AN0023
Application Note

Getting Started with AT32F403A/AT32F407

Introduction

The purpose of this application note is to provide a user guide on how to expedite AT32F403A/AT32F407-based application development. Compared to AT32F403A series, the AT32F407 series offers an addiontal feature – EMAC.

Note: The codes in this application note are written based on the V2.x.x version of BSP developed by ARTERY. For other versions of BSP, please pay attention to the differences between versions when in use.

Applicable products:

Part number	AT32F403A series AT32F407 series

Contents

1 Environment requirements 6

1.1 Set up AT32 MCU development environment 6

1.1.1 Debugging tool and evaluation board 6

1.1.2 Programming tools and software 7

1.1.3 AT32 MCU-based development environment 7

1.1.4 AT32 Work Bench 12

1.1.5 How to quickly replace SXX with AT32F403A_407 18

1.2 AT32F403A/F407 MCU function enhancement 19

1.2.1 PLL clock settings 19

1.2.2 How to enable FPU function (Floating point unit) 21

1.2.3 ZW/NZW Flash and embedded SRAM configuration 22

1.2.4 Encryption (access /erase and program protection /external Flash encryption) 27

1.2.5 Distinguish AT32 MCU from other MCUS 31

2 Frequently asked questions 32

2.1 Enter Hard Fault Handler 32

2.2 J-Link fails to identify IC in Keil 32

2.3 Problems occurred in program download 33

2.3.1 Error warning: Flash Download failed–“Cortex-M4” 33

2.3.2 No Debug Unit Device found 33

2.3.3 RDDI-DAP Error 33

2.3.4 ISP serial port gets stuck during download 33

2.3.5 How to resume download 34

3 Security Library (sLib) 35

3.1 Introduction 35

3.2 Principles of application 35

3.3 How to use Security library 36

4 Revision history 37

List of tables

Table 1. Document revision history..... 37

List of figures

Figure 1. AT-START-F407 evaluation board with AT-Link-EZ 6
Figure 2. AT-START-F407 evaluation board package 6
Figure 3. ICP/ISP/AT-Link-Family 7
Figure 4. BSP package at Artery's official website 7
Figure 5. Keil_v5 templates 8
Figure 6. Pack download 8
Figure 7. Set up ArteryTek.AT32F403A_F407 _DFP 9
Figure 8. Set up Keil4_AT32MCU_AddOn 9
Figure 9. Pack Installer icon in Keil 9
Figure 10. Set up IAR_AT32MCU_AddOn 10
Figure 11. Keil Debug option 10
Figure 12. Keil Debug Settings 11
Figure 13. Keil Utilities 11
Figure 14. IAR Debug option 11
Figure 15. IAR CMSIS-DAP option 12
Figure 16. AT32 Work Bench files at ARTERY official website 12
Figure 17. Enter command line 13
Figure 18. AT32 Work Bench graphic installation 13
Figure 19. AT32 Work Bench graphic installation 14
Figure 20. Project configuration interface 14
Figure 21. Mode and configuration window 15
Figure 22. Peripheral mode selection 15
Figure 23. Parameters settings 15
Figure 24. GPIO settings 16
Figure 25. DMA settings 16
Figure 26. NVIC settings 16
Figure 27. Right click and left click 17
Figure 28. Clock configuration window 17
Figure 29. Code view 17
Figure 30. Project manager 18
Figure 31. Project file structure 18
Figure 32. Use SXX32F103 to output 240 MHz clock on AT32F403A/F407 19
Figure 33. AT32F403A/F407 crm_pll_output_range parameter 19

Figure 34. New Clock Configuration tool on Artery official website 19
Figure 35. SXX PLL auto step-by-step switch configuration 20
Figure 36. AT32 PLL auto step-by-step switch configuration 20
Figure 37. Enable FPU in Keil 21
Figure 38. Add FPU enabling codes in Keil 21
Figure 39. Enable FPU in IAR 21
Figure 40. Add FPU enabling codes in IAR 22
Figure 41. User system data in ICP Programmer 23
Figure 42. Select SRAM size in user system data 23
Figure 43. Edit user system data in ISP Programmer 24
Figure 44. User system data setting in ISP Multi-Port Programmer 24
Figure 45. Extend_SRAM(void) in SXX program 25
Figure 46. Extend_SRAM(void) in AT32 program 25
Figure 47. Change SRAM in Keil 26
Figure 48. Change SRAM in IAR 26
Figure 49. Enable access protection with ISP 27
Figure 50. Disable access protection with ISP 28
Figure 51. Enable erase and program protection with ICP 29
Figure 52. Disable erase and program protection with ICP 29
Figure 53. External memory encryption with ICP 30
Figure 54. External memory encryption with ISP 31
Figure 55. Read Cortex ID 31
Figure 56. Read PID and UID 31
Figure 57. Add FPU enable code 32
Figure 58. Flash Download failed–“Cortex- M4” 33
Figure 59. Debug interface in Keil 34
Figure 60. CMSIS DAP in IAR 35

1 Environment requirements

Download link:

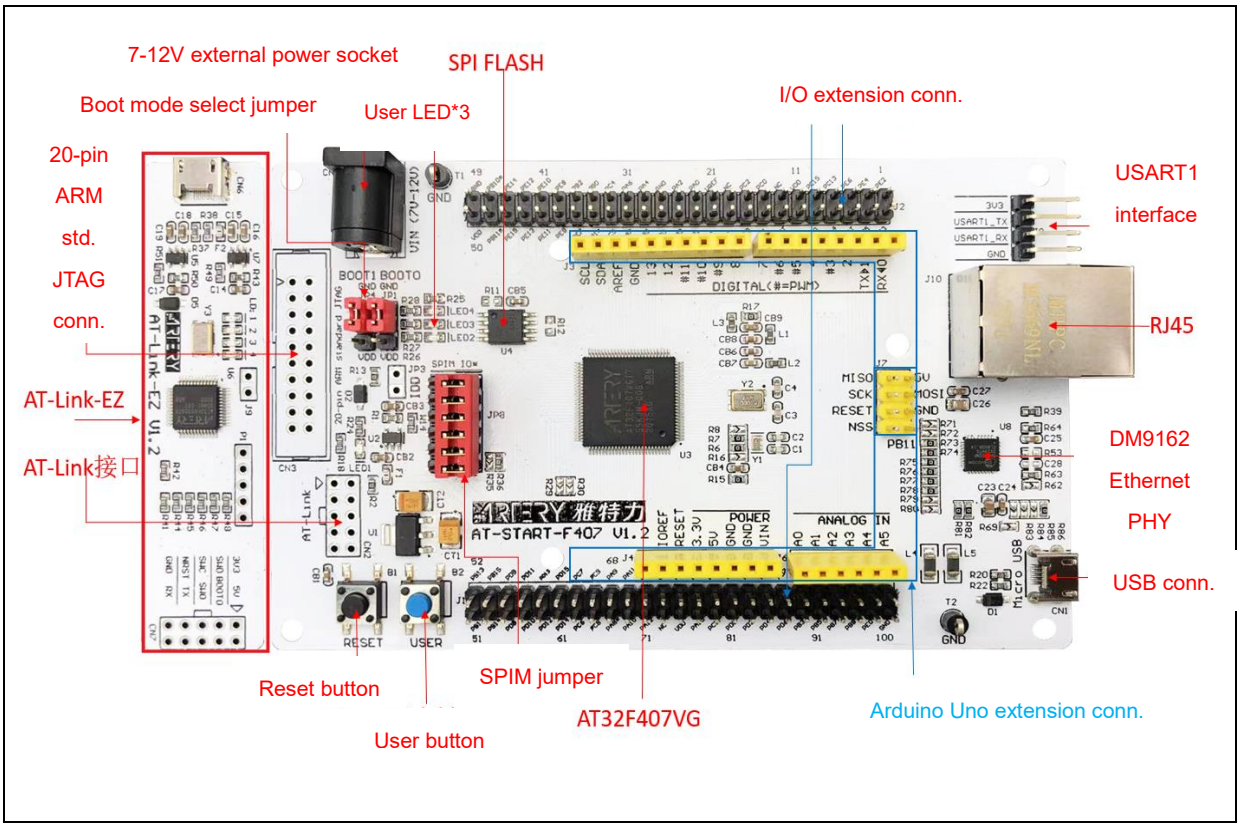
- ARTERY's official website: https://www.arterychip.com/

1.1 Set up AT32 MCU development environment

1.1.1 Debugging tool and evaluation board

The AT32F403A/F407 evaluation board is equipped with the AT-Link-EZ tool for debugging purpose, as shown in the left side of Figure 1 (marked in red box) below. This AT-Link-EZ can be disassembled from the board and used with other circuit boards. It supports IDE online debugging, online programming and USB-to-serial port.

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



Note: For details on the AT-START evaluation board, please refer to UM_AT_START_F403A/407_Vx.x, which can be found at ARTERY's official website → PRODUCTS → Mainstream → AT32F403A/F407 → Evaluation Board → AT_START_F403A/F407.

Figure 2. AT-START-F407 evaluation board package

Evaluation Board

Download	Description	Version
AT-START-F407	AT32F407 evaluation board supporting Arduino standard interfaces	V1.3

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.
- For ICP operation instruction, please refer to *UM_ICP_Programmer* (located at ARTERY's official website → PRODUCTS → Mainstream → AT32F4xx → Tool → "Artery_ICP_Programmer_Vx.x.xx\Document\UM_ICP_Programmer").
- For ISP operation instruction, please refer to *UM_ISP_Programmer* (located at ARTERY's official website → PRODUCTS → Mainstream → AT32F4xx → Tool "Artery_ISP_Programmer_Vx.x.xx\Document\UM_ISP_Programmer").
- For AT-Link operation instruction, please refer to *UM0004_AT-Link_User_Manual* (located at ARTERY's official website → PRODUCTS → Mainstream → AT32F4xx → Tool → and "AT_Link_CH_ Vx.x.x\05_Documents\UM0004_AT-Link_User_Manual_EN_Vx.x.x").

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

Tool

Download	Description	Version	Date
01_AT32 IDE AT32 IDE_Linux AT32 IDE_Win	AT32 IDE User Manual A software development environment for cross-platform ARM embedded system based on Eclipse development supporting AT32 MCU	V1.0.10	2024/04/28
02_AT32 Work Bench AT32 Work Bench_Linux AT32 Work Bench_Win	AT32 Work Bench The AT32 Work Bench can generate initialization C code and corresponding IDE project through MCU graphical configuration.	V1.0.08	2024/04/29
03_AT32 IDE_Project Generate_Linux AT32 IDE_Project Generate_Win	AT32 IDE Project Generate	V1.0.01	2024/01/05
04_AT-Link Family AT-Link Family	AT-Link Family Emulation and online/offline programming tools supporting AT32 MCU	V2.1.2	2024/01/02
05_AT-Link Console AT-Link Console_Linux AT-Link Console_Win	AT-Link Console In-Circuit-Programming Console tool supporting AT32 MCU	V3.0.9	2024/04/26
06_ICP ICP	ICP Programmer In-Circuit-Programming tool supporting AT32 MCU	V3.0.14	2024/03/12
07_ISP ISP	ISP Programmer In-System-Programming tool supporting AT32 MCU	V2.0.13	2024/04/26

1.1.3 AT32 MCU-based development environment

1.1.3.1 Template project

IDE template projects can be found in BSP which is available at ARTERY's official website → PRODUCTS → Mainstream → AT32F4xx → BSP.

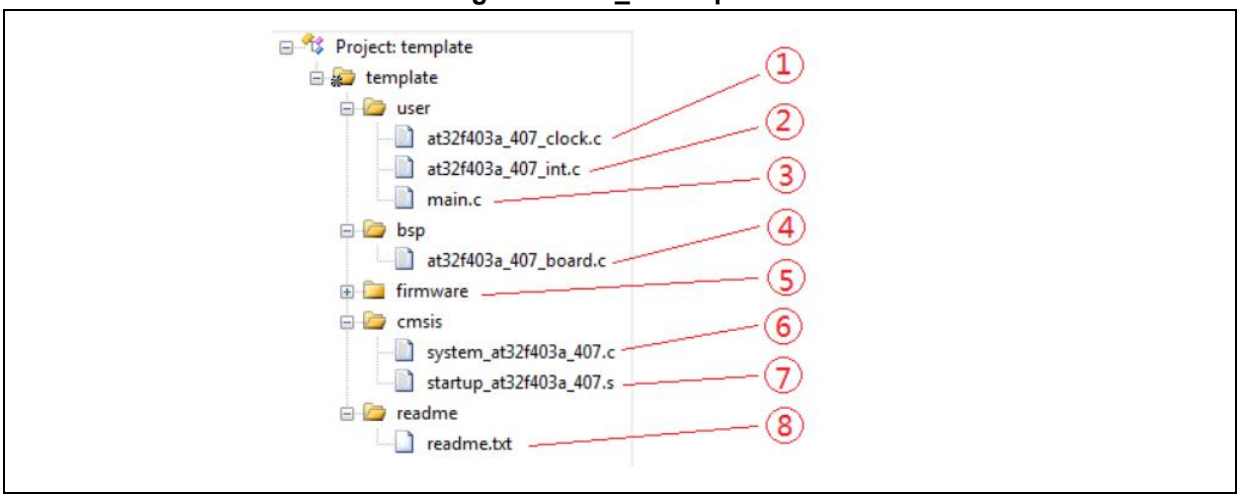
Figure 4. BSP package at Artery's official website

BSP

Download	Description	Version
Firmware Library	AT32F413 firmware library BSP user guide	V2.0.9

BPS offers such template projects as Keil_v5, Keil_v4, IAR_6.10, IAR_7.4, IAR_8.2, eclipse_gcc, at32_ide. They are located at AT32F403A/F407_Firmware_Library_V2.x.xproject\at_start_f4xx\templates. Here is an example of Keil_v5 template project.

Figure 5. Keil_v5 templates



A template project contains the following content:

① at32f403a_407_clock.c: clock configuration file, with default clock frequency and clock path being programmed

② at32f403a_407_int.c: interrupt file that contains interrupt function-related code

③ main.c: refers to the main code file

④ at32f403a_407_board.c: board-level configuration file that contains general hardware configuration such as AT-START on-board buttons and LEDs

⑤ at32f403a_407_xx.c under firmware: peripheral driver files

⑥ system_at32f403a_407.c: system initialization files

⑦ startup_at32f403a_407.s: startup file

⑧ readme.txt: statement file that is used to introduce general operations and settings and application notes related to the template project.

In addition to templates, BSP contains a variety of code examples (Keil_v5 project files) for uses' reference. They are located at AT32F403A_F407_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_F407 Firmware BSP&Pack User Guide by visiting ARTERY's official website → PRODUCTS → Mainstream → AT32F4xx → BSP → "\AT32F403A_F407_Firmware_Library_Vx.x.x\document".

1.1.3.2 Installing PACK

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

Figure 6. Pack download

Pack

Download	Description	Version	Date
Keil 4 Keil 5	Supports AT32 MCU to run in Keil MDK	V2.2.0 V2.2.3	2023.07.07
IAR	Supports AT32 MCU to run in IAR EWARM	V2.1.6	2023.07.07
Segger	Supports Segger tools to identify AT32 MCU	V2.0.7	2023.03.17
ConfigJLink	How to resume download for AT32 series	V1.0.1	2019.02.26

For Keil compiling system, keil 4.74 /5.23 or above is recommended. For Keil_v5, users need to unzip the Keil5_AT32MCU_AddOn and then install the corresponding ArteryTek.AT32F403A_F407_DFP.

For Keil_v4, users can directly install Keil4_AT32MCU_AddOn. By default, the Keil installation path can be recognized automatically. In case of path recognition failure, users need to manually select the path.

Figure 7. Set up ArteryTek.AT32F403A_F407 _DFP

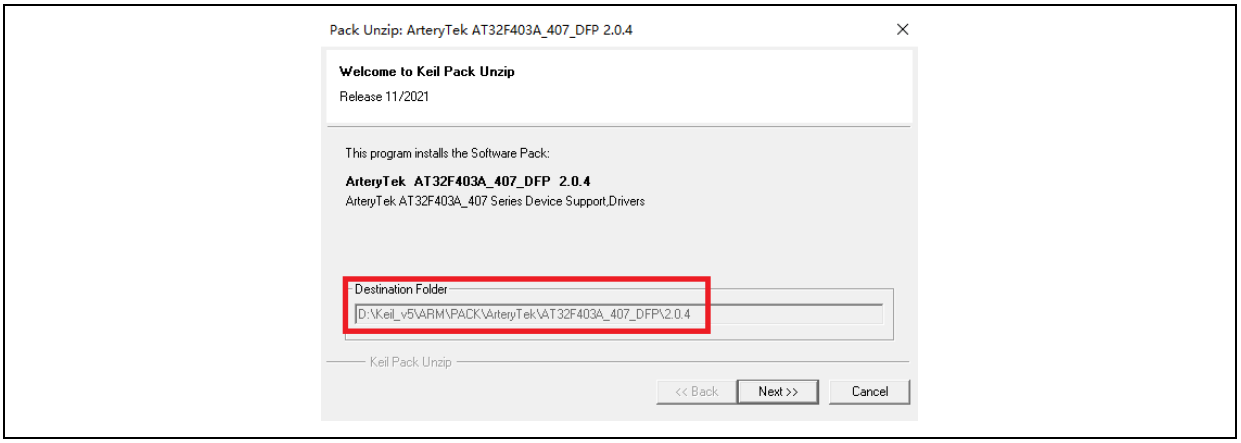
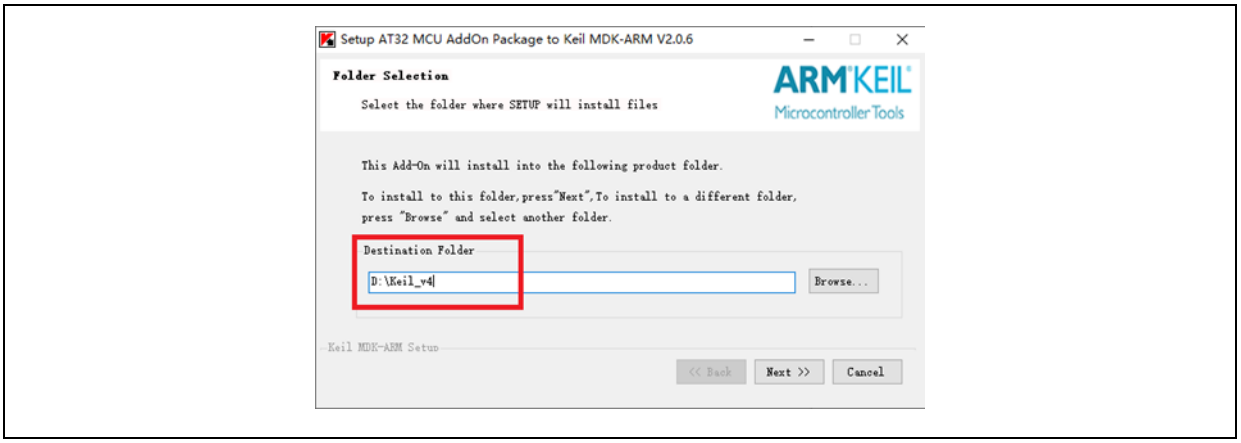
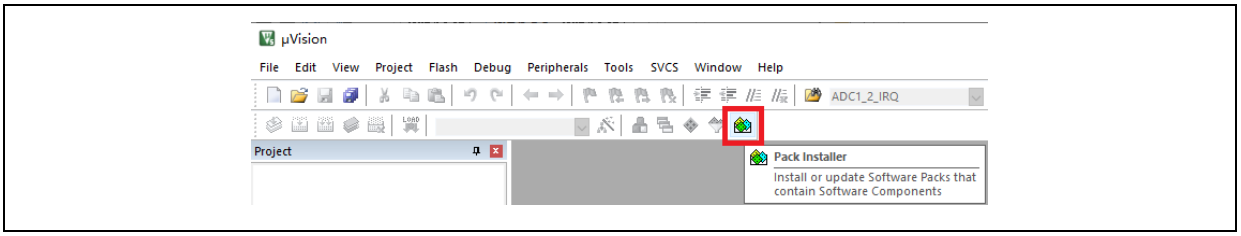


Figure 8. Set up Keil4_AT32MCU_AddOn



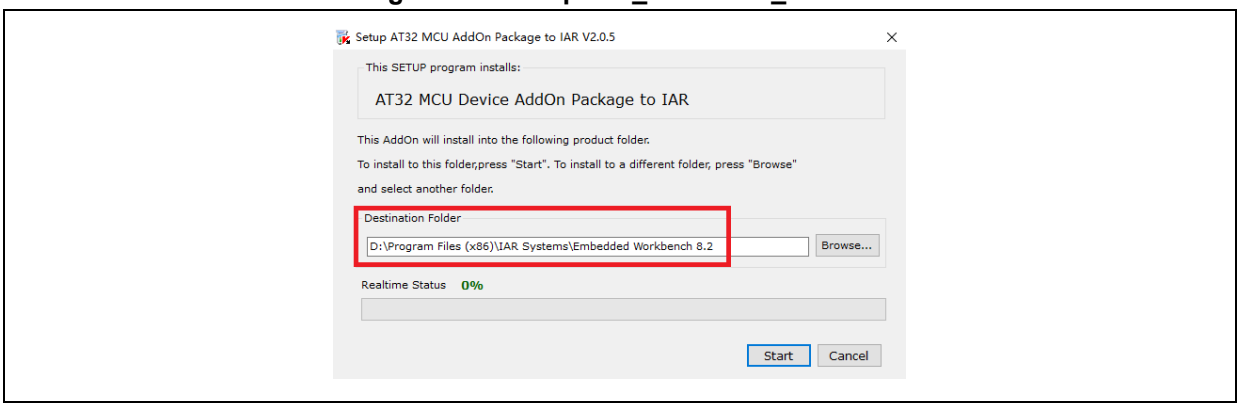
Another way to install PACK is as follows: Open keil and click on “Pack Installer” icon, click on “file” and select “import” to import the corresponding pack downloaded from ARTERY's official website.

Figure 9. Pack Installer icon in Keil



For IAR compiling system, IAR7.0 / IAR6.1 or above is recommended. The IAR_AT32MCU_AddOn needs to be installed. By default, the IAR installation path can be recognized automatically during installation. In case of path recognition failure, users need to manually select the path.

Figure 10. Set up IAR_AT32MCU_AddOn

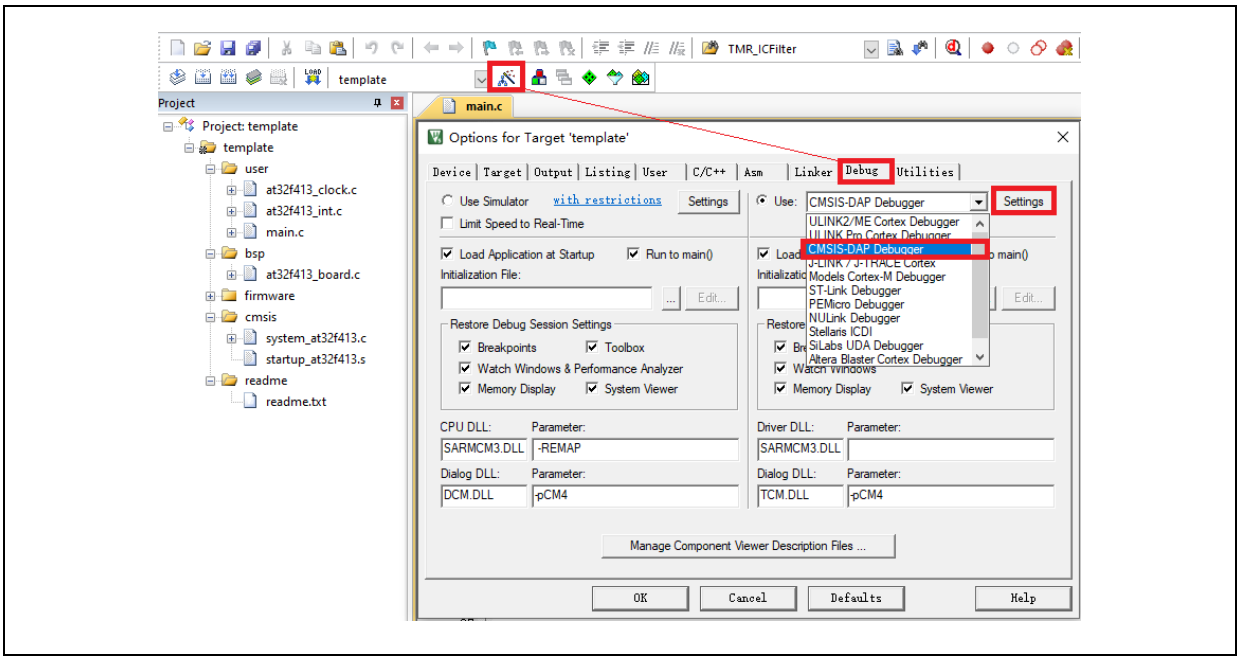


Note: For more details about PACK settings, please refer to "Section 2 Pack setup" of the AT32F403A_F407 Firmware BSP&Pack User Guide by visiting ARTERY's official website → PRODUCTS → Mainstream → AT32F4xx → BSP → "\AT32F403A_F407_Firmware_Library_Vx.x.x\document".

1.1.3.3 Debug and download with AT-Link

To use AT-Link in IAR, select CMSIS-DAP in Debugger option.

Figure 11. Keil Debug option



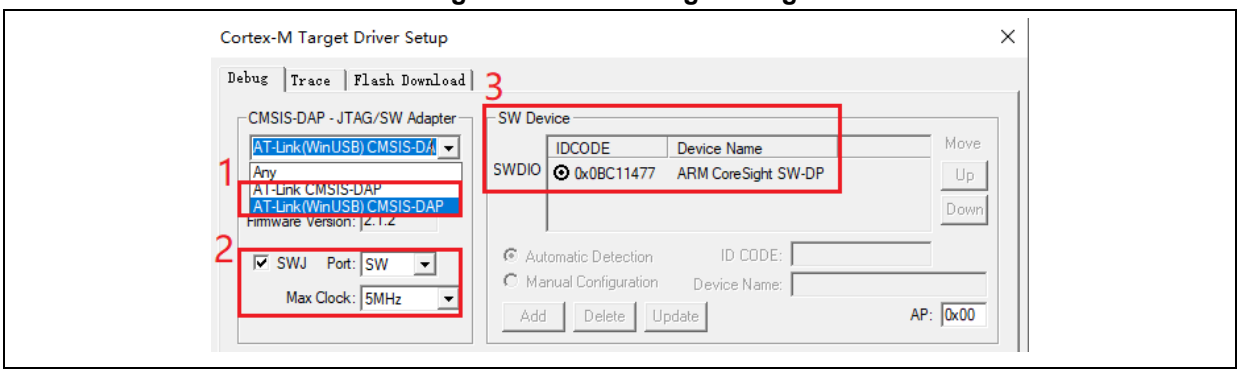
Go to "Debug" and click on "Settings" to enter "Cortex-M Target Driver Setup" interface.

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 by visiting ARTERY's official website → SUPPORT → FAQ → FAQ0136.

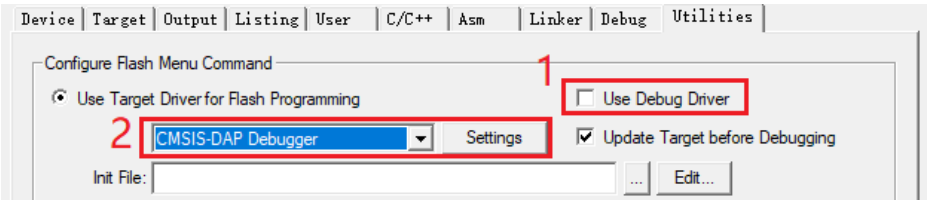
2. Select "SW" in Port, and tick "SWJ"
3. Confirm that the ARM SW-DP debug module is recognized.

Figure 12. Keil Debug Settings



Click on “Utilities”, untick “Use Debug Driver” option (Step 1 below), choose “CMSIS-DAP Debugger” in Settings (Step 2 below), and then re-tick Step 1 (This option must be first unticked and then re-ticked)

Figure 13. Keil Utilities



If users want to use AT-Link in IAR, click on “Project” and “Options”, then select “CMSIS-DAP”, and select “SWD”.

Figure 14. IAR Debug option

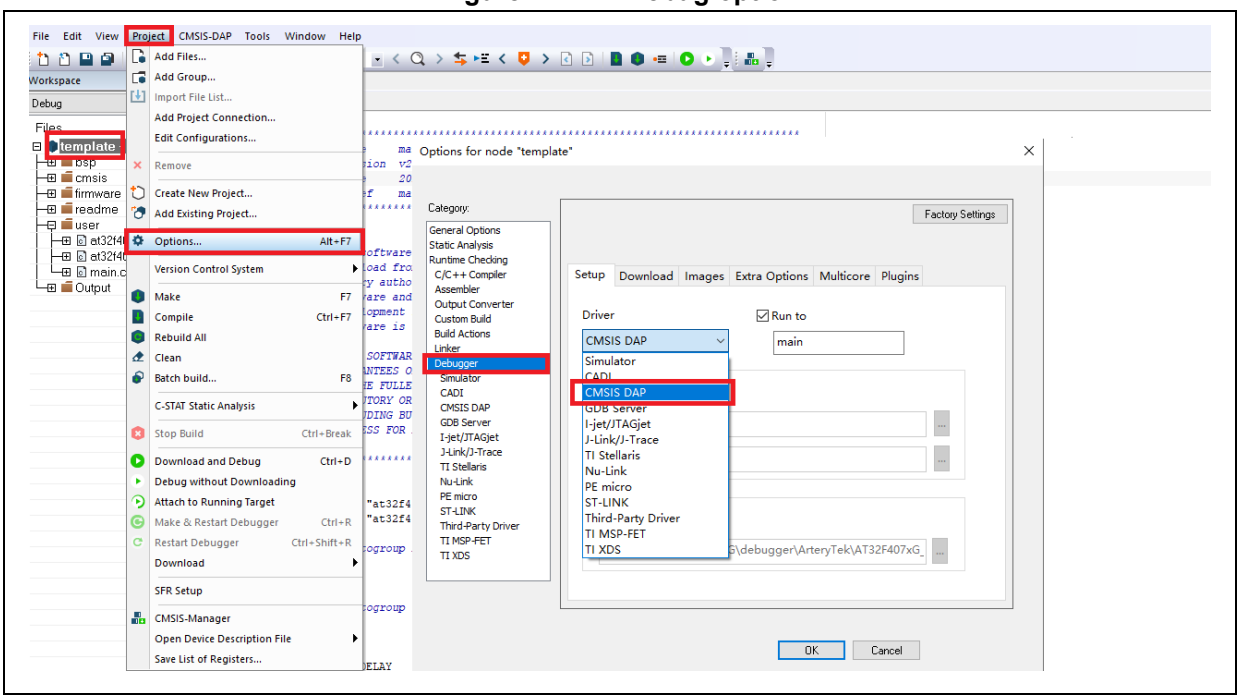
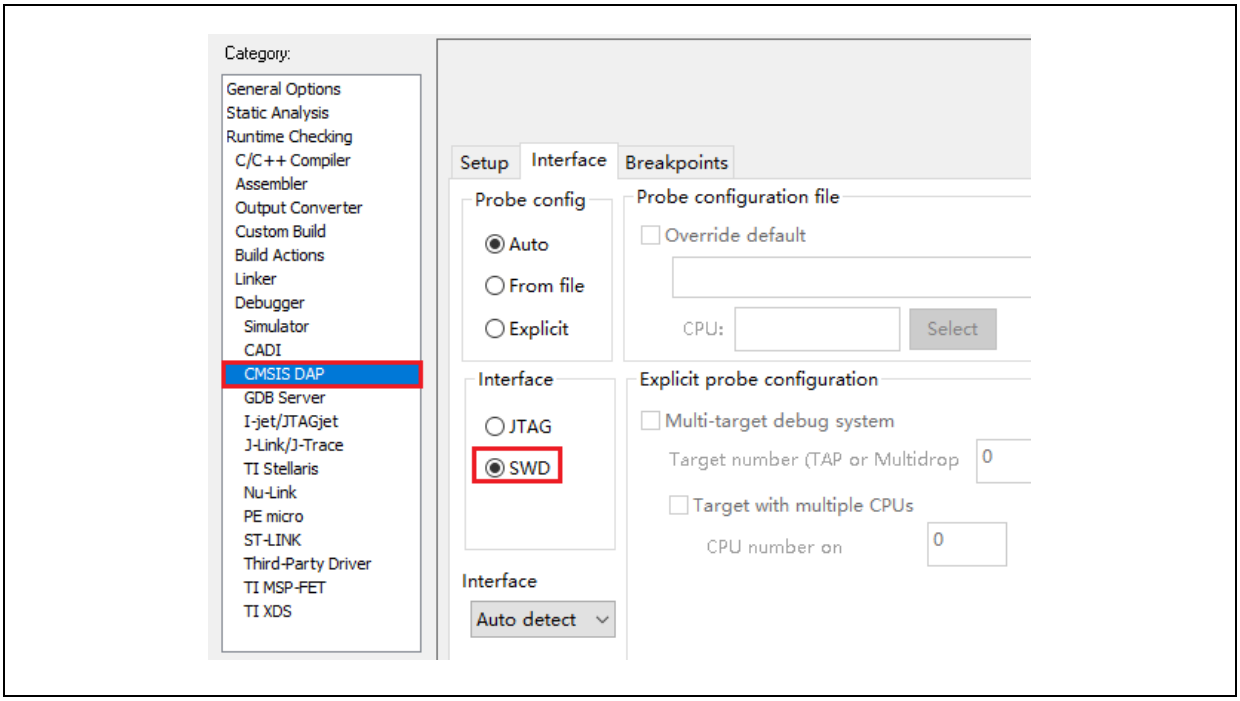


Figure 15. IAR CMSIS-DAP option



Note: For full details, please refer to AT32F403A_F407 Firmware BSP&Pack User Guide by visiting ARTERY's official website → PRODUCTS → Mainstream → AT32F4xx → BSP → "\AT32F403A_F407_Firmware_Library_Vx.x.x\document".

1.1.4 AT32 Work Bench

1.1.4.1 Introduction

This section describes how to use AT32 Work Bench. The AT32 Work Bench is used to generate the initialized C code and corresponding IDE projects through graphic MCU configuration, with the aim of expediting development cycles for engineers.

- Environment setup

  Software resources:

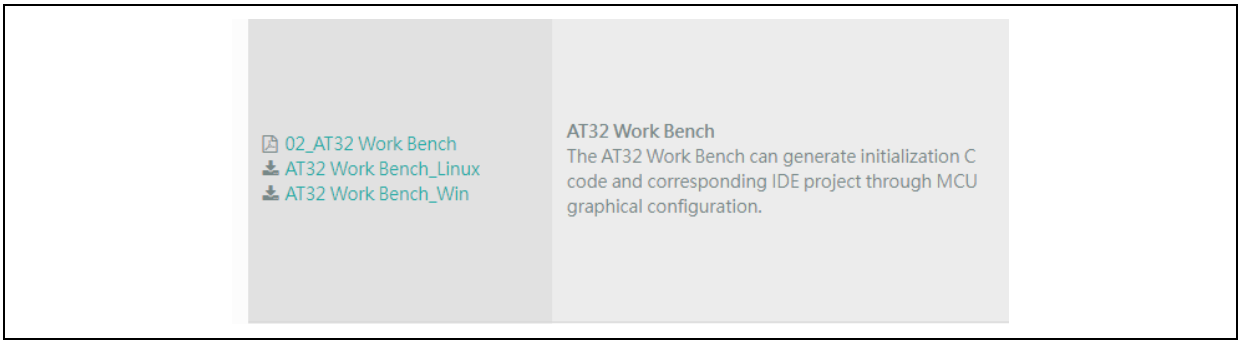
  Windows system: Windows 7 or above operating system

  Linux system: Ubuntu, Fedora and others that support x86_64 architecture

  Hardware resources: the minimum internal memory is 2GB and the minimum hard drive is 4GB

Note: For full details on AT32 Work Bench, please refer to UM_AT32_Work_Bench by visiting Artery's official website → Support → Tool. (Address link: https://www.arterychip.com/en/support/index.jsp)

Figure 16. AT32 Work Bench files at ARTERY official website



1.1.4.2 Installing AT32 Work Bench

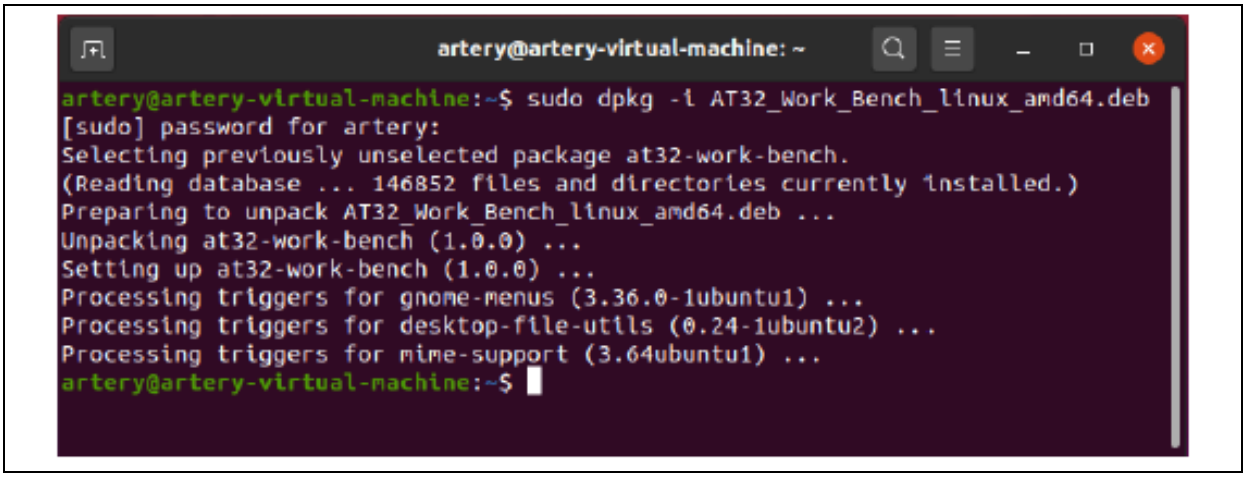
For Windows operating systems, users can directly run the AT32_Work_Bench.exe without needing to install AT32 Work Bench.

For Linux operating systems, Ubuntu 16.4 or above version is supported. Here are two ways to install AT32 Work Bench.

- Use “dpkg” command

As shown in Figure 17, input command line: *sudo dpgk –i AT32_Work_Bench_Linux-x86_64_Vx.x.xx.deb*

Figure 17. Enter command line



- Graphic installation

Copy the “AT32_Work_Bench_Linux-x86_64_Vx.x.xx.deb” to Linux system, and double click it. When a window interface pops up, click “install” button, as shown in Figure 18.

Figure 18. AT32 Work Bench graphic installation

After installation is complete, click “display all programs” button on the left bottom column, and find the “AT32 Work Bench” in the prop-up list, click it and start “AT32 Work Bench”.

1.1.4.3 Project configuration

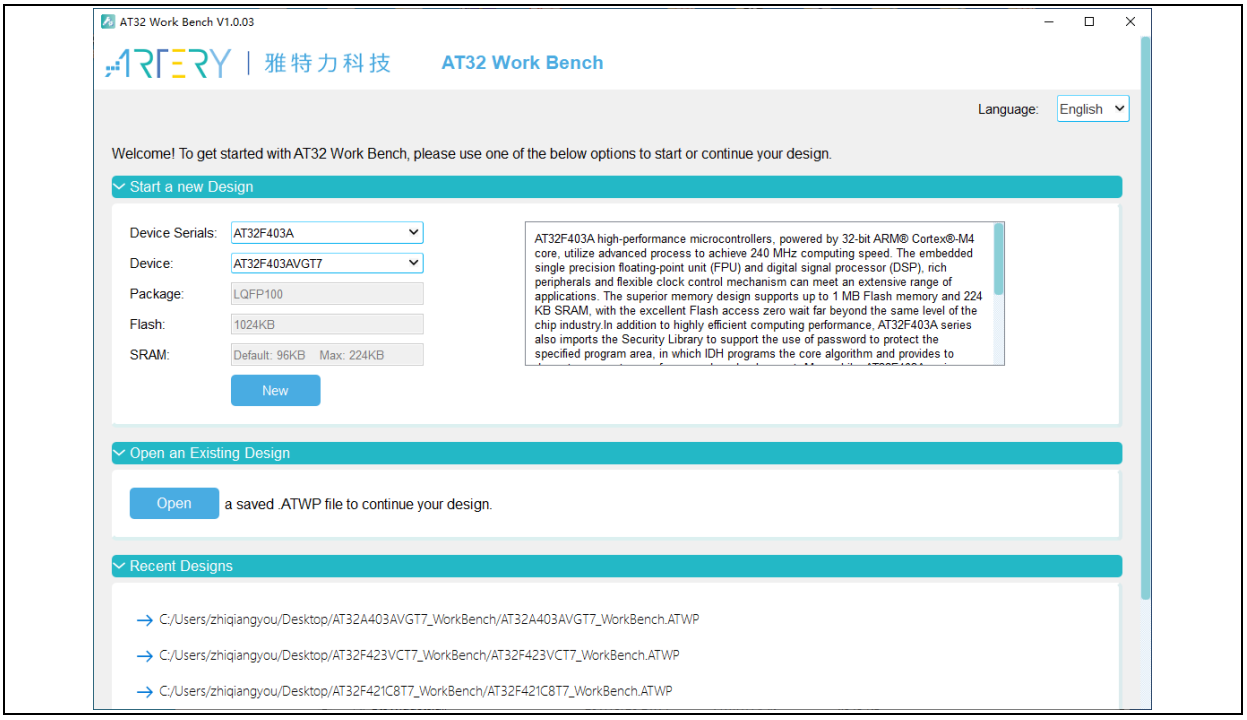
This section uses AT32F403AVGT7 to create an USART project as an example of demonstrating how to generate the initialized C codes and corresponding IDE projects with AT32_Work_Bench. After starting AT32_Work_Bench, the first window interface is the Introductory Page. In this page, there are three options to choose from.

First option: Start a new design

Second option: Open an existing design

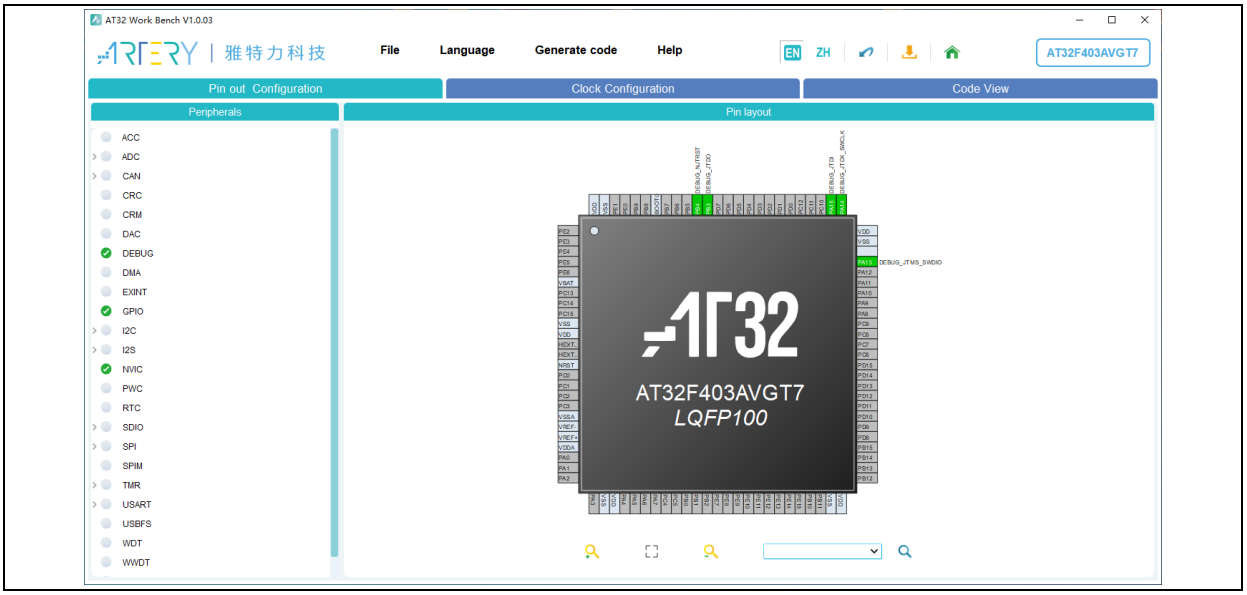
Third option: Recent designs

Figure 19. AT32 Work Bench graphic installation



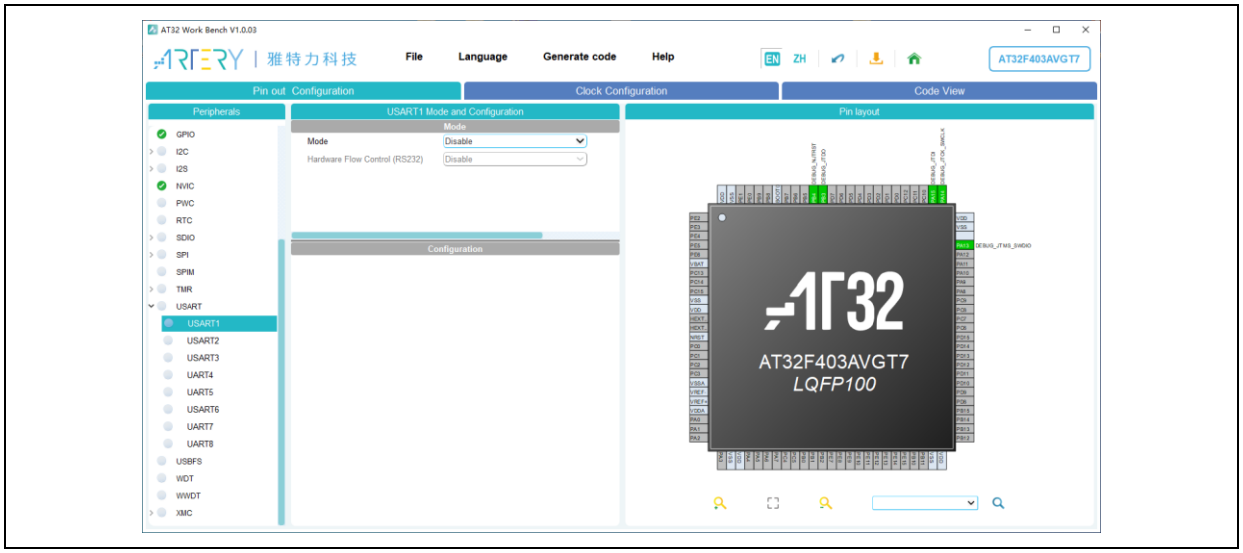
In Introductory Page, we can select a specific MCU device. Click “New” to enter project configuration interface. In here we can see “Pin out configuration”, “Clock configuration” and “Code view”. They are three major functions used to set up AT32_Work_Bench project.

Figure 20. Project configuration interface



(1) In “Pin out configuration”, it is used to configure the desired peripherals and pins. It offers three windows: peripherals, mode and configuration, and pin out.
Users can select a peripheral from the list of “Peripherals”, then go to the “Mode and Configuration” window to select mode and configure parameters. Uses can also choose operating mode of peripherals and pins to be used.
As MCU allows for different peripherals and multiple functions to use the same pin, after a mode is selected, the system automatically configure the pin out that is best suitable for the selected peripherals. As USART1 as an example, USART1 mode and configuration is as follows:

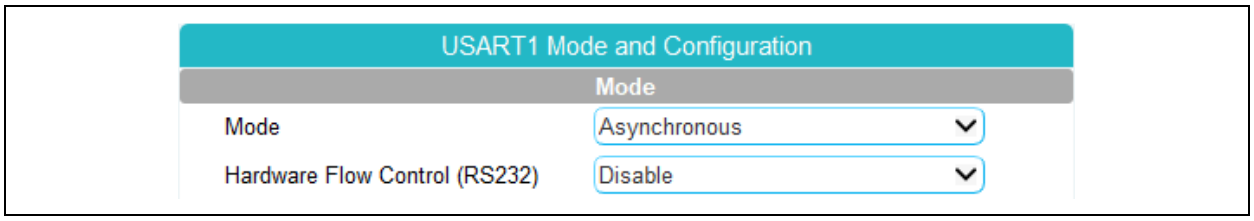
Figure 21. Mode and configuration window



As USART1 as an example, the following procedure is used to configure USART1 peripheral.

- Peripheral mode

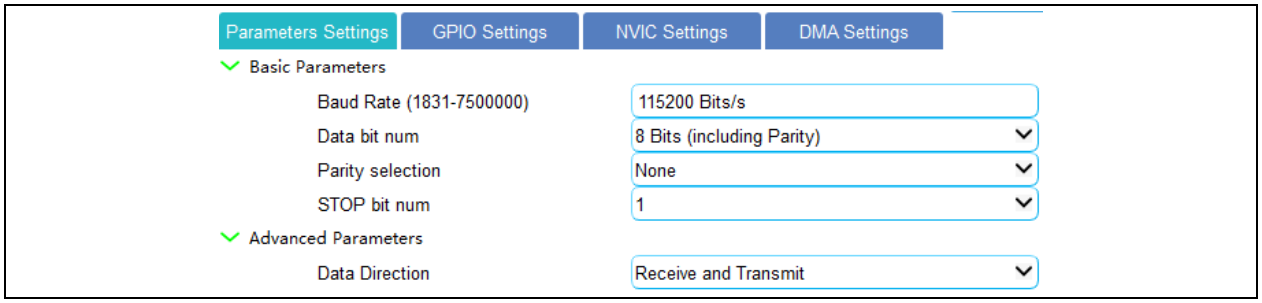
Figure 22. Peripheral mode selection



As shown in Figure 22, we select “Asynchronous” mode for USART1. So in “Pin out and configuration” window, PA9 and PA10 are automatically mapped on to the USART1_TX and USART1_RX signals.

- Parameters settings

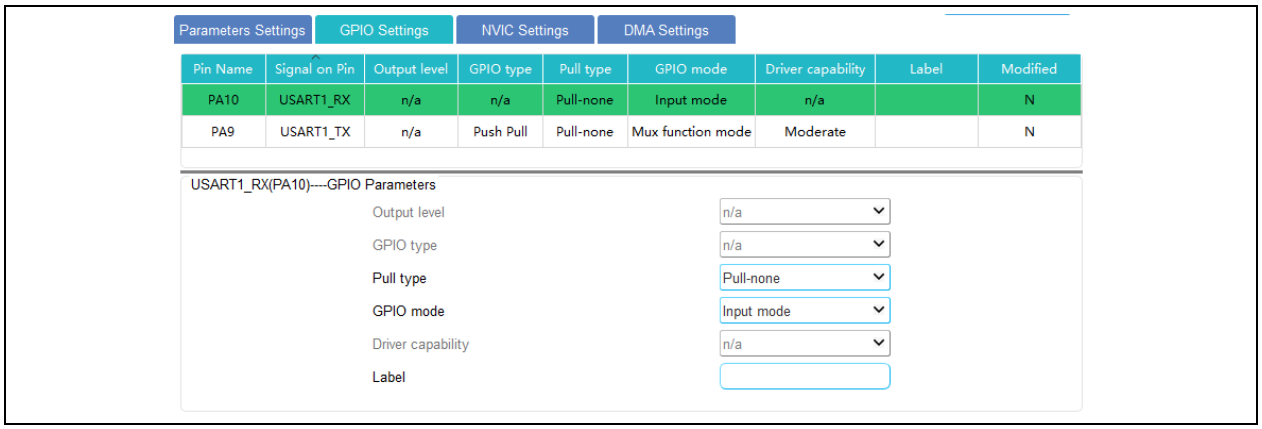
Figure 23. Parameters settings



In “Parameters settings” window, we can set such parameters as baud rate, data-bit-num, data direction.

- GPIO settings

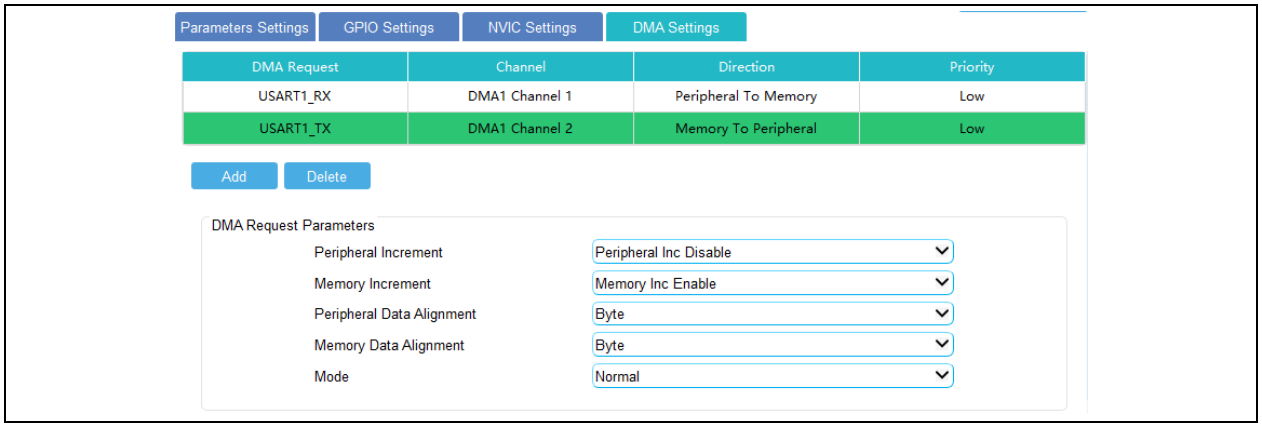
Figure 24. GPIO settings



In “GPIO settings” window, users can set GPIO parameters of USART1_TX and USART1_RX.

- DMA settings

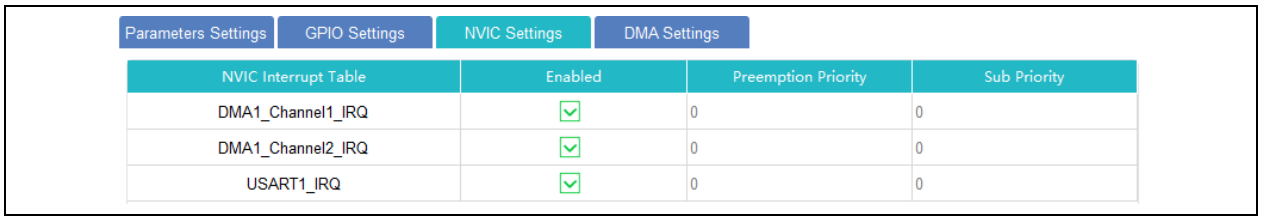
Figure 25. DMA settings



In “DMA settings”, users can configure DMA request parameters of USART1_TX and USART1_RX.

- NVIC settings

Figure 26. NVIC settings

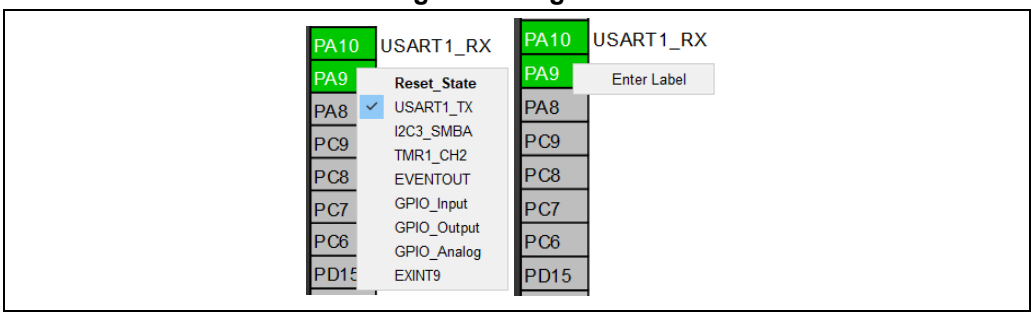


In “NVIC settings”, we can see the NVIC interrupts to be configured. After the corresponding DMA channel of the selected peripheral is enabled, we can also configure corresponding DMA channel interrupts. “Preemption priority” and “Sub priority” needs to be configured in the general NVIC window.

In “Pin out” window, we can the pin out of the selected package type (such as LQFP48, QFN32, TSSOP20,etc). We can view pin name (like PA9), the current state and signal destruction of each pin.

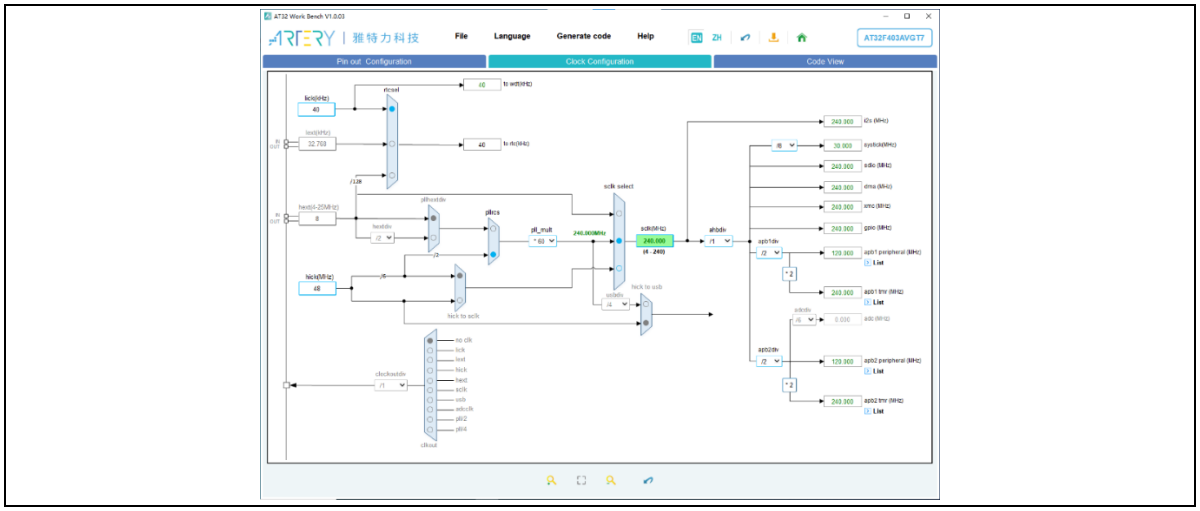
As shown in Figure 27, left click on a pin (except those with fixed mode), and a menu pops up showing the signals corresponding to the pin; For those whose functions have been configured, right click on the pin, a “enter label” button pops up. Users can click on this button to customize a label.

Figure 27. Right click and left click



(2) "Clock configuration" window is used to configure clock path and parameters. Users can use drop-down menu and enter box (shown in Figure 28) to change clock tree configuration parameters to meet actual demands.

Figure 28. Clock configuration window



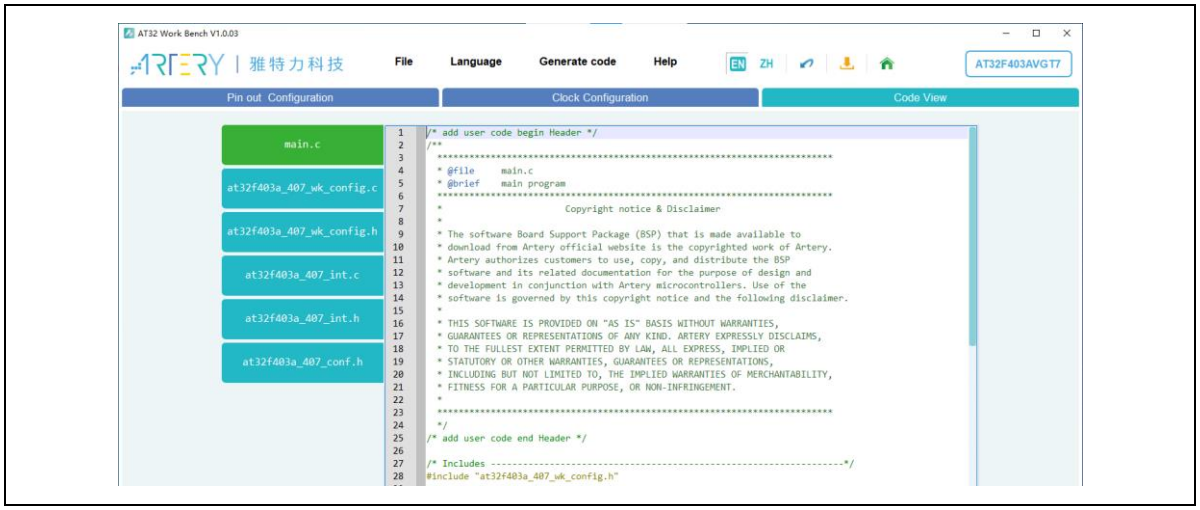
Note 1: To enable LEXT, it is necessary to set low-speed external clock (LEXT) mode in CRM mode window.

Note 2: To enable HEXT, it is necessary to set high-speed external clock (HEXT) mode in CRM mode window.

Note 3: To enable clockout, it is necessary to select "Clock output" in CRM mode window.

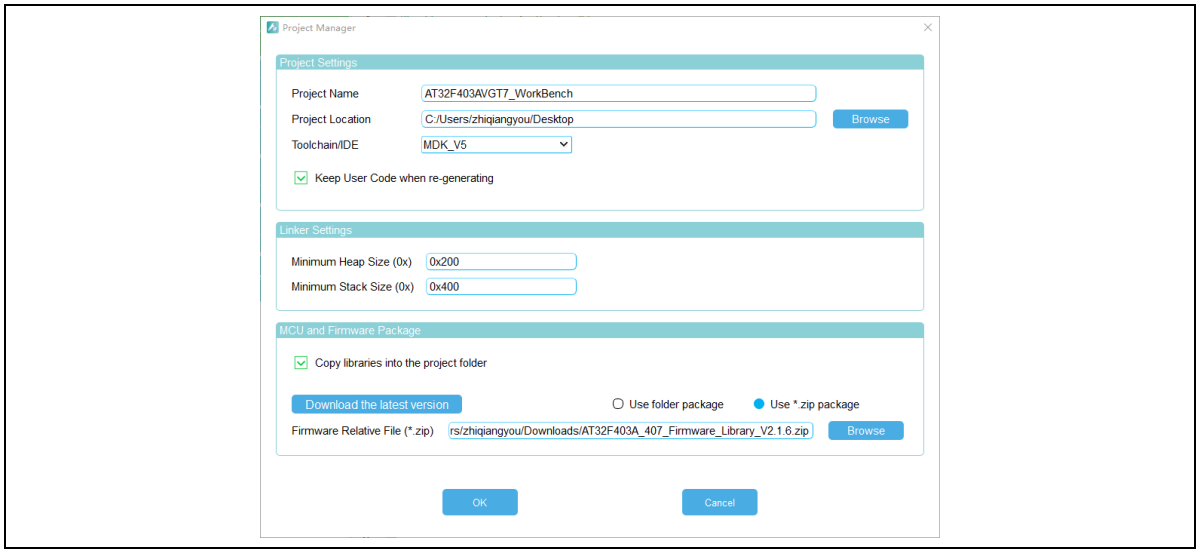
(3) Click on "Code view" button, the system will generate and display the configured code. On the left side, we can see the generated code files. On the right side, we can see the detailed code content.

Figure 29. Code view



Click on “generate code” button, a project manager window will be displayed, as shown in Figure 30.

Figure 30. Project manager



For heap size and stack size, it is recommended to set 0x200 and 0x400 for them respectively. However, these values may need to be greater when the application uses the stack of middleware. After selecting and copying a library file to the project folder, the library files in the firmware package are automatically copied into the project folder when generating code. After the completion of project configuration, click on “confirm” button, user code is generated and IDE project file is selected. Figure 31 shows the structure of a project file.

Figure 31. Project file structure

1.1.5 How to quickly replace SXX with AT32F403A_407

- Step 1: Based on peripheral specifications, Flash size and SRAM size and other features, unsolder SXX32F103 and replace it with the corresponding AT32F403A/F407 part number;
- Step 2: Use ARTERY ICP/ISP or KEIL/ IAR to download SXX32F103 HEX files or BIN files;
- Step 3: If needed, download other materials in addition to 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 and MG0009_ Migrating from SXX32F103 to AT32F407* (visit ARTERY’s official website → PRODUCTS → Mainstream → AT32F4xx);
- Step 6: If the program fails to run after above steps, please refer to other sections of this application note or contact local agents or ARTERY technicians for help.

Note: Due to the flexible memory expansion design of AT32F403A/F407, its internal Flash memory has non-zero wait area, which may affect operational performance of some SXX32F103 programs on the AT32F403A/F407 series. In this case, please refer to AN0004_Performance_Optimization (visit ARTERY’s official website → SUPPORT → AP Note → AN0004).

1.2 AT32F403A/F407 MCU function enhancement

1.2.1 PLL clock settings

1.2.1.1 PLL greater than 72 MHz

The internal PLL in the AT32F403A/F407 has a maximum output of 240 MHz. As a result, settings are slightly different when the clock is greater than 72 MHz. Users need to set the PLLRANGE register according to the output frequency (PLL>72 MHz, PLLRANGE=1; PLL≤72 MHz, PLLRANGE=0).

PLL setting example for SXX32F403 BSP (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 clock greater than 72 MHz on the AT32F403A/F407 series, it is necessary to configure the PLLRANGE bit.

As PLL=240 MHz an example, proceed as follows:

Figure 32. Use SXX32F103 to output 240 MHz clock on AT32F403A/F407

```
#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);
```

PLL setting example for AT32F403A/F407 BSP (HEXT=8 MHz)

Figure 33. AT32F403A/F407 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;
```

As PLL=72 MHz an example, proceed as follows:

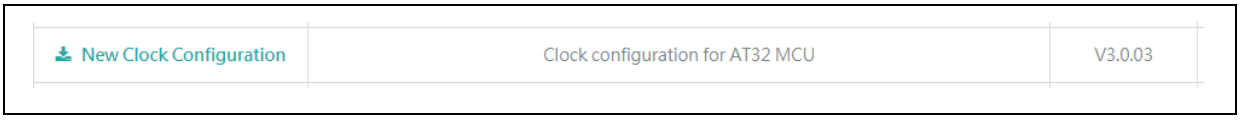
```
crm_pll_config(CRM_PLL_SOURCE_HEXT_DIV, CRM_PLL_MULT_18, CRM_PLL_OUTPUT_RANGE_LE72MHZ);
```

As PLL=240 MHz an example, proceed as follows:

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

For full details on clock configuration, please refer to AN0082_AT32F403A_407_CRM_Start_Guide (visit ARTERY's official website→SUPPORT→AP Note→ AN0082) to learn how to configure and modify AT32F403A/F407 clock source code and how to use New Clock Configuration tool to generate the desired clock code (visit ARTERY's official website→PRODUCTS→ Mainstream→AT32F4xx).

Figure 34. New Clock Configuration tool on Artery official website



1.2.1.2 Auto step mode

When the internal PLL of AT32F403A/F407 is set to 108 MHz and above, it is necessary to enable auto step mode.

For SXX32F103 program, users need to open “system_Sxx32f10x.c” to find the current system clock frequency configuration function (need to first finish Section 1.2.1.1 PLL settings), and add the following code marked in Italic black to the *static void SetSysClockToxxM(void)* function.

Figure 35. SXX PLL auto step mode configuration

```
/* Wait till PLL is ready */
while((RCC->CR & RCC_CR_PLLRDY) == 0)
{
}
*((unsigned int   *)0x40021054) |= (0x30);   // Enable auto step mode
/* 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 mode
```

For AT32F403A/F407 BSP, here is an example of PLL auto step-by-step switch function.

Figure 36. AT32 PLL auto step mode

```
/* 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();
```

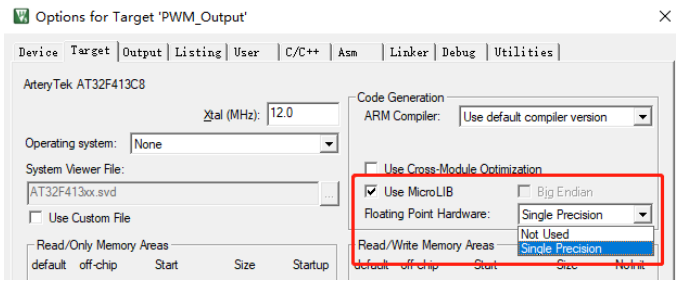
Note: If the auto step mode is enabled, it must be disabled after clock switch. This means that enable and disable must operate in pair.

1.2.2 How to enable FPU function (Floating point unit)

There are two ways to enable FPU in Keil:

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

Figure 37. Enable FPU in Keil



② As SXX32F10X does not support FPU function, users need to follow the procedure below if they want to enable FPU in the developed project under SXX library.

- 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 code at the beginning of the SystemInit function of the system_stm32f10x.c, and add cm4.h to the project.

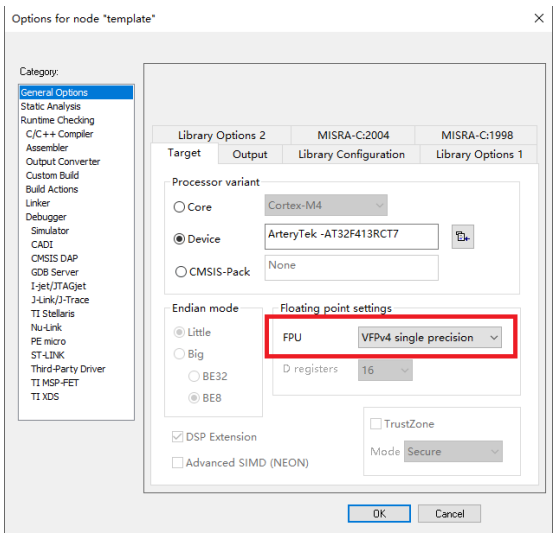
Figure 38. Add FPU enabling codes in Keil

```
/* 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 ways to enable FPU in IAR:

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

Figure 39. Enable FPU in IAR



② As SXX32F10X does not support FPU function, users need to follow the procedure below if they want to enable FPU in the developed project in SXX library.

- 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 the SystemInit function of the system_stm32f10x.c, and add cm4.h to the project.

Figure 40. 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 see AN0037_How_to_use_FPU (visit ARTERY's official website→ SUPPORT→AP Note→AN0037).

1.2.3 ZW/NZW Flash and embedded SRAM configuration

The configuration and distribution of the internal Flash memory and SRAM are done through user system data.

As AT32F403AVGT7 as an example, its internal Flash memory and SRAM can be configured as follows:

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

The core reads code stored in zero-wait Flash without any latency, which means that there is no need to insert any wait state due to CPU running at a fast frequency. For example, if the system clock is 240 MHz and AT32F403A zero-wait area is 256 KB, then the first 96 KB of a 512 KB bin file is executed at 240 MHz, and the remaining 256 KB bin file stored in the non-zero wait area is processed at 96 MHz. Despite, it is faster than 72 MHz 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

SRAM can be selected in any of the following ways:

The SRAM size of AT32F403A series is configured through EOPB0 in the user system data area at address 0x1FFF_F810.

EOPB0=0xFF indicates that the on-chip SRAM is 96 KB.

EOPB0=0xFE indicates that the on-chip SRAM is 224 KB.

To take EOPB0 enable operation into account, a power-off or RESET must be performed once.

The following section takes AT32F403AVGT7 as an example to introduce how to configure SRAM size through ICP/ISP. For more information about SRAM extension, please refer to *AN0026_Extending_SRAM_in_User's_Program* (visit ARTERY's official website → SUPPORT→ AP Note → AN0026).

① ICP/ISP tool

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

Connect AT-Link-EZ/AT-Link /J-Link to MCU — Target —User system data — Select **96 KB/224 KB** EOPB0 (complete other settings if needed) — Apply to device.

Figure 41. User system data in ICP Programmer

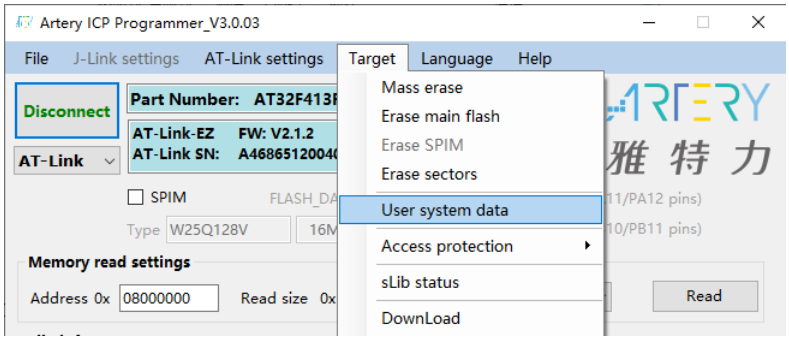
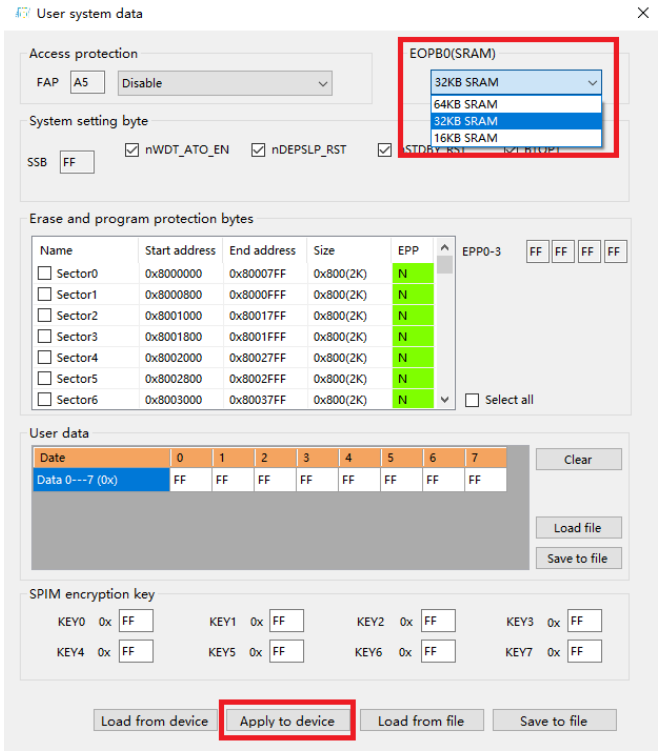


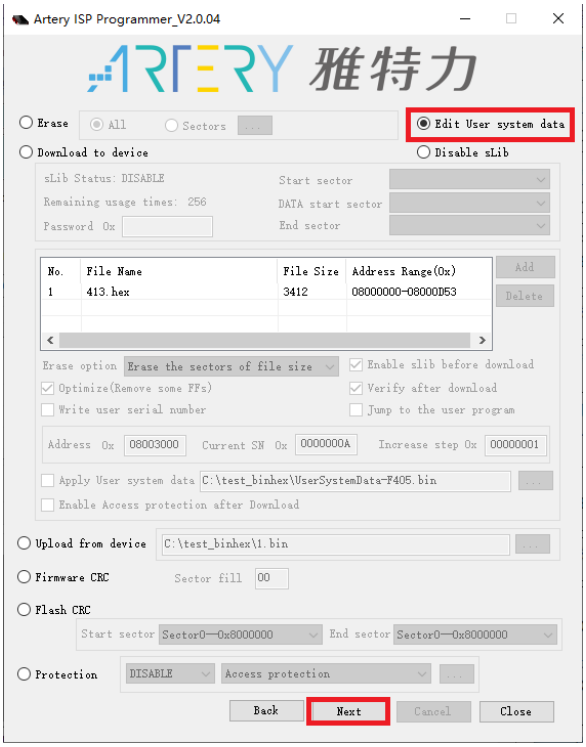
Figure 42. Select SRAM size in user system data



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

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

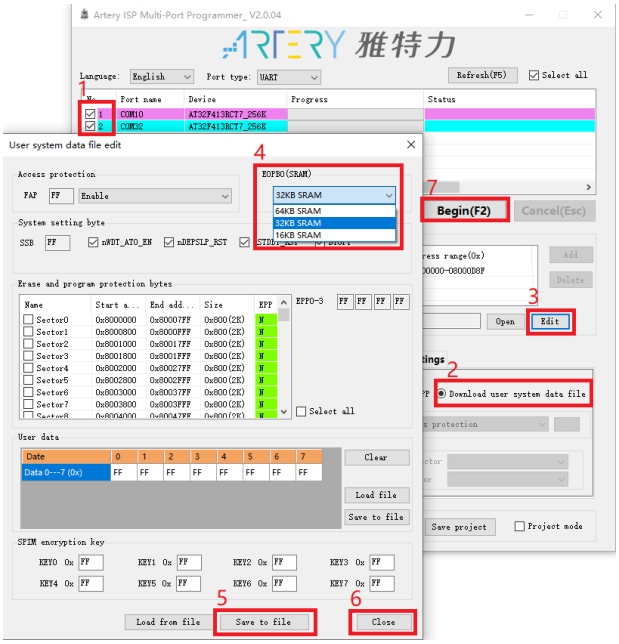
Figure 43. Edit user system data in ISP Programmer



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

Select the desired Port — download user system data file — Edit — Select **96 KB/224 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 44. User system data setting in ISP Multi-Port Programmer



② Users can also change SRAM size through Bootloader program (IAP). Note that the SRAM size in the compiler must be the same as the modified SRAM size (see AN0026)

Figure 45 is an example of using SXX32F103 program:

Figure 45. Extend_SRAM(void) 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 EO
PB0
    {
      FLASH_Unlock();
      FLASH_EraseOptionBytes();
      FLASH_ProgramOptionByteData(0x1FFFF810, SRAM_224k);
      …                    //set other user system data
      FLASH_Lock();
      NVIC_SystemReset();
    }
}
```

Figure 46 shows an example of using AT32F403A/F407 program:

Figure 46. Extend_SRAM(void) 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();
    }
}
```

Note that the user system data area must be erased prior to changing them. If the other items in the user system data area are already configured and need to be read, they can be erased and written again when setting SRAM size.

③ How to change SRAM size in the startup file?

The SRAM is loaded when the startup file is being run. If without IAP and the application uses the SRAM larger than 96KB, then SRAM loading will fail and enter hardfault, resulting in the application to fail to run. Therefore it is necessary to set the SRAM to 224KB before loading SRAM.

In Keil environment, please add the following code marked in **BOLD** font.

Figure 47. Change SRAM in Keil

```
; 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
```

In IAR environment, please add the following code marked in **BOLD** font.

Figure 48. Change SRAM in IAR

```
; 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 aforementioned configuration, it is also necessary to add declaration and define the Extend_SRAM function, please refer to ② in the Section 1.2.3 for details.

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

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

1.2.4.1 Access protection

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

The ICP/ISP tool can be used to enable/ disable access protection based on the following procedures:

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

  Enable access protection: Target — Access protection — Enable (Y).

  Disable access protection: Target — Access protection — Disable (Y).

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

  Enable access protection: Enable/disable access protection — enable Access protection – Next —Yes

  Disable access protection: Enable/disable access protection — disable Access protection - Next—Yes

Figure 49. Enable access protection with ISP

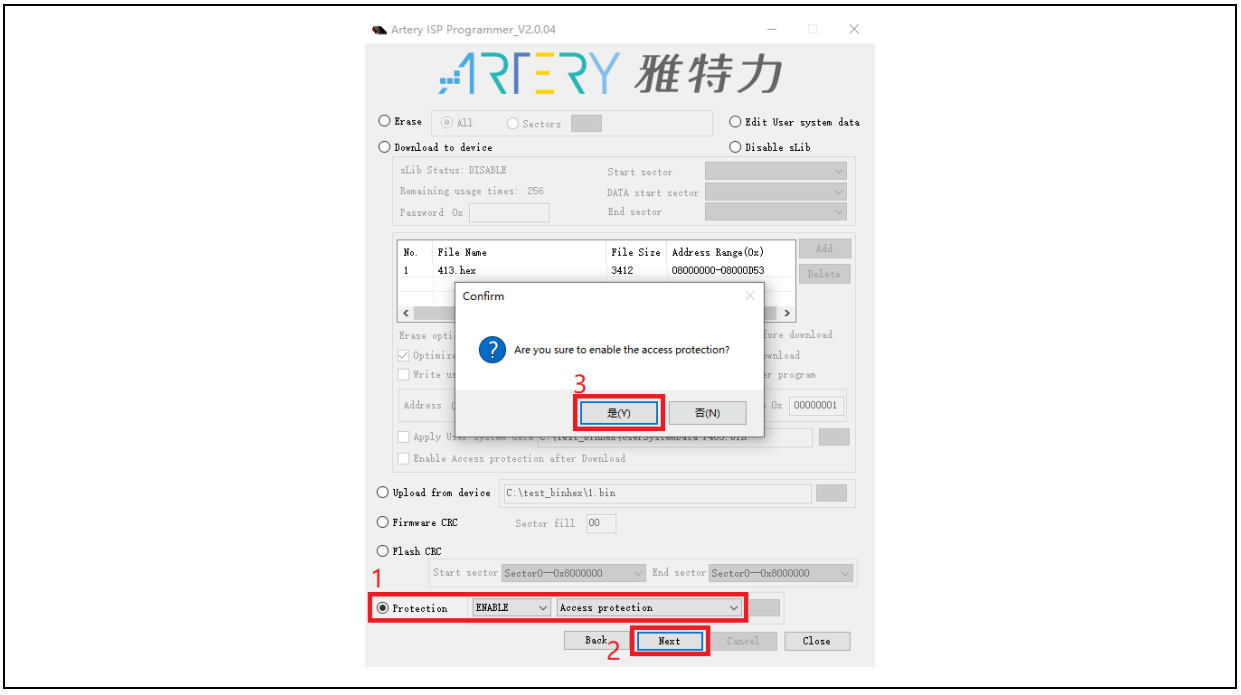
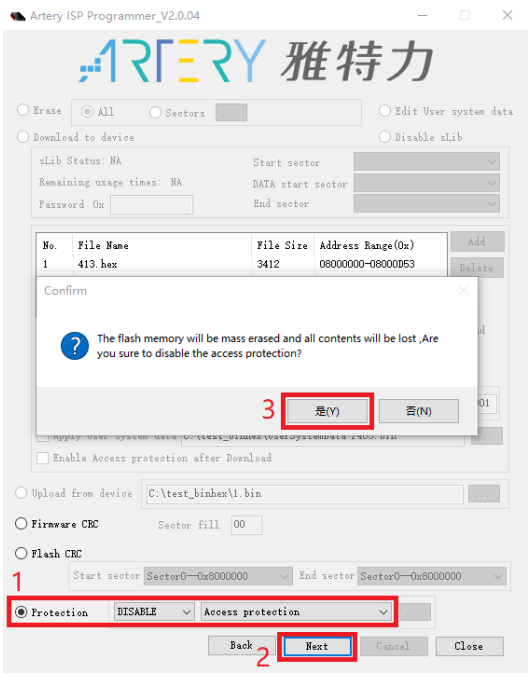


Figure 50. Disable access protection with ISP



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

Enable access protection: Enable/disable access protection — enable access protection — Start —Yes

Disable access protection: Enable/disable access protection — disable access protection — Start — Yes

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

1.2.4.2 Erase and program protection

Write protection is applied to the entire Flash memory area or a given page in Flash memory. Once the Flash write protection is enabled, the internal Flash memory area cannot be written in any way.

The ICP/ISP programmer can be used to enable/disable erase and program protection based on the following procedures.

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

Enable erase and program protection: Target — User system data — Tick “erase and program protection bytes” — Apply to device.

Disable erase and program protection: Target — User system data — Untick “erase and program protection bytes” — Apply to device.

Figure 51. Enable erase and program protection with ICP

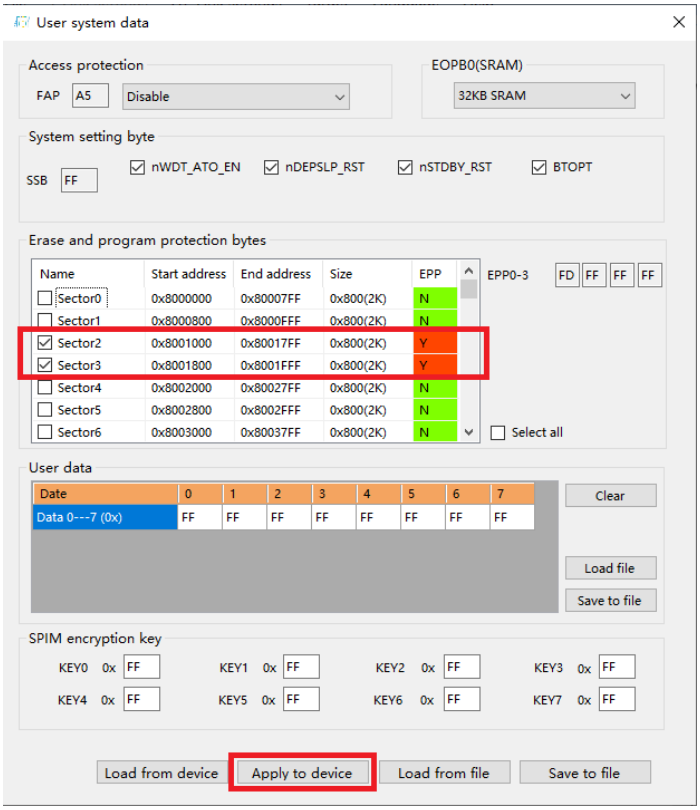
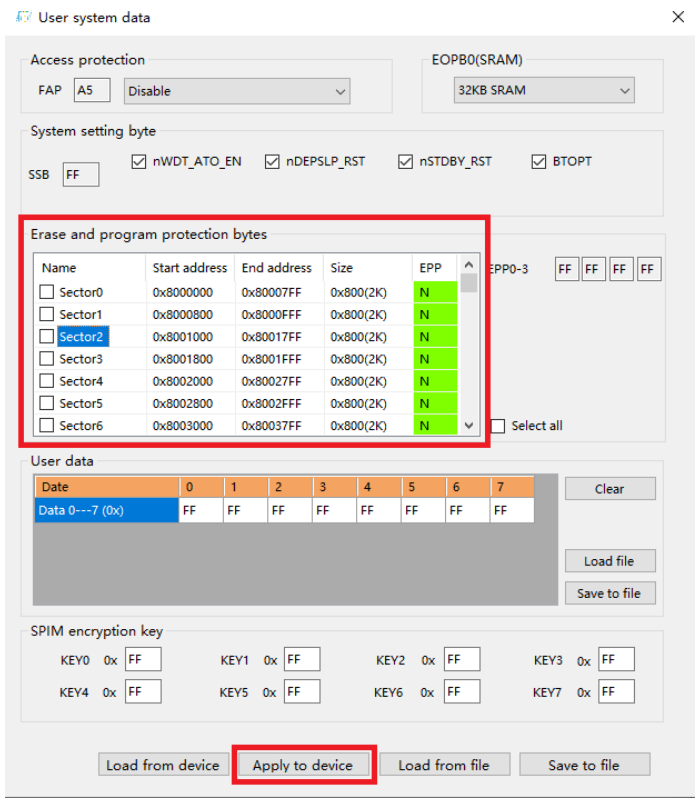


Figure 52. Disable erase and program protection with ICP



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

  Enable erase and program protection: Enable/disable protection — enable Erase and program protection — Next — Yes

  Disable erase and program protection: Enable/disable protection — disable Erase and program protection — Next —Yes

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

  Enable erase and program protection: Enable/disable protection — enable Erase and program protection — Start —Yes

  Disable erase and program protection: Enable/disable protection — disable Erase and program protection — Start —Yes

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

1.2.4.3 External Flash encryption (download and read encrypted data)

To perform encryption on external Flash memory, it is necessary to set the encryption range and password before programming and enabling access protection.

The encryption range refers to the area to be encrypted starting from 0x08400000. If the encryption key is all 0xFF or 0x00, the external Flash memory is not encrypted else it is encrypted.

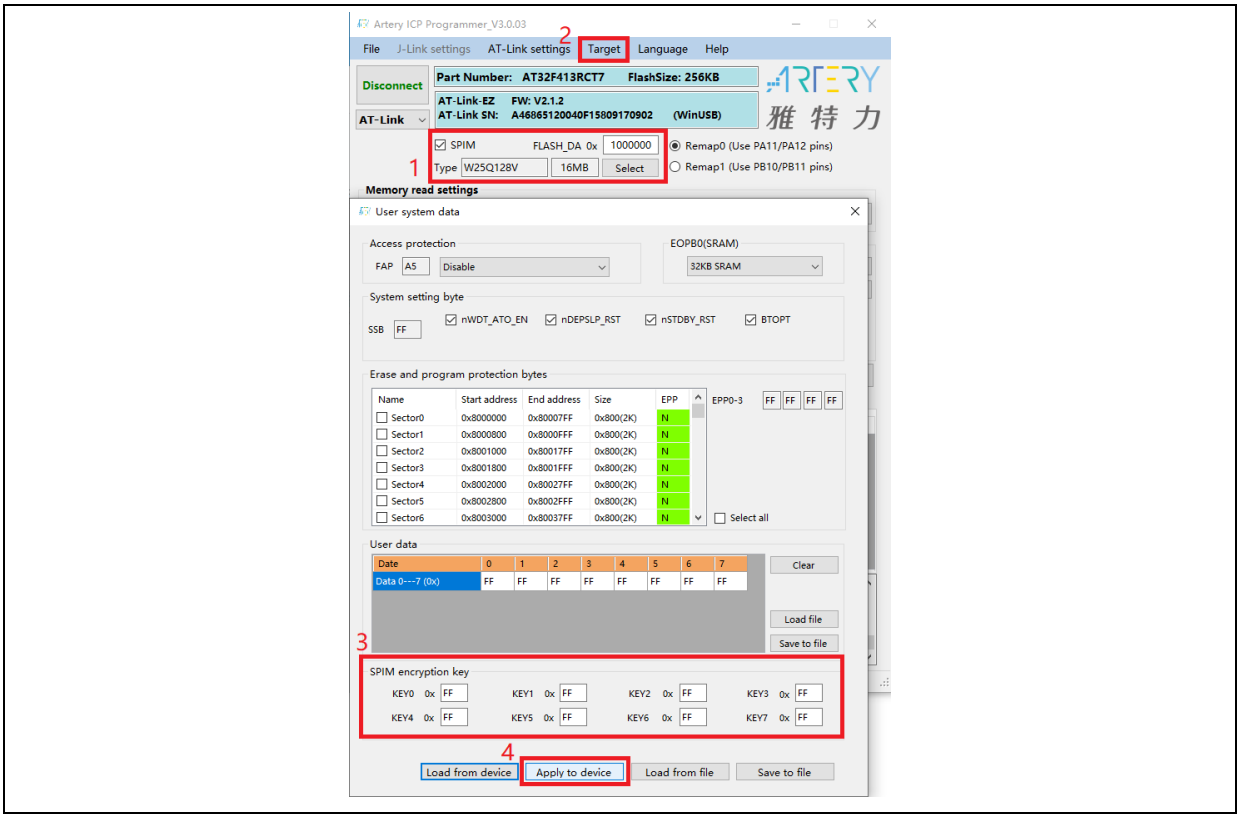
Disabling access protection sets the external Flash encryption key to all 0xFF.

Use ICP/ISP to encrypt external memory:

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

  Tick SPIM → Select SPIM type → Set Flash_DA 0x → Target → User system data → Modify SPIM encryption key → Apply to device.

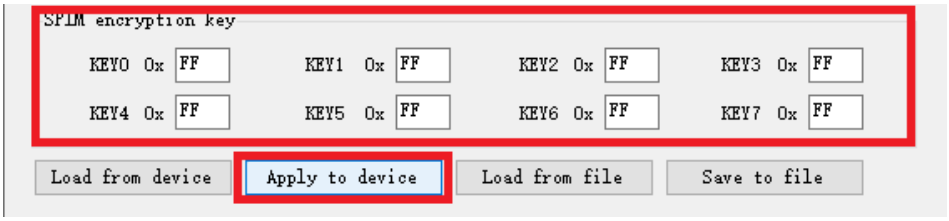
Figure 53. External memory encryption with ICP



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

Edit user system data → Next → Modify SPIM encryption key → Apply to device.

Figure 54. External memory encryption with ISP



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

Download user system data file → Edit → Modify SPIM encryption key → Save to file → Start.

1.2.5 Distinguish AT32 MCU from other MCUS

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

Figure 55. Read Cortex ID

```
cortex_id = *(uint32_t *)0xE000ED00;// Read Cortex model
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 56. 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. By packing the obtained ID information into a 64-bit data, users can distinguish AT32 MCU from other ones. For details, refer to the corresponding Reference Manual (DEBUG section) and AN0016_Recognize_AT32_MCU (visit ARTERY's website → SUPPORT→ AP Note → AN0016).

2 Frequently asked questions

2.1 Enter Hard Fault Handler

- SRAM size you are using exceeds the SRAM size configured in user system data

  See Section 1.2.3 and use ICP/ISP to extend 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 57. Add FPU enable code

```
  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 outside the boundary

  Find the location where data boundary is violated and change them back to normal range.

- System clock setting out of spec

2.2 J-Link fails to identify IC in Keil

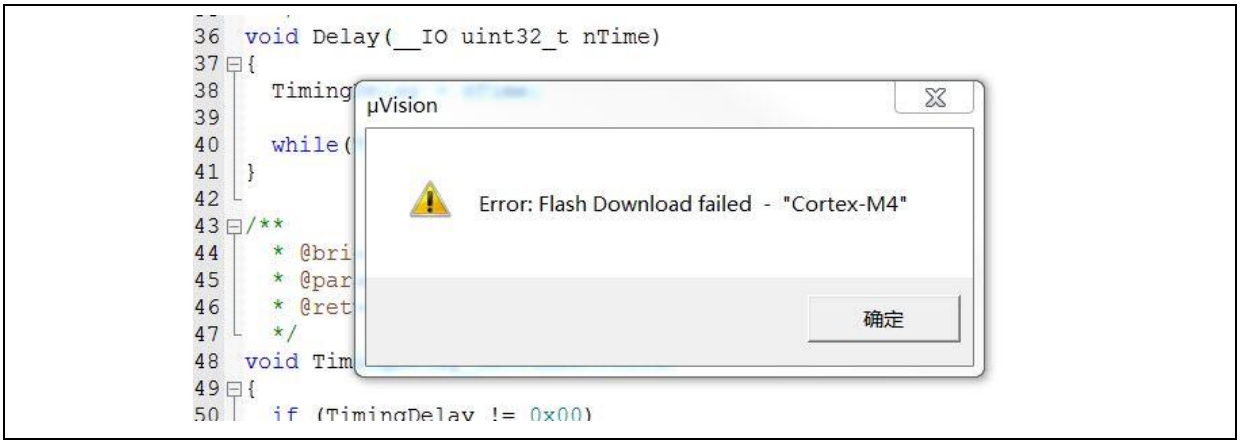
- Please refer to *FAQ0008_ J-Link cannot find IC* (visit ARTERY's official website → SUPPORT → FAQ → FAQ0008)
- Please refer to *FAQ0132_Add Artery MCU to J-Link* (visit ARTERY's official website → SUPPORT → FAQ → FAQ0132).

2.3 Problems occurred in program download

2.3.1 Error warning: Flash Download failed–“Cortex-M4”

The following warning message pops up during Keil emulation or download:

Figure 58. Flash Download failed–“Cortex- M4”



Here are the possible reasons behind this problem:

- Access protection is enabled. Please disable access protection before download
- Flash file algorithm is incorrect or Flash file algorithm is not loaded. Please add a correct Flash file algorithm
- BOOT0 and BOOT1 settings are incorrect. Set BOOT0=0 and BOOT1=0 to allow MCU to boot from the main Flash memory.
- Use older version of J-Link. Please use 6.20C or above.
- JTAG/SWD PIN is disabled. 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, select “–Oz” as the default optimization level of Keil AC6. This actually should be changed to -O0/-O1.
- JTAG/SWD PIN is disabled. Please refer to “2.3.5 Resume download”.

2.3.4 ISP serial port gets stuck during download

When the ISP serial port is used to download, it may be stuck so that it cannot be released. Following the procedure below:

- Check whether the power supply is stable
- Use a high-quality USB-to-serial port tool, i.e., CH340 chip.

2.3.5 How to resume download

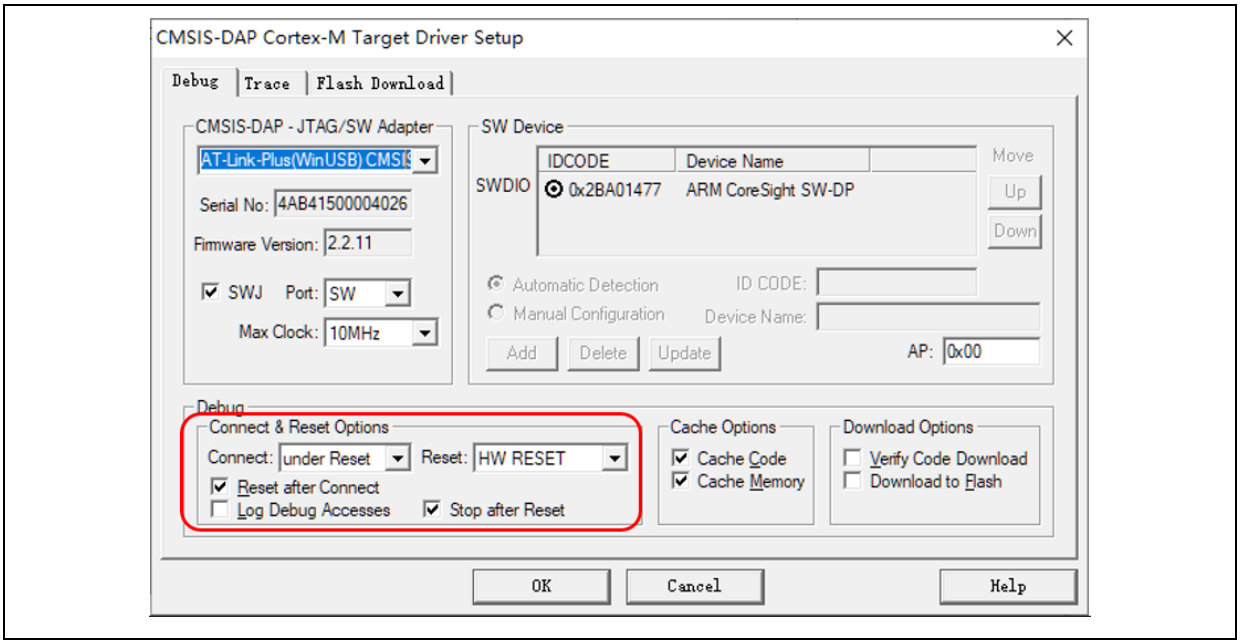
When users fail to download programs following the operations below on the AT32F403A/407 series:

- After JTAG/SWD PIN is enabled, the program cannot be downloaded and JTAG/SWD device cannot not be identified
- After Standby mode and other low-power mode entry, the program cannot be downloaded and JTAG/SWD device cannot be identified

The solution to this problem is to halt MCU before program execution. Here are several methods for reference.

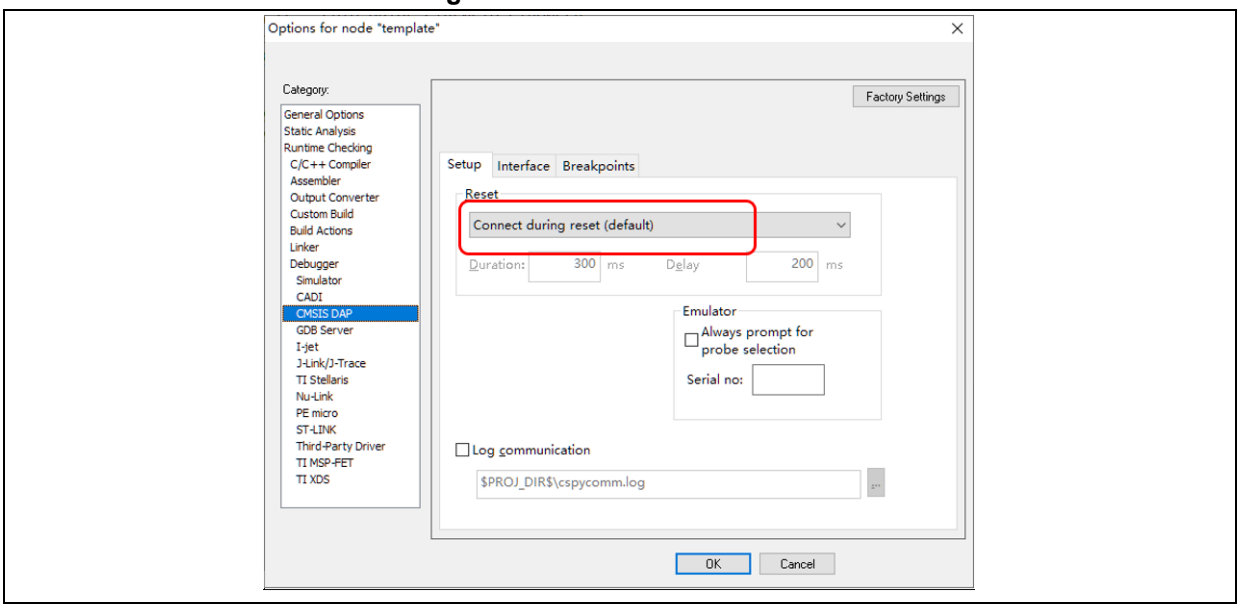
1. Change boot mode so that the microcontroller boots from Flash memory or SRAM, and then reset the microcontroller with Reset pin. By doing so, program can be erased and download operation is resumed
2. Use ICP tool and AT-Link debugger. Connect the AT-Link RST pin to the reset pin of the microcontroller. Click “connect” button on ICP interface. After successful connection, erase program to resume download
3. Use Keil and AT-Link debugger. Connect the AT-Link RST pin to the reset pin of the microcontroller. On Keil “Debug” window interface, select the following options marked in red box in Figure 41. This operation will erase program and resume download.

Figure 59. Debug interface in Keil



4. Use IAR and AT-Link debugger. Connect the AT-Link RST pin to the reset pin of the microcontroller. On IAR “CMSIS DAP” window interface, select the following options marked in red box in Figure 42. This operation will erase program and resume download.

Figure 60. CMSIS DAP in IAR



3 Security Library (sLib)

3.1 Introduction

As more and more MCU applications require complex algorithms and middleware solutions, how to protect core algorithms and related intellectual property rights of (such as core algorithms) software solution providers is emerging as an important topic and concern.

To provide robust protection function, the AT32F403A/F407 series is equipped with a security library (sLib) to protect IP-Codes against being changed or read by end user's programs.

3.2 Principles of application

- Security library (sLib) is a secure area protected by an encryption key in the main memory. This area is used for software solution providers to store their core algorithms for better protection purpose. In the meantime, the rest of the area can be used for secondary-level development by terminal customers.
- Security library is divided into the instruction security library (SLIB_INSTRUCTION) and data security library (SLIB_DATA). Users can select part of or the whole security library for storing instructions. However, it is not possible to use the whole security library for storing data.
- Program codes in the instruction security library (SLIB_INSTRUCTION) can only be fetched (can only be executed) by MCU through I-Code bus but cannot be read through D-Code in the way it reads data (such as ISP/ICP debug mode or boot program from internal RAM). When accessing the SLIB_INSTRUCTION in the form of reading data, values return all 0xFF.
- Data in the data security library (SLIB_DATA) can only be read through D-Code bus but cannot be written.
- The program code and data in security library cannot be erased unless the correct code is keyed in. If a wrong password is entered in an attempt to write or erase the security library, a warning message will be issued by setting EPPERR=1 in the FLASH_STS register.
- The program code and data in security library are not erased when end users perform a mass erase on the main Flash memory.

- After sLib protection is enabled, it is possible to unlock it by writing the previously preset password in the SLIB_PWD_CLR register. After the security library protection is unlocked, the MCU will perform a mass erase on the main Flash memory (including the contents of security library). As a result the program code will not be leaked even if the password set by the software solution provider is leaked.

3.3 How to use Security library

For details on security library, refer to *AN0040_AT32F403A_407_Security_Library_Application_Note* (visit 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.
2024.01.05	2.0.4	Updated the descriptions on Resume AT32 MCU download; Added AT32 Work Bench description.

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 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.