AN0040

Application Note

AT32F403A/407 Security Library Application Note

Introduction

This application note mainly introduces the security library (sLib) application principle of AT32F403A/407 MCUs, operation methods and example projects.

Applicable products:

Part number	AT32F403A
	AT32F407

Contents

1	Ove	/erview				
2	Арр	Application principles				
	2.1	Application principle of sLib				
	2.2	How to enable sLib protection				
	2.3	How t	to disable sLib protection	9		
	2.4	Comp	bile and execute program in sLib	10		
		2.4.1	Setting interrupt vector table as security library not allowed	11		
		2.4.2	Correlation between sLib area and user code area	11		
3	Exa	mple a	pplications of sLib	14		
	3.1	Exam	ple application requirements			
		3.1.1	Hardware requirements	14		
		3.1.2	Software requirements	14		
	3.2	Overv	<i>r</i> iew	14		
	3.3	3.3 SLIB protected code: FIR low-pass filter				
	3.4	Projec	ct_L0: example for solution providers			
		3.4.1	Generate execute-only code	16		
		3.4.2	Compile security library address	18		
		3.4.3	Enable sLib protection	22		
		3.4.4	Project_L0 execution process	23		
		3.4.5	Generate header file and symbol definition file	25		
	3.5	Projec	ct_L1: example for end users			
		3.5.1	Create user application project	27		
		3.5.2	Add symbol definition file to project	27		
		3.5.3	Call functions in SLIB-protected area	29		
		3.5.4	Project_L1 execution process	29		
		3.5.5	SLIB protection in debug mode	29		
4	Inte	Integrate codes and download				
	4.1	.1 Program codes separately				
	4.2	Integrate and program codes				
5	Rev	ision history				

List of tables

Table 1. Flash size of AT32F403A/407	8
Table 2. Document revision history	

List of figures

Figure 1. Mapping of main Flash memory featured with sLib	8
Figure 2. Literal pool example (1)	10
Figure 3. Literal pool example (2)	11
Figure 4. Function in sLib area calling the function in user code area	12
Figure 5. Example of user-defined function	13
Figure 6. Example application process	15
Figure 7. Example application	15
Figure 8. FIR low-pass filter	15
Figure 9. Enter Option interface in Keil	17
Figure 10. Select Execute-only Code in Keil	17
Figure 11. Enter Option interface in IAR	17
Figure 12. Set C/C++ options in IAR	18
Figure 13. Main Flash memory mapping and RAM partition	19
Figure 14. Set Linker option in Keil	19
Figure 15. Modify scatter in Keil	20
Figure 16. Modify SLIB RAM address in KEIL	20
Figure 17. Modify SLIB constant address in KEIL	20
Figure 18. SLIB address definition in icf	21
Figure 19. Address assignment in icf file	21
Figure 20. Modify IP-Code RAM in icf file	21
Figure 21. Modify SLIB constant address in IAR	22
Figure 22. Configure ICP Programmer	22
Figure 23. Set parameters in Download Form	23
Figure 24. Project_L0 execution process	24
Figure 25. Set Misc controls in Keil	25
Figure 26. Contents of modified fir_filter_symbol.txt	25
Figure 27. Set Build Actions in IAR	26
Figure 28. Edit steering_file.txt content	26
Figure 29. Modified scatter file	27
Figure 30. Modified icf file	27
Figure 31. Add symbol definition file in Keil	28
Figure 32. Modify symbol definition file type to "Object file"	28
Figure 33. Add symbol definition file in IAR	28
Figure 34. Project_L1 execution process	29

Figure 35. Enter Show Disassembly at Address	30
Figure 36. Set Show Code at Address	30
Figure 37. View codes	30
Figure 38. View codes in Memory	31
Figure 39. View SLIB_DATA start sector in Memory	31
Figure 40. SLIB write test	31
Figure 41. Write protection error interrupt	32
Figure 42. Save SLIB codes	33
Figure 43. Generate bin file of SLIB code	34
Figure 44. Online programming to MCU in ICP	34
Figure 45. Offline programming to MCU via AT-Link	35
Figure 46. End users program codes to MCU	36
Figure 47. Set offline project	37
Figure 48. Add project file	38

1 Overview

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 MCUs are designed with a security library (sLib) to protect important IP-Codes against being changed or read by the end user's program.

This application note details the sLib application principle and operation methods of AT32F403A/407 MCUs.

2 Application principles

2.1 Application principle of sLib

- Security library is a defined area protected by a code in the main memory, so that solution providers can program core algorithm into this area, and the rest of the area can be used for secondary development by end customers.
- Security library includes 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 (can only be executed) by MCU through I-Code bus 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.

The mapping of main Flash memory featured with sLib is shown in Figure 1. The program codes in security library can be easily called and executed by end users, but cannot be read directly.

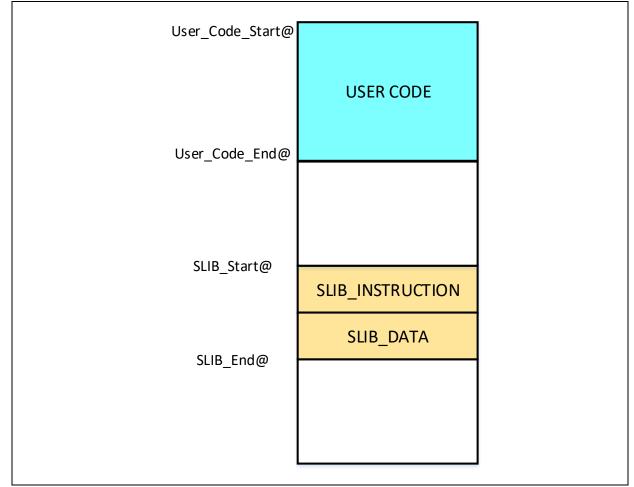


Figure 1. Mapping of main Flash memory featured with sLib

The range of sLib is set by sector, and the size of each sector is subject to the specific MCUs. Table 1 lists the main Flash size, sector size and configurable range of AT32F403A/407 series MCUs.

Table 1. Flash size of AT32F403A/4	07
------------------------------------	----

Part number	Internal Flash size(Byte)	Sector size (Byte)	Configurable range
AT32F403AxC AT32F407xC	256K	2К	Sector 2 ~ 63 (0x08001000 ~ 0x0801FFFF)
AT32F403AxE AT32F407xE	512K	2К	Sector 2 ~ 63 (0x08001000 ~ 0x0801FFFF)
AT32F403AxG AT32F407xG	1024K	2К	Sector 2 ~ 63 (0x08001000 ~ 0x0801FFFF)

2.2 How to enable sLib protection

By default, security library setting register is unreadable and write protected. To enable write access to this register, security library should be unlocked first. Write 0xA35F6D24 to the SLIB_UNLOCK register, and check the SLIB_ULKF bit in the SLIB_MISC_STS register to verify if it is unlocked successfully. Then, set values can be written into the security library setting register.

The steps to enable security library are as follows:

- Check the OBF bit in the FLASH_STS register to confirm that there is no other ongoing programming operation;
- Write 0xA35F6D24 to the SLIB_UNLOCK register to unlock security library;
- Check the SLIB_ULKF bit in the SLIB_MISC_STS register to verify that it is unlocked successfully;
- Set the sectors to be protected in the SLIB_SET_RANGE register, including the address of instruction and data areas;
- Wait until the OBF bit becomes "0";
- Set a security library password in the SLIB_SET_PWD register;
- Wait until the OBF bit becomes "0";
- Program the code to be saved in security library;
- Perform system reset, and then reload security library setting word;
- Read the SLIB_STS0/STS1 register to verify the security library settings.

Notes:

- It is allowed to set security library in the main Flash memory only, and refer to <u>Table 1</u> for the configurable range;
- The security library code must be programmed by sectors, with its start address aligned with the main memory address;
- The interrupt vector table is in data type and usually placed in the first sector (sector0) of the main Flash memory, which should not be configured as security library;
- Codes to be protected by the security library should not be placed in the first 4 KB of main Flash memory.

For details of security library setting register, please refer to AT32F403A/407 Series Reference Manual.

The program to start security library can be found in the slib_enable() function in main.c file of project_I0. In addition, users can use Artery ICP/ISP tools for configuration.

2.3 How to disable sLib protection

The security library protection can be disabled by writing the previously defined password to the SLIB_PWD_CLR register. While disabling security library protection, MCU will perform mass erase operation to the main Flash memory (including the contents of security library).

The steps to disable main Flash security library are as follows:

- Check the OBF bit in the FLASH_STS register to ensure that there is no other ongoing programming operation;
- Write the previously defined password to the SLIB_PWD_CLR register;
- Perform system reset, and then reload security library setting word;
- Read the SLIB_STS0 register to check the security library settings.

2.4 Compile and execute program in sLib

As aforementioned, program codes in the instruction security library (SLIB_INSTRUCTION) can be fetched by MCU via I-Code bus but cannot be read via D-Code bus in the manner of reading data, which means that program codes in SLIB_INSTRUCTION cannot read the data saved in the same SLIB_INSTRUCTION. For example, literal pool, branch table or constant compiled from C program code in the SLIB_INSTRUCTION cannot be read via D-Code bus.

Only instructions rather than data can be placed in the instruction security library. Therefore, when compiling program codes to be placed in the instruction security library, the user must configure the compiler to generate execute-only codes to avoid generating the above mentioned data.

Figure 2 and Figure 3 show the examples of literal pool and branch table.

The "switch()" is a jump instruction in C program, and the "sclk_source" variable is used to read the CRM_CFG register. As shown in Figure 2, the compiled assembly code "LDR R7, [PC, #288]" obtains the address of CRM_CFG register in a PC (program counter) indirect addressing manner, and the address of CRM_CFG register is saved as a constant in the adjacent instruction area (within the instruction security library); therefore, the data is read when the switch() instruction is executed. An error will occur during execution if there is such program code in the instruction security library.

The example program in Section 3 introduces how to configure compiler settings to avoid error.

0x0	8004798	2600	MOVS	r6,#0x00
		sclk_sour	ce = (crm_s	sclk_type)CRM->cfg_bit.sclksts;
-No-	80: 800479A	4520		
1 V	800479A 800479C		LDR LDR	r7,[pc,#228] ; @0x08004880 r7,[r7,#0x04]
		F3C70381		r3, r7, #2, #2
			lk source)	
	82:	(_	
	83:	case CR	M_SCLK_HICH	Χ:
•				
	main.c	📄 startup_	at32f403a_407.s	at32f403a_407_clock.c
	77			
	78	/* ge	et sclk	source */
\triangleright	79			= (crm sclk type)CRM->cfg bit.sclksts;
	80			(orm_born_t, pp) orall / org_bret bornots,
	81	owite	h(sclk	source)
	82 🖻	SWILL	II (SCIK_	Source)
		l	CDM C	OLK HICK.
	83			CLK_HICK:
	84	1		->misc3_bit.hick_to_sclk) != RESET) && ((CRM->misc1_bit.hickdiv
	85		•	_core_clock = HICK_VALUE * 6;
	86	e	else	
	87		system	_core_clock = HICK_VALUE;
	88	ł	preak:	

Figure 2. Literal pool example (1)

137. 3	sascem_cor	e_crock - syst	em_core_clock >> div_value;
x0800486E	4F06	LDR	r7,[pc,#24] ; @0x08004888
x08004870	683F	LDR	r7,[r7,#0x00]
x08004872	40F7	LSRS	r7,r7,r6
x08004874	F8DFC010	LDR.W	r12,[pc,#16] ; @0x08004888
x08004878	F8CC7000	STR	r7,[r12,#0x00]
138: }			
0x0800487C	BDF0	POP	{r4-r7,pc}
0x0800487E	0000	DCW	0x0000
0x08004880	1000	DCW	0x1000
x08004882	4002	DCW	0x4002

Figure 3. Literal pool example (2)

2.4.1 Setting interrupt vector table as security library not allowed

The interrupt vector table contains entry point address of each interrupt handler, which is read by MCU via D-Code bus. Generally, the interrupt vector table is located in the first sector (sector0, starting address: 0x08000000). Therefore, the following rules must be followed when setting the instruction security library:

- Do not configure the first sector of the main Flash as security library;
- Program codes to be protected by security library should not be placed in the first sector.

2.4.2 Correlation between sLib area and user code area

Program code (IP-code) protected by sLib area can call functions from the function library located in user code area (outside the sLib area). In this case, these function addresses are contained in the IP-Code, allowing PC (program counter) to jump to these functions when IP-Code is executed. Once the sLib area is enabled, function address cannot be changed. At this point, addresses of functions in the user code area must be fixed; otherwise, PC will jump to a wrong address and not work properly. Therefore, when configuring the sLib area, all functions related to IP-Code should be compiled into the sLib area. Figure 4 gives an example of the protected Function_A() being called to Function_B() in the user code area.



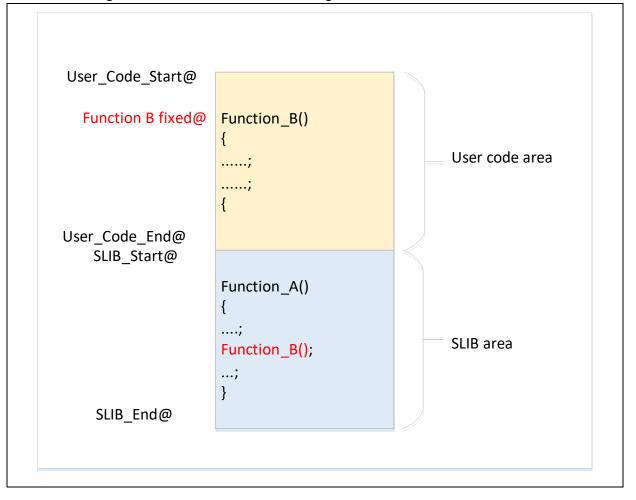


Figure 4. Function in sLib area calling the function in user code area

In addition, the standard function library of C programming language is commonly used, such as memset() and memcpy() functions. If both IP-Code and user area code call such functions, the above mentioned error may occur. The two solutions are recommended:

- 1) Compile into the sLib area (refer to Keil or IAR documents for details about implementation).
- 2) Do not use the standard function library of C programming language in IP-Code. If it is necessary to use in IP-Code, functions to be used must be renamed. Figure 5 shows an example of writing the my_memset() function to replace the original memset() in IP-Code.



Figure 5. Example of user-defined function

```
void* my_memset(void *s, int c, size_t n);
void arm_fir_init_f32(
    arm_fir_instance_f32 * S,
    uint16_t numTaps,
    float32_t * pCoeffs,
    float32_t * pState,
    uint32_t blockSize)
{ /* Assign filter taps */
    S->numTaps = numTaps;
    /* Assign coefficient pointer */
    S->pCoeffs = pCoeffs;
    /* Clear state buffer and the size of state buffer is (blockSize + numTaps = 1) */
    my_memset(pState, 0, (numTaps + (blockSize - lu)) * sizeof(float32_t));
    /* Assign state pointer */
    S->pState = pState;
    }
    void* my_memset(void *s, int c, size_t n)
    {
        while (n>0)
           *( (char*)s + n-- -1 ) = (char)c;
           return (s);
    }
```

3 Example applications of sLib

This section introduces example applications of sLib and how to complete these applications step by step.

The AT32F407 and AT32F403A series MCUs have the same SLIB feature. In this application note, the AT32F403A series is used for demonstration.

3.1 Example application requirements

3.1.1 Hardware requirements

- AT-START-F403A demo board with AT32F403AVGT7 chip
- AT-Link emulator for debugging example applications

3.1.2 Software requirements

- Keil® μvision IDE (μvision V5.18.0.0 is used in this example) or IAR Embedded workbench IDE (IAR V8.22.2 is used in this example)
- Artery ICP/ISP programming tools for enabling or disabling sLib

3.2 Overview

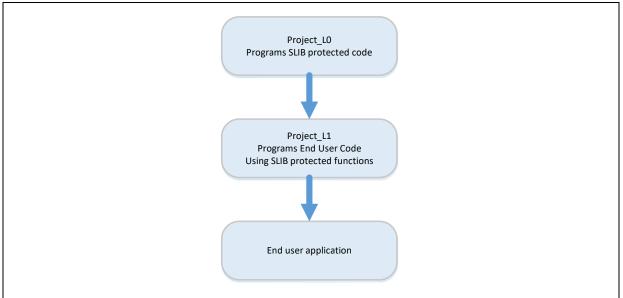
This application note provides two example projects to demonstrate that software developers develop IP-Code for end-user applications.

- Project_L0: Solution provider develops algorithm and compiles to sLib
- Project_L1: Apply algorithm for end users

The algorithm completed in Project_L0 will be pre-downloaded and pre-burned to AT32F403A chip and configured as sLib protected. In addition, the following settings are available for the end-user applications.

- Main Flash memory mapping, showing the area occupied by sLib and the area where users can develop programs
- Header file that contains algorithm function definitions, allowing end users to call relevant functions;
- Symbol definition file, which contains the actual address of each IP-Code function, so that functions can be called properly by the end-user application.

Figure 6. Example application process



Software solution providers can refer to the Project_L0 to develop algorithm code and refer to Project_L1 for end-user application.

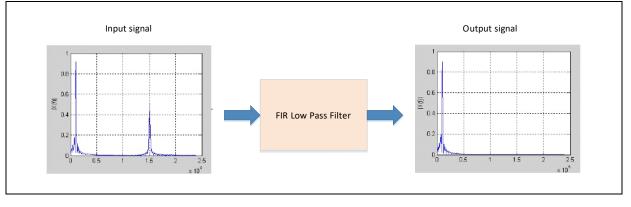
Figure 7. Example application

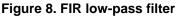


3.3 SLIB protected code: FIR low-pass filter

This example uses FIR low-pass filter algorithm provided by CMSIS-DSP library as the sLib protected IP-Code. For details about FIR low-pass filter algorithm, refer to CMSIS-DSP relevant documents. This application note mainly introduces how to configure sLib to protect this algorithm and how it is called by the end-user program code.

The low-pass filter input signal in this example is a combination of two sine waves with frequencies of 1 KHz and 15 KHz, while the low-pass filter cut-off frequency is about 6 KHz. A 15 KHz signal is filtered through the low-pass filter and outputs 1 KHz sine wave. Figure 8 shows the FIR low-pass filter functions.





CMSIS DSP library functions and files to be used are:

arm_fir_init_f32()

It is used for initialization of filter function, which is included in "arm_fir_init_f32.c" file.

• arm_fir_f32()

It is the main part of filter algorithm, which is included in "arm_fir_f32.c" file.

• FIR_lowpass_filter()

It is a FIR low-pass filter global function written by using the above two functions for the end user to call, which is included in "fir_filter.c" file.

• fir_coefficient.c

This C file contains coefficients (read-only constants) used by FIR filter functions, and these coefficients are placed in data security library in the example.

In this example, FPU and DSP instructions in the MCU are used for signal processing and floating point arithmetic to realize accurate calculation and correct output signal.

3.4 **Project_L0: example for solution providers**

The following projects are completed in this example:

- Compile the algorithm-related functions to execute-only code;
- Place the algorithm program code to the main Flash memory sector 2;
- Place the coefficients of filter functions to the main Flash memory sector 4;
- Execute FIR_lowpass_filter() in the main program to verify its correctness;
- If correct, set sector 2/3 as the instruction security library and sector 4/5 as the data security library, which can be completed by calling the *slib_enable()* function in the main program or using Artery ICP Programmer (it is recommended to use ICP tools to complete configurations);
- Generate the header file and symbol definition files that are used by end-user program to call low-pass filter functions.

3.4.1 Generate execute-only code

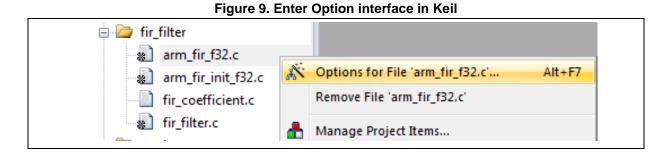
Each toolchain has specific setting options to prevent the compiler generating literal pools and branch table that can read data while executing instructions, such as "LDR Rn, [PC, #offset]". Section <u>2.4</u> lists examples of literal pools and branch table.

For Keil® µvision, it has the Execute-only Code option, which can be set as follows:

Keil® µvision: Set Execute-only Code option

Operate as follows:

- Select C file group or individual C file (in this example, the C files to be protected are placed in "fir_filter");
- Right click and select the corresponding files (for example, the *Option for File 'arm_fir_f32.c'*), as shown in Figure 9;



 Tick "Execute-only Code" in the C/C++ interface, and the "--execute_only" instruction is added to the compiler control string, as shown in Figure 10;

Figure 10. Select Execute-only Code in Keil

Options for File 'arm_fir_f32.c'			
Properties C/C++			
Preprocessor Symbols Define: Undefine:			
Language / Code Generation ✓ Execute-only Code Optimization: <default> ✓ Optimize for Time ✓ Optimize for Time ✓ Split Load and Store Multiple ✓ One <u>E</u>LF Section per Function</default>	Image: Strict ANSI C Image: Enum Container always int Image: Plain Char is Signed Image: Read-Only Position Independent Image: Read-Write Position Independent	Warnings: All Warnings	
Include Paths Misc Controls Compiler -execute_onlycpu Contex control \Ubraries\drivers\inc -1\	<-M4fp -DMICROLIB -g -O0 -apcs=interwo \ibiraries∖cmsis∖cm4\core_support -l\	k -split_sections -1	
OK	Cancel Defaults	Help	

• The *arm_fir_f32.c*, *arm_fir_init_f32.c* and *fir_filter.c* files are in the SLIB_INSTRUCTION area, and these files need to be set as generating execute-only code.

IAR: Set No data read in code memory option

Operate as follows:

• Select the corresponding file in the fir_filter group; right click and select Option;

Figure 11. Enter Option interface in IAR

├₽ 🛑 fir_filter	
arm_fir_f32.c	Options
arm_fir_init_f32.d	Options
⊢⊞ ⊚ fir_coefficient.c └─⊞ ⊚ fir_filter.c	Make

• Enter "C/C++" interface and tick "Override inherited settings" and "No data read in code memory", as shown in Figure 12;



Options for node "arm_fir_	f32.c*
Exclude from build	
Category:	Override inherited settings Factory Settings
Static Analysis Runtime Checking	
C/C++ Compiler Custom Build	Preprocessor Diagnostics MISRA-C:2004
	MISRA-C:1998 Encodings Extra Options Language 1 Language 2 Code Optimizations Output List
	Processor mode Arm Thumb Position-independence Code and read-only data (ropi) Read/write data (rwpi) No dynamic read/write initializati. No data reads in code memory OK Cancel

Figure 12. Set C/C++ options in IAR

• The *arm_fir_f32.c*, *arm_fir_init_f32.c* and *fir_filter.c* files are in the SLIB_INSTRUCTION area, and these files need to be set as generating execute-only code.

3.4.2 Compile security library address

As aforementioned, the first sector (sector0) of the main Flash memory is used to store interrupt vector table. Therefore, the security library is set from sector 2 in this example, with sector 2 and sector 3 being set as instruction security library, and sector 4 and sector 5 being set as data security library. Figure 13 shows the main Flash memory mapping and RAM partition. The main purpose of RAM partitioning is to avoid the same RAM being used by sLib-protected code and end-user code.

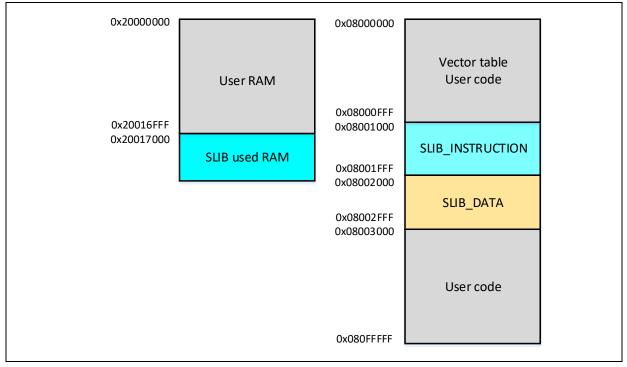


Figure 13. Main Flash memory mapping and RAM partition

Keil® µvision: scatter file

Operate as follows:

• Click Project → Optios for Target→Linker, untick "Use memory layout from Target Dialog" and click "Edit" to open and modify *slib-w-xo.sct* file, as shown in Figure 14:

Figure 14. Set Linker option in Keil

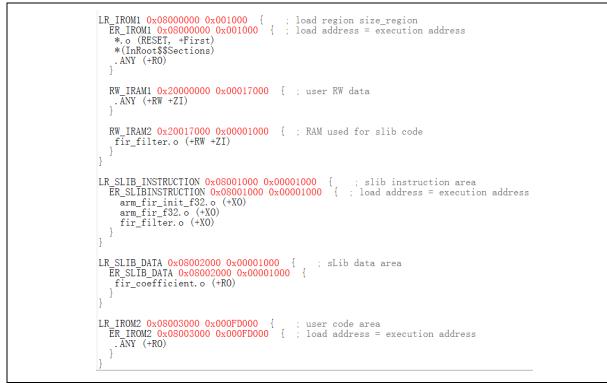
Options for Target 'at_start_f403a'	x
Device Target Output Listing User C/C++ Asm Linker Debug	Vtilities
Use Memory Layout from Target Dialog X/O Base:	
Make RW Sections Position Independent R/O Base: Dx0800 Make RO Sections Position Independent Dx080 Dx0800 Dx0800	
Don't Search Standard Libraries	0000
Report 'might fail' Conditions as Errors disable Warnings:	
Scatter \slib-w-xo.sct	Edit
Misc controls ──symdefs=fir_filter_symbol.txt	* *
Linker control -library_type=microlib -strict -scatter ".\slib-w-xo.sct" string	•
OK Cancel Defaults	Help

Open scatter file, load the object file of the code to be placed in SLIB_INSTRUCTION area to "LR_SLIB_INSTRUCTION" (a dedicated loading area that starts from sector 2 and occupies two sectors), and modify the label to "execute-only (+XO)". In addition, place the area occupied by SLIB_Data to a dedicated loading area named "LR_SLIB_DATA" to avoid the compiler compiling other non-IP-code functions to the SLIB area. The RW_IRAM2 assigns the region from 0x20017000 to 0x20017FFF to the algorithm functions to avoid the same RAM region



being used by end-user project, causing fault or error in program execution.

Figure 15. Modify scatter in Keil



 In addition to modifying the scatter file, for the RAM used by IP-Code, users can also use the Keil "__attribute__((at(address)))" descriptor to load variables to 0x20017000, as shown in Figure 16.

Figure 16. Modify SLIB RAM address in KEIL

```
61 =#if defined ( __ICCARM__)
62 static float32_t firStateF32[BLOCK_SIZE + NUM_TAPS - 1] @ 0x20017000 ;
63 #elif defined ( __CC_ARM )
64 static float32_t firStateF32[BLOCK_SIZE + NUM_TAPS - 1]
__attribute__((at(0x20017000)));
65 #endif
```

The start address of data security library is sector 4 (0x08002000). To compile the constants used by FIR low-pass filter functions to this address, users can modify the scatter file as aforementioned, or use the Keil "__attribute__((at(address)))" descriptor to load the constants to a fixed address, as shown in Figure 17.

Figure 17. Modify SLIB constant address in KEIL

```
57 ₽#if
              defined (
                                ICCARM
58 const float32_t firCoeffs32[NUM_TAPS] @ 0x08002000 ={
59 | #elif defined ( __CC_ARM )
60 const float32_t firCoeffs32[NUM_TAPS]
                                                                _attribute__((at(0x08002000)))
                                                                                                             = {
61 -#endif
        -0.0018225230f, -0.0015879294f, +0.0000000000f, +0.0036977508f, +0.0080754303f, +0.00
-0.0341458607f, -0.0333591565f, +0.0000000000f, +0.0676308395f, +0.1522061835f, +0.22
+0.1522061835f, +0.0676308395f, +0.0000000000f, -0.0333591565f, -0.0341458607f, -0.01
62
63
64
        +0.0080754303f, +0.0036977508f, +0.0000000000f, -0.0015879294f, -0.0018225230f
65
66
     };
```

IAR: ICF file

Operate as follows:

 Open the icf file in \project_I0\IAR_V8.2\, and add three new SLIB loading areas as shown in Figure 18. The SLIB_RAM region reserves the RAM (0x20017000 ~ 0x20017FFF) for the algorithm functions.

Figure 18. SLIB address definition in icf

```
/* SLIB INSTRUCTION area */
define symbol __ICFEDIT_region_SLIB_INSTRUCTION_start__ = 0x08001000;
define symbol __ICFEDIT_region_SLIB_INSTRUCTION_end__ = 0x08001FFF;
/* SLIB DATA area */
define symbol __ICFEDIT_region_SLIB_DATA_start__ = 0x08002000;
define symbol __ICFEDIT_region_RAM_start__ = 0x20000000;
define symbol __ICFEDIT_region_RAM_end__ = 0x20017FFF;
/* SLIB RAM region */
define symbol __ICFEDIT_region_SLIB_RAM_start__ = 0x20017000;
define symbol __ICFEDIT_region_SLIB_RAM_end__ = 0x20017FFF;
```

 In the icf file, the area occupied by SLIB is reserved to avoid the compiler compiling other non-IP-code functions to the SLIB area, and the RAM region used by IP-Code is reserved.

```
Figure 19. Address assignment in icf file
```

• For the RAM used by IP-Code, users can use the IAR @ descriptor to load variables to a fixed address 0x20017000 or modify the icf file, as shown in Figure 20.

Figure 20. Modify IP-Code RAM in icf file

 The start address of data security library is sector 4 (0x08002000). To compile the constants used by FIR low-pass filter functions to this address, users can modify the icf file as aforementioned, or use the IAR @ descriptor to load the constants to a fixed address, as shown in Figure 21.

Figure 21. Modify SLIB constant address in IAR

```
#if defined ( __ICCARM__)
Const float32_t firCoeffs32[NUM_TAPS] @ 0x08002000 ={
    #elif defined ( __CC_ARM )
    Const float32_t firCoeffs32[NUM_TAPS] __attribute__((at(0x08002000))) = {
    #endif
    -0.0018225230f, -0.0015879294f, +0.0000000000f, +0.0036977508f, +0.0080754303f,
    -0.0341458607f, -0.0333591565f, +0.0000000000f, +0.0676308395f, +0.1522061835f,
    +0.1522061835f, +0.0676308395f, +0.000000000f, -0.0333591565f, -0.0341458607f,
    +0.0080754303f, +0.0036977508f, +0.000000000f, -0.0015879294f, -0.0018225230f
    -};
```

3.4.3 Enable sLib protection

There are two methods to enable sLib protection:

(1) Use Artery ICP Programmer (recommended)

It is recommended to use the ICP Programmer as follows:

- Connect AT-Link emulator to AT-START-F403A board and power on;
- Open ICP Programmer, select AT-Link for connection, and add the HEX or BIN file generated by Project_L0, as shown in Figure 22.

Artery IC		<u> </u>															_	
File J-	Link s	etting	s A	T-Lir	nk set	ttings	s Ta	arget	t L	angu	age	He	lp					
Disconn	ect	Part N	luml	ber:	AT3	2F40	3AV	ST 7	F	lashS	ize:	1024	КВ		. 1	2	Ξ	7Y
		AT-Li				V1.5.												
AT-Link	•	AT-Li	NK SN	1: 8	D109	C110	08045	4105	1780	02				,	猚	1	守	力
		SPI	м		FL	ASH_	DA Ox	c 🗌	0		@ Re	emap	0 (Us	e PA1	L1/PA	12 pi	ns)	
	-	Туре							Select		C Re	emap	1 (Us	e PB1	LO/PB	11 pi	ns)	
Memory	read	settin	as															
Address	_						0x 00	0000	000)ata b	. [0 _:+-		1		Read	d
Address	UX U	80000	00	K	ead s	size		0003	scc		ata b	oits [o Dits	•	ļ		Near	<u> </u>
-File info																		
No.	File na	ame				File	size	Ad	dres	s ran	ge(0x)					A	dd
1	at32f4	103a_p	roject	_l0.he	x	923	2	08	0000	80-00	30003	CB,08	0010	00-08	30011	17,08	De	elete
•																•		
									Flas	h CRO	c	File	+ CRC	verif	fy	Do	wnLo	oad
Flash info	File	at32f4	03a i	oroie	t 10.H	nex												
Address				-			dress r	ange:	[0x08	00100	0 0x08	00111	7] A	ddres	s rang	ge:[0x	08002	000
0x080020		Address	rance	30x01:	00300	0 0x0	8004EE	371 (check	sum: (0x000E	06B92						
Address 0x080000	00	0 40	1 1B	2	3 20	4	5 30	6 00	7	8 1B	9 36	A 00	B 08	C 13	D 36	E 00	F 08	A: ^ @(
0x080000		17	36	00	20	F3	35	00	08	1D 99	36	00	08	00	00	00	00	
0,000000	10	1/	30		00	F.J			00	33			00					
0x080000	20	00	00	00	00	00	00	00	00	00	00	00	00	1F	36	00	08	

Figure 22. Configure ICP Programmer

 Click "Download" and the "Download Form" pops up, which shows SLIB status and relevant parameters. Set sector 2 as the start sector, sector 4 as the data start sector and sector 5 as the end sector; set the enable password as "0x55665566" (user-defined), tick "Enable SLIB", and then click "Start Download" to complete programming and enable SLIB, as shown in Figure 23.

sLib status: Disable	Remaining usage t	r
Enable password 0x 55665566	sLib position:	Main Flash
Disable password 0x	Start sector	Sector 20x08001000
Disable sLib	DATA start sector	Sector 40x08002000
	End sector	Sector 50x08002800
Extra options		
Erase the sectors of file size	▼ Dis	able sLib before download
Verify Custom encryption key	y for verify: 👿 Ena	able sLib
	Dis	able FAP before download
Jump to the user program	Ena	able FAP after download
Write software serial number(SN)		
Write address 0x 08010000	🔲 But	ton free mode
Current SN 0x 00000001		
Increase step 0x 00000001		
🔲 Write user system data		
User system data file path		

Figure 23. Set parameters in Download Form

For details about ICP Programmer, refer to ICP Programmer User Manual.

(2) Use slib_enable() function in main.c

After the slib_enable() function is verified correct by low-pass filter function and then executed, the sLib protection can be enabled. To execute this function, enable the "#define USE_SLIB_FUNCTION" in main.c.

3.4.4 Project_L0 execution process

In this example, FIR low-pass filter calculates the input signal (testInput_f32_1kHz_15kHz) mixed with 1 KHz and 15 KHz sine waves, and the output 1 KHz sine wave data is saved in testOutput, which will be compared with the data calculated by MATLAB and saved in refOutput. If the error value is smaller than expected (SNR larger than the preset threshold), the green LED on the board blinks; otherwise, the red LED blinks. Figure 24 shows the Project_L0 execution process.

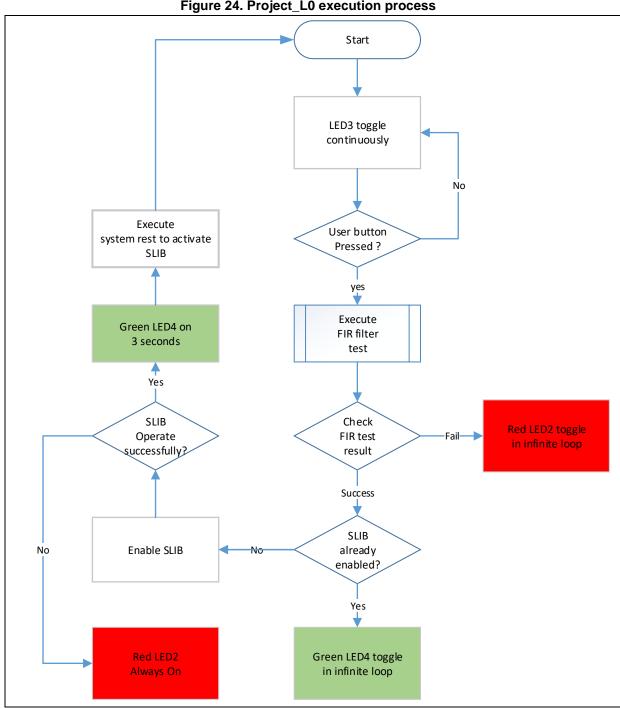


Figure 24. Project_L0 execution process

Go through the following steps to execute this example program:

- (1) Use Keil® µvision to open the Project_L0 under \utilities\at32f403a_407_slib_demo\project_I0\mdk_v5\, and then compile;
- (2) Before downloading the code, check whether the chip on AT-START-F403A board is SLIBprotected or write/read-protected (FAP/EPP). If it is protected, use ICP programmer to disable protection and then download the code;
- (3) After successful download, start to execute the code, and the on-board LED3 keeps blinking rapidly;
- (4) Press the on-board USER button to perform operation of low-pass filter;
- (5) Compare the computation result. If it is correct, the green LED4 keeps blinking; otherwise, the

red LED2 keeps blinking;

(6) After obtaining the correct result, if the USE_SLIB_FUNCTION in main.c is defined and the SLIB is not enabled, the slib_enable() function will be executed to set SLIB. If SLIB setting fails, the red LED2 will be always ON; if SLIB setting succeeds, the green LED4 will be ON for about three seconds and then perform system reset to enable SLIB; then, go to step (3).

3.4.5 Generate header file and symbol definition file

The header file and symbol definition file are used when the Project_L1 calls FIR low-pass filter functions, which is the *fir_filter.h* file in main.c in this example.

The generation of symbol definition file is related to the specific toolchain being used.

Use Keil® µvision to generate symbol definition file

Operate as follows:

- Enter Options for Target → Linker interface;
- Add "--symdefs=fir_filter_symbol.txt" command in the Misc controls, as shown in Figure 25;

Figure 25. Set Misc controls in Keil

Options for Target 'at_start_f403a'
Device Target Output Listing User C/C++ Asm Linker Debug Utilities
□ Use Memory Layout from Target Dialog X/O Base: □ Make RW Sections Position Independent R/O Base: □ Make RO Sections Position Independent R/W Base □ Don't Search Standard Libraries 0x20000000 □ Report 'might fail' Conditions as Errors disable Warnings:
Scatter .\slib-w-xo.sct Edit
Miscsymdefs=fir_filter_symbol.txt
Linker control string
OK Cancel Defaults Help

- After compiling the project, a symbol definition file named "fir_filter_symbol.txt" is generated under project_I0\mdk_v5\Objects;
- This symbol definition file contains all symbol definitions of the project, and it needs to be modified to only remain the definitions of low-pass filter functions to be called by end users. The modified *fir_filter_symbol.txt* is shown in Figure 26;

Figure 26. Contents of modified fir_filter_symbol.txt

0x08001001 T FIR_lowpass_filter

Use IAR to generate symbol definition file

Operate as follows:

● Select Project→Option→Build Actions

Options for node "project, Category: General Options Static Analysis Runtime Checking C/C++ Compiler Assembler Output Converter	Do*
Custom Build Build Actions Linker Debugger Simulator CADI CMSIS DAP GDB Server I.jet/JTAGjet J-Link/J-Trace TI Stellaris NU-Link	Fost build command line: \$TOOLKIT_DIR\$\bin\isymexport.exeedit "\$PROJ_DIR\$\st.
Nu-Link PE micro ST-LINK Third-Party Driver TI MSP-FET TI XDS	OK Cancel

Figure 27. Set Build Actions in IAR

- Input the following commands to the Post-build command line: \$TOOLKIT_DIR\$\bin\isymexport.exe --edit "\$PROJ_DIR\$\steering_file.txt" "\$TARGET_PATH\$" "\$PROJ_DIR\$\fir_filter_symbol.o"
- The *fir_filter_symbol.o* is the symbol definition file to be generated, and the *steering_file.txt* is saved under project_I0\iar_v8.2, which is used to select function symbols to be generated. Users can manually edit according to the contents called by sLib. As shown in Figure 28, the "show" is the command used to select functions.

show FIR_lowpass_filter

3.5 **Project_L1: example for end users**

Project_L1 uses the FIR low-pass filter function that is debugged in Project_L0, programmed to AT32F403A MCU main Flash memory and SLIB-protected According to the header file, symbol definition file and the main Flash memory mapping of Project_L0, end users can complete the followings for Project_L1:

- Create an application project;
- Add the header file and symbol definition file provided by Project_L0 to the project;
- Call the FIR low-pass filter function;

• Develop and debug user's program.

Notes:

Project_L1 must use the same toolchain and the same version of the compiler as that of Project_L0; otherwise, incompatibility problem may occur and the code provided by Project_L0 cannot be used properly. For example, Project_L0 uses Keil® µvision V5.18.0.0; therefore, Project_L1 need to use the same version.

3.5.1 Create user application project

The security library enabled in Project_L0 occupies some specific main Flash memory sectors; therefore, the address for Project_L1 code storage should be compiled according to the main Flash memory mapping of Project_L0, as shown in Figure 13. In the main Flash memory, sector 2~ sector 5 are occupied by the security library, and end users need to isolate this region by using linker control file to avoid code being compiled to this region.

Keil® µvision: scatter file

Refer to the *end_user_code.sct* file under project_I1\mdk_v5\ and divide the main Flash memory into two regions, and the middle part is the SLIB-protected area. In addition, the area after 0x20017000 needs to be reserved for the RAM area, as shown in Figure 29.

Figure 29. Modified scatter file

```
{
LR_IROM1 0x08000000 0x00001000
                                load region size_region
  ER_IROM1 0x08000000 0x00001000 {
                               ; load address = execution address
  *.o (RESET, +First)
  *(InRoot$$Sections)
   .ANY (+RO)
  RW IRAM1 0x20000000 0x00017000 { : RW data
   .ANY (+RW +ZI)
  0x20017000 ~ 0x20017FFF
                       RAM reserved for SLIB code
}
; 0x08001000 ~ 0x08002FFF is SLIB area
. ANY (+RO)
  }
}
```

IAR: ICF file

Refer to the *enduser.icf* file under project_I1\iar_V8.2\, as shown in Figure 30.

Figure 30. Modified icf file

<pre>/* Reserved SLIB area */ define region ROM_region</pre>	<pre>= mem:[fromICFEDIT_region_ROM_start toICFEDIT_region_ROM_end] -mem:[fromICFEDIT_region_SLIB_start toICFEDIT_region_SLIB_end];</pre>
define region RAM_region	<pre>= mem:[fromICFEDIT_region_RAM_start toICFEDIT_region_RAM_end] - mem:[fromICFEDIT_region_SLIB_RAM_start toICFEDIT_region_SLIB_RAM_end];</pre>

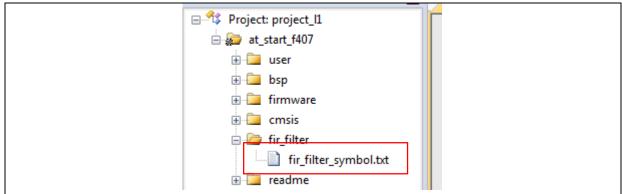
3.5.2 Add symbol definition file to project

The symbol definition *file fir_filter_symbol.txt* generated in Project_L0 must be added to Project_L1, so that it can be correctly compiled and linked to the SLIB-protected area code.

Add symbol definition file in Keil® µvision

Add *fir_filter_symbol.txt* to the project, as shown in Figure 31.

Figure 31. Add symbol definition file in Keil



After adding this file to fir_filter, modify its file type from "text" to "Object", as shown in Figure 32.

Options for File 'fir_filter_symbol.txt'	X	
Properties		
Path- Nir filter symbol bt		
File Type: Object file	☑ Include in Target Build	
Size: 217 Bytes	🗹 Always Build	
last change: Fri May 21 11:14:16 2021	🔽 Generate Assembler SRC File	
	Assemble SRC File	
Stop on Exit Code: Not specified	☑ Image File Compression	
Custom Arguments:		
Memory Assignment:		
Code / Const: ">		
Zero Initialized Data: <a> 		
Other Data:		
OK Cancel D	efaults Help	

Figure 32. Modify symbol definition file type to "Object file"

Add symbol definition file in IAR

Add the *fir_filter_symbol.o* (Object) file to fir_filter, as shown in Figure 33.

Figure 33. Add symbol definition file in IAR

riles	¥	-
🖂 🌒 project_l1 - at_start_f	~	
⊢⊕ 🖬 bsp		•
⊢⊞ 🛋 cmsis		•
├₽ <mark>,=fir_filter</mark>		
🛛 🖵 🗋 fir_filter_symbol.o		
⊡ firmware		•
- 🕀 💼 readme		
⊢⊞ 📠 user		•
🖵 🖬 🛑 Output		

3.5.3 Call functions in SLIB-protected area

When the filter.h header file is referred in main.c and the symbol definition file is added to the project, the low-pass filter function in the protection area can be called, as shown below:

FIR_lowpass_filter(inputF32, outputF32, TEST_LENGTH_SAMPLES);

Where,

- *inputF32*: pointer containing input signal data table;
- *outputF32*: pointer storing output signal data table;
- TEST_LENGTH_SAMPLES: the number of signal samples to be processed.

3.5.4 Project_L1 execution process

Figure 34 shows the execution process of Project_L1:

- Start execution, and LED3 will keep blinking;
- Press the USER button on AT-START board, and the FIR_lowpass_filter() starts operation;
- If the result is correct, the green LED4 will keep blinking; otherwise, the red LED2 will keep blinking.

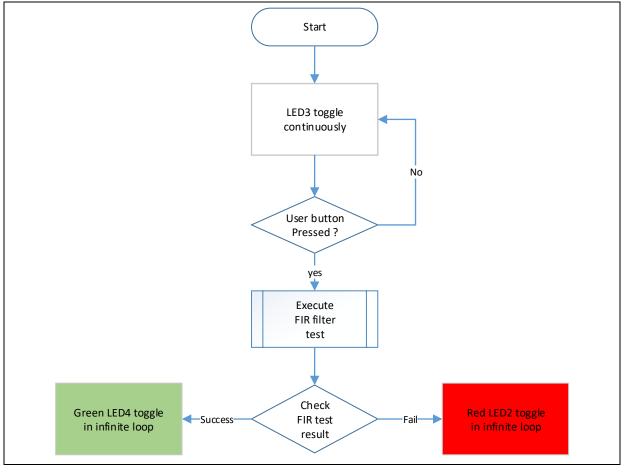


Figure 34. Project_L1 execution process

3.5.5 SLIB protection in debug mode

Development tools are used by end users to debug codes when developing applications. This section takes Keil® µvision as an example to introduce how to protect codes in the SLIB-protected

area from being read as data in debug mode.

- Open Project_L1 and compile;
- Click "Start/Stop Debug Session" to enter debug mode;
- Right click in the "Disassembly" interface and select "Show Disassembly at Address", as shown in Figure 35.

l	→ T T T T T → → //= //R → T T T T T T T T T T T T T T T T T T	V 🗰 💷 🔪 🥆	
	🔯 🖬 🚍 🛵 🚚 • 🔲 • 📴 • 📾 • 💷 • 💆 •		
×	Disassembly		д <mark>Е</mark>
<u> </u>	0x08003E52 4770 EX 1r 94: AT32_Board_Init(); 95:		^
- 1	96: /* Configure Flash to generate 🗸 Mixed Mode	error occur */	
- 1	0x08003E54 2804 CHP r0, #0 Assembly Mode		
	OXOBOO3E56 D106 ENE OXOBO 97: Enable Flash INT(); Address Range	•	
- 1	98:		
	99: Show Disassembly at Address	-	
	100: /* Wait for KEY button to be p Set Program Counter		
- 1	0x08003E58 490A LDR r1, [p */ } Run to Cursor line Ctrl+F1	0	
	0x08003E5A 6809 LDR r1,[r 101: while(AT32_BUTTON_State(BUTTON 102: (- 1
- 1	0x08003E5C F0510104 ORRS r1,r1 O Enable/Disable Breakpoint Ctrl+F	9	
	0x08003E60 4A08 LDR r2,[p		
	Ox08003E62 6011 STR r1, [r Insert Tracepoint at '0x08003E54'		
	104: Delay_ms(300); Enable/Disable Tracepoint		
	105:) 106: Inline Assembly		
	107: /* Turn Off LED3 */ Load Hex or Object file		
	0x08003E64 E005 B 0x080		
	Ox08003E66 4907 LDR r1, [p Instruction Trace	•	~
-	Execution Profiling	•	>
<u>•</u>	main.c		▼ ×
		2	• ^
	🕂 🖬 Call Stack 🗈 Copy Ctrl+	c	д 🛛
	b_V1.x.x\\Utilities\\AT32F403A_S Name Location/V Type		

Enter the address "0x08001000" of SLIB_INSTRUCTION start sector (sector 2);

Figure 36. Set Show Code at Address

Show Code at Address	×
Address: 0x08001000	Go To

• As shown in Figure 37, codes from 0x08001000 are all 0xFFFFFFF;

Figure 37. View codes

	• 💷 • 📓 • 🔀 •	
Disassembly		P 🗵
76: (^
77: uint32_t i;		
Ox08001000 FFFFFFFF DCD	OxFFFFFFF	
0x08001004 FFFFFFFF DCD		
	OxFFFFFFFF	
	locks = testlengthsamples/BLOCK_SIZE;	
80: arm_fir_instance_f3	2 5;	
81:		
82:		
	nction to initialize the instance structure	2. */
	OxFFFFFFFF	
84: arm_fir_init_f32(&S 85: /*	<pre>, NUM_TAPS, (float32_t *)&firCoeffs32[0], -</pre>	GfirStateF32[U], blockSize
	cess function for every blockSize samples	
87: **	cess function for every processive samples	*/
0x08001010 FFFFFFF DCD		,
0x08001014 FFFFFFF DCD		
	OxFFFFFFF	
0x0800101C FFFFFFFF DCD		
0x08001020 FFFFFFFF DCD	OxFFFFFFFF	
0x08001024 FFFFFFFF DCD	OxFFFFFFFF	
0x08001028 FFFFFFFF DCD	OxFFFFFFFF	
0x0800102C FFFFFFFF DCD	OxFFFFFFFF	
0x08001030 FFFFFFFF DCD	OxFFFFFFFF	
0x08001034 FFFFFFFF DCD	OxFFFFFFFF	~
· <		>
k		

 Similarly, enter address 0x08001000 in "Memory" window, and codes are all 0xFF, as shown in Figure 38;

Figure 38	. View	codes	in	Memory
-----------	--------	-------	----	--------

Address: 0x08001000 0x08001000: FF	Memory 1																			д	x
0x08001013: FF	Address: 0x080	0100)																		^
0×08001026: FF	0x08001000:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	
0x08001039: FF	0x08001013:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	
0x0800104C: FF	0x08001026:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	
0x0800105F: FF F	0x08001039:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	
0x08001072: FF	0x0800104C:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	
0x08001085: FF	0x0800105F:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	
OxO8001098: FF	0x08001072:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	
	0x08001085:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	
Call Stack + Locals Memory 1 Memory 2	nxn8nn1n98:	FF	ন ন_	न न	FF	नन	FF	FF	FF	ŦŦ	FF	FF	FF	FF	FF	FF	ŦŦ	FF	FF	FF	~
	🚰 Call Stack + L	ocals		Mer	nory	1	M	lemo	ry 2												

 In the "Memory" window, enter the address 0x08002000 of SLIB_DATA start sector (sector 4); this region is allowed to be read through D-Code bus, so original values can be found, as shown in Figure 39;

Figure 39. View SLIB	_DATA start	sector in	Memory
----------------------	-------------	-----------	--------

Memory 1														Ļ	x
Address: 0x08002000															^
0x08002000: B9 E1	EE BA 1	12 22	DO E	3A 00	00	00	00	F7	55	72	ЗB	CF	4E	04	
0x08002013: 3C 58	C2 OB 3	3C 00	00 0	00 80	9E	85	8E	ВC	88	DC	OB	BD	9C	АЗ	
0x08002026: 08 BD	00 00 0	00 00	OA 8	32 8A	ЗD	FO	DB	1B	3 E	5F	46	64	3 E	06	
0x08002039: 41 80	3E 5F 4	46 64	3E F	O DB	1B	ЗE	ΟA	82	8Å	ЗD	00	00	00	00	
0x0800204C: 9C A3	08 BD 8	38 DC	OB B	3D 9E	85	8E	BC	00	00	00	80	58	C2	OB	
0x0800205F: 3C CF	4E 04 3	3C F7	55 7	72 3B	00	00	00	00	12	22	DO	BA	В9	E1	
0x08002072: EE BA	FF FF H	FF FF	FF F	FF FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	
0x08002085: FF FF	FF FF H	FF FF	FF F	FF FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	
<u>0x08002098: FF FF</u>	म जज जज	<u> </u>	न नन	नन नन	नन	नन	नन	নন	नन	नन	नन	नन	नन	नन	Y
Call Stack + Locals	Memory 1	M	emory	2											

 In the "Memory" window, double click to modify the value of 0x08002000, and a warning pops up by setting EPPERR=1 in the FLASH_STS register, indicating the protection is enabled;

	Figure 40. SLI	B write test	
Pr	operty	Value	
	PSR	0x00000030	
	UNLOCK	0	
	USD_UNLOCK	0	Ì.
	STS	0x00000010	Ì.
	ODF		
	EPPERR		
	PRGMERR		
	OBF		

 In case of enable erase/program protection error interrupt, continuing execution will enter the interrupt program;



i igure i ii tinte preteenen en en interrapt
115 void FLASH_IRQHandler(void) 116 □ {
<pre>117 if(flash_flag_get(FLASH_EPPERR_FLAG)) 118 □ {</pre>
110 flash_flag_clear(FLASH_EPPERR_FLAG); 120 delay_ms(500); 121 } 122 }

Figure 41. Write protection error interrupt

4 Integrate codes and download

After codes of the solution provider and end user are well designed, download to the same MCU on the premise of guaranteeing code security. Project_L0 and Project_L1 are used to introduce two downloading methods for reference.

This operation involves offline downloading mode of AT-Link. For details, refer to operation manuals of ICP and AT-Link.

4.1 Program codes separately

Firstly, the solution provider programs SLIB codes to MCU; then, the end user programs application codes to MCU. The process is as follows:

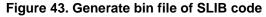
(1) Method A: The solution provider uses ICP tool to save the SLIB code in the compiled project as BIN or HEX file: download the complete project to MCU (do not configure SLIB and FAP), read the corresponding SLIB codes (0x08001000~0x08002FFF) by using the memory access function, and then click "File-Save Flash data as" to save the codes as BIN or HEX file. In this example, it is named "slib.bin", as shown in Figure 42.

	-Link setti	ngs	AT-Li	nk se	tting	s Ta	arget	L	angu	age	He	lp					
Sav	e file as			ATS	2F40)3AV0	5T7	F	ashS	Size:	1024	КВ		.1	זכ		7 Y
Sav	e flash da	ta as			V1.5												
Ma	ke encryp	tion file)			08045	41051	17BD	02					雅	1	等	力
Exit	:			FL	ASH	DA 0x		0		(@) R	emap	0 (Us			12 pir	_	
-	Тур	e						elect	_						11 pi		
Memor	y read set																
	s 0x 0800			head	cize.	0x 20	00) ata k	site (8 bits				Read	4
Addres	s 0x 0800	1000		leau	SIZE	0x 20	00				nis (o bits	, ,	ļ		Reat	<u> </u>
File inf	D																
No.	File name				File	size	Ad	dres	s ran	ge(0x	:)					A	dd
1	at32f403a	_projec	t_l0.he	×	923	2	080	0000	00-08	30003	CB,08	30010	00-00	30011	17,08	De	elete
										_							
				III											•		
•																	
•								Flas	h CR	с	Fil	e CRO	C veri	fy	Do	wnLo	oad
Flash inf	File:at3	2f403a	proje	ct_10.1	nex			Flas	h CR	с	File	e CR(C veri	fy	Do	wnLo	oad
Flash inf	File:at3			_		ecksum	: 0x00			c	File	e CRO	C veri	y	Do	wnLo	oad
Flash inf	range:[0x0			_		ecksum 5	: 0x00			9	Fil	e CRO	C veri	b D	Dov	F	oad
Flash inf Address	range:[0x0	8001000	0x0800	2FFF]	che		6)15062	2E								
Flash inf Address	; range:[0x0	8001000 D 1	0x0800	2FFF]	che 4	5	6 0F	015062	2E 8	9	A	В	С	D	E	F	A! ^
Flash inf Address Address 0x080010	range:[0x0	8001000 0 1 2D E9	0x0800 2 FF	2FFF] 3 47	che 4 06	5 46	6 0F 01	015062 7 46	2E 8 90	9 46	A 20	B 25	C 4F	DEA	E 58	F 19	A! ^ -?(
Flash inf Address Address 0x080010 0x080010	5 range:[0x0	8001000 0 1 2D E9 47 F2	0x0800 2 FF 00	2FFF] 3 47 03	Che 4 06 C2	5 46 F2	6 0F 01 00	015062 7 46 03	2E 8 90 42	9 46 F2	A 20 00	B 25 02	C 4F C0	D EA F6	E 58 00	F 19 02	A: ^ -?(G?

Figure 42. Save SLIB codes

Method B: The solution provider uses the compiled project to generate a bin file directly, and take the corresponding section in the SLIB area. For example, in the KEIL project, add "fromelf.exe

--bin --output .\Listings\@L.bin !L" in the "user" option to generate a bin file of the corresponding firmware, and add a suffix ".bin" to the SLIB area file. In this example, they are "ER_SLIB.bin" and "ER_SLIB_DATA.bin", corresponding to the SLIB-INSTRUCTION file (0x08001000) and SLIB-DATA file (0x08002000), respectively, as shown in Figure 43.



Command Items	User Command		Stop on Exi	S	
Before Compile C/C++ File					
Run #1		1	Not Specified		
🗌 🗌 Run #2		1	Not Specified		
Before Build/Rebuild					ER_IROM1
Run #1		2	Not Specified		
🗌 🗌 Run #2		1	Not Specified		ER_IROM2
After Build/Rebuild					
🔽 Run #1	fromelf.exebinoutput .\Listings\@L.bin !L	1	Not Specified		ER_SLIB_DATA
Run #2		2	Not Specified		

(2) Use ICP Programmer to program the ER_SLIB_INSTRUCTION.bin and ER_SLIB_DATA.bin to MCU, as shown in Figure 44.

Artery ICP Programmer_V2.4.24	😥 DownLoad Form
File J-Link settings AT-Link settings Target Language Help Disconnect Part Number: AT32F403AVGT7 FlashSize: 1024KB 1024KB	sLib status SLib status: Disable Remaining usage times: 246
Disconnect AT-Link EZ FW: V15.10 AT-Link N: 8D109C11008045410517BD02 雅特力。	Enable password 0x 55665566 sLib position: Main Flash v Disable password 0x Start sector Sector 20x08001000 v Disable sLib DATA start sector Sector 40x08002000 v
SPIM FLASH_DA 0x 0 @ Remap0 (Use PA11/PA12 pins) Type Select © Remap1 (Use PB10/PB11 pins)	End sector Sector 50x08002800
Memory read settings	Extra options
Address 0x 08000000 Read size 0x 000003CC Data bits 8 bits 🔻 Read	Erase the sectors of file size 🔹 🗖 Disable sLib before download
File info	Verify Custom encryption key for verify Disable FAP before download
No. File name File size Address range(0x) Add 1 ER SLIB INSTRUCTION.bin 280 08001000-08001117 Delete	Jump to the user program Enable FAP after download
1 ER_SLB_INSTRUCTION.bin 280 08001000-08001117 Delete 2 ER_SLB_DATA.bin 116 08002000-08002073 Delete	Write software serial number(SN)
	Write address 0x 08010000 Button free mode Current SN 0x 00000001
Flash CRC File CRC verify DownLoad	Increase step 0x 00000001
Address range:[0x08002000 0x08002073] checksum: 0x00002C29	🔲 Write user system data
Address 0 1 2 3 4 5 6 7 8 9 A B C D E F A: -	User system data file path
0xx08002000 B9 E1 EE BA 12 22 D0 BA 00 00 00 F7 55 72 3B 要	
0x08002010 CF 4E 04 3C 58 C2 0B 3C 00 00 00 80 9E 85 8E BC 🐲 🗏	
0x08002020 88 DC 0B BD 9C A3 08 BD 00 00 00 0A 82 8A 3D 現金	
0x08002030 F0 DB 1B 3E 5F 46 64 3E 06 41 80 3E 5F 46 64 3E 17:	
0x08002040 F0 DB 1B 3E 0A 82 8A 3D 00 00 00 9C A3 08 BD 7:	Start Download Cancel Close
4 m	

Figure 44. Online programming to MCU in ICP

(3) End users also can use ICP Programmer to set an offline project and save to AT-Link, and then complete offline programming to MCU through AT-Link, as shown in Figure 45.

Offli	ne project		• Delete		Creat
Proj	ect name slib_dewnload		Device AT32F403A	AT32F403AVGT7	-
No	. File name	File size	Address range(0x)	Storage loca	. Add
1	ER_SLIB_INSTRUCTION.bin	280	08001000-0800111	7	Delete
2	ER_SLIB_DATA.bin	116	08002000-0800207	3	_
•				•	
Era	se option Erase the sectors of fil	e size	-		
	Download times		Download interfac	e SWD	_
	nonvention transmit	Verify			•
] Verify	Reset and run		
	Encryption transmit	Verify			
□ \ □ E	Nrite user system data				· · · · · · · · · · · · · · · · · · ·
So	Nrite user system data Enable FAP after download		ELID settings	Main Flash	
So	Nrite user system data Enable FAP after download ftware serial number(SN) SPIM	settings S	Reset and run Lib settings Lib position		
So.	Arite user system data Enable FAP after download ftware serial number(SN) SPIM Enable sLib	settings S	ELib settings Start sector	Main Flash	
So sL	Write user system data Enable FAP after download ftware serial number(SN) SPIM Enable sLib ib enable password 0x 556655	settings S	Reset and run Lib settings Lib position Start sector DATA start sector	Main Flash Sector 20x08001000	
So sL	Arite user system data Enable FAP after download ftware serial number(SN) SPIM Enable sLib ib enable password 0x 556655 Disable sLib before download	settings S	Reset and run Lib settings Lib position Start sector DATA start sector	Main Flash Sector 20x08001000 Sector 40x08002000 Sector 50x08002800	
So St	Arite user system data Enable FAP after download ftware serial number(SN) SPIM Enable sLib ib enable password 0x 556655 Disable sLib before download	settings S	Reset and run Lib settings Lib position Start sector DATA start sector End sector	Main Flash Sector 20x08001000 Sector 40x08002000 Sector 50x08002800	
So'	Arite user system data Enable FAP after download ftware serial number(SN) SPIM Enable sLib ib enable password 0x 556655 Disable sLib before download	settings S	Reset and run Lib settings Lib position Start sector DATA start sector End sector Load param	Main Flash Sector 20x08001000 Sector 40x08002000 Sector 50x08002800	

Figure 45. Offline programming to MCU via AT-Link

(4) After completing step 2/3, end users can get the MCU with programmed SLIB area (SLIB status: enabled), and program the application code to MCU through online or offline programming, as shown in Figure 46.



	securigs	AT-Lin	nk settin	97 H	arget	Lung	uage	Help						sLib status			
	Part Nu	nher:	AT326	103AV	GT7	Flash	Size: 1	024KB		.1	זכ		עכ	sLib status: Enable	Remain	ing usage	times: 245
isconnect	AT-Link		FW: V1.							, i I	Z I	Ξ	X I	Enable password 0x 5566556	i6 sLib po	sition:	Main Flash 👻
T-Link 🔻	AT-Link		D109C11		41051	7BD02				雅	- 1	は	力	Disable password 0x	Start se	ctor	Sector 20x08001000 -
· Link						-						-	//	Disable	sLib DATA s	tart sector	Sector 40x08002000 -
	SPIM		FLASH	H_DA 0>		0		nap0 (l							End sec	tor	Sector 50x08002800 -
	Туре				Se	elect	Rei	nap1 (l	lse PB		11 pi	ins)					
Memory read	d settings													Extra options		_	
Address 0x	08001000	R	ead size	0x 00	000011	.8	Data bi	s 8 bi	ts 🔻			Read		Erase the sectors of file size		- 📃 Die	able sLib before download
ile info														Verify Custom encry	yption key for verify	: 🔽 En	able sLib
												Ac				📃 Dis	able FAP before download
No. File r	ame f403a_proj			e size 128		dress rai 00000-0		0000	000.0	00040	47			Jump to the user program		En	able FAP after download
1 at52	i405a_proj	scilitue	x 74	20	0800	00000-0	000036	5,08003	000-0	00049	47	Del	ete	Write software serial numb			
															Jer(SIN)		tton free mode
1											•			Write address 0x 080	010000	Bu	tton free mode
						_					_	_	_	Current SN 0x 000	000001		
						Flash CF	RC	File CF	lC veri	fy	Do	wnLo	ad	Increase step 0x 000	000001		
lash info Fil																	
Flash info Fil Address range				ddress r	ange:[0	1x080030	00 0x080	04947]	checks	sum: 0:	k000A	BAA7		🔲 Write user system data			
		0 0x0800		Address r		0x080030		04947] A B	checks C	sum: 0:	E	F	A!	Write user system data User system data file path			
Address range	::[0x0800000	0 0x0800 2 0 00	03BB] A 3 4 20 01	5 30	6		9 35	A B	C EF	D 35	Е 00	F 08	80				
Address range Address x08000000 x08000010	0 1 38 1 F3 3	0 0x0800 2 3 00 5 00	03BB] A 3 4 20 01 08 CF	5 30 35	6 00 00 00	7 8 08 F7 08 75	9 35 36	A B 10 08 10 08	C EF 00	D 35 00	E 00 00	F 08 00					
Address range Address x08000000 x08000010 x08000020	E:[0x0800000 0 1 38 1 F3 3 00 0	0 0x0800 2 3 00 5 00 0 00	03BB] A 3 4 20 01 08 CF 00 00	5 30 35 00	6 00 00 00 00 00 00 00 00 00 00 00 00 00	7 8 08 F7 08 75 00 00	9 35 36 00	A B 10 08 10 08 10 00	C EF 00 FB	D 35 00 35	E 00 00 00	F 08 00 08	80				
Address range Address	0 1 38 1 F3 3	2 3 00 3 00 3 00 3 00 3 00 3 00 3 00 3 00	03BB] A 3 4 20 01 08 CF	5 30 35 00 00	6 00 00 00 00 00 00 00 00 00 00 00 00 00	7 8 08 F7 08 75	9 35 36 00 35	A B 10 08 10 08	C EF 00	D 35 00	E 00 00	F 08 00 08 08	80				

Figure 46. End users program codes to MCU

4.2 Integrate and program codes

Integrate the SLIB code of solution provider and the end user application code to an offline project, and then download the integrated code to MCU through AT-Link offline programming. The process is as follows:

- (1) The solution provider handles the compiled project as aforementioned to get a slib.bin file;
- (2) The solution provider uses ICP Programmer to generate an offline project and save it to PC. Parameters (such as number of download, project files binding to AT-Link and enable FAP after download) can be configured as needed. Save the offline project as follows.

Note: The offline project is encrypted. To enhance security, the solution provider also can set the slib.bin file to an encrypted slib.benc and then add it to the offline project. In this case, the offline project can only be used on the AT-Link with the corresponding encryption key.



Offline project	AT-Link offline download status	
Project name slib_project No. File name File 1 ER_SLIB_INSTRUCTION.bin 280 2 ER_SLIB_DATA.bin 116	Device AT32F403A AT32F403AVGT7 size Address range(0x) O8001000-08001117 O8002000-08002073 Device AT32F403A Add Delete Dete SWD Download interface SWD	
Software serial number(SN) SPIM setting	s Lib settings	
Enable sLib sLib enable password 0x 55665566 Disable sLib before download sLib disable password 0x	stib position Main Flash ▼ Start sector Sector 20x08001000 ▼ DATA start sector Sector 40x08002000 ▼ End sector Sector 50x08002800 ▼ Load parameters Save parameters	Image: Second state of the settings Image: Second state of the settings Image: Second state of the setting second state of the

Figure 47. Set offline project

(3) After obtaining the offline project, the end user should use ICP Programmer to open the project file and add the application codes to the offline project; then save to PC or AT-Link, and perform offline download. Figure 48 shows how to add the project file.

Note: To protect codes from being leaked or decoded, do not other change settings when adding code file to the offline project, which requires the solution provider to configure the final settings in advance.

F-Link set				
Offline p	oroject		▼ Delete	Creat
Project n	ame slib_project	D	evice AT32F403A	▼ AT32F403AVGT7
No. F	ile name	File size	Address range(0)	() Storage loci ^ Add
1 E	R_SLIB_INSTRUCTION.bin	l.bin 280 08001000-08001117		17 Delete
	R_SLIB_DATA.bin			
۰ ۲	+20f402a project 1 her	056	0000000 00000	
Erase o	ption Erase the sectors of fi	ile size	-	
Down	load times 0	1	Download interf	ace SWD
	L			
	ption transmit 🔽	Verify	Reset and ru	
Write	L	Verify		
Urite	ption transmit viser system data			n
Write	ption transmit viser system data		Reset and ru	n
O Write Enabl	ption transmit user system data e FAP after download re serial number(SN) SPIM	settings sL	Reset and ru	n
O Write C Enable Softwa ✓ Ena sLib er	ption transmit user system data e FAP after download re serial number(SN) SPIM ble sLib	settings sL	Reset and ru Start sector	n Main Flash Sector 20x08001000
 Write Enabl Softwa ✓ Ena sLib er Dis. 	ption transmit user system data e FAP after download re serial number(SN) SPIM ble sLib able password 0x ****** able sLib before download	settings sL	Reset and ru Bettings sLib position Start sector DATA start sector	Main Flash • Sector 20x08001000 • Sector 40x08002000 •
 Write Enabl Softwa ✓ Ena sLib er Dist 	ption transmit user system data e FAP after download re serial number(SN) SPIM ble sLib sable password 0x ******	settings sL	Reset and ru Start sector	n Main Flash Sector 20x08001000
 Write Enabl Softwa ✓ Ena sLib er Dis. 	ption transmit user system data e FAP after download re serial number(SN) SPIM ble sLib able password 0x ****** able sLib before download	settings sL	Reset and ru Bettings sLib position Start sector DATA start sector	Main Flash • Sector 20x08001000 • Sector 40x08002000 • Sector 50x08002800 •
 Write Enabl Softwa ✓ Ena sLib er Dis. 	ption transmit user system data e FAP after download re serial number(SN) SPIM ble sLib able password 0x ****** able sLib before download	settings sL	Reset and ru Acceleration Start sector DATA start sector End sector	Main Flash • Sector 20x08001000 • Sector 40x08002000 • Sector 50x08002800 •
 Write Enabl Softwa ✓ Ena sLib er Dis. 	ption transmit user system data e FAP after download re serial number(SN) SPIM ble sLib able password 0x ****** able sLib before download	settings sL	Reset and ru Acceleration Start sector DATA start sector End sector	Main Flash • Sector 20x08001000 • Sector 40x08002000 • Sector 50x08002800 •
 Write Enabl Softwa ✓ Ena sLib er Dis. 	ption transmit user system data e FAP after download re serial number(SN) SPIM ble sLib able password 0x ****** able sLib before download	settings sL	Reset and ru Acceleration Start sector DATA start sector End sector	Main Flash • Sector 20x08001000 • Sector 40x08002000 • Sector 50x08002800 •
 Write Enable Softwa ✓ Ena sLib er Disissibility 	ption transmit user system data e FAP after download re serial number(SN) SPIM ble sLib able password 0x ****** able sLib before download	settings sL	Reset and ru	Main Flash • Sector 20x08001000 • Sector 40x08002000 • Sector 50x08002800 •

Figure 48. Add project file

5 Revision history

Table 2. Document revision history

Date	Version	Revision note
2021.05.21	2.0.0	Initial release.
2021.11.01	2.0.1	Optimized format.
2022.4.13	2.0.2	Modified the configurable range of sLib in the Flash.

IMPORTANT NOTICE – PLEASE READ CAREFULLY

Purchasers are solely responsible for the selection and use of ARTERY's products and services, and ARTERY assumes no liability whatsoever relating to the choice, selection or use of the ARTERY products and services described herein.

No license, express or implied, to any intellectual property rights is granted under this document. If any part of this document deals with any third party products or services, it shall not be deemed a license grant by ARTERY for the use of such third party products or services, or any intellectual property contained therein, or considered as a warranty regarding the use in any manner whatsoever of such third party products or services or services or any intellectual property contained therein.

Unless otherwise specified in ARTERY's terms and conditions of sale, ARTERY provides no warranties, express or implied, regarding the use and/or sale of ARTERY products, including but not limited to any implied warranties of merchantability, fitness for a particular purpose (and their equivalents under the laws of any jurisdiction), or infringement of any patent, copyright or other intellectual property right.

Purchasers hereby agrees that ARTERY's products are not designed or authorized for use in: (A) any application with special requirements of safety such as life support and active implantable device, or system with functional safety requirements; (B) any air craft application; (C) any automotive application or environment; (D) any space application or environment, and/or (E) any weapon application. Purchasers' unauthorized use of them in the aforementioned applications, even if with a written notice, is solely at purchasers' risk, and is solely responsible for meeting all legal and regulatory requirement in such use.

Resale of ARTERY products with provisions different from the statements and/or technical features stated in this document shall immediately void any warranty grant by ARTERY for ARTERY products or services described herein and shall not create or expand in any manner whatsoever, any liability of ARTERY.

© 2021 Artery Technology -All rights reserved