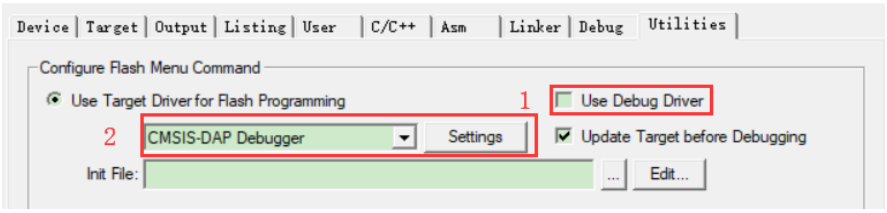
In "***Debug***", click on "***Settings***" to enter "cortex – M Target Driver Setup", as shown in the figure below:
Step 1: select "***AT-Link-EZ-CMSIS-DAP***"
Step 2: select "***SW***" at Port and tick "***SWJ***"
Step 3: ARM SWD debug module is identified.

Figure 3. Keil Debug settings



In "***Utilities***", first uncheck the box "***Use Debug Driver***" (step 1), select "***CMSIS-DAP Debugger*** "from the drop-down menu (step 2), and then tick the box "***Use Debug Driver***" (need to first uncheck and then tick)

Figure 4. Keil Utilities option



② For IAR compiling system, IAR7.0, IAR6.1 or above is recommended;
When using AT-Link-EZ in IAR environment, select "***CMSIS-DAP***" in "***Debugger***".

Figure 5. IAR Debug option

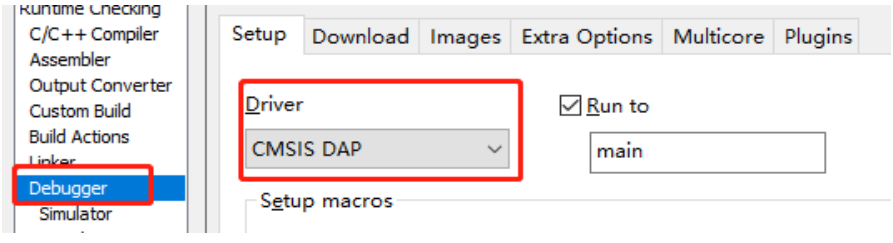
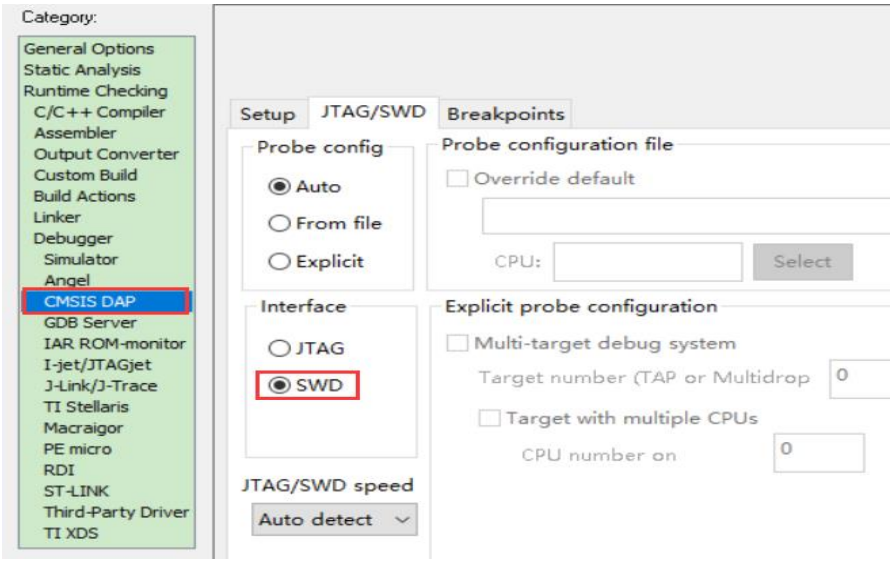


Figure 6. IAR CMSIS-DAP option



③ Selection of BSP and PACK (six scenarios are described as below)

Table 1. Selection of BSP and PACK

No.	BSP/Pack used	Whether to use AT32F403A_407 new features	Solutions
1	AT32F403A_407 BSP/Pack	Y	Refer to BSP related routines
2	SXX32F103 BSP/Pack	N	Download and run the program directly
3	SXX32F103 register operations	N	Download and run the program directly
4	SXX32F103 BSP + AT32 Pack	N	Modify the FPU settings
5	SXX32F103 register operations	Y	1. If the system max frequency is set above 108 MHz, need to configure an automatic step-by-step frequency switching function. 2. Refer to *Migration guide from SXX32F103 to AT32F403A_407_v1.0.0* for modifying related program
6	SXX32F103 BSP/Pack	Y	1. If the system max frequency is set above 108 MHz, need to configure an automatic step-by-step frequency switching function. 2. Refer to *Migration guide from SXX32F103 to AT32F403A_407_v1.0.0* for modifying related program

For detailed information on BSP and Pack, please refer to *AT32F4XX standard library BSP&Pack Application note*, which is located in the folder of *AT32F4xx_StdPeriph_Lib_V1.x.x* available from website.

Note: *Because AT has different Flash mechanism from SXX, there is no need to set the following two Flash wait bits for AT products:*

Figure 7. SXX Flash wait bits

```
/* Enable Prefetch Buffer */
FLASH->ACR |= FLASH_ACR_PRFTBE;
/* Flash 2 wait state */
FLASH->ACR &= (uint32_t)((uint32_t)~FLASH_ACR_LATENCY);
FLASH->ACR |= (uint32_t)FLASH_ACR_LATENCY_2;
```

This means that the Flash of SXX need to wait for 1-2 clock at high speed to fetch instructions, but AT has no such restriction.

1.1.4 How to quickly replace SXX

- Step 1: Based on peripheral specifications, Flash capacity, SRAM size, etc., de-solder SXX32F103 and replace it with the corresponding AT32F403A part no
- Step 2: Download SXX32F103 HEX file or BIN file using Artery ICP, ISP, KEIL or IAR
- Step 3: If necessary, download information other than SXX32F103HEX file or BIN file or perform system calibration
- Step 4: Check if the program can run normally.
- Step 5: For other issues, please refer to Migration guide from *SXX32F103 to AT32F403A_407*
- Step 6: if the program still cannot run normally after following the above steps, please refer to other chapters in this document, or contact your agent for assistance
- *Note:* because *AT32F403A_407 applies flexible memory expansion design, the internal Flash memory has a non-zero wait area, which will cause some SXX32F103 programs to run poorly on AT32F403A_407.*

  For how to improve operating efficiency, please refer to AN0004_Performance_Optimization_V1.0.x.pdf.

1.2 Enhanced functions of AT32F403A_407

1.2.1 PLL setting when the clock is greater than 72 MHz

AT32F403A_407 embeds a PLL that can output up to 240 MHz clock, the setting is slightly different

Thus the PLLRANGE register must be set depending on the output frequency.

```
#define   RCC_CFG_PLLRANGE_GT72MHZ                ((uint32_t)0x80000000
```

For example, AT32F403A PLL setting example: (HSE = 8 MHz, PLL = 240 MHz)

```
RCC->CFG |= (uint32_t)(RCC_CFG_PLLRC_HSE | RCC_CFG_PLLMULT30 | RCC_CFG_PLLRANGE_GT72MHZ);
```

Enable StepModeCmd:

```
RCC_StepModeCmd(ENABLE);
```

Disable StepModeCmd:

```
RCC_StepModeCmd(DISABLE);
```

Function definition of enable/disable StepModeCmd:

```
void RCC_StepModeCmd(FunctionalState NewState)
{
  assert_param(IS_FUNCTIONAL_STATE(NewState));
  if(NewState == ENABLE)
    {
      RCC->MISC2 |= RCC_MISC2_AUTO_STEP_EN;
    }
    else
    {
      RCC->MISC2 &= ~RCC_MISC2_AUTO_STEP_EN;
    }
}
```

Note: when the PLL is equal to 72 MHz, the setting procedure is the same as that of SXX32F103.

SXX32F103 PLL setting example: (HSE = 8 MHz, PLL = 72 MHz)

```
RCC->CFGR |= (uint32_t)(RCC_CFGR_PLLSRC_HSE | RCC_CFGR_PLLMULL9);
```

AT32F403A PLL setting example :(HSE = 8 MHz, PLL = 72 MHz)

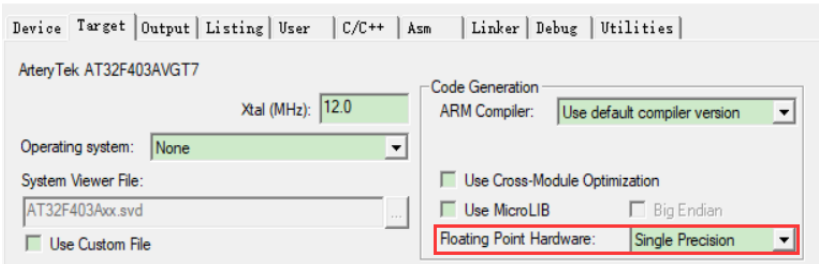
```
RCC->CFG |= (uint32_t)(RCC_CFG_PLLRC_HSE | RCC_CFG_PLLMULT9);
```

1.2.2 How to enable FPU (Floating Point Unit)

There are two situations in Keil environment:

① Use AT32F403A_407 BSP/Pack to directly select "***Floating Point Hardware***" as shown below:

Figure 8. Select FPU in ATK BSP/Pack Keil environment

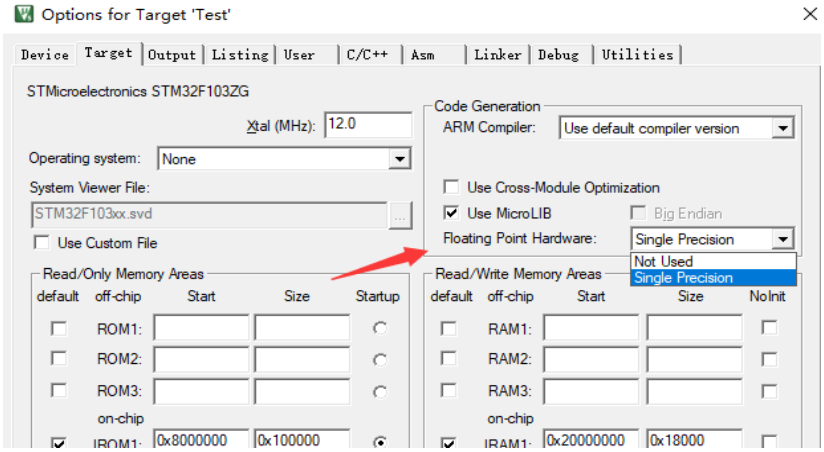


② Because SXX32F10X series does not support FPU function, if users want to enable FPU in the project developed under SXX library, the following procedures are required:

● Firstly, refer to *AT32F4xx standard library BSP & Pack Application note* to install Keil Environment Pack, and modify related header files.

● Then, select the corresponding AT parts, and then select in "***Options---Target***" as shown below:

Figure 9. Enable FPU in SXX BSP / ATK Pack Keil environment



● Finally, add the following configuration in the SystemInit function of system_stm32f10x.c, and add cm4.h to the project.

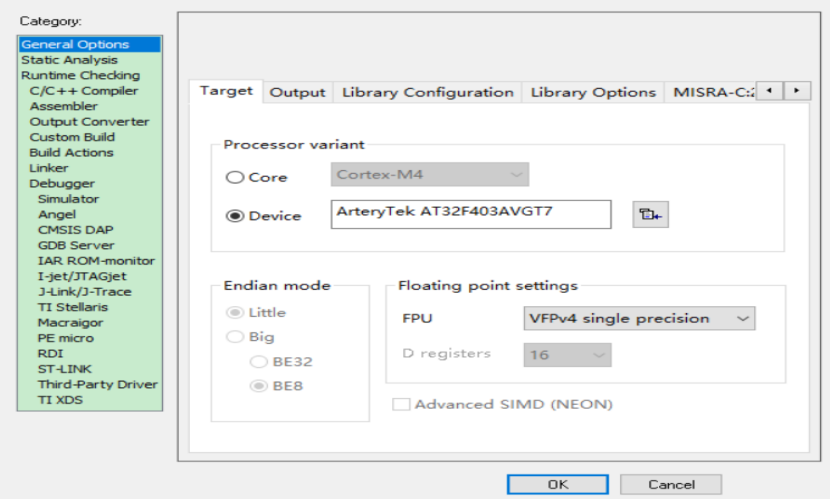
Figure 10. Add the code to enable FPU in Keil environment

```
void SystemInit (void)
{
  /* Enable FPU*/
#if defined (__FPU_USED) && (__FPU_USED == 1U)
  SCB->CPACR |= ((3U << 10U * 2U) |          /* set CP10 Full Access */
                 (3U << 11U * 2U)  );        /* set CP11 Full Access */
#endif
```

There are also two situations in IAR environment as follows:

① Use AT32F403A_407 BSP/Pack or modified SXX BSP/Pack to directly modify Floating Point Hardware, as shown in the figure below:

Figure 11. Enable FPU in IAR environment



② Because SXX32F10X series does not support FPU function, if users want to enable FPU in the project developed under SXX library, the following procedures are required:

- Firstly, refer to *AT32F4xx standard library BSP & Pack Application note* to install the Pack of IAR environment, and modify related header files;
- Then, select the corresponding AT parts, and then select in "***General Options---Target***";
- Finally, add the following configuration in the SystemInit function of system_stm32f10x.c, and add cm4.h to the project.

Figure 12. Add the code to enable FPU in IAR environment

```
#if defined (__FPU_USED) && (__FPU_USED == 1U)
  SCB->CPACR |= ((3U << 10U * 2U) |          /* set CP10 Full Access */
                 (3U << 11U * 2U)  );        /* set CP11 Full Access */
#endif
```

1.2.3 Configuration for AT32F403A_407 ZW/NZW Flash and embedded SRAM Size

Except for AT32F403ACBT7, other products support the allocation of internal Flash memory and SRAM through Option Bytes configuration. Taking AT32F403AVGT7 as an example, the internal Flash memory and SRAM can be configured as follows:

- ZW: 256 KB, NZW: 768 KB, SRAM: 96 KB (default)
- ZW: 128 KB, NZW: 896 KB, SRAM: 224 KB

The core reads the instruction code stored in the zero-wait Flash without any delay, and will not insert the wait clock because the CPU frequency is too fast and Flash cannot keep up with it. Assuming the system clock is 240 MHz, the AT32F403A has 256 KB of zero wait. The first 256 KB of the 512 KB bin file can be executed at 240 MHz, and the last 256 KB bin file is stored in the non-zero wait area with the execution rate of 96 MHz, which is still faster than the max frequency 72 MHz of SXX32F10X. The operating rate of non-zero wait is 0.4 times of that of zero wait.

Embedded SRAM 96KB (default)/224 KB can be selected in any of the following ways:
The AT32F403A SRAM size configuration involves the FMC option byte, which is selected by configuring the EOPB0. The address is 0x1FFF_F810.
EOPB0=0xFF means that the on-chip SRAM is 96 KB, while, EOPB0=0xFE for 224 KB SRAM.

It must be powered down or reset once to enable the EOPB0 to be valid.

① Using ICP/ISP

■ ICP tool

"***Target***"---select "***96 KB/224 KB***" in option byte---"***Apply to device***".

Figure 13. Select SRAM size in ICP tool option byte

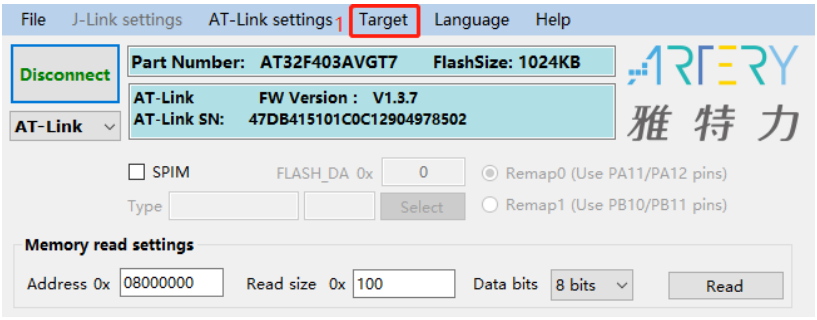
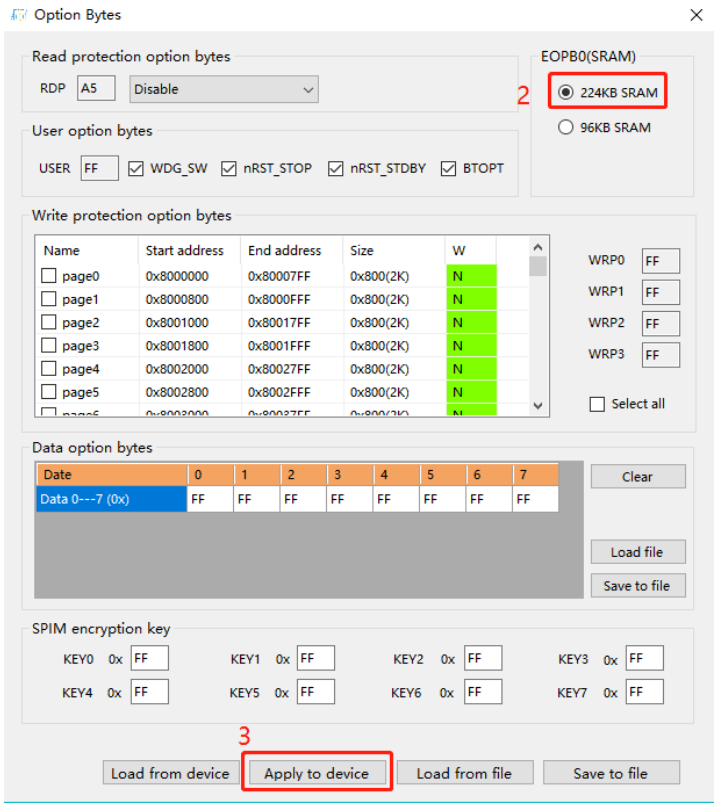


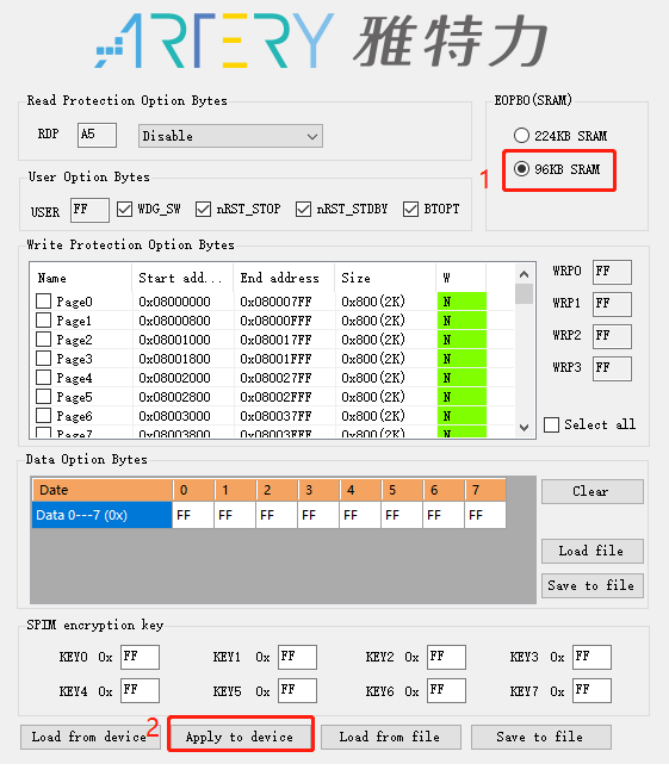
Figure 14. Select SRAM size in ICP tool option byte



■ Artery ISP Programmer tool

Enter the final interface, select "***96 KB/224 KB***"---"***Apply to device***"

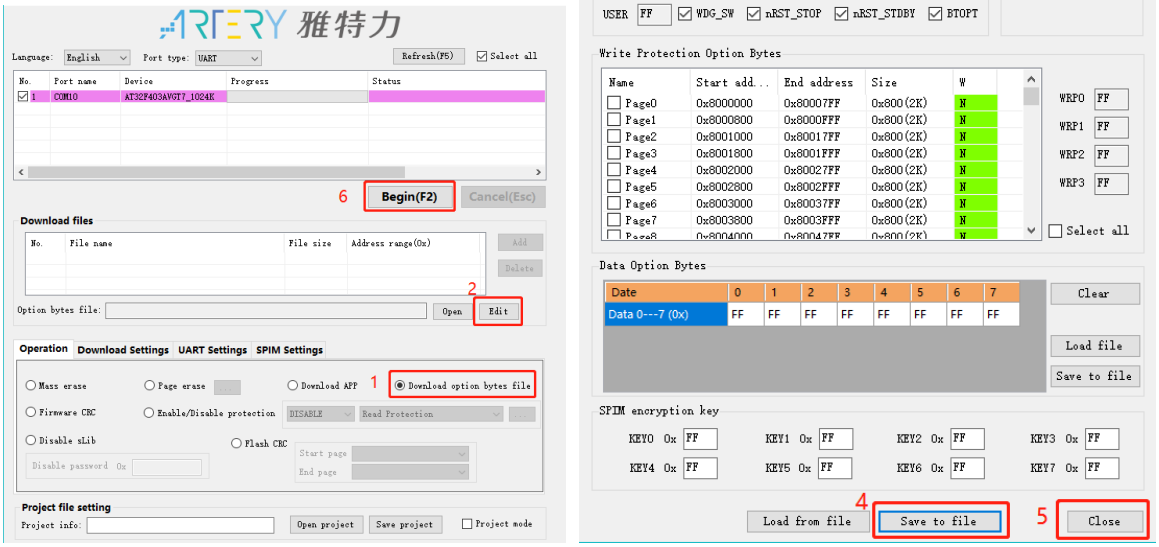
Figure 15. Select SRAM size in ISP tool option byte



■ Artery ISP Multi-Port Programmer tool

"***Download option byte file***"---"***Edit***"---select "***96 KB/224 KB***"---"***Save to file***" (create a new option byte programming file)---"***Close***"---"***Begin***", or "***Download option byte file***"---"***Open***" (open the saved option byte programming file) ---"***Begin***"

Figure 16. Select option bytes in ISP Multi-Port Programmer tool



② Customers can also modify SRAM size in the Bootloader program (IAP), the function is defined as follows and calls it, as shown in the figure below:

```
void Extend_SRAM(void)
{
  // check if RAM has been set to 224K, if not, change EOPB0
  if(((UOPTB->EOPB0) & 0xFF) != 0xFE)
  {
    /* Unlock Option Bytes Program Erase controller */
    FLASH_Unlock();

    /* Erase Option Bytes */
    FLASH_EraseUserOptionBytes();

    /* Change SRAM size to 224KB */
    FLASH_ProgramUserOptionByteData((uint32_t)&UOPTB->EOPB0,0xFE);
    NVIC_SystemReset();
  }
}
```

③ Modify the SRAM of AT32F403A_407 to 224 KB in the startup file

SARM will be loaded when the startup file is running. If the program has no IAP and the SRAM used by the application is greater than 96 KB, then the loading would fail and enter a hardfault, causing the application to fail to run. So you can configure the SRAM size to 224 KB before loading the SRAM in the starup file. Add the following code in red to the startup file in Keil compilation environment.

```
; Reset handler
Reset_Handler    PROC
                 EXPORT  Reset_Handler                    [WEAK]
                 IMPORT  __main
                 IMPORT  SystemInit

                  IMPORT      Extend_SRAM
                 MOV32    R0, #0x20001000
                 MOV      SP, R0
                 LDR      R0, =Extend_SRAM
                 BLX      R0
                 MOV32    R0, #0x08000000
                 LDR      SP, [R0]

                 LDR      R0, =SystemInit
                 BLX      R0
                 LDR      R0, =__main
                 BX       R0
                 ENDP
```

Add the following red font to the startup file in IAR environment.

```
; Default interrupt handlers.
        THUMB

        PUBWEAK Reset_Handler
        SECTION .text:CODE:REORDER:NOROOT(2)

        EXTERN    Extend_SRAM
Reset_Handler
        MOV32    R0,#0x20001000
        MOV      SP,R0
        LDR      R0,=Extend_SRAM
        BLX      R0
        MOV32    R0,#0x08000000
        LDR      SP,[R0]

        LDR      R0, =SystemInit
        BLX      R0
        LDR      R0, =__iar_program_start
        BX       R0
```

Add declaration and define Extend_SRAM function in the application

```
void Extend_SRAM(void)
{
  // check if RAM has been set to 224K, if not, change EOPB0
  if(((UOPTB->EOPB0) & 0xFF) != 0xFE)
  {
      /* Unlock Option Bytes Program Erase controller */
    FLASH_Unlock();

    /* Erase Option Bytes */
    FLASH_EraseUserOptionBytes();

    /* Change SRAM size to 224KB */
    FLASH_ProgramUserOptionByteData((uint32_t)&UOPTB->EOPB0,0xFE);
    NVIC_SystemReset();
  }
}
```

Note: *It is not recommended to use APP to change the SRAM size; if the SRAM space used in APP is greater than the modified SRAM size, the program will enter a Hardfault.*

1.2.4 Encryption (read protection and external Flash encrypted)

1.2.4.1 Read protection

Read protection, commonly referred to as “encryption”, acts on the entire Flash storage area. Once the read protection is set in the Flash, the embedded Flash storage area can only be read through the normal execution of the program, instead of JTAG or SWD. When the read protection is disabled using ISP/ICP tool, the chip will erase the Flash.
ISP/ICP tool can be used to enable/disable read protection as follows:

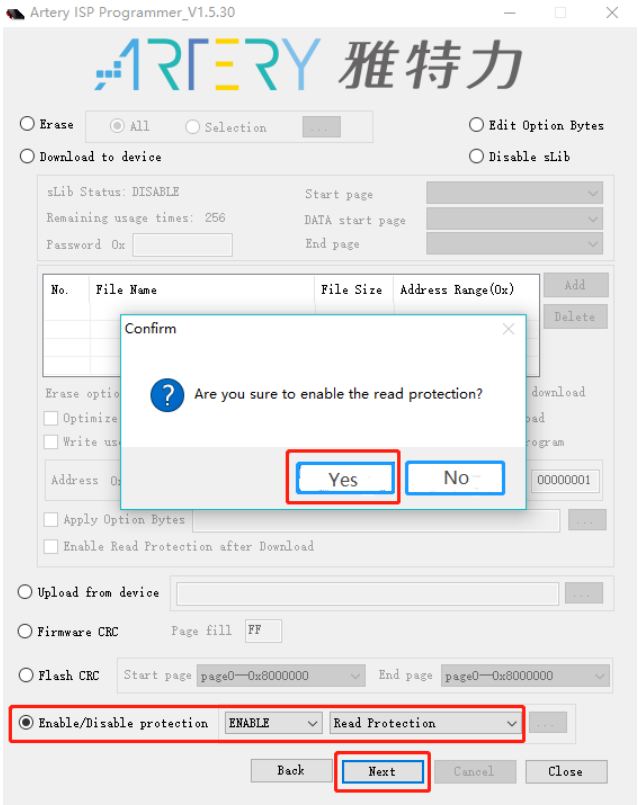
■ ICP tool

Read protection: “***Target***”---“***read protection***”---“***enable protection***”
Disable read protection: “***Target”***---“***read protection***”---“***disable read protection***”

■ Artery ISP Programmer tool

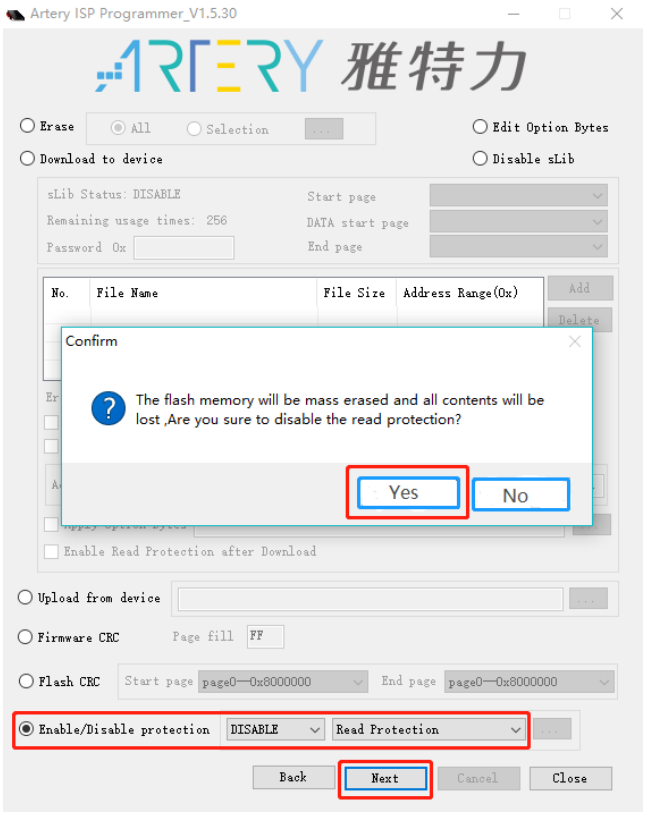
Read protection: “***enable/disable protection***”---“***enable---read protection***”---“***Next***”---“***Yes***”, and the program is encrypted.

Figure 17. Enable protection using ISP



Disable read protection: "***enable/disable protection***"---"***disable-read protection***"---"***Next***"—"***Yes***", and then the Flash can be unencrypted.

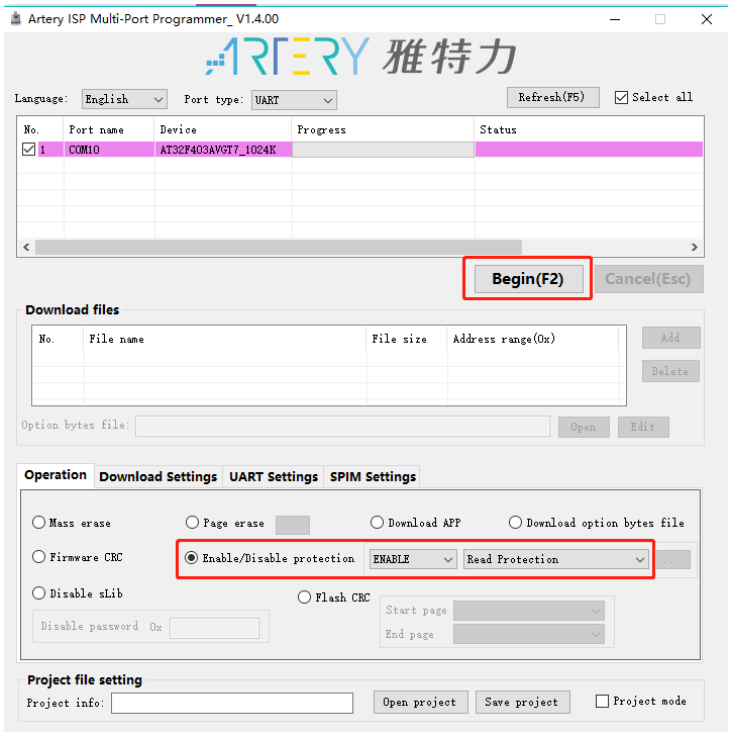
Figure 18. Disable read protection using ISP



■ Artery ISP Multi-Port Programmer

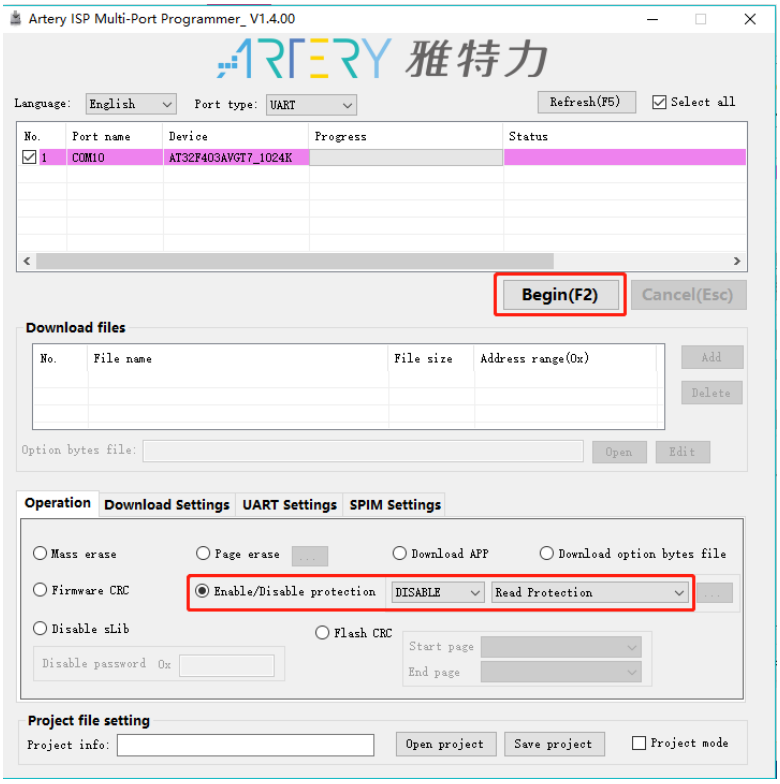
Read protection: "***enable/disable protection***"---"***enable-read protection***"---click on "**Begin**", and then the program is encrypted.

Figure 19. Enable read protection using ISP Multi-Port Programmer



Disable read protection: "***enable/disable protection***"--- "***disable-read protection***"—click on "***Begin***", then the Flash is unencrypted. The read protection cannot be disabled by the erase operation.

Figure 20. ISP Multi-Port Programmer disable read protection



1.2.4.2 Encryption of external Flash

To encrypt the external Flash, you need to set the encryption range and encryption password before programming the user program, and then enable read protection.

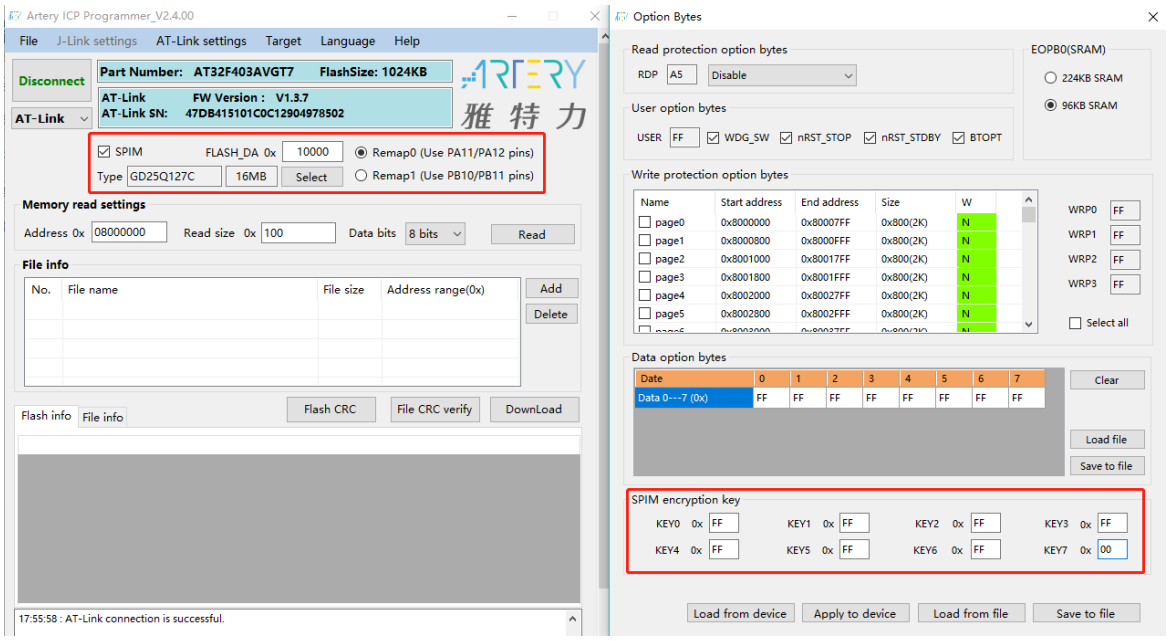
The encryption range refers to the size of the space that needs to be encrypted starting from the address 0x08400000. If the encryption key of the external memory is all 0xFF or 0x00, it will not be encrypted, otherwise it will be. Disabling the read protection will set the encryption key of external memory to all 0xFF.

The following example shows how to encrypt the external memory using ICP/ISP tool:

■ ICP tool:

Tick **"*SPIM*"**---select SPIM type---set **"*FLASH_DA*"**---click on **"*Target*"**---**"*Option Byte*"**---Modify "**SPIM encryption key**"---"***Apply to device***".

Figure 21. SPIM encryption operation using ICP tool



■ Artery ISP Programmer tool

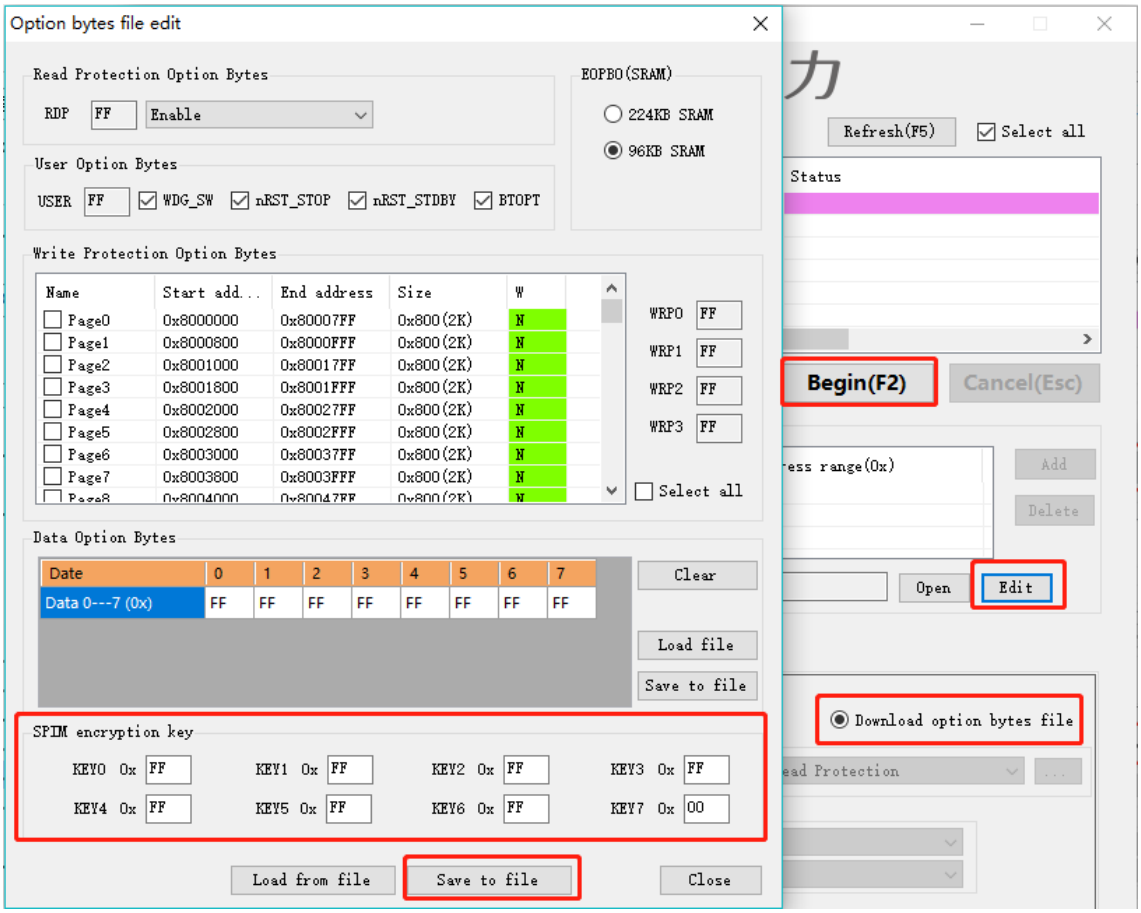
Click on "***Edit option byte***"---Modify "***SPIM encryption key***"---"***Apply to device***".

Figure 22. SPIM encryption operation through ISP tool

■ Artery ISP Multi-Port Programmer tool

"***Download option byte file***"---"***Edit***"--- Modify "***SPIM encryption key***"---"***Save to file***"---"***Begin***"

Figure 23. SPIM encryption operation by ISP Multi-Port Programmer tool



1.2.5 How to distinguish AT and other IC in the program

1.2.5.1 Using the UID/PID to distinguish

■ Read the Cortex-M CPU ID to distinguish M0, M3 and M4

```
cortex_id = *(uint32_t *)0xE000ED00;                    //read cortex

if((cortex_id == 0x410FC240) || (cortex_id == 0x410FC241))
{
    printf("This chip is Cortex-M4.\r\n");
}
else
{
    printf("This chip is Other Device.\r\n");
}
```

■ Read PID and UID to distinguish

```
/* Read the base address of    PID/UID of AT32 MCU*/
#define DEVICE_ID_ADDR1 0x1FFFF7F3
#define DEVICE_ID_ADDR2 0xE0042000

/* AT32F403A MCU type table */
const uint64_t AT32_MCU_ID_TABLE[] =
{
    0x0000000270050242,          //AT32F403ARCT7    256KB    LQFP64
    0x00000002700502CA,          //AT32F403ARET7    512KB    LQFP64
    …
};

/* Read PID/UID */
ID[0] = *(int*)DEVICE_ID_ADDR1;
ID[1] = *(int*)(DEVICE_ID_ADDR2+3);
ID[2] = *(int*)(DEVICE_ID_ADDR2+2);
ID[3] = *(int*)(DEVICE_ID_ADDR2+1);
ID[4] = *(int*)(DEVICE_ID_ADDR2+0);

/* Combine PID/UID */
AT_device_id =
((uint64_t)ID[0]<<32)|((uint64_t)ID[1]<<24)|((uint64_t)ID[2]<<16)|((uint64_t)ID[3]<<8)|((uint64_t)ID[4]<<0);

/* judge AT32 MCU */
for(i=0;i<sizeof(AT32_MCU_ID_TABLE)/sizeof(AT32_MCU_ID_TABLE[0]);i++)
{
    if(AT_device_id == AT32_MCU_ID_TABLE[i])
    {
        printf("This chip is AT32F4xx.\r\n");
    }
    else
    {
        printf("This chip is Other Device.\r\n");
    }
}
```

Note: there are multiple ID codes in the AT32F4xx microcontrollers, they assemble the acquired ID information into a 64-bit data to distinguish which type of MCU is.

PROJECT ID: the access address is 0x1FFF F7F3 [7:0], which defines the project model of Artery MCU.

DEVICE ID: the access address is 0xE004 2000 [31:0], which defines the device model of MCU.

1.2.5.2 Take 32-bit as the simplified unique UID code

When the users do not want to read 96-bit UID (base address 0x1FFF F7E8, 0x1FFF F7EC, 0x1FFF F7F0), only use 32-bit as the simplified unique UID code, they can just read [87:79] [33:28][16:0] bits.

2 FAQs during download and compiling

2.1 The program enters Hard Fault Handler at startup

2.1.1 Abnormal circumstances triggering Hardfault

- The SRAM exceeds the SRAM size set by option bytes, please use ICP/ISP to program.
- The single precision function is enabled in Keil or IAR, but the M4 core FPU register is not enabled in the code, so the use needs to enable FPU the code.

Figure 24. Add a code to enable FPU

```
void SystemInit (void)
{
  /* Enable FPU*/
#if defined (__FPU_USED) && (__FPU_USED == 1U)
  SCB->CPACR |= ((3U << 10U * 2U) |          /* set CP10 Full Access */
                 (3U << 11U * 2U)  );        /* set CP11 Full Access */
#endif
```

- Access data is out of bounds

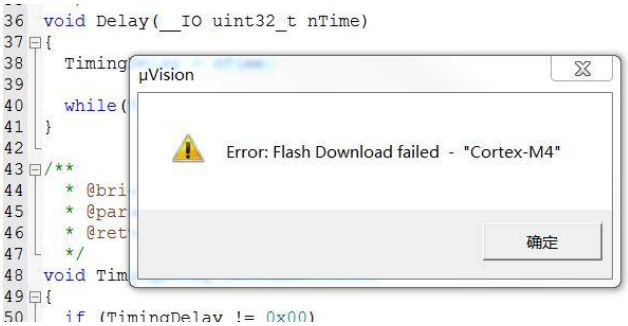
Find out what the problem is, and change it to the normal data area.

2.2 Error occurred during download

2.2.1 Error: Flash Download failed – “Cortex-M4”

An error pops up during KEIL emulation or download:

Figure 25. Flash Download failed – “Cortex- 4”



The possible reasons are as follows:

- Read protection is enabled: you need to first disable MCU read protection before download;
- Select a wrong Flash file algorithm or not to load the Flash file algorithm: you need add a correct Flash file algorithm at FlashDownload
- J-Link driver version is too old: the driver 6.20C or above is recommended.

2.2.2 No Debug Unit Device found

- The download port is occupied, for example, ICP is connecting to the target device.
- JTAG/SWD connection error or no connection.

2.2.3 RDDI-DAP Error

- Disable the JTAG/SWD PIN, please refer to “2.2.5 AT32 resume download for solutions”.

2.2.4 ISP serial interface gets stuck during download

When the ISP serial interface is used to download, it occasionally gets stuck, causing the PC not to release the serial port.

Solutions:

- Power supply is not stable;
- Use a better USB-to-serial interface tool, such as CH340 chip.

2.2.5 AT32 resume download

When using AT32F403A_407, users may not be able to download the program after the following operations:

- After the JTAG/SWD PIN is disabled, the program cannot be downloaded and the JTAG/SWD device cannot be found.
- After entering Standby mode, the program cannot be downloaded and JTAG/SWD device cannot be found.

Here we provide the solutions in KEIL and IAR environment:

- Solution 1: switch boot mode

Switch the boot mode to Boot[1:0]=01b or Boot[1:0]=11b, and press the reset button to resume download. In the same way, ISP can also resume download.

- Solution 2: ICP tool and AT-Link-EZ

AT-Link-EZ is specially designed for AT32, so ICP and AT-Link-EZ can use resume download.

3 Revision history

Table 2. Document revision history

Date	Revision	Changes
2020.3.16	1.0.0	Initial release
2020.12.3	1.0.1	Deleted the descripton in IAR, the pack does not support bank 3 operation in *Section 1.1.3*

IMPORTANT NOTICE – PLEASE READ CAREFULLY

Purchasers understand and agree that purchasers are solely responsible for the selection and use of Artery's products and services.

Artery's products and services are provided "AS IS" and Artery provides no warranties express, implied or statutory, including, without limitation, any implied warranties of merchantability, satisfactory quality, non-infringement, or fitness for a particular purpose with respect to the Artery's products and services.

Notwithstanding anything to the contrary, purchasers acquires no right, title or interest in any Artery's products and services or any intellectual property rights embodied therein. In no event shall Artery's products and services provided be construed as (a) granting purchasers, expressly or by implication, estoppel or otherwise, a license to use third party's products and services; or (b) licensing the third parties' intellectual property rights; or (c) warranting the third party's products and services and its intellectual property rights.

Purchasers hereby agrees that Artery's products are not authorized for use as, and purchasers shall not integrate, promote, sell or otherwise transfer any Artery's product to any customer or end user for use as critical components in (a) any medical, life saving or life support device or system, or (b) any safety device or system in any automotive application and mechanism (including but not limited to automotive brake or airbag systems), or (c) any nuclear facilities, or (d) any air traffic control device, application or system, or (e) any weapons device, application or system, or (f) any other device, application or system where it is reasonably foreseeable that failure of the Artery's products as used in such device, application or system would lead to death, bodily injury or catastrophic property damage.