AN0023
Application Note

Getting Started with AT32F403A & AT32F407

Introduction

This application note is written to help users with rapid project development using AT32F403Axx /AT32F407xx (compared with AT32F403A, AT32F407 is featured with Ethernet media access control (EMAC) additionally).

Note: The corresponding code in this application note is developed on the basis of V2.x.x BSP provided by Artery. For other versions of BSP, please pay attention to the differences in usage.

Applicable products:

Part number	AT32F403Axx
	AT32F407xx

Contents

List of Tables

List of Figures

1 Preliminary environment requirements

Download development environment

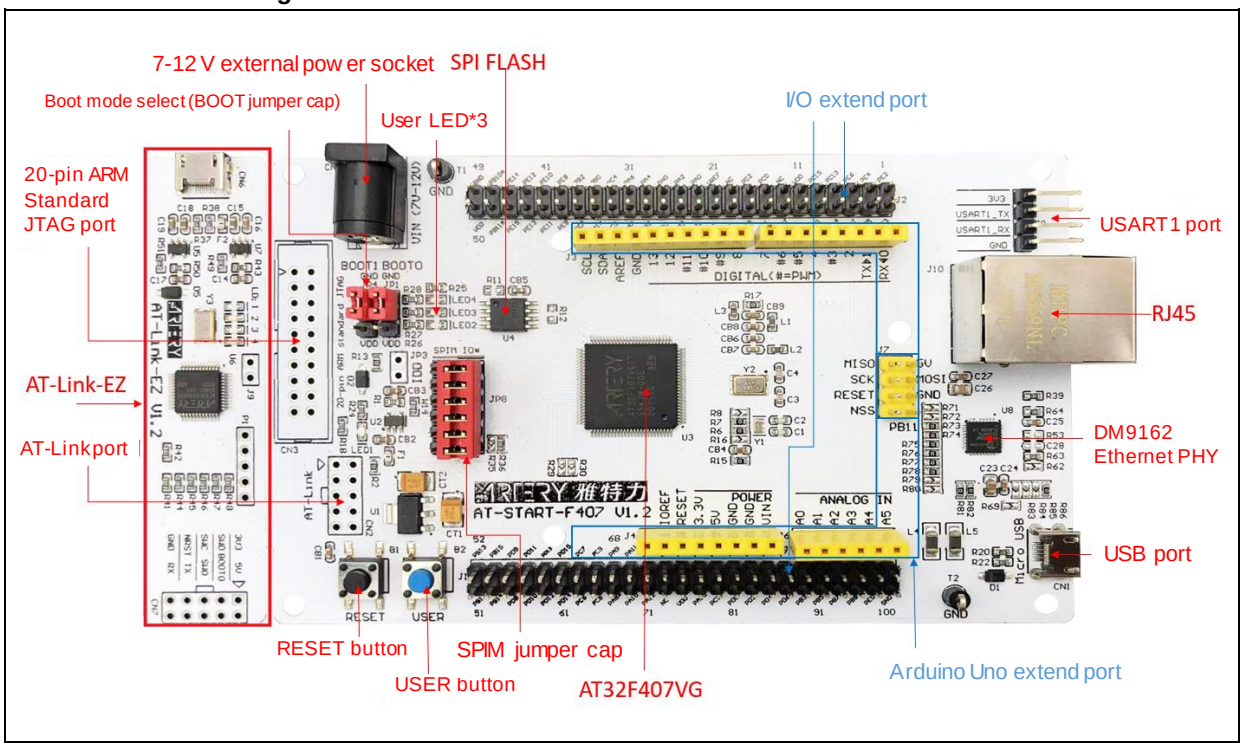
- ARTERY's official website

1.1 Build AT32 development environment

1.1.1 Debug tools and evaluation board

The AT32F403A /AT32F407 evaluation board has an AT-Link-EZ debug tool, as shown in the red box in Figure 1 below. The AT-Link-EZ can be disassembled and used with other circuit boards, supporting IDE online debugging, online programming and USB-to-serial port.

Figure 1. AT-START-F407 evaluation board with AT-Link-EZ



Note: For details about resources for AT-START evaluation board, please refer to UM_AT_START_F40x_Vx.x. Path: ARTERY's official website→PRODUCTS→Mainstream→AT32F4xx; download and unzip Evaluation Board package, and get the "\AT_START_F40x_Vx.x\03_Documents".

Figure 2. AT-START-F407 evaluation board package

Evaluation Board

Download	Description	Version
AT-START-F407	AT32F407 evaluation board supporting Arduino standard interfaces	V1.2

1.1.2 Programming tools and software

- AT programming tools and software: AT-Link / AT-Link+ /AT-Link-Pro / AT-Link-ISO /AT-Link-EZ, ICP/ISP.

- 3rd party programming tools: J-Link, Armfly, Alientek, XWOPEN, ICWORKSHOP, ZLG, MaxWiz, Amomcu, Acroview, Forcreat, Galecomm, Prosystems, Rx-prog, Sinaen, XELTEK, Zhifeng, etc.

Note: For more information, please visit ARTERY's official website→SUPPORT→Hardware Development Tool and 3RD Party Writer.

- For ICP usage instructions, please refer to *UM_ICP_Programmer*. Path: ARTERY's official website→PRODUCTS→Mainstream→AT32F4xx; download and unzip ICP tool, and get the "Artery_ICP_Programmer_Vx.x.xx\Document\UM_ICP_Programmer.
- For ISP usage instructions, please refer to *UM_ISP_Programmer*. Path: ARTERY's official website→PRODUCTS→Mainstream→AT32F4xx; download and unzip ISP tool, and get the "Artery_ISP_Programmer_Vx.x.xx\Document\UM_ISP_Programmer.
- For AT-Link usage instructions, please refer to *UM0004_AT-Link_User_Manual*. Path: ARTERY's official website→PRODUCTS→Mainstream→AT32F4xx; download and unzip AT-Link-Family, and get the "AT_Link_CH_ Vx.x.x\05_Documents\UM0004_AT-Link_User_Manual_ZH_Vx.x.x".

Figure 3. ICP/ISP/AT-Link-Family

Tool

Download	Description	Version
AT32 IDE_Linux AT32 IDE_Windows	支持AT32 MCU的基于Eclipse开发的跨平台ARM嵌入式系统的软件开发环境	V1.0.02
AT-Link Family	支持AT32 MCU 仿真与在线/离线烧录工具 (包含AT-Link-EZ/AT-Link/AT-Link-Pro/AT-Link-ISO四种工具)	V2.1.1
AT-Link Console	支持AT32 MCU「在电路编程」Console工具	V3.0.02
ICP	支持AT32 MCU「在电路编程」工具	V3.0.05
ISP	支持AT32 MCU「在系统编程」工具	V2.0.06
ISP Multi-Port	支持AT32 MCU一对多设备「在系统编程」工具	V2.0.06

1.1.3 AT32 development environment

1.1.3.1 Template projects

The commonly used IDE template projects are available from BSP provided by ArteryTek. BSP path: ARTERY's official website→PRODUCTS→Mainstream→AT32F4xx.

Figure 4. BSP package

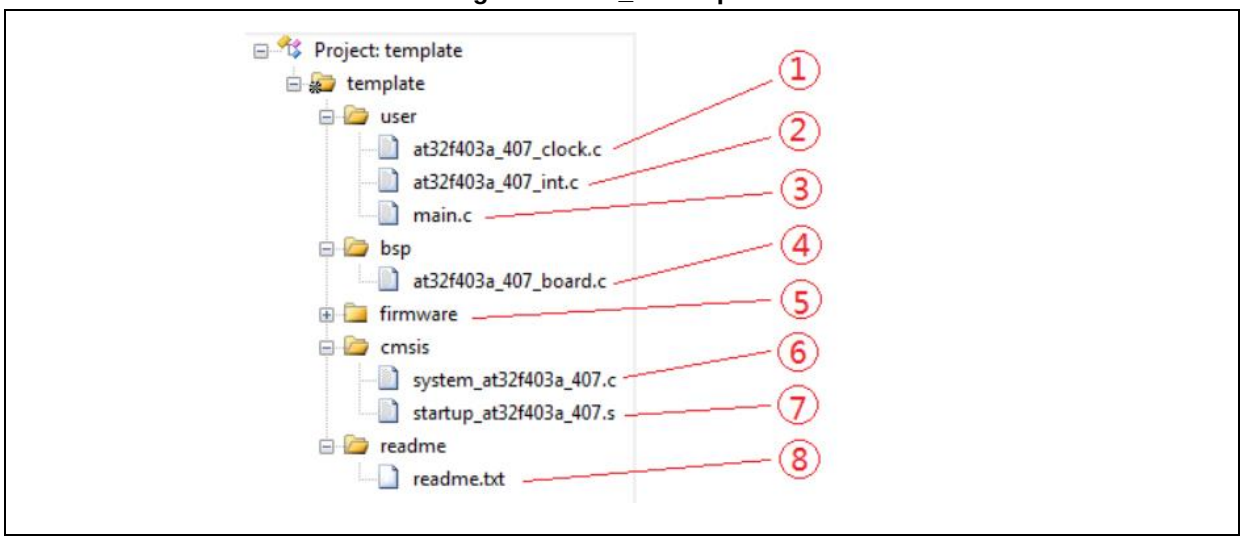
BSP

Download	Description	Version
Firmware Library	AT32F407 firmware library BSP user guide	V2.1.3

For AT32F403A/407 series, template projects of Keil_v5/Keil_v4/IAR_6.10/IAR_7.4/IAR_8.2/eclipse_gcc/at32_ide are created in BSP. Path:

AT32F403A_407_Firmware_Library_V2.x.x\project\at_start_f4xx\templates. Open the project folder and click on the project file to open the corresponding IDE project. The example of Keil_v5 template project is shown below.

Figure 5. Keil_v5 templates



Contents in the project:

① at32f403a_407_clock.c: clock configuration file, contains default clock frequency and clock path;

② at32f403a_407_int.c: interrupt file, part of core interrupt function code flow is written by default;

③ main.c: main code file of the template project;

④ at32f403a_407_board.c: board-level configuration file, contains settings of AT-START on-board buttons and LEDs;

⑤ at32f403a_407_xx.c under *firmware*: driver file of on-chip peripherals;

⑥ system_at32f403a_407.c: system initialization file;

⑦ startup_at32f403a_407.s: startup file;

⑧ readme.txt: project documentation, contains application functions, setting methods and associated application notes (ApNote) of the template project.

Except for templates, BSP also includes code examples (Keil_v5 project files) in terms of peripherals for reference.

Path: AT32F403A_407_Firmware_Library_V2.x.x\project\at_start_f4xx\examples.

Note: For more details about BSP, please refer to "Section 4 BSP application" of AT32F403A_407 Firmware BSP&Pack User Guide. Path: ARTERY's official website→PRODUCTS→Mainstream→AT32F4xx; download and unzip BSP, and get the "\AT32F403A_407_Firmware_Library_Vx.x.x\document".

1.1.3.2 Pack installation

Install Pack and add the AT32 MCU part number to Keil/IAR. You can download Pack from ARTERY's official website→PRODUCTS→Mainstream→AT32F4xx.

Figure 6. Pack download

Pack

Download	Description	Version
Keil 4 Keil 5	Supports AT32 MCU to run in Keil MDK	V2.2.0 V2.2.3
IAR	Supports AT32 MCU to run in IAR EWARM	V2.1.6
Segger	Supports Segger tools to identify AT32 MCU	V2.0.7

For Keil compiling system, keil 4.74 /5.23 or above is recommended. If Keil_v5 is used, please unzip Keil5_AT32MCU_AddOn and install the corresponding ArteryTek.AT32F403A_407_DFP. If Keil_v4 is used, please install Keil4_AT32MCU_AddOn. By default, the Keil installation path can be recognized automatically during installation. If the path is not recognized or incorrect, you need to manually select the Keil installation path.

Figure 7. Set up ArteryTek.AT32F403A_407_DFP

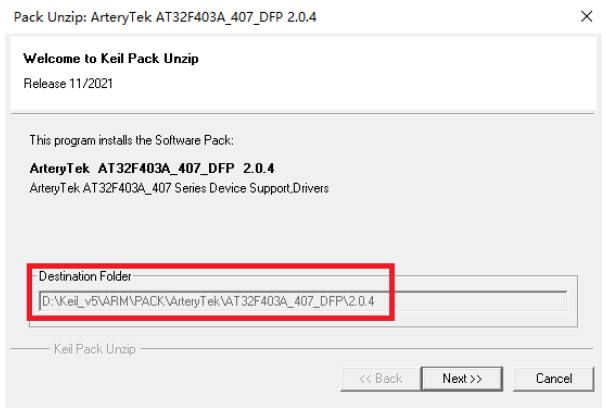
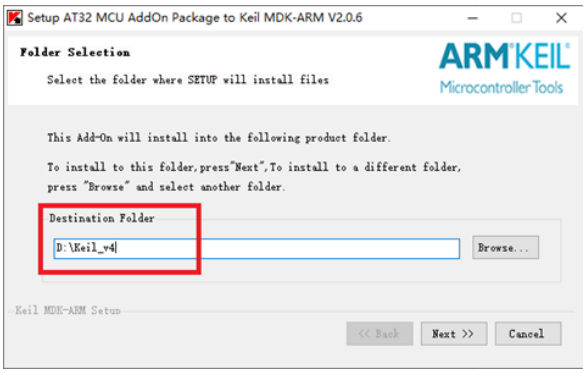
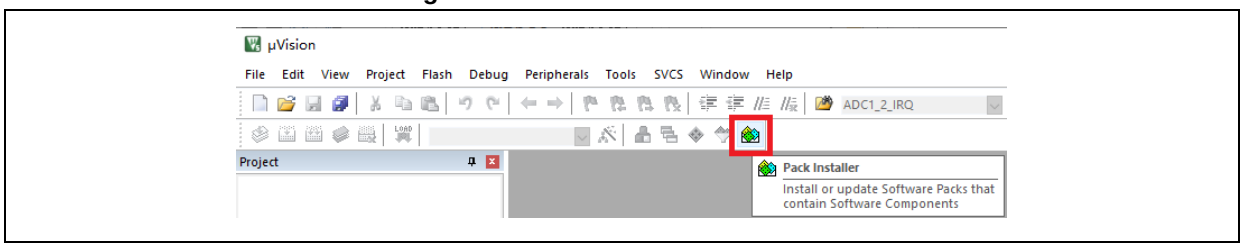


Figure 8. Set up Keil4_AT32MCU_AddOn



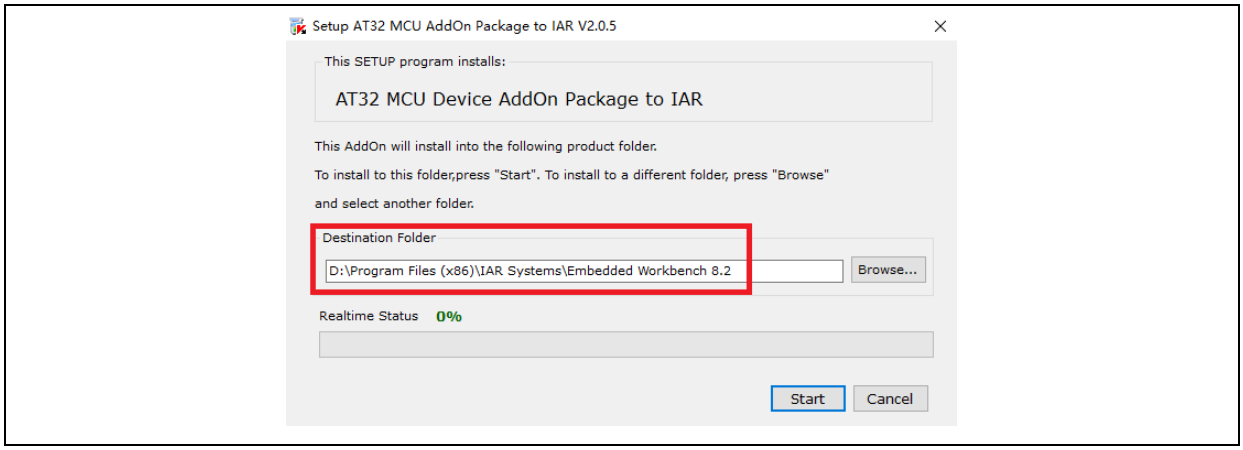
You can also open keil and click on “Pack Installer” icon; then click on the top left “file” and select “import” to import the corresponding pack downloaded from ARTERY's official website.

Figure 9. Pack Installer icon in Keil



For IAR compiling system, IAR7.0 or IAR6.1 above is recommended. It is necessary to install IAR_AT32MCU_AddOn. By default, the IAR installation path can be recognized automatically during installation. If the path is not recognized or incorrect, you need to manually select the IAR installation path.

Figure 10. Set up IAR_AT32MCU_AddOn

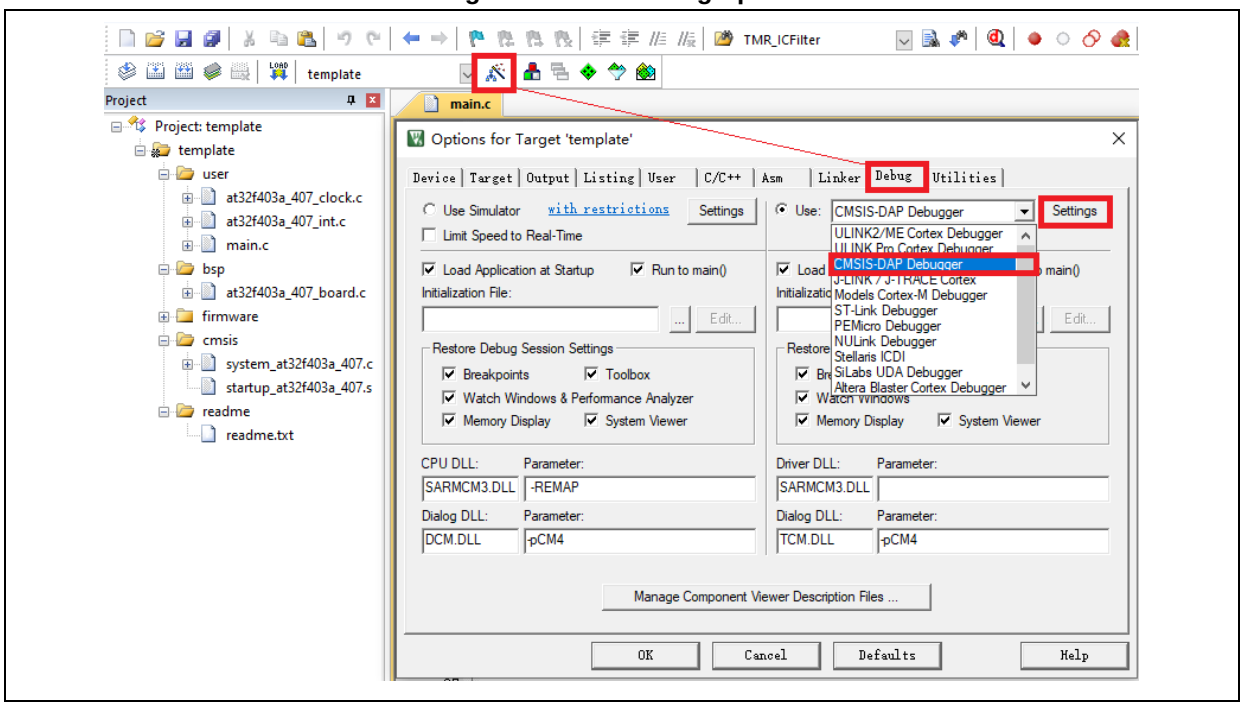


Note: For more details about Pack setup, please refer to "Section 2 Pack setup" of AT32F403A_407 Firmware BSP&Pack User Guide. Path: ARTERY's official website→PRODUCTS→Mainstream→AT32F4xx; download and unzip BSP, and get the "\AT32F403A_407_Firmware_Library_Vx.x.x\document".

1.1.3.3 Use AT-Link for debug and download

If you want to use AT-Link in IAR, select CMSIS-DAP in Debugger option.

Figure 11. Keil Debug option



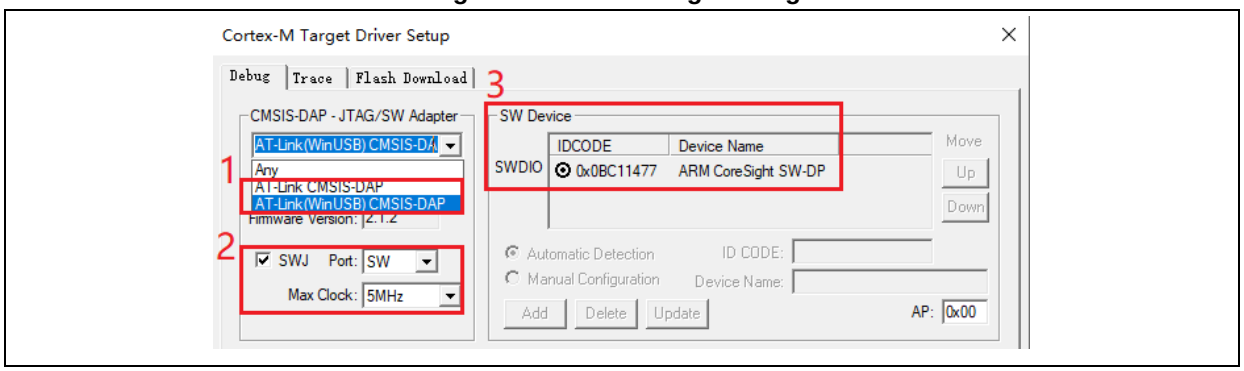
Go to Debug and click on Settings to enter the Cortex-M Target Driver Setup interface.

1. Select AT-Link(WinUSB)-CMSIS-DAP/AT-Link-CMSIS-DAP;

Note: For details about WinUSB, please refer to FAQ0136_How to use AT-LINK WinUSB to improve download speed (ARTERY's official website→SUPPORT→FAQ→FAQ0136).

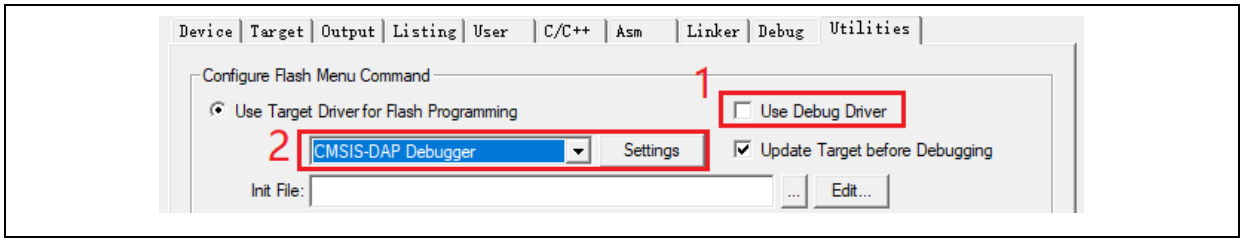
2. Find Port, select SW and then tick SWJ;
3. Confirm that the ARM SW-DP debug module is recognized.

Figure 12. Keil Debug Settings



Click on Utilities and untick option box 1; then select CMSIS-DAP Debugger in option box 2, and finally tick option box 1 (it should be unticked first and then ticked).

Figure 13. Keil Utilities



If you want to use AT-Link in IAR, please click on Project and select Options; then select CMSIS-DAP in Debugger option, and select SWD in CMSIS DAP option.

Figure 14. IAR Debug option

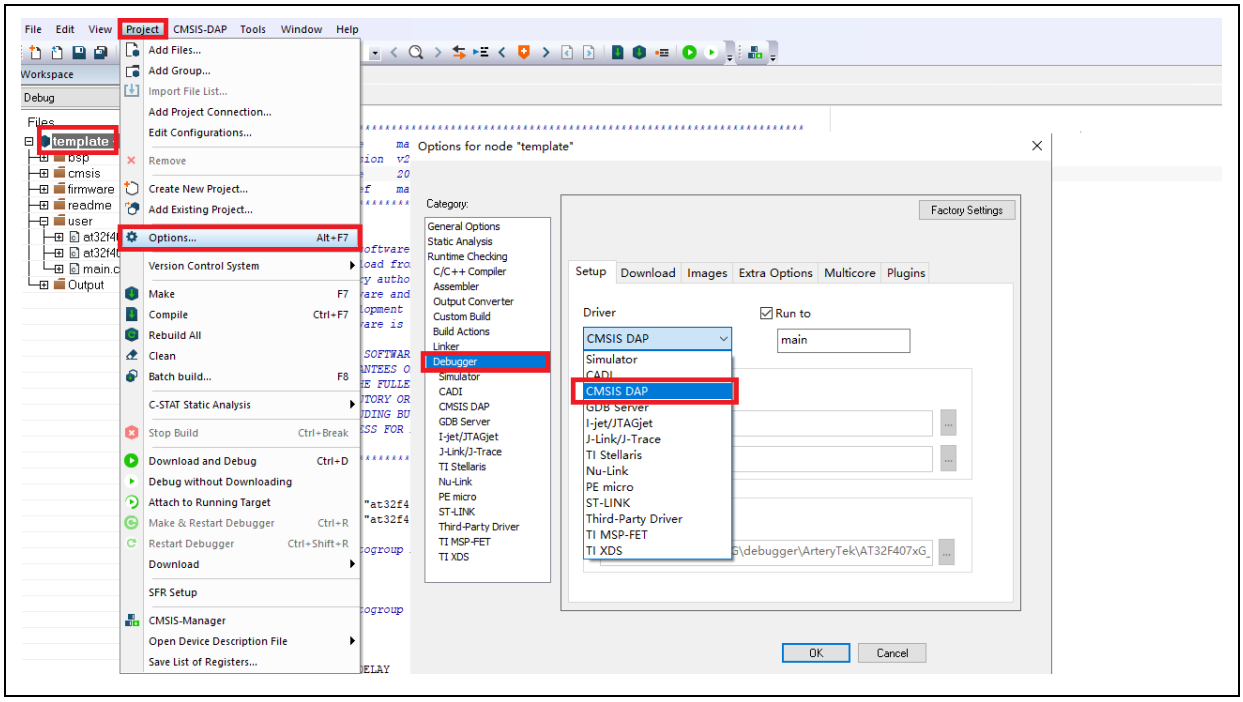
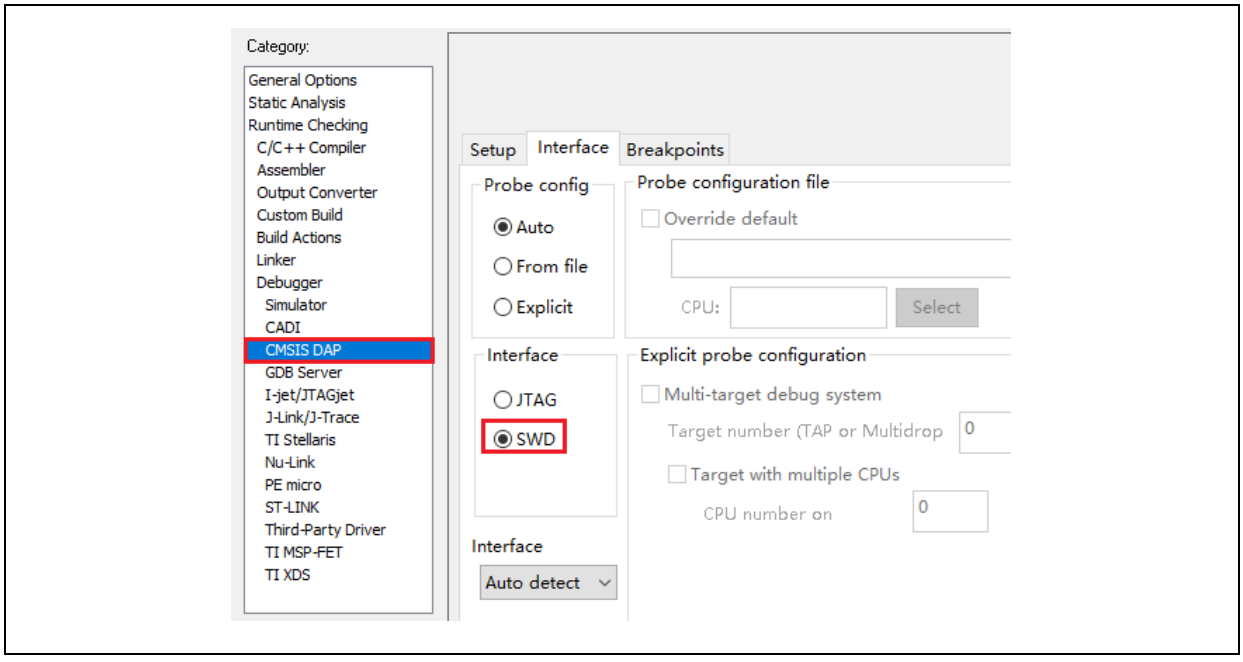


Figure 15. IAR CMSIS-DAP option



Note: For details about Flash algorithm file, MCU switch and solutions for "J-Link cannot find MCU", please refer to AT32F403A_407 Firmware BSP&Pack User Guide. Path: ARTERY's official website→PRODUCTS →Mainstream→AT32F4xx; download and unzip BSP (\AT32F403A_407_Firmware_Library_Vx.x.x\document).

1.1.4 How to replace SXX

- Compare the peripheral specifications, Flash size and SRAM size, etc.; unsolder SXX32F103 and replace it with the corresponding AT32F403A/AT32F407 part;

- Step 2: Use ARTERY ICP/ISP or KEIL/ IAR and download SXX32F103 HEX files or BIN files;
- Step 3: If necessary, download other materials together with SXX32F103 HEX files or BIN files, or perform system calibration;
- Step 4: Check if the program can run normally;
- Step 5: For other issues, please refer to *MG0007_ Migrating from SXX32F103 to AT32F403A* (ARTERY's official website→PRODUCTS→Mainstream→AT32F4xx);
- Step 6: If the program still cannot run properly after completing the above steps, please refer to other sections of this application note or contact the agent and ARTERY technicians for help.

Note: Due to the flexible memory expansion design of AT32F403A/AT32F407, its internal Flash memory has non-zero wait area, which may cause poor performance of some SXX32F103 programs on AT32F403A /AT32F407. In this case, please refer to AN0004_Performance_Optimization (ARTERY's official website→ SUPPORT→AP Note→AN0004).

1.2 AT32F403A /AT32F407 chip enhanced functions

1.2.1 PLL clock settings

1.2.1.1 PLL greater than 72 MHz

The internal PLL of AT32F403A/AT32F407 has a maximum output frequency of 240 MHz, and settings are slightly different when the clock is greater than 72 MHz. The user needs to set the PLLRANGE register according to the output frequency (PLL>72 MHz, PLLRANGE=1; PLL≤72 MHz, PLLRANGE=0).

When the SXX32F103 BSP is used, the PLL setting example (HEXT=8 MHz, PLL=72 MHz)

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

If the user wants to use SXX32F103 program to output a lock greater than 72 MHz on AT32F403A /AT32F407, it is necessary to configure the PLLRANGE bit.

When PLL=240 MHz, the corresponding setting is as follows:

Figure 16. Use SXX to output 240 MHz clock on AT32F403A /AT32F407

```
#define RCC_CFG_PLLRANGE_GT72MHZ ((uint32_t)0x80000000)
/*!< When PLL frequency is greater than 72MHz */
RCC->CFGR |= (uint32_t)(RCC_CFGR_PLLSRC_HSE | RCC_CFGR_PLLMULL30 |
RCC_CFG_PLLRANGE_GT72MHZ);
```

When the AT32F403A /AT32F407 BSP is used, the PLL setting example (HEXT=8 MHz)

Figure 17. AT32F403A /AT32F407 crm_pll_output_range parameter

```
typedef enum
{
  CRM_PLL_OUTPUT_RANGE_LE72MHZ= 0x00, /*!< pll clock output range less than or equal to 72mhz */
  CRM_PLL_OUTPUT_RANGE_GT72MHZ= 0x01 /*!< pll clock output range greater than 72mhz */
```

```
} crm_pll_output_range_type;
```

When PLL=72 MHz, the corresponding setting is as follows:

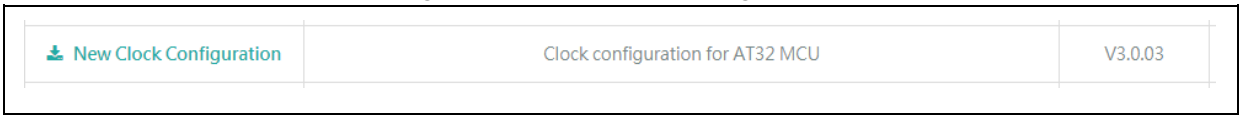
```
crm_pll_config(CRM_PLL_SOURCE_HEXT_DIV, CRM_PLL_MULT_18, CRM_PLL_OUTPUT_RANGE_LE72MHZ);
```

When PLL=240 MHz, the corresponding setting is as follows:

```
crm_pll_config(CRM_PLL_SOURCE_HEXT_DIV, CRM_PLL_MULT_60, CRM_PLL_OUTPUT_RANGE_GT72MHZ);
```

For details about clock configurations, refer to AN0082_AT32F403A_407_CRM_Start_Guide (ARTERY's official website→SUPPORT→AP Note→ AN0118) to learn how to configure and modify AT32F403A/AT32F407 clock source codes and use New Clock Configuration (ARTERY's official website→PRODUCTS→ Mainstream→AT32F4xx) to generate the desired clock code and apply it to the project.

Figure 18. New Clock Configuration



1.2.1.2 Auto step-by-step clock switch

When the internal PLL of AT32F403A /AT32F407 is set to 108 MHz and above, it is necessary to perform auto step-by-step clock switch function.

When the SXX32F103 program is used, the user needs to open system_Sxx32f10x.c, find out the current system clock frequency configuration function (go through Section 1.2.1.1 PLL settings), and add the following codes in Italic black to the *static void SetSysClockToxxM(void)* function.

Figure 19. SXX PLL auto step-by-step switch configurations

```
/* Wait till PLL is ready */
while((RCC->CR & RCC_CR_PLLRDY) == 0)
{
}
*((unsigned int *)0x40021054) |= (0x30); // Enable auto step-by-step clock switch function
/* Select PLL as system clock source */
RCC->CFGR &= (uint32_t)((uint32_t)~(RCC_CFGR_SW));
RCC->CFGR |= (uint32_t)RCC_CFGR_SW_PLL;
/* Wait till PLL is used as system clock source */
while ((RCC->CFGR & (uint32_t)RCC_CFGR_SWS) != (uint32_t)0x08)
{
}
*((unsigned int *)0x40021054) &=~ (0x30); // Disable auto step-by-step clock switch function
```

When AT32F403A /AT32F407 BSP is used, the example of PLL auto step-by-step switch is shown below:

Figure 20. AT32 PLL auto step-by-step switch configurations

```
/* enable auto step mode */
crm_auto_step_mode_enable(TRUE);
/* select pll as system clock source */
crm_sysclk_switch(CRM_SCLK_PLL);
/* wait till pll is used as system clock source */
while(crm_sysclk_switch_status_get() != CRM_SCLK_PLL)
{
}
/* disable auto step mode */
crm_auto_step_mode_enable(FALSE);
/* update system_core_clock global variable */
system_core_clock_update();
```

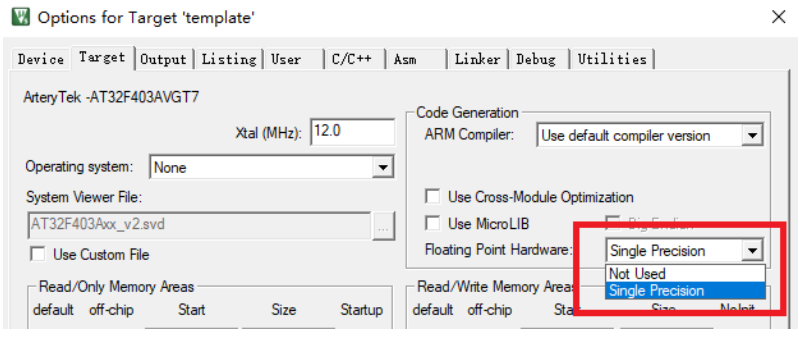
Note: If the auto step-by-step system clock switch is enabled, it must be disabled after clock switch.

1.2.2 How to enable FPU function

There are two conditions in Keil:

① Use AT32F403A_407 BSP/Pack or the modified SXX BSP/Pack to modify the Floating Point Hardware.

Figure 21. Enable FPU in Keil



② SXX32F10X does not support FPU function. If the user wants to enable FPU function in the developed project in SXX library, the following operations should be done:

- Refer to *AT32F403A_407 Firmware BSP&Pack User Guide* to set up Keil PACK and modify the relevant header files;
- Select the corresponding AT part number in Options—Target;
- Add the following configuration codes at the beginning of SystemInit function of system_stm32f10x.c, and add cm4.h to the project.

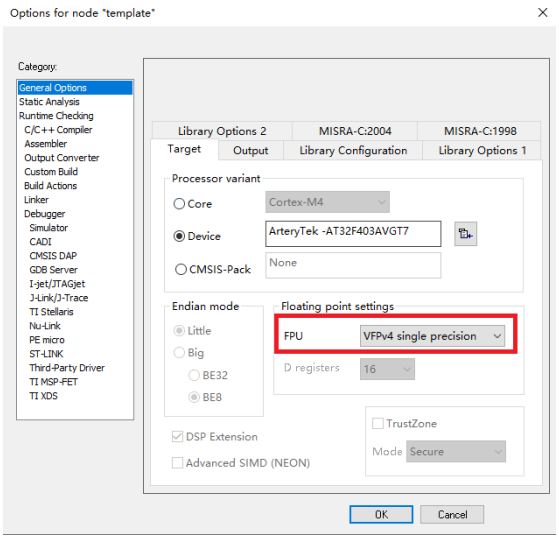
Figure 22. Add FPU enabling codes in Keil

```
/* 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 two conditions in IAR:

① Use AT32F403A_407 BSP/Pack or the modified SXX BSP/Pack to modify the Floating Point Hardware.

Figure 23. Enable FPU in IAR



② SXX32F10X does not support FPU function. If the user wants to enable FPU function in the developed project in SXX library, the following operations should be done:

- Refer to *AT32F403A_407 Firmware Library BSP&Pack User Guide* to set up IAR PACK and modify the relevant header files;
- Select the corresponding AT32 part number in General Options—Target;
- Add the following configuration codes at the beginning of SystemInit function of system_stm32f10x.c, and add cm4.h to the project.

Figure 24. Add FPU enabling codes in IAR

```
/* 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
```

For more information, please refer to AN0037_How_to_use_FPU (ARTERY's official website→SUPPORT→AP Note→AN0037), which details how to use FPU function on AT32 MCU and configurations in Keil / IAR.

1.2.3 AT32F403A /AT32F407 zero-wait/non-zero-wait Flash and embedded SRAM configurations

Configure the user system data to allocate the internal Flash memory and SRAM.

Taking AT32F403AVGT7 as an example, the internal Flash memory and SRAM can be configured as one of the followings:

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

The core reads the command code stored in zero-wait Flash without any latency, and it is unnecessary to insert any wait state. For example, if the system clock is 240 MHz and AT32F403A zero-wait Flash is 256 KB, the first 256 KB of 512 KB bin file can be executed at 240 MHz, and the later 256 KB bin file is stored in the non-zero wait area and executed at 96 MHz, which is faster than 72 MHz (the maximum frequency of SXX32F10X). The execution rate of non-zero-wait area is 0.4 times that of the zero-wait area.

Embedded SRAM 96 KB (default)/224 KB, which can be selected in any of the following ways:

To configure the AT32F403A SRAM, the user needs to refer to the description of FLASH user system data and set EOPB0 (address: 0x1FFF_F810). EOPB0=0xFF indicates that the on-chip SRAM is 96 KB, and EOPB0=0xFE indicates that the on-chip SRAM is 224 KB. **To enable EOPB0, a power-off or RESET must occur once.**

The following section takes AT32F403AVGT7 as an example to introduce how to configure SRAM size. For more information about the principle and method of SRAM expansion, please refer to *AN0026_Extending_SRAM_in_User's_Program* (ARTERY's official website→ SUPPORT→ AP Note→AN0026).

① ICP/ISP Programmer

■ Artery ICP Programmer (BOOT0=0, BOOT1=0)

Connect AT-Link-EZ /AT-Link /J-Link to MCU—Open Artery ICP Programmer—User system data—Select EOPB0 **96 KB/224 KB** (complete other relevant settings if any) — Apply to device

Figure 25. User system data in ICP Programmer

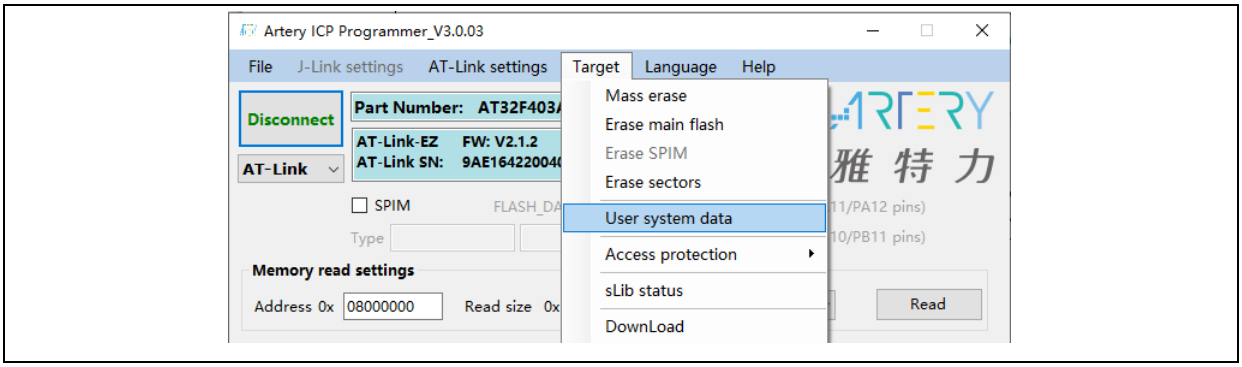
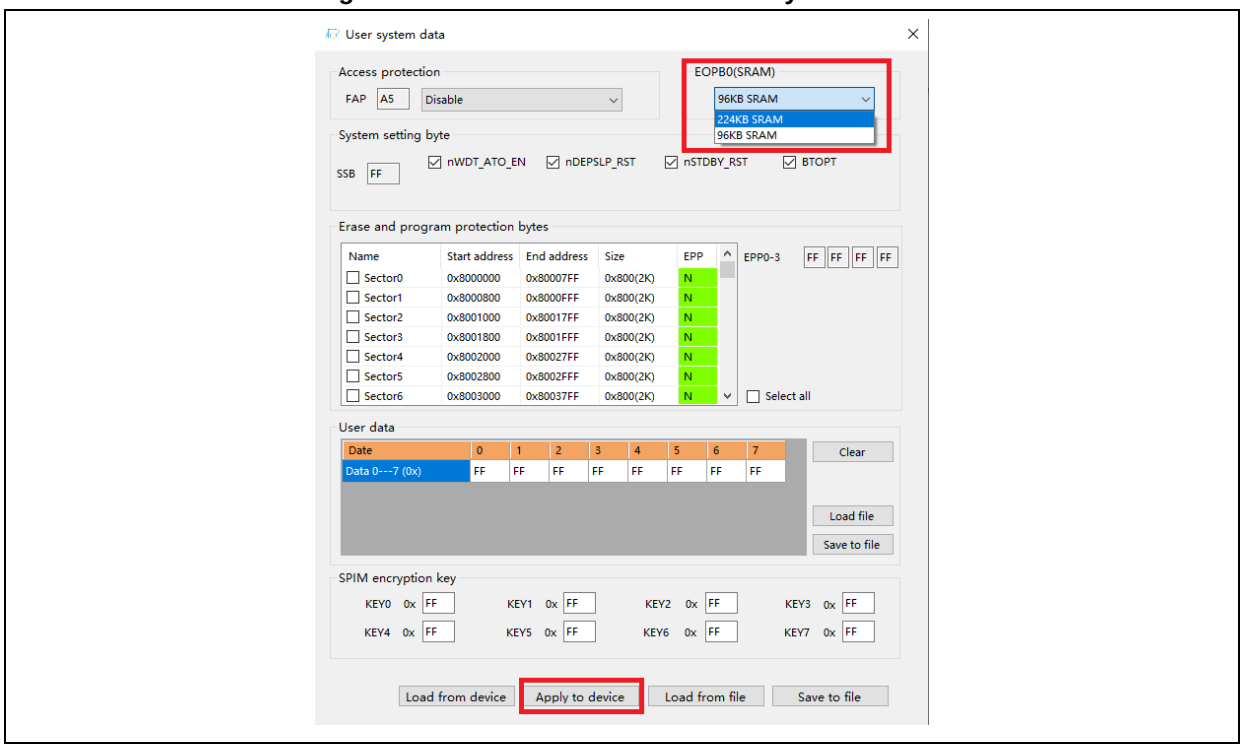


Figure 26. Select SRAM size in user system data



■ Artery ISP Programmer (BOOT0=1, BOOT1=0)

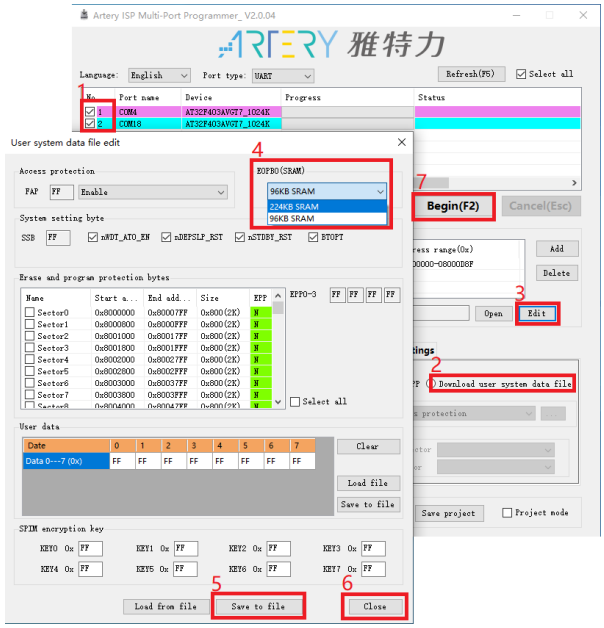
Connect UART or USB to MCU—Click on “Next” until enter the final interface—Select “Edit User system data”—Next— Select EOPB0 **96 KB/224 KB** (complete other relevant settings if any) — Apply to device.

Figure 27. Edit user system data in ISP Programmer

■ Artery ISP Multi-Port Programmer (BOOT0=1, BOOT1=0)

Select the port to be used—download user system data file after the port is recognized successfully—Edit—Select **16 KB/32 KB/64 KB**—Save to file (create a user system data program file)—Close—Start, or download user system data file—Open (the saved user system data program file) —Start.

Figure 28. User system data setting in ISP Multi-Port Programmer



② The user can also modify SRAM size in the Bootloader program (IAP). Note that the SRAM should be set to the same size as the modified SRAM (refer to AN0026).

For example, when the SXX32F103 is used:

Figure 29. Define Extend_SRAM(void) function in SXX program

```
#define  SRAM_96k   0xFF
#define  SRAM_224k   0xFE
static uint32_t F_EOPB0;
F_EOPB0=*(uint32_t*)(0x1FFFF810);
void Extend_SRAM(void)
{
    if((F_EOPB0 & 0xFF) != 0xFE) // check if RAM has been set to 224K, if not, change EOPB0
    {
       FLASH_Unlock();
       FLASH_EraseOptionBytes();
       FLASH_ProgramOptionByteData(0x1FFFF810, SRAM_224k);
       …                    // Set other user system data
       FLASH_Lock();
       NVIC_SystemReset();
     }
```

```
}
```

When the AT32F403A/ AT32F407 is used:

Figure 30. Define Extend_SRAM(void) function in AT32 program

```
#define   SRAM_96k   0xFF
#define   SRAM_224k   0xFE
static uint32_t f_eopb0;
f_eopb0=*(uint32_t*)(0x1FFFF810);
void Extend_SRAM(void)
{
    if((f_eopb0 & 0xFF) != 0xFE) // check if RAM has been set to 224K, if not, change EOPB0
    {
      flash_unlock();
      flash_user_system_data_erase();
      flash_user_system_data_program(0x1FFFF810, SRAM_224k);
      …                    // Set other user system data
      flash_lock();
      nvic_system_reset();
    }
}
```

It is necessary to erase the user system data before modification. If other options in the user system data are set, these settings should be read and erased, and then written together with the configured SRAM size.

③ Modify SRAM size in the startup file.

When running the startup file, the SRAM will be loaded. If there is no IAR in the program and the SRAM is greater than 96 KB, SRAM cannot be loaded successfully, and then a hardfault occurs, causing the program to fail to run. To avoid this, you can set SRAM=224 KB before loading it to the startup file.

Add the codes in bold (as shown in Figure 31) to the Keil startup file.

Figure 31. Modify SRAM size in Keil startup file

```
; 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 codes in bold (as shown in Figure 32) to the IAR startup file.

Figure 32. Modify SRAM size in IAR startup file

```
; 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
```

After completing the above configurations, add a declaration and define Extend_SRAM function to the program, as shown in ② in Section 1.2.3, and define Extend_SRAM(void) function to modify the SRAM size.

④ It is not recommended to use APP program to modify the SRAM size, for the reason that if SRAM used in APP is larger than the modified SRAM, the program will enter hardfault.

1.2.4 Encryption mode (access protection /erase and program protection /external Flash encryption)

1.2.4.1 Access protection

The access protection is commonly known as “encryption”, which is applied to the entire Flash storage area. Once the Flash access protection is enabled, the embedded Flash storage area can only be read through normal execution of the program, not through JTAG or SWD. When ICP/ISP tool is used to disable the access protection, the chip will perform erase operation on the Flash.

The ICP/ISP tool can be used to enable and disable IC access protection, as shown below:

- Artery ICP Programmer (BOOT0=0, BOOT1=0)

  Enable access protection: Open Artery ICP Programmer—Access protection—Enable (Y).

  Disable access protection: Open Artery ICP Programmer-- Access protection—Disable (Y).

- Artery ISP Programmer (BOOT0=1, BOOT1=0)

  Enable access protection: Protection/Enable/Access protection--Next—Yes, encrypted.

  Disable access protection: Protection/Disable/Access protection--Next—Yes, Flash decrypted.

Figure 33. Enable access protection in ISP Programmer

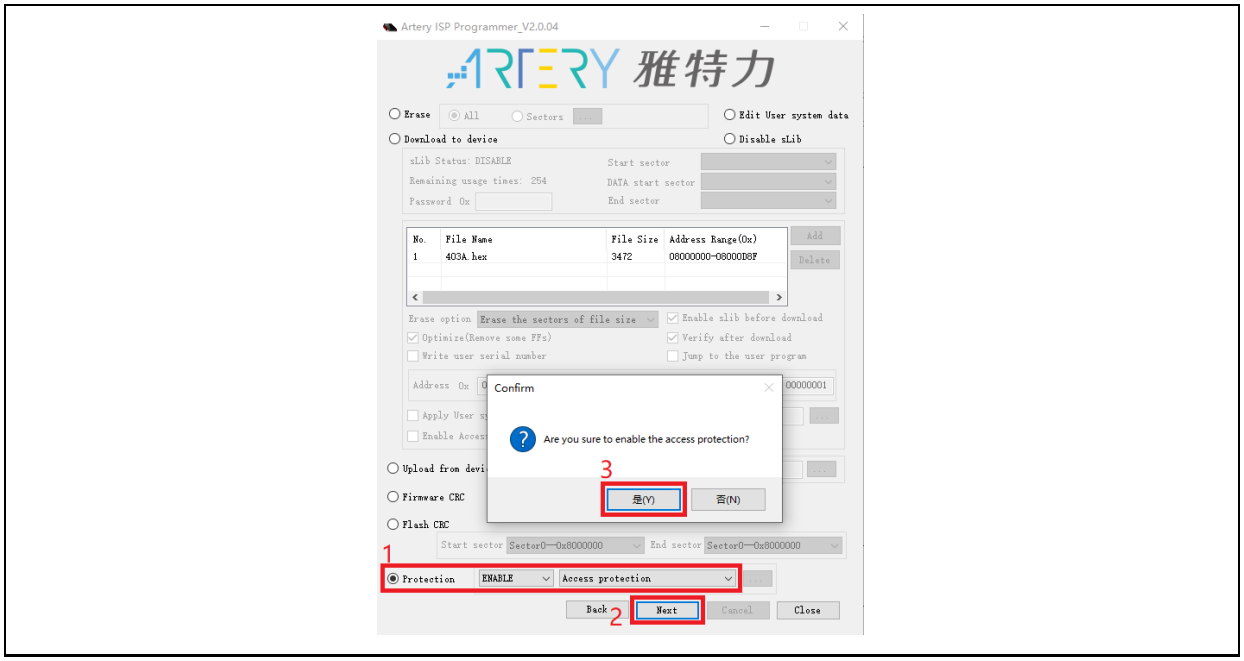
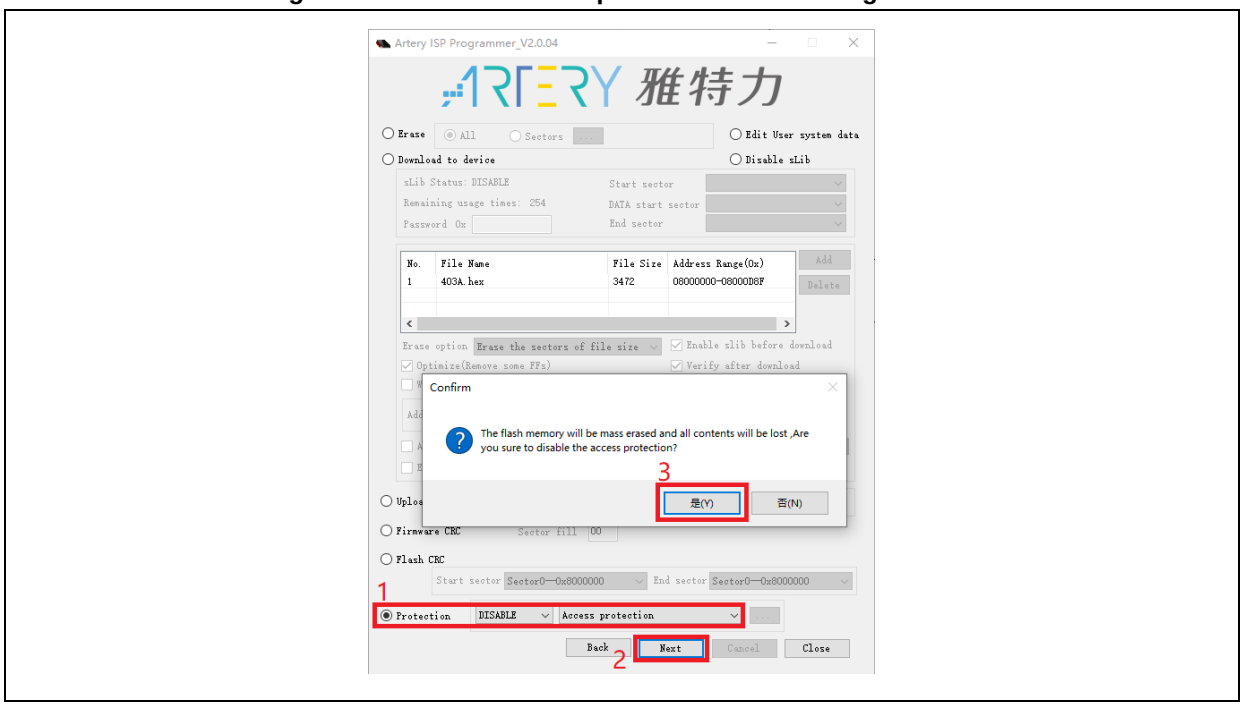


Figure 34. Disable access protection in ISP Programmer



■ Artery ISP Multi-Port Programmer (BOOT0=1, BOOT1=0)

Enable access protection: Protection/Enable/Access protection—Start—Yes, encrypted.

Disable access protection: Protection/Disable/Access protection—Start—Yes, Flash decrypted.

Note: Once enabled, the access protection cannot be disabled through erase operation.

1.2.4.2 Erase and program protection

The program protection is applied to the entire Flash storage area or certain pages in the Flash storage area. Once the Flash program protection is enabled, the internal Flash storage area cannot be programmed in any way.

The user can use ICP/ISP programmer to enable/disable erase and program protection, as shown below:

■ Artery ICP Programmer (BOOT0=0, BOOT1=0)

Enable erase and program protection: Open Artery ICP Programmer—User system data—Tick the sectors that require erase and program protection—Apply to device.

Disable erase and program protection: Open Artery ICP Programmer—User system data—Untick the sectors that do not require erase and program protection—Apply to device.

Figure 35. Enable erase and program protection in ICP Programmer

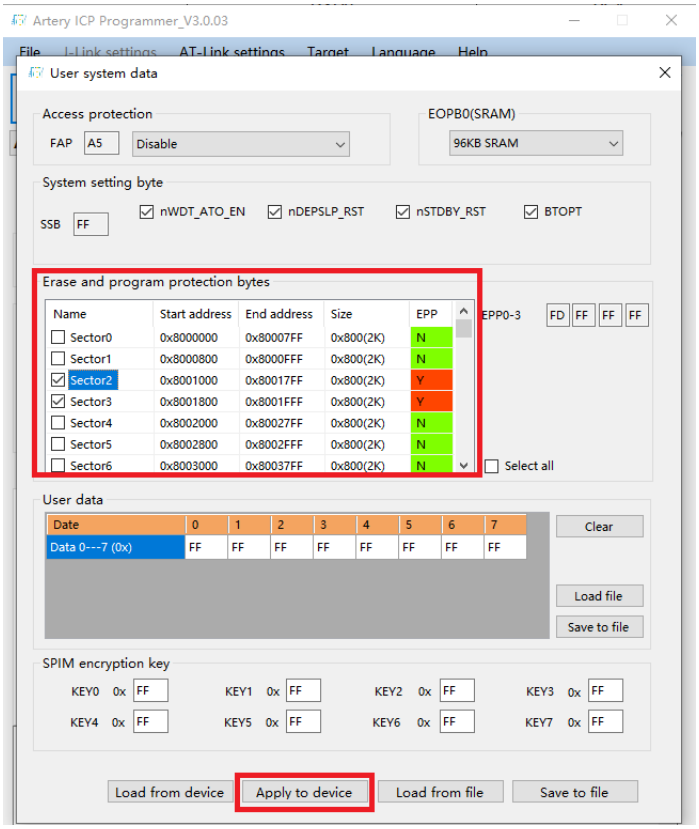
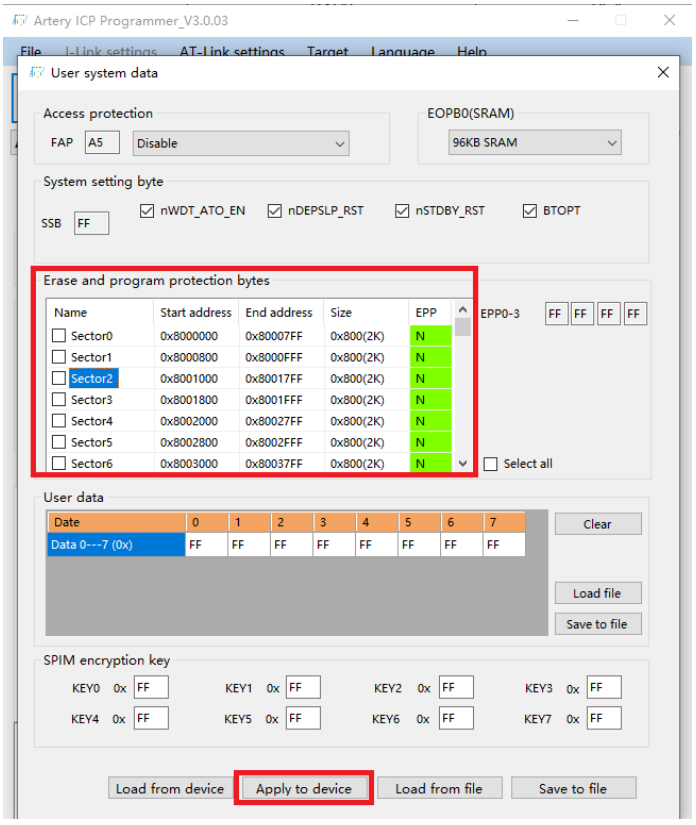


Figure 36. Disable erase and program protection in ICP Programmer



- Artery ISP Programmer (BOOT0=1, BOOT1=0)

  Enable erase and program protection: Protection/Enable/Erase and program protection—Next—Yes, erase and program protection enabled.

  Disable erase and program protection: Protection/Disable/Erase and program protection—Next—Yes, erase and program protection disabled.

- Artery ISP Multi-Port Programmer (BOOT0=1, BOOT1=0)

  Enable erase and program protection: Protection/Enable/Erase and program protection—Start—Yes, erase and program protection enabled.

  Disable erase and program protection: Protection/Disable/Erase and program protection—Start—Yes, erase and program protection disabled.

Note: Once enabled, the erase and program protection cannot be disabled through erase operation.

1.2.4.3 External Flash encryption (encryption of downloading and reading the data within the encrypted range of external memory)

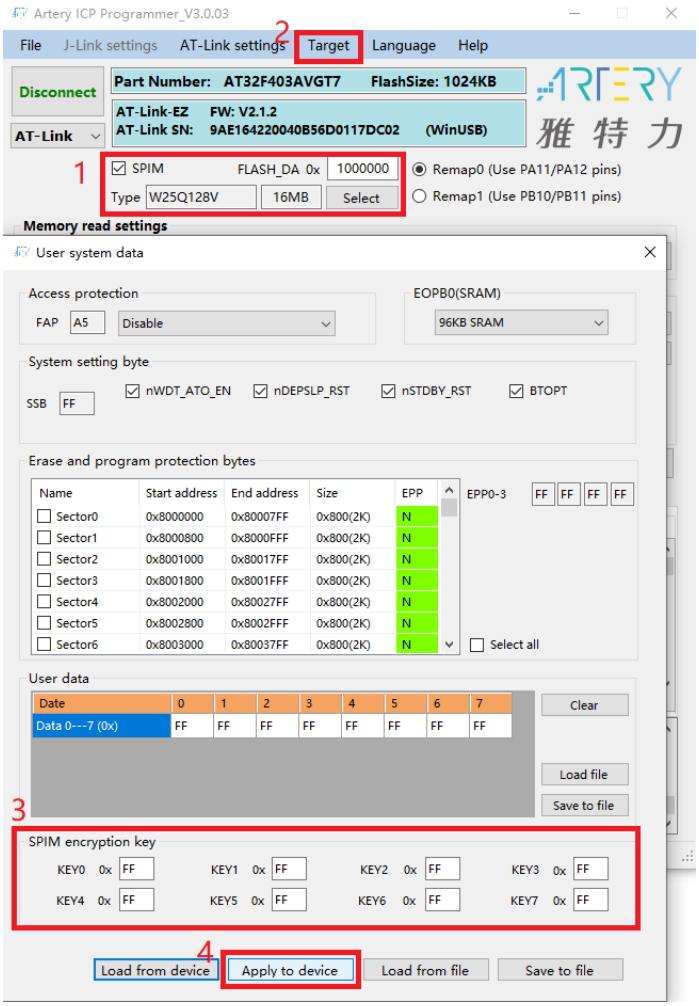
For external Flash encryption, the user needs to set the encryption range and password as shown below, program the user program, and then enable access protection. The encryption range indicates the size of the space to be encrypted starting from 0x08400000. If the encryption keys are all 0xFF or 0x00, do not encrypt the external Flash; otherwise, encrypt the external Flash. Disabling access protection will set the external Flash encryption keys all to 0xFF.

Use ICP/ISP programmer to encrypt the external memory as follows:

- Artery ICP Programmer (BOOT0=0, BOOT1=0)

  Tick the external memory—Select the external memory type—Set the encryption range—Device operation—User system data—Modify the encryption key—Apply to device.

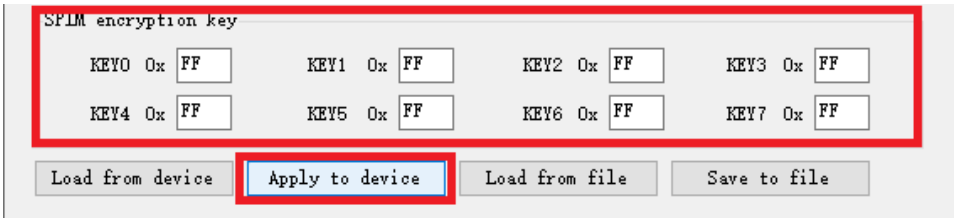
Figure 37. Encrypt external memory in ICP Programmer



■ Artery ISP Programmer (BOOT0=1, BOOT1=0)

Edit user system data—Next—Modify the encryption key—Apply to device.

Figure 38. Encrypt external memory in ISP Programmer



■ Artery ISP Multi-Port Programmer (BOOT0=1, BOOT1=0)

Download user system data file—Edit—Modify the encryption key—Save to file—Start.

1.2.5 Recognize AT32 MCU in program

■ Read Cortex-M CPU ID to identify M0, M3 and M4 core

Figure 39. Read Cortex ID

```
cortex_id = *(uint32_t*)0xE000ED00;// Read Cortex ID
if((cortex_id == 0x410FC240) || (cortex_id == 0x410FC241))
{
        printf("This chip is Cortex-M4F.\r\n");
}
else
{
        printf("This chip is Other Device.\r\n");
}
```

■ Read PID and UID

Figure 40. Read PID and UID

```
/* Get the base address of AT32 MCU PID/UID */
#define DEVICE_ID_ADDR1 0x1FFFF7F3          // Define Artery MCU part number, UID base address
#define DEVICE_ID_ADDR2 0xE0042000          // Define MCU device number, PID base address

/* Used to store ID */
uint8_t     ID[5] = {0};

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

  /* Get 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: AT32F4xx MCU contains multiple ID codes. Combine the obtained ID information to a 64-bit data to help identify the specific MCU model. For more information, please refer to the Reference Manual (DEBUG section) of each series and *AN0016_Recognize_AT32_MCU* (ARTERY's official

website→SUPPORT→AP Note→AN0016).

2 FAQs in downloading/compiling

2.1 Hard Fault Handler

- The SRAM being used exceeds the specified SRAM in user system date.

  Please refer to Section 1.2.3 and use ICP/ISP programmer or 3rd party programmer to configure a larger SRAM space before programming.

- The single precision function is enabled in Keil or IAR, while the M4 core FPU register is not enabled in the code. Please enable FPU function in the code:

Figure 41. Add FPU enabling codes

```
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
```

- The data being accessed is out of range.

  Find out the access violation point and modify the data to the normal range.

- The system clock configuration is out of specification.

2.2 J-Link cannot find IC

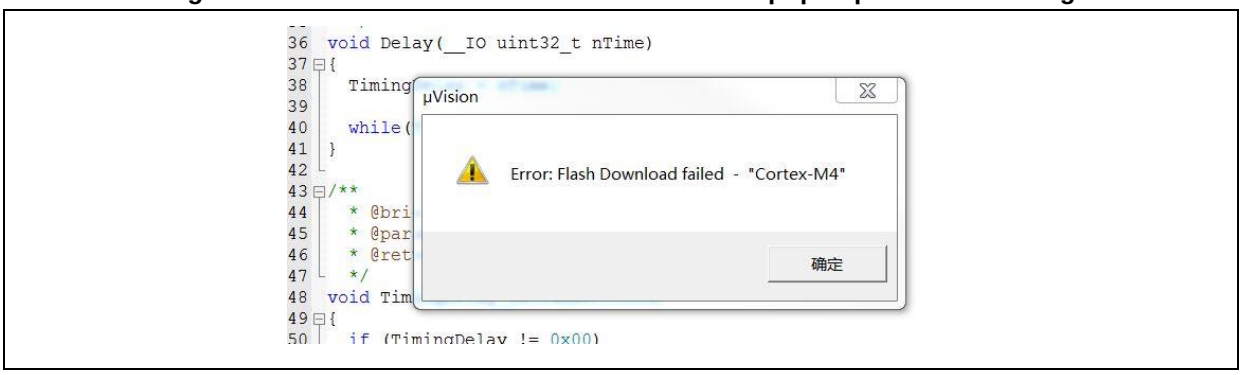
- Please refer to *FAQ0008_ J-Link cannot find IC* (ARTERY's official website→ SUPPORT→FAQ→FAQ0008).
- Please refer to *FAQ0132_Add Artery MCU to J-Link* (ARTERY's official website→ SUPPORT→FAQ→FAQ0132).

2.3 Problems in program downloading

2.3.1 Error: Flash Download failed–"Cortex-M4"

The following pop-up window appears in KEIL emulation or downloading:

Figure 42. Flash Download failed–“Cortex- M4” pops up in downloading



This pop-up window may appear in one of the following conditions:

- The access protection is enabled. Please disable MCU access protection and then download.
- The selected Flash file algorithm is incorrect or no Flash file algorithm is loaded. Please add the proper Flash file algorithm in “Flash Download”.
- BOOT0 and BOOT1 are incorrect. Set BOOT0=0 and BOOT1=0 to allow MCU to boot from the main Flash memory.
- The J-Link version is outdated. Please use 6.20C and above.
- JTAG/SWD PIN is disabled in the program. Please refer to “2.3.5 Resume download”.

2.3.2 No Debug Unit Device found

- The download port is occupied. For example, ICP is connecting to the target device.
- JTAG/SWD connection error or not connected.

2.3.3 RDDI-DAP Error

- Compiler optimization level is too high. For example, the default optimization level of Keil AC6 is –Oz, which should be changed to -O0/-O1.
- JTAG/SWD PIN is disabled in the program. Please refer to “2.3.5 Resume download”.

2.3.4 ISP serial port gets stuck in downloading

When the ISP serial port is used for downloading, it may be stuck occasionally, and the PC cannot release serial port in this case.

The following operations are recommended:

- Check whether the power supply is stable;
- Replace with better USB-to-serial port tools, i.e., CH340 chip.

2.3.5 Resume download

When using AT32F403A/AT32F407, the user may not be able to download the program again in the following conditions:

- After disabling JTAG/SWD PIN in the program, the program cannot be downloaded and

JTAG/SWD device is not found.

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

Solutions in KEIL and IAR are as follows:

- Solution 1: Use the ConfigureJLink.exe provided by ARTERY.
- Solution 2: Switch boot mode: switch to Boot[1:0]=01b or Boot[1:0]=11b, and then press “Reset” button to resume download (note: switch back to Boot[1:0]=00b). Similarly, ISP download can be resumed.
- Solution 3: ICP tool + AT-Link
  AT-Link is specially designed for AT32 MCUs. The ICP can be used together with AT-Link to resume download. Solution 2 and Solution 3 require Boot PIN or device (AT-Link) additionally; therefore, this application note mainly introduces Solution 1.

2.3.5.1 Solutions in KEIL

Use the ConfigureJLink.exe provided by ARTERY.

The following procedures are recommended:

- Place the ConfigJLink_V1.0.0.exe into the directory where the project file (*.uvprojx) is located;
- Double click the ConfigJLink_V1.0.0.exe , and the following window (as shown in Figure 43) will pop up;
- Tick “Agree” and then click on “OK”; then the progress bar will pop up (as shown in Figure 44), and wait for the erase progress is completed, and then the program can be downloaded.

Figure 43. Operate ConfigJLink_V1.0.0 in KEIL

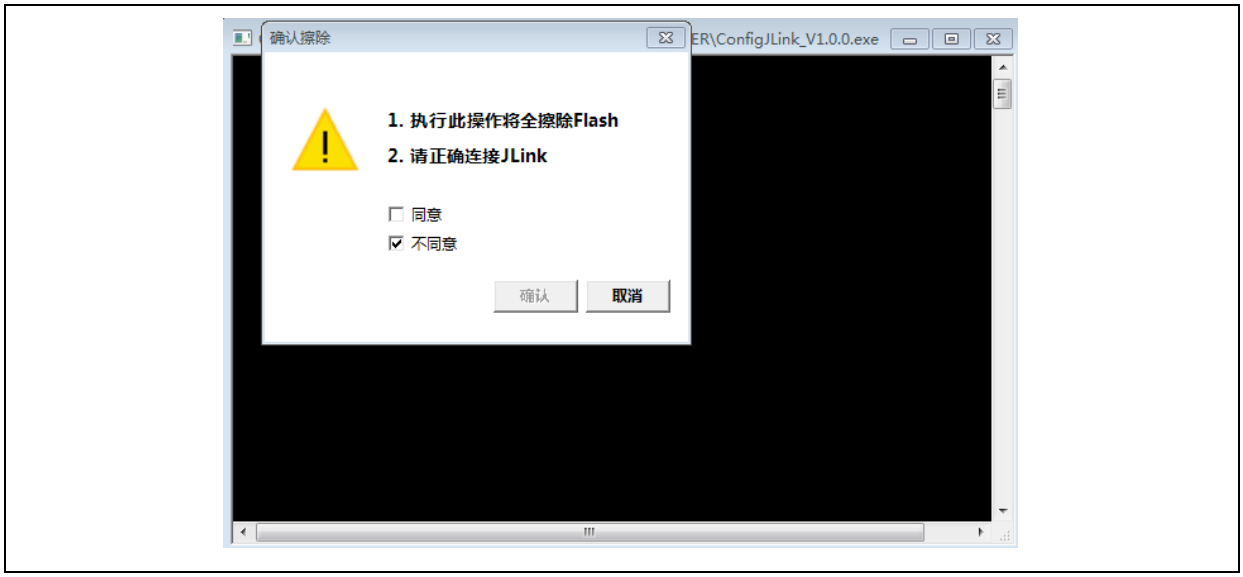
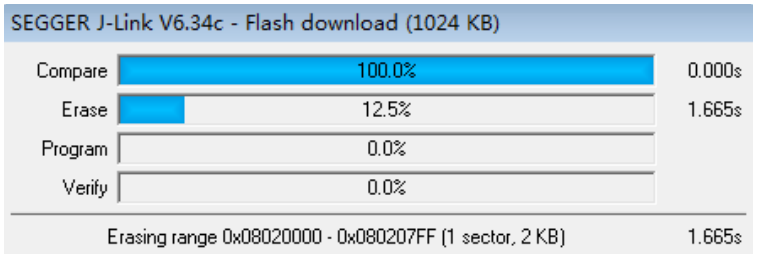


Figure 44. ConfigJLink_V1.0.0 execution progress in KEIL



Note 1: Make sure that SEGGER J-Link interface DLL is not lower than V6.14.

Note 2: If JTAG/SWD PIN is disabled every time the program is downloaded, perform the preceding steps before downloading the program.

Note 3: If the MCU enters Standby mode every time the program is downloaded, perform the preceding steps before the chip is powered on.

Note 4: In Keil, AT32F403A /AT32F407 enters Standby mode, the solution using ConfigureJLink.exe is invalid.

2.3.5.2 Solutions in IAR

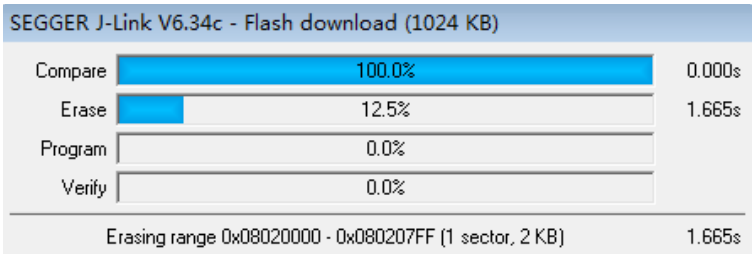
Use the ConfigJLink_V1.0.0.exe provided by ARTERY.

The following procedures are recommended:

- Place the ConfigJLink_V1.0.0.exe to the *settings* folder under the project directory.
- Double click the ConfigJLink_V1.0.0.exe, and the following window (as shown in Figure 45) will pop up.
- Tick “Agree” and then click on “OK”; then the progress bar will pop up (as shown in Figure 46), and wait for the erase progress is completed, and then the program can be downloaded.

Figure 45. Operate ConfigJLink_V1.0.0 in IAR

Figure 46. ConfigJLink_V1.0.0 execution progress in IAR



Note 1: Make sure that SEGGER J-Link interface DLL is not lower than V6.14.

Note 2: If JTAG/SWD PIN is disabled every time the program is downloaded, perform the preceding steps before downloading the program.

Note 3: If the MCU enters Standby mode every time the program is downloaded, perform the preceding steps before the chip is powered on.

3 Security Library (sLib)

3.1 Introduction

As more and more MCU applications require complex algorithms and middleware solutions, it has become an important issue that how to protect IP-Codes (such as core algorithms) developed by software solution providers.

The AT32F403A/407 series is designed with a security library (sLib) to protect important IP-Codes against being changed or read by the end user's program.

3.2 Principles of application

- Security library (sLib) is a defined area protected by a code in the main memory. Software solution providers store core algorithms in sLib for protection. The rest of the area can be used for secondary development by terminal customers.
- Security library includes the instruction security library (SLIB_INSTRUCTION) and data security library (SLIB_DATA). Users can select part of or the whole security library for instruction storage, but using the whole security library for storing data is not supported.
- Program codes in the instruction security library (SLIB_INSTRUCTION) can only be fetched by MCU through I-Code bus (can only be executed) and cannot be read through D-Code (including ISP/ICP debug mode and programs that boot from internal RAM). When accessing the SLIB_INSTRUCTION in the manner of reading data, values are all read 0xFF.
- Data in the data security library (SLIB_DATA) can only be read through D-Code bus and cannot be programmed.
- The program code and data in security library cannot be erased unless the correct code is keyed in. If a wrong code is keyed in, in an attempt of writing or erasing the security library, a warning message will be issued by EPPERR=1 in the FLASH_STS register.
- The program code and data in security library are not erased when the end users perform a mass erase on the main Flash memory.
- Users can write the previously defined password in the SLIB_PWD_CLR register to disable

security library protection. When the security library protection is disabled, the chip will perform a mass erase on the main Flash memory (including the contents of security library). Therefore, even if the code defined by the software solution provider is leaked, the program code will not be leaked.

3.3 Security library application

For details, please refer to *AN0040_AT32F403A_407_Security_Library_Application_Note* (ARTERY's official website→SUPPORT→AP Note→AN0040).

4 Revision history

Table 1. Document revision history

Date	Version	Revision note
2021.12.27	2.0.0	Initial release
2022.08.03	2.0.1	Updated 3rd party programming tools.
2022.10.08	2.0.2	Added description of development environment and file path.
2022.10.21	2.0.3	Optimized description of UID and PID.

IMPORTANT NOTICE – PLEASE READ CAREFULLY

Purchasers are solely responsible for the selection and use of ARTERY's products and services; ARTERY assumes no liability for r 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) Auto-motive application or environment; (D) Aerospace applications or environment, and/or (E) weapons. 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.