

AN0071

Application Note

AT32F421 Security Library Application Note

Introduction

This application note introduces the security library (sLib) application principle of AT32F421 MCUs, operation methods and example projects.

Applicable products:

Partnumber	AT32F421
Faithuilibei	AI 32F421

Contents

1	Ove	rview.		7
2	Арр	licatio	n principles	8
	2.1	Applic	ation principle of sLib	8
	2.2	How to	o enable sLib protection	9
	2.3	How to	o disable sLib protection	10
	2.4	Comp	ile and execute program in sLib	11
		2.4.1	Setting interrupt vector table as sLib instruction area not allowed	12
		2.4.2	Correlation between sLib area and user code area	12
		2.4.3	Use and compile software floating-point arithmetic library	14
3	Exa	mple a	pplications of sLib	16
	3.1	Exam	ple application requirements	16
		3.1.1	Hardwar requirements	16
		3.1.2	Software requirements	16
	3.2	Overv	iew	16
	3.3	SLIB p	protected code: FIR low-pass filter	17
	3.4	Projec	ct_L0: example for solution providers	18
		3.4.1	Generate execute-only code	
		3.4.2	Compile security library address	20
		3.4.3	Enable sLib protection	24
3		3.4.4	Project_L0 execution process	25
		3.4.5	Generate header file and symbol definition file	27
	3.5	Projec	ct_L1: example for end users	
		3.5.1	Create user application project	29
		3.5.2	Add symbol definition file to project	
		3.5.3	Call functions in SLIB-protected area	31
		3.5.4	Project_L1 execution process	31
		3.5.5	SLIB protection in debug mode	32
4	Inte	grate c	odes and download	35
	4.1	Progra	am codes separately	35



5

4.2	Integrate and program codes	38
Revis	sion history	41



List of tables

Table 1. Flash size of AT32F421 MCUs	9
Table 2. Document revision history	. 41

<u>FARERY</u> List of figures

Figure 1. Mapping of main Flash memory featured with sLib)
Figure 2. Literal pool example (1)1	I
Figure 3. Literal pool example (2)12	2
Figure 4. Example of function in sLib area calling the function in user code area	3
Figure 5. Example of user-defined function14	1
Figure 6. Assembly codes in division functions14	1
Figure 7. Example application process16	3
Figure 8. Example application17	7
Figure 9. FIR low-pass filter	7
Figure 10. Enter Option interface in Keil	3
Figure 11. Tick Execute-only Code in Keil)
Figure 12. Enter Option interface in IAR)
Figure 13. Set C/C++ options in IAR 20)
Figure 14. Main Flash memory mapping and RAM partition)
Figure 15. Set Linker option in Keil	l
Figure 16. Modify scatter in Keil	2
Figure 17. Modify SLIB RAM address in Keil	2
Figure 18. Modify SLIB constant address in KEIL 22	2
Figure 19. SLIB address definition in icf file	3
Figure 20. Address assignment in icf file	3
Figure 21. Modify SLIB used RAM in icf file	3
Figure 22. Modify SLIB read-only area in icf file	1
Figure 23. Configure ICP Programmer24	1
Figure 24. Set parameters in Download Form25	5
Figure 25. Project_L0 execution process	3
Figure 26. Set Misc controls in Keil 27	7
Figure 27. Contents of modified fir_filter_symbol.txt	7
Figure 28. Set Build Actions in IAR	3
Figure 29. Edit steering_file.txt contents	3
Figure 30. Modified scatter file)
Figure 31. Modified icf file)
Figure 32. Add symbol definition file in Keil)
Figure 33. Modify symbol definition file type to "Object file")
Figure 34. Add symbol definition file in IAR	l

<u>, 175575</u>

Figure 35. Project_L1 execution process	32
Figure 36. Enter Show Disassembly at Address	33
Figure 37. Set Show Code at Address	33
Figure 38. View codes	33
Figure 39. View codes in Memory	34
Figure 40. SLIB_READ_ONLY start sector in Memory	34
Figure 41. SLIB write test	34
Figure 42. Write protection error interrupt	34
Figure 43. Save SLIB codes	35
Figure 44. Generate .bin file of SLIB code	36
Figure 45. Online programming to MCU in ICP	36
Figure 46. Offline programming to MCU via AT-Link	37
Figure 47. End users program codes to MCU	38
Figure 48. Set offline project	39
Figure 49. Add project file	40

<u>Y7=</u>]71:

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 AT32F421 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 AT32F421 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 contains data security library (SLIB_READ_ONLY) and instruction security library (SLIB_INSTRUCTION); users can set part of or the whole security library as SLIB_READ_ONLY or SLIB_INSTRUCTION.
- Data in the SLIB_READ_ONLY area can only be read through I-Code and D-Code and cannot be programmed.
- Program code 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 bus (including ISP/ICP/debug mode and programs that boot from internal RAM). When reading the SLIB_INSTRUCTION area, values are all read 0xFF or 0x00.
- 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 programmed 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 AT32F421 series MCUs.

In addition, once the boot memory is programmed as memory extension area, the entire 4 KB area can be used as security library.

Table	1.	Flash	size	of	AT32F421	MCUs
-------	----	-------	------	----	----------	------

Model	Internal Flash size (Byte)	Sector size (Byte)	Configurable range
AT32F421x4	16K	1К	Sector 0 ~ 15 ⁽¹⁾ (0x08000000 ~ 0x08003FFF)
AT32F421x6	32К	1К	Sector 0 ~ 31 ⁽¹⁾ (0x08000000 ~ 0x08007FFF)
AT32F421x8	64K	1К	Sector 0 ~ 63 ⁽¹⁾ (0x08000000 ~ 0x0800FFFF)

(1) Sector 0 cannot be configured as the instruction security library.

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 by writing 0xA35F6D24 to the SLIB_UNLOCK register. Then check the SLIB_ULKF bit in the SLIB_MISC_STS register to verify if it is unlocked successfully. If successful, write the programmed value 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 if it is unlocked successfully;
- Set the sectors to be protected in the SLIB_SET_RANGE register, including the SILB start/end addresses and the start address of instruction security library;
- 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 a system reset, and then reload the security library setting words;
- Read the SLIB_STS0/STS1 register to verify the security library settings.

Note:

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

For details about security library setting register, refer to *AT32F421 Series Reference Manual*. The security library can be enabled by the *slib_enable()* function in *main.c* file of project_I0. In addition, users can use Artery ICP or ISP tools for configuration.

2.3 How to disable sLib protection

The security library protection can be disabled by writing the previously programmed 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 programmed password to the SLIB_PWD_CLR register;
- Perform a system reset, and then reload security library setting words;
- Read the SLIB_STS0 register to verify 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, 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 the CRM_CFG register in a PC (program counter) indirect addressing manner, and the address of the 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 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.

				0	
0x08	004798	2600	MOVS	r6,#0x00	
	/9: s 80:	CIK_SOUR	ce = (crm_s	<pre>scik_type)ckm->cig_</pre>	DIL.BCIKSUS;
0x08	00479A	4F39	LDR	r7,[pc,#228]	; @0x08004880
0x08	00479C	687F	LDR	r7,[r7,#0x04]	
0x08	00479E	F3C70381	UBFX	r3,r7,#2,#2	
	81: s	witch(sc	lk_source)		
	82: {	anan CD	A SCIN HICK		
	03:	case ch		•	
	main.c	startup_a	at32f403a_407.s	at32f403a_407_clock.	system_at32f403a_407.c at32f403a_407_crm.c at32f403a_407_gpio.c
	77				
	78	/* ae	t solk	source */	
	70	- colle	COURGO T	= (arm call ti	ma) CPM-) of a hit callesta:
·V	19	SCIK_	source -	- (CIM_SCIK_U	/pe/orm /cig_bit.sciksts,
	80			×	
	81	swite	h(sclk_	source)	
	82 申	{			
	83	cas	e CRM SO	CLK HICK:	
	84		f((CRM)	->misc3 hit hi	ick to selk) != RESET) && ((CRM-)misel bit bickd
	01	-		one alask -	UTCV VALUE + 6.
	00		system	_core_clock -	$\operatorname{HICK}_{\operatorname{VALUE}} * 0,$
	86	e	else		
	87		system	_core_clock =	HICK_VALUE;

Figure 2. Literal pool example (1)



	137: 5	system_core	e_clock = syste	em_core_clock >> div_value;
	0x0800486E	4F06	LDR	r7,[pc,#24] ; @0x08004888
	0x08004870	683F	LDR	r7,[r7,#0x00]
	0x08004872	40F7	LSRS	r7,r7,r6
	0x08004874	F8DFC010	LDR.W	r12,[pc,#16] ; @0x08004888
	0x08004878	F8CC7000	STR	r7,[r12,#0x00]
	138: }			
R	0x0800487C	BDF0	POP	{r4-r7,pc}
	0x0800487E	0000	DCW	0x0000
	0x08004880	1000	DCW	0x1000
	0x08004882	4002	DCW	0x4002

Figure 3. Literal pool example (2)

2.4.1 Setting interrupt vector table as sLib instruction area 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 memory as sLib instruction area.

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 cannot 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. Example of 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.

Recommended solutions:

- 1) Compile into the sLib area (refer to Keil or IAR documents for details).
- 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



2.4.3 Use and compile software floating-point arithmetic library

Since the AT32F421 series does not have a hardware floating point unit (FPU), the Keil or IAR compiler will use ARM[®] software floating-point arithmetic library for floating-point operations. The software floating-point arithmetic library functions are compiled codes that cannot be modified, and some of them are in the literal pool format as mentioned before; therefore, these functions cannot be compiled to the SLIB_CODE but must be placed in the SLIB_READ_ONLY area. As shown in Figure 6, the division functions in Keil floating-point arithmetic library have assembly codes in literal pool format.

Disassembly			
0x0803C776	F3EF8000	MRS	rO, APSR
0x0803C77A	4770	BX	lr
	a	eabi_fdiv	1
>0x0803C77C ↓	F44F0C7F	MOV	r12,#0xFF0000
0x0803C780	EA1C12DO	ANDS	r2,r12,r0,LSR #7
0x0803C784	BF1E	ITTT	NE
0x0803C786	EA1C13D1	ANDS	r3,r12,r1,LSR #7
0x0803C78A	EA920FOC	TEQ	r2,r12
0x0803C78E	EA930FOC	TEQ	r3,r12
0x0803C792	F0008085	BEQ.W	0x0803C8A0
0x0803C796	EA900F01	TEQ	r0,r1
0x0803C79A	BF48	IT	MI
0x0803C79C	F4427280	ORR	r2,r2,#0x100
0x0803C7A0	F4400C00	ORR	r12,r0,#0x800000
0x0803C7A4	F4410000	ORR	r0,r1,#0x800000
0x0803C7A8	F02C417F	BIC	r1,r12,#0xFF000000
0x0803C7AC	F020407F	BIC	r0,r0,#0xFF000000
0x0803C7B0	B500	PUSH	{lr}
0x0803C7B2	4281	CMP	r1,r0
0x0803C7B4	EBA20203	SUB	r2,r2,r3
0x0803C7B8	F20F1C08	ADR.W	r12,{pc}+0x10C ; @0x0803C8C4
0x0803C7BC	EBAC4E50	SUB	lr,r12,r0,LSR #17
0x0803C7C0	F1C00000	RSB	r0,r0,#0x00
0x0803C7C4	F89EE000	LDRB	lr,[lr,#0x00]
0x0803C7C8	EA4FUE4E	LSL	lr, lr, #1
0x0803C7CC	FBOOFCOE	MUL	r12.r0.lr



After the sLib protection is enabled, all contents in the sLib protected area cannot be changed, including the address of floating-point arithmetic library functions called by SLIB_CODE. Section 3 introduces how to compile the floating-point arithmetic library functions to be used into the SLIB_READ_ONLY area, so that programs in SLIB_CODE can be called properly after the sLib protection is enabled.

For details about Keil floating-point arithmetic library, refer to the ARM DUI0378G ARM® Compiler v5.06 for $\mu Vision$ ® ARM C and C++ Libraries and Floating-Point Support User Guide under the installation directory.

For details about IAR floating-point arithmetic library, refer to the EWARM_DevelopmentGuide *IAR C/C++ Development Guide* (PREBUILT RUNTIME LIBRARIES section) under the installation directory.



3 Example applications of sLib

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

3.1 Example application requirements

3.1.1 Hardwar requirements

- AT-START-F421 demo board with AT32F421C8T7 chip
- AT-Link emulator for debugging

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 or ISP programming tools for enabling and disabling sLib

3.2 Overview

This application note provides two sample 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 by end users

The algorithm completed in Project_L0 will be pre-downloaded and pre-burned to AT32F421 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, and end users can 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 7. 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.



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 at 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 9 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, 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. It is called by the end user and 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 read-only area in the example.

Since the AT32F421 series does not have an embedded hardware FPU, the floating-point arithmetic library functions in this example are used for signal processing and floating-point operations.

3.4 **Project_L0: example for solution providers**

The following projects are completed in this level:

- Compile the algorithm-related functions to execute-only code;
- Place the algorithm program code to the main Flash memory sector 12 to sector 19 (address: 0x08003000 ~ 0x08004FFF);
- Place the filter function coefficients, floating-point arithmetic library and C library used by the program to the main Flash memory sector4 to sector11 (address: 0x08001000 ~ 0x08002FFF);
- Execute *FIR_lowpass_filter()* in the main program to verify its correctness;
- If correct, configure sector12 to sector19 as instruction security library and sector4 to sector11 as read-only area, which can be completed by calling *slib_enable()* function in the main program or using Artery ICP Programmer (recommended);
- 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 pool and branch table.

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

Keil® $\mu\nu$ ision: 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 10;



Figure 10. Enter Option interface in Keil

• Tick "Execute-only Code" in the C/C++ and the "--execute_only" instruction is added to the compiler control string, as shown in Figure 11;

Options for File 'arm_fir_f32.c'		×
Properties C/C++		
Preprocessor Symbols		
Define:		
Undefine:		
Language / Code Generation		
Execute-only Code	Strict ANSI C	Warnings:
Optimization: <default> ▼</default>	Enum Container always int	All Warnings 🗨
	Plain Char is Signed	Thum <u>b</u> Mode
Split Load and Store Multiple	Read-Only Position Independent	No Auto Includes
One ELF Section per Function	Read-Write Position Independent	C99 Mode
Include Paths		
Controls		
Compiler control string	x-M4.fp -DMICROLIB -g -O0apcs=interwo \Vibraries\cmsis\cm4\core_support -I\.	rksplit_sections -l\\ A
08	Cancel Defaults	Help
		nerp

Figure 11. Tick Execute-only Code in Keil

• The a*rm_fir_f32.c*, *arm_fir_init_f32.c* and *fir_filter.c* 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;

|--|

├ 📮 🛋 fir_filter	
−⊞ 🗟 arm_fir_init_f32.c	Options
ir_coefficient.c	Make
└─⊞ 💿 fir_filter.c	

 Enter "C/C++" interface and tick "Override inherited settings" and "No data read in code memory", as shown in Figure 13;



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 Optimirations Output List Processor mode Arm Optimirations Output List Position-independence Ecde and read-only data (ropi) Read/write data (rwpi) No dynamic read/write initializati. Image: Model and read in code memory Image: Model and read in code memory Image: Model and read in code memory
	OK Cancel

Figure 13. Set C/C++ options in IAR

• The *arm_fir_f32.c*, *arm_fir_init_f32.c* and *fir_filter.c* are in the SLIB_INSTRUCTION area, and these files need to be configured 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 4 in this example, with sectors 12-19 being set as instruction security library and sector 4-11 being set as read-only area. Figure 14 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 14. Main Flash memory mapping and RAM partition



Keil® µvision: scatter file

Operate as follows:

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

Figure 15. Set Linker option in Keil

Device Tar;	et Output Listing Vser C/C++	Asm Linker Debug Utiliti	ies	
Use Men Make Don't Repo	ory Layout from Target Dialog RW Sections Position Independent RO Sections Position Independent Search Standard Libraries t 'might fail' Conditions as Errors	X/O Base:		
Sca <mark>t</mark> er He	\slib-w xo.sct		<u>Edt</u>	
Misc controls Linker control	-symdefs=fir_filter_symbol.bt -cpu Cortex-M4.fp *.o -library_type=microlib -strict -scatter ".\slib-w	∵xo.sct"	* *	
string	0K C	ncel 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 12 and occupies eight sectors) and modify the label to "execute-only (+XO)". Place the area occupied by SLIB_READ_ONLY to a dedicated loading area named "LR_SLIB_READ_ONLY" to avoid the compiler compiling other non-IP-Code functions to the SLIB area. The RW_IRAM2 assigns the region from 0x20003000 to 0x20003FFF to the sLib algorithm functions to avoid the same RAM region being used by end-user project, causing fault or error in program execution process.

Figure 16. Modify scatter in Keil



Note: Use *armlib* to compile the floating-point arithmetic library functions and C standard library to LR_SLIB_READ_ONLY area.

 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 0x20003000, as shown in Figure 17.



```
#if defined (__ICCARM__)
static float32_t firStateF32[BLOCK_SIZE + NUM_TAPS - 1] @ 0x20003000 ;
#elif defined (__CC_ARM )
static float32_t firStateF32[BLOCK_SIZE + NUM_TAPS - 1] __attribute_((at(0x20003000)));
#endif
```

 The start address of read-only area is sector 4 (0x08001000). To compile the constants used by FIR low-pass filter to this address, users can modify the scatter file as aforementioned, or use Keil "__attribute__((at(address)))" descriptor to load the constants to a fixed address, as shown in Figure 18.

Figure 18. Modify SLIB constant address in KEIL

```
##if defined ( __ICCARM__ )
#const float32_t firCoeffs32[NUM_TAPS] @ 0x08001000 ={
    #elif defined ( __CC_ARM )
#const float32_t firCoeffs32[NUM_TAPS] __attribute__((at(0x08001000))) = {
    #endif
```



IAR: ICF file

Operate as follows:

 Open the *icf* file in "\project_I0\IAR_V8.2\", and add three new loading areas as shown in Figure 19. The SLIB_RAM region reserves the corresponding RAM (0x20003000 to 0x20003FFF) for the algorithm functions.

Figure 19. SLIB address definition in icf file

```
/* SLIB read-only area */
define symbol __ICFEDIT_region_SLIB_READ_ONLY_start__ = 0x08001000;
define symbol __ICFEDIT_region_SLIB_READ_ONLY_end__ = 0x08002FFF;
/* SLIB instruction area */
define symbol __ICFEDIT_region_SLIB_INST_start__ = 0x08003000;
define symbol __ICFEDIT_region_RAM_start__ = 0x20000000;
define symbol __ICFEDIT_region_RAM_end__ = 0x2000000;
define symbol __ICFEDIT_region_RAM_end__ = 0x20002FFF;
/* SLIB RAM region */
define symbol __ICFEDIT_region_SLIB_RAM_start__ = 0x20003000;
define symbol __ICFEDIT_region_SLIB_RAM_end__ = 0x20003000;
```

• 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 20. 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 (0x20003000) or modify the *icf* file, as shown in Figure 21.

Figure 21. Modify SLIB used RAM in icf file

 The start address of read-only area is sector 4 (0x08001000). To compile the constants, m7Mx_tl.a (floating-point arithmetic library) and rt7Mx_tl.a (C library) used by FIR low-pass filter to this address, as shown in Figure 22. Figure 22. Modify SLIB read-only area in icf file

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 Artery ICP Programmer as follows:

- Connect AT-Link emulator to AT-START-F421 board and then 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 23.

-		Part	Num	ber:	AT3	2F42	21C8	T7	Fla	shSi	ze: 64	4KB			.1	וכ		C
Discor	nnect	AT-Li	nk-F2	2	FW.	V1.5	12								;;;	ZI	-	Z
AT-Lin	ık 🔻	AT-Li	nk SM	- 1: 6	C985	03200	000B3	32905	9709	02					雅	1	恃	5
		SPI	м		FL	ASH	DA 0	x	0		@ R	emap	0 (U:	se PA	11/PA	12 p	ins)	-
		Туре							Selec	t	R	emap	1 (U:	se PB	10/PE	11 p	ins)	
Memo	ory read	l settin	gs															
Address 0x 08000000 Read size 0x 00002000 Data bits 8 bits Read																		
ella la	<i>6</i> -																	
File in	T ^C					c :1			1.1			2						\dd
1	File n	ame				File	size	Ac	ares	s ran	ige(Ux 20002	18.09	0010	00-0	8001	07.0		kua
1	proje	ct_i0.ne	x			022	4	UC	0000	00-04	60005	10,00	SOOTC	00-0	50011	,097,0	C De	elete
•					ш											Þ		
									Flag	-h CR		Fil		veri	6 7	De		d
The she i	- (-) E il	oproie	-+ 10 k								<u> </u>	<u> </u>	0 0.10		.,			ouu
Addre	ss range	:[0x0800	00000 (0x0800	0031B]	Ad	dress	range	[0x08	00100	0 0x08	3001D	97] /	Addre	ss ran	ge:[0x	08003	3000
0x080	0310B1	Addres	rand	e:[0x0	80050	00 0x0	08005E	5F1	check	sum:	0x000	C2944	B	6	D	F	E	1
0x0800	0000	40	18	00	20	01	50	00	08	4D	50	00	08	45	50	00	08	@:
0x0800	0010	49	50	00	08	25	50	00	08	C1	50	00	08	00	00	00	00	IP.
0x0800	0020	00	00	00	00	00	00	00	00	00	00	00	00	51	50	00	08	
0x0800	0030	29	50	00	08	00	00	00	00	4F	50	00	08	53	50	00	08)P.
0x0800	0040	1B	50	00	08	1B	50	00	08	1B	50	00	08	2B	50	00	08	DP
						III												4
•		k conn	ection	is suc	cessfu	ul.												
4	t : AT-Lir	ik com																
14:00:04	l : AT-Lir l : Part N	umber:	AT32	F4210	8T7	Flas	hSize:	64KB										

Figure 23. Configure ICP Programmer



 Click "Download" and the "Download Form" pops up, which shows sLib status and relevant parameters. Set the start sector, INSTR start sector and end sector; set the enable password as "0x55665566" (user-defined) and tick "Enable sLib"; then click "Start Download" to complete programming and enable sLib successfully, as shown in Figure 24.

Disable password 0x Start sector Sector40x08001000 Disable sLib INSTR start sector Sector120x08003000 End sector Sector190x08004C00 Extra options Erase the sectors of file size	isable password 0x Start sector Sector120x08001000 Disable sLib INSTR start sector End sector tra options	_
End sector End sector End sector End sector Extra options Erase the sectors of file size Verify Custom encryption key for verify: Disable sLib Disable FAP before download Disable FAP before download Disable FAP after download Write software serial number(SN) Write address 0x 08010000 Current SN 0x 00000001	End sector Sector190x08004C00	-
Extra options Erase the sectors of file size Verify Custom encryption key for verify: Disable sLib Disable FAP before download Jump to the user program Write software serial number(SN) Write address 0x 08010000 Current SN 0x 00000001 Disable FAP Disable sLib Disable sLib	tra options	•
Erase the sectors of file size Disable sLib before download Verify Custom encryption key for verify: Enable sLib Disable FAP before download Jump to the user program Enable FAP after download Write software serial number(SN) Basic access protection Button free mode 		
✓ Verify Custom encryption key for verify: ✓ Enable sLib □ Disable FAP before download □ Jump to the user program □ Enable FAP after download □ Write software serial number(SN) □ Basic access protection ○ Write address 0x 08010000 Current SN 0x 0000001	Erase the sectors of file size 🔹 🗾 Disable sLib before downl	bad
Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constraint of the user program Image: Constrai	✓ Verify Custom encryption key for verify:	
Jump to the user program Enable FAP after download Write software serial number(SN) Basic access protection Write address 0x 08010000 Current SN 0x 00000001	Disable FAP before downl	ad
Write software serial number(SN) Basic access protection Write address 0x 08010000 Current SN 0x	Jump to the user program Enable FAP after download	l i
Write address 0x 08010000 Current SN 0x	Write software serial number(SN) Basic access protection	-
Current SN 0x 00000001	Write address 0x 08010000 🔲 Button free mode	
	Current SN 0x 00000001	
Increase step 0x 0000001	Increase step 0x 00000001	
Write user system data	Write user system data	
User system data file path		
	User system data file path	

Figure 24. Set parameters in Download Form

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

(2) Use *slib_enable()* 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 25 shows the Project_L0 execution process.



Figure 25. 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\AT32F421_slib_demo\project_I0\mdk_v5\, and then compile;
- (2) Before downloading the code, check whether the chip on AT-START-F421 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 \rightarrow Linker interface;
- Add "--symdefs=fir_filter_symbol.txt" command in the "Misc controls", as shown in Figure 26;

Figure 26. Set Misc controls in Keil

Levice Targ □ Use Mem □ Make □ Dont □ Report	et Output Listing User C/C4 ny Layout from Target Dialog RW Sections Position Independent RO Sections Position Independent Search Standard Libraries t 'might fail' Conditions as Errors	+ Asm Linker <u>X</u> /O Base: <u>R</u> /O Base: R/ <u>W</u> Base gisable Warnings:	Debug Utilities 		
Scaţter File	\slib-w-xo.sct			Edit	
Г	symdefs=fir_filter_symbol.txt			<u> </u>	
<u>M</u> isc controls	-cou Cottex-M4 * o			-	1

- After compiling the project, a symbol definition file named "fir_filter_symbol.txt" is generated under "project_l0\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 27;

Figure 27. Contents of modified fir_filter_symbol.txt

0x08003001 T FIR_lowpass_filter



Use IAR to generate symbol definition file

Operate as follows:

● Select Project → Option → Build Actions

Figure 28. Set Build Actions in IAR							
Options for node "project_10"							
Category: General Options Static Analysis Runtime Checking Q/C++ Compler Assembler Output Converter Custom Build Build Actions Linker Debugger Simulator CADI CMSIS DAP GDB Server I-jet/JTAGjet Jink/J-Trace TJ Stellaris Nu-Link Penicro ST 4LINK Thrid-Party Driver TI MSP-FET TI XDS OK							

- 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 the contents called by sLib. As shown in Figure 29, the "show" is the command used to select functions.

Figure	29.	Edit	steering	file.txt	contents
i igui o	20.		olooning_		0011101110

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 AT32F421 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.

Note:

Project_L1 must use the same toolchain and the same version of the compiler as those 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. In the main Flash memory, sector 4 to sector 19 are occupied by security library, which should be isolated by using the linker control file to avoid code being compiled to this region.

Keil® µvision: scatter file

Refer to the *end_user_code.sct* 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 region behind 0x20003000 in the RAM should be reserved, as shown in Figure 30.

Figure 30. 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 0x00003000 { ; RW data
   .ANY (+RW +ZI)
  0x20003000 ~ 0x20003FFF
                              RAM reserved for SLIB code
}
; 0 \times 08001000 ^{\sim} 0 \times 08004 FFF is SLIB area
LR_IROM2 0x08005000 0x0000B000 {
                                       ; load region size region
  ER IROM2 0x08005000 0x0000B000 { ; load address = execution address
   .ANY (+RO)
}
```

IAR: ICF file

Refer to the *enduser.icf* under "project_l1\iar_V8.2\", as shown in Figure 31.

_	Figure 31. Modified ict file
define region ROM_region	<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 32.

Figure 32. Add symbol definition file in Keil



Add this file to fir_filter, and then modify its file type from "text" to "Object", as shown in Figure 33.

🛛 Options for File 'fir_filter_symbol.txt'	
Properties	
Path: Nir_filter_symbol bd	
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: <default></default>	
OK Cancel	Defaults Help

Figure 33. Modify symbol definition file type to "Object file"

Add symbol definition file in IAR

Add the fir_filter_symbol.o (Object) to fir_filter, as shown in Figure 34.



🗆 🌒 project_l1 - at_start_f	~	
⊨ 🖬 💼 bsp		•
⊨ 🖬 🛋 cmsis		•
- 🖓 📫 fir_filter		
📔 🛏 🗋 fir_filter_symbol.o		
⊢⊕ 🖬 firmware		•
🗕 🕀 💼 readme		
⊨ 🕀 🛋 user		•
🖵 🖬 🛑 Output		

Figure 34. Add symbol definition file in IAR

3.5.3 Call functions in SLIB-protected area

After 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:

- *inputF3:* pointer to input signal data table;
- *outputF32*: pointer to output signal data table;
- *TEST_LENGTH_SAMPLES*: the number of signal samples to be processed.

3.5.4 Project_L1 execution process

Figure 35 shows the execution process of Project_L1:

- Start execution and LED3 keeps 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 35. 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 36.



2	Disassembly				^				п 🔽
	Disassembly								* ^
-	0x08003E5 94: 95:	2 4770 AT32_Board	<pre>BX d_Init();</pre>	Ir					^
	96:	/* Configu	re flash to ge	nerate	\checkmark	Mixed Mode		error occur */	
		4 2804 6 D106	ENF	ru,#U		Assembly Mode			
	97:	Enable Fla	ash INT():	0,000		Address Range	•		
	98:								
	99:					Show Disassembly at Address			
	100:	/* Wait fo	or KEY button	to be p		Set Program Counter			
	0x08003E5	8 490A	LDR	r1,[p	*()	Run to Cursor line	Ctrl+F10		
	UXU8003E5 101: 102:	while(AT32	LDR 2_BUTTON_State	r1,[r (BUTTON		Insert/Remove Breakpoint			
	0x08003E5	C F0510104	ORRS	r1,r1	0	Enable/Disable Breakpoint	Ctrl+F9		
	0x08003E6	2 6011	STR	r1,[r		Insert Tracepoint at '0x08003E54'			
	104: 105:	Delay_r	ms(300);			Enable/Disable Tracepoint			
	106:					Inline Assembly			
	107: 0x08003E6	/* Turn 4 E005	Off LED3 */ B	0x080		Load Hex or Object file			
	0x08003E6	6 4907	LDR	r1,[p		Instruction Trace	+		~
•	<	0 2000	IND	~1 F~		Execution Profiling	•		>
	📄 main.c				P	Insert/Remove Bookmark	Ctrl+F2		▼ ×
			д 🕅	Call Stark	Ph.		61. C		л 🗙

Figure 36. Enter Show Disassembly at Address

Enter the address "0x08003000" of SLIB_INSTRUCTION start sector (sector 12);

Figure 37. Set Show Code at Address

Show Code at Address
Address: 0x08003000 <u>G</u> o To

• As shown in Figure 38, codes from 0x08003000 are all 0x00000000;

Disassembly			
0x08003000	0000	MOVS	r0,r0
0x08003002	0000	MOVS	r0,r0
0x08003004	0000	MOVS	r0,r0
0x08003006	0000	MOVS	r0,r0
0x08003008	0000	MOVS	r0,r0
0x0800300A	0000	MOVS	r0,r0
0x0800300C	0000	MOVS	r0,r0
0x0800300E	0000	MOVS	r0,r0
0x08003010	0000	MOVS	r0,r0
0x08003012	0000	MOVS	r0,r0
0x08003014	0000	MOVS	r0.r0

 Similarly, enter address "0x08001000" in "Memory" window, and codes are all 0x00, as shown in Figure 39.

Figure 39. View codes in Memory

Me	mory 1																
A	ddress:	0x0800	03000														
0 x	0800	3000:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x	0800	3022:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0 x	0800	3044:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0 x	0800	3066:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x	0800	3088:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

 In the "Memory" window, enter the address 0x08001000 of SLIB_READ_ONLY start sector (sector 1); this region is allowed to be read through D-Code bus, so that original values can be found, as shown in Figure 40.

Figure 40. SLIE	_READ_C	NLY start	sector in	n Memory
-----------------	---------	-----------	-----------	----------

Address:	0x0800	1000														
0x08001	.000:	40	EA	01	03	9B	07	03	DO	09	EO	08	C9	12	1F	08
0x08001	.022:	70	47	D2	B2	01	EO	00	F8	01	2B	49	1E	FB	D2	70
0x08001	.044:	20	46	10	BD	FO	В4	80	EA	01	02	D4	OF	42	00	B2
0 x 08001	066:	C1	F3	C7	52	AD	1A	20	2D	35	DA	C1	F3	16	01	41
0x08001	.088:	B3	EB	DO	5F	23	DO	C4	B1	01	2D	AO	EB	C3	50	09

• Click to modify the value of 0x08003000 in the code, and a warning message will be issued by setting EPPERR=1 in the FLASH_STS register, indicating the protection is enabled.

	Figure 41. SI	_IB write test	
	opp_oncolor	×	
	STS	0x0000010	
	ODF		
	EPPERR		
	PRGMERR		
	OBF		
1			

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

Figure 42. Write protection error interrupt





4 Integrate codes and download

After codes of the solution provider and end user are configured, 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 (0x08001000~0x08004FFF) by using the memory access function, and then click "File-Save Flash data as" to save codes as BIN or HEX file. In this example, it is named "slib.bin", as shown in Figure 43.

Exit FLASH_DA 0x 0 © Remap0 (Use PA11/PA12 pins) Type Select Remap1 (Use PB10/PB11 pins) Memory read settings Address 0x 08001000 Read size 0x 00004000 Data bits 8 bits Read File info Read Add Project_l0.hex Add Bits Add File info Project_l0.hex 8224 08000000-0800031B,08001000-08001D97,00 Delete Image: The size in the size		ncryptic	on file		985	03200	000B3	2905	9709	02					雅	1	恃	ノ
Type Select Remap1 (Use PB10/PB11 pins) Memory read settings Address 0x 00004000 tata bits 8 bits Read File info Add No. File name File size Address range(0x) Add I project_I0.hex 8224 08000000-0800031B,08001000-08001D97,0E Delete I Image: [0.hex Read Flash info File:project_I0.hex Address range: [0x08001000 0x08004FFF] checksum: 0x0037582A Address 0 1 2 3 4 6 7 8 2 1 File CRC verify DownLoad Flash info File:project_I0.hex Address range: [0x08001000 Address range: [0x08001000 A 6 7 6 7 8 6 7	Exit				FL	ASH_	DA 0	< 🗌	0		@ R	lema	50 (U:	se PA	11/PA	12 p	ins)	
Memory read settings Address 0x 08001000 Read size 0x 00004000 Data bits 8 bits Read File info Add Add Delete No. File name File size Address range(0x) Add 1 project_ 0.hex 8224 08000000-0800031B,08001000-08001D97,0E Delete III IIII Flash CRC File CRC verify DownLoad Flash info File:project_ 0.hex Address range:[0x08001000 0x08004FFF] checksum: 0x0037582A Address 0 1 2 3 4 5 6 7 8 9 A B C D E F A Address range:[0x08001000 0x08004FFF] checksum: 0x0037582A 0x08001000 40 EA 01 03 9B 07 03 D0 9E 0.8 C9 12 1F 08 C0 02 0x08001000 40 EA 01 03 9B 07 03 D0 9E 0.8 C9 12 1F 08 C0 02 0		Туре							Select	t	R	ema	p1 (U:	se PB	10/PE	311 p	ins)	
Address 0x 08001000 Read size 0x 00004000 Data bits 8 bits Read File info	Memory re	ad settir	ngs								1				_			
File info No. File name File size Address range(0x) Add 1 project_l0.hex 8224 08000000-0800031B,08001000-08001D97,0E Delete Image: 10.hex B224 08000000-0800031B,08001000-08001D97,0E Delete Flash CRC File CRC verify DownLoad Flash info File:project_l0.hex Address 0 1 2 3 4 5 6 7 8 9 A B C D E F A Address 0 1 2 3 4 5 6 7 8 9 A B C D E F A 6 7 8 9 A B C D E 7 8 7 9 A B C D E 7 8 9 A B C D E 7 8 9 A B C D E 7 8 9 A B C D E	Address 0x	08001	000	F	Read	size	0x 0	0004	000	()ata I	bits	8 bit	s 🔻]		Rea	d
No. File name File size Address range(0x) Add 1 project_l0.hex 8224 08000000-0800031B,08001000-08001D97,00 Delete III III IIII IIII IIII IIII IIII Delete	File info			_	_						1							
1 project_I0.hex 8224 08000000-0800031B,08001000-08001D97,00 Delete III IIII IIIII IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII	No. File	name				File	size	A	ddres	is ran	ge(0)	<)					4	Add
Image:	1 pro	ject_l0.h	ex			822	4	08	30000	00-0	80003	31B,0	80010	00-00	80010	097,0	E De	elete
Image:																		
Image:																		
Flash CRC File CRC verify DownLoad Flash info File:project_0.hex Address range:[0x08001000 0x08004FFF] checksum: 0x0037582A Address 0 1 2 3 4 5 6 7 8 9 A B C D E F A Address 0 1 2 3 4 5 6 7 8 9 A B C D E F A 6 Address 0 1 2 3 4 5 6 7 8 9 A B C D E F A 6 0x08001000 40 EA 01 03 9B 07 03 D0 9E 0 8 C9 12 1F 08 C0 @ C0 @ C0 @ C0 0 F D1 3B 52 1E D* Q<	•				III											+		
Flash info File:project_0.hex Address range:[0x08001000 0x08004FFF] checksum: 0x0037582A Address 0 1 2 3 4 5 6 7 8 9 A B C D E F A Address 0 1 2 3 4 5 6 7 8 9 A B C D E F A ^ Address 0 1 2 3 4 5 6 7 8 9 A B C D E F A ^ Address 0 1 2 3 4 5 6 7 8 9 A B C D E F A ^ Address 0 1 03 9B 07 03 D0 09 E0 08 C9 12 15 08 C0 © © 1 1 1 1 1 1 13 13									Flas	sh CR	с	Fi	le CR(C veri	fy	Do	wnL	oad
Address range:[0x08001000 0x08004FFF] checksum: 0x0037582A Address 0 1 2 3 4 5 6 7 8 9 A B C D E F Af 0x08001000 40 EA 01 03 9B 07 03 D0 09 E0 08 C9 12 1F 08 C0 ©? 0x08001000 40 EA 01 03 9B 07 03 D0 09 E0 08 C9 12 1F 08 C0 ©? 0x08001000 04 2A FA D2 03 E0 11 F8 01 3B 00 F8 01 3B 52 1E □* 0x08001020 F9 D2 70 47 D2 B2 01 E0 00 F8 01 2B 49 1E FB D2 p 0x08001030 70 47 00 22 F6 E7 10 B5 13	Flash info	ile:proje	ct_10.1	nex														
Address 0 1 2 3 4 5 6 7 8 9 A B C D E F A 0x08001000 40 EA 01 03 9B 07 03 D0 09 E0 08 C9 12 1F 08 C0 09 0x08001010 04 2A FA D2 03 E0 11 F8 01 3B 00 F8 01 3B 52 1E □* 0x08001020 F9 D2 70 47 D2 B2 01 E0 00 F8 01 2B 49 1E FB D2 r 0x08001030 70 47 00 22 F6 E7 10 B5 13 46 0A 46 19 46 pG 0x08001040 FF F7 F0 FF 20 46 10 BD F0 B4 80 EA 01 02 D4 9 F <th>- Address ran</th> <th>ge:[0x080</th> <th>01000</th> <th>0x0800</th> <th>04FFF]</th> <th> che</th> <th>ecksun</th> <th>n: 0x0</th> <th>03758</th> <th>2A —</th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th>	- Address ran	ge:[0x080	01000	0x0800	04FFF]	che	ecksun	n: 0x0	03758	2A —								
0x08001000 40 EA 01 03 9B 07 03 D0 09 E0 08 C9 12 1F 08 C0 @? 0x08001010 04 2A FA D2 03 E0 11 F8 01 3B 00 F8 01 3B 52 1E □* 0x08001020 F9 D2 70 47 D2 B2 01 E0 00 F8 01 2B 49 1E FB D2 r 0x08001030 70 47 00 22 F6 E7 10 B5 13 46 0A 46 19 46 pG 0x08001040 FF F7 F0 FF 20 46 10 BD F0 B4 80 EA 01 02 D4 0F I 0x08001040 FF F7 F0 FF 20 46		0	1	2	3	4	5	6	7	8	9	Α	В	С	D	E	F	A:
Ox08001010 04 2A FA D2 03 E0 11 F8 01 3B 00 F8 01 3B 52 1E IF 0x08001020 F9 D2 70 47 D2 B2 01 E0 00 F8 01 3B 52 1E IF 0x08001020 F9 D2 70 47 D2 B2 01 E0 00 F8 01 2B 49 1E FB D2 1 0x08001030 70 47 00 22 F6 E7 10 B5 13 46 0A 46 04 46 19 46 pG 0x08001040 FF F7 F0 FF 20 46 10 BD F0 B4 80 EA 01 02 D4 0F 1 <td>Address</td> <td></td> <td>EA</td> <td>01</td> <td>03</td> <td>9B</td> <td>07</td> <td>03</td> <td>D0</td> <td>09</td> <td>EO</td> <td>08</td> <td>C9</td> <td>12</td> <td>1F</td> <td>08</td> <td>C0</td> <td>@1</td>	Address		EA	01	03	9B	07	03	D0	09	EO	08	C9	12	1F	08	C0	@1
Ox8001020 F9 D2 70 47 D2 B2 01 E0 00 F8 01 25 49 12 F8 D2 F8 01 25 49 12 F8 D2 F 0x08001030 70 47 00 22 F6 E7 10 B5 13 46 04 46 19 46 pG 0x08001040 FF F7 F0 FF 20 46 10 BD F0 B4 80 EA 01 02 D4 0F 1 4 IIII IIII	Address 0x08001000	40			D2	03	EO	11	F8	01	3B	00	F8	01	3B	52	1E	•
Ox08001040 FF F7 F0 FF 20 46 10 BD F0 B4 80 EA 01 02 D4 0F I 4 III IIII III III IIII IIII IIII IIII IIII IIII IIII IIII IIII IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII	Address 0x08001000 0x08001010	40	2A	FA 70	47	02	60	01	EO		1 6	101	20	49	10	FD	102	F
Image: state	Address 0x08001000 0x08001010 0x08001020 0x08001030	40 04 F9 70	2A D2 47	70 00	47	D2	B2 F7	01 10	E0 B5	13	46	0A	46	04	46	19	46	pG
14-50-27 - Verification successfully 1.1	Address 0x08001000 0x08001010 0x08001020 0x08001030 0x08001040	40 04 F9 70 FF	2A D2 47 F7	70 70 00 F0	47 22 FF	D2 F6 20	B2 E7 46	01 10 10	E0 B5 BD	13 F0	46 B4	0A 80	46 EA	04 01	46 02	19 D4	46 0F	pG
IA 17 1/ VELULAUUU NULESSUUUVI I	Address 0x08001000 0x08001010 0x08001020 0x08001030 0x08001040	40 04 F9 70 FF	2A D2 47 F7	FA 70 00 F0	47 22 FF	D2 F6 20	B2 E7 46	01 10 10	E0 B5 BD	13 F0	46 B4	0A 80	46 EA	04 01	46 02	19 D4	46 0F	pG
	Address 0×08001000 0×08001000 0×08001020 0×08001030 0×08001040 	40 04 F9 70 FF	2A D2 47 F7 Juccess	FA 70 00 F0 	47 22 FF 	D2 F6 20 	B2 E7 46	01 10 10	E0 B5 BD	00 13 F0	46 B4	0A 80	46 EA	04 01	46 02	19 D4	46 0F	pG

Figure 43. 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_INST.bin" and "ER_SLIB_READ_ONLY.bin", corresponding to the SLIB-INST file (0x08003000) and SLIB-READ-ONLY file (0x08001000), as shown in Figure 44.

		_					
Figure	44.	Generate	.bin	tile	ot	SLIB	code

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		1	Not Specified		
🗌 Run #2		1	Not Specified		
After Build/Rebuild					
🔽 Run #1	fromelf.exebinoutput .\Listings\@L.bin !L	1	Not Specified		EK_SLIB_INST
Run #2		1	Not Specified		

(2) Use ICP Programmer to program the .bin file to MCU, as shown in Figure 45.

ne y carac	setting	s A	T-Lir	ık set	tings	T	arget	La	ngua	ge H	elp						sLi	b status								
Disconnect	Part I	Numb	er:	AT3	2F42	1C81	7	Flas	hSize	64KB			47	5 L:	= 1	7 7		ib status: Dis	sable							
	AT-Li	nk-EZ		FW:	V1.5.1	L2						 				<u>.</u>	En	able passwor	d 0x 5566	5566	sLi	o positio	on:	Main Flas	h	V
T-Link 🔻	AT-Li	nk SN	: 6	C 9 850	3200	00B3	29059	7090	2			1	脽	キ	Ŧ	力	Di	sable passwo	rd 0x		Sta	rt secto	r s	Sector4	0x080010	• 00
	SPI	м		FL	ASH_E	DA 0		0	(Rema	p0 (Us	e PA1	1/PA1	2 pins	s)				Dis	able sLib	INS	STR star	t sector S	Sector12-	-0x08003	• 000
	Type							elect		Rema	p1 (Us	e PB1	0/PB1	· 1 pins	s)						End	d sector		Sector19-	-0x08004	C00 -
Memory rea	d settin	gs															Ex	tra options								
Address 0x	080000	- 00	R	ead s	ize ()x 0	00003	1C	Da	a bits	8 bits	, ,		R	Read		E E	Erase the sect	ors of file s	ize		•	🔲 Disal	ble sLib b	efore do	wnload
															_			Verify	Custom e	nervation k	rev for v	erify:	V Enab	le sLib		
File info																		, Г	castom e	nei)puon i	,	ciniy.	Direl		afava dav	اسمام
No. File :	ame							Fil	e size	Ade	ress r	ange(0x)		A	dd							Disa	DIE FAP D	elore dov	whitead
1 50 5	LIB_REA	D_ON	ILY.bi	n				34	80	080	01000	-0800	1D97		De	lata	E	Jump to the	user prog	ram			📃 Enab	le FAP af	ter downl	oad
1 11_3																iete j										
2 ER_S	LIB_INS	T.bin						26	8	080	03000	-0800	310B		_	iete		Write softw	are serial n	umber(SN)			Basi	ic access	protectio	n 🔻
2 ER_S	LIB_INS	T.bin						26	8	080	03000	-08003	310B					Write softw	are serial n	umber(SN)			Basi	ic access on free m	protectio ode	n 🔻
2 ER_S	LIB_INS	T.bin						26	8	080	03000	-08003	310B					Write softw Write add	dress 0x	umber(SN) 08010000			Basi	ic access on free m	protectio ode	n 🔻
2 ER_S	LIB_INS	T.bin						26 Flash	8 CRC	080	03000	-08003	310B	Dow	mLc	bad		Write softwo Write add Current S	are serial n dress 0x	umber(SN) 08010000 00000001			Basi	ic access on free m	protectio ode	n 🔻
2 ER_S	LIB_INS	T.bin	ST bir					26 Flash	8 CRC	080	03000 le CRG	-0800: Cverify	310B	Dow	'nLo	ad		Write softw Write add Current S Increase	are serial n dress 0x [:N 0x [step 0x [umber(SN) 08010000 00000001 00000001			Basi	ic access on free m	protectio ode	n v
2 ER_S	le:ER_SL	T.bin IB_IN 3000 0	ST.bir	31081	che	cksun	n: 0x00	26 Flash	8 CRC	080	03000 le CRO	-0800: C verify	310B	Dow	'nLc	ad		Write softw Write add Current S Increase	are serial n dress 0x [:N 0x [step 0x [umber(SN) 08010000 00000001 00000001			Basi	ic access on free m	protectio ode	n 🔻
2 ER_S	LIB_INS	T.bin IB_IN 3000 0	ST.bir x0800	1 310B]	che	cksun	n: 0x00	26 Flash	8 CRC	080	le CRC	-0800: C verify	310B	Dow	'nLc	ad		Write softw Write add Current S Increase	are serial n dress 0x [N 0x [step 0x]	umber(SN) 08010000 00000001 00000001			Basi	ic access on free m	protectio ode	n 🔻
2 ER_S Elash info Fi Address range	LIB_INS	T.bin IB_IN 3000 0	ST.bir x0800	310B]	che 4	cksun 5	n: 0x00	26 Flash 000718 7	8 CRC 2 8	080	le CRO	-0800: C verify	310B	Dow	/nLc	ad		Write softw Write add Current S Increase Write user s User system	are serial n dress 0x [N 0x [step 0x] system data n data file p	umber(SN) 08010000 00000001 00000001			Basi	ic access	protectio	n v
2 ER_S 2 ER_S Flash info Fi Address range Address	liB_INS le:ER_SL e:[0x0800 2D 2D	T.bin IB_INS 3000 0 1 E9	ST.bir x0800 2 FF	310B] 3 47	che 4 06	cksun 5 46	n: 0x00	26 Flash 00718 7 46	8 CRC 2 8 90 4	080	B 25	C verify	310B	Dow E 1 58 1	rnLa F	As A		Write softw Write add Current S Increase Write user e	are serial n dress 0x [iN 0x [step 0x] system data n data file p	umber(SN) 08010000 00000001 00000001			Basi	ic access on free m	protectio ode	n v
Elash info Fi Address Address 0x08003000 0x08003010 0x08003010	LIB_INS	T.bin IB_INS 3000 0 1 E9 F2 21	ST.bir x0800 2 FF 00	310B] 347 03	che 4 06 C2	cksun 5 46 F2	n: 0x00 6 0F 00	26 Flash 000718 7 46 03	8 CRC 2 8 90 4 41 F	080 Fi	B 25 42 24	-0800: verify C 4F C0 00	D EA F6	Dow E 1 58 1 00 0	rnLc F 19	A: ^ -7(E C?:	C	Write softw Write add Current S Increase Write user s User system	are serial n dress 0x (iN 0x (step 0x (system data n data file p	umber(SN) 08010000 00000001 00000001			Basi	ic access on free m	protectio ode	n v
1 EIG3 2 ER_S 2 ER_S 4 EIG3 6 EIG3 7 EIG3 8 EIG3	le:ER_SL 2D 43 1D	T.bin IB_INS 3000 0 E9 F2 21 E0	2 FF 00 01	310B] 3 47 03 A8	che 4 06 C2 00	cksun 46 F2 95	6 0F 00 00	26 Flash 000718 7 46 03 F0	8 CRC 2 8 90 4 41 F 5C F	080 Fi 6 20 6 E0 8 00 0 06	B 25 42 24 EB	-0800: C verify C 4F C0 0C 80	D EA E0	Dow E 1 58 1 00 0 04 F	rnLc F 19 22 FB	At ^ -?(= C?:		Write softw Write add Current S Increase Write user s User system	are serial n dress 0x (iN 0x (step 0x (system data n data file p	umber(SN) 08010000 00000001 00000001			Basi	ic access on free m	protectio ode	n
I ER-3 ER-S ER-S	LIB_INS	T.bin IB_INS 3000 0 1 E9 F2 21 F0 A8	ST.bir x0800 FF 00 01 07 00	310B] 347 03 A8 EB E0	che- 4 06 C2 00 80 05	cksun 5 46 F2 95 02 F8	n: 0x00 6 0F 00 00 00 04 64	26 Flash 1000718 7 46 03 F0 FB	CRC 2 8 990 4 41 F 55C F 505 F	080 Fi 6 20 6 E0 8 00 0 06 5 E0	B 25 42 24 EB D3	-08003 C verify 4F C0 0C 80 BD	D EA F6 01 F8	Dow E 1 58 1 00 0 04 F 2B 4 FF 8	F 19 22 FB 46	At ^ -?(E C?; D?; D?;	E	Write softw Write add Current S Increase Write user s	are serial n dress 0x [N 0x [step 0x [system data n data file p	umber(SN) 08010000 00000001 00000001			Basi	ic access on free m	protectio ode	n v

Figure 45. Online programming to MCU in ICP

(3) End users also can use ICP Programmer to set an offline project and save it to AT-Link, and then complete offline programming to MCU through AT-Link, as shown in Figure 46.

Offline project		• Delete	Creat	
Project name slib_project_l0		Device AT32F421	▼ AT32F421C8T7	•
No. File name	File size	Address range(0x	() Storag e loca Add	
1 ER_SLIB_READ_ONLY.bin	3480	08001000-080010	D97 Delete	,
2 ER_SLIB_INST.bin	268	08003000-080031	LOB	
•			4	
Erase option Erase the sectors of fil	e size	•		
Download times		Download interf	ace SWD -	
Encryption transmit	Verify	Reset and ru	n	
Write user system data				ר
Enable FAR after download		Sustem mem	on: AP mode	
Basic access protection		Kev:(0x)	(0vA25E6D2	A)
	_	(integration)	(0,4551052	"
Software serial number(SN) SPIM	settings	sLib settings		_
☑ Enable sLib		slib position	Main Flash	_
sLib enable password 0x 55665	566	Start cactor	Sector40x08001000	
Disable sLib before download		INCTR start as star	Sector120x08003000	
sLib disable password 0x		INSTR start sector	Sector12-0x08003000 +	
		End sector	Sector190x08004c00	
		Load para	meters Save parameters	
Open project Save project file]	Save	project to AT-Link Close	ן ר
	J		p	

Figure 46. 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 47.



	1													clib status. Enable
Disconnect	Part N	lumber	AT3	2F421	C8T7	Flas	Size:	64KB		;	<i>i</i> v	21 =	<u>ک</u>	Enable password 0x 55665566 sLib position: Main Flash
T Link	AT-Lir	ık-EZ ık SN:	FW: 6C985	V1.5.12 032000) B3290	5970902				1	I	化 主	-	Disable password 0x Start sector Sector40x08001000
I-LINK ¥											IE	1न		Disable sLib INSTR start sector Sector120x08003000
		Л	FL	ASH_D	\ 0x	0								End sector Sector190x08004C00
	Туре					Select								
Memory rea	d setting	JS									_			Extra options
Address 0x	0800100	00	Read	size Os	00000	D98	Data	bits	8 bits	•		Re	ad	Erase the sectors of file size
File info														Verify Custom encryption key for verify: Enable sLib
No File	name			File ci	· •	ddrace	rangel	∩~ 1					Add	Disable FAP before download
1 pro	ect l1.he:	< .		7000	0	800000	-0800	0307.08	300500	00-080	0684F) elete	Jump to the user program Enable FAP after download
	-				-								Velete	Write software serial number(SN)
														Button free mode
														Write address 0x 08010000
•												P.		
•			III		ſ	fleek	CRC] [=:	- 000			•		Current SN 0x 00000001
•		. 14.1			[Flash	CRC	Fil	e CRC	verify	· [) Down	Load	Current SN 0x 00000001 Increase step 0x 00000001
Flash info F	le:projec	t_11.hex		Addr	[Flash	CRC	Fil	e CRC	verify		Down	Load	Current SN 0x 00000001 Increase step 0x 00000001
Flash info F Address rang	le:projec e:[0x08000	t_l1.hex	000307]	Addre	ss range	Flash :[0x0800	CRC 5000 0x	Fil 0800684	e CRC IF] ch	verify	r (Down	Load	Current SN 0x 00000001 Increase step 0x 00000001
 Flash info F Address rang Address 	le:projec e:[0x08000	t_l1.hex 0000 0x08 1 2	000307]	Addre 4	ss range	Flash :(0x0800	CRC 5000 0x 8 9	Fil 0800684	e CRC IF] ch B	verify necksur	m: 0x00	Down D0A4D7 F	Load	Current SN 0x 00000001 Increase step 0x 00000001 Write user system data User system data file path
Flash info F Address rang Address 0x08000000	le:projec e:[0x08000 38 67	t_l1.hex 0000 0x08 1 2 13 00	111 0000307] 3 20 08	Addre 4 01 5	ss range	Flash :[0x0800 7 08	CRC 5000 0x 8 9 38 57	Fil 0800684 A 00	e CRC IF] ch B 08	verify	m: 0x00 D E 57 0	Down D0A4D7 F 0 08 0	Load 7 80 #	Current SN 0x 00000001 Increase step 0x 00000001 Write user system data User system data file path
Flash info F Address rang Address 0x0800000 0x08000010	le:projec e:[0x08000 38 C7 00	t_l1.hex 0000 0x08 1 2 13 00 57 00 00 00	111 0000307] 3 20 08 00	Addre 4 01 5 A3 5	ss range 6 6 0 00 7 00	Flash :[0x0800 7 08 08	CRC 5000 0x 8 9 33 57 11 58 0 00	Fil 0800684 A 00 00	e CRC IF] ch B 08 08	verify necksur C3 00	m: 0x00 D E 57 0 00 0	Down 00A4D7 F 0 08 0 00 0 00	Load 7 名: 登.	Current SN 0x 00000001 Increase step 0x 00000001 Write user system data User system data file path
Flash info F Address rang Address 0x0800000 0x08000010 0x08000020 0x0800000000 0x080000000000	lle:projec e:[0x08000 38 C7 00 A7	t_l1.hex 0000 0x08 1 2 13 00 57 00 00 00 57 00	111 0000307] 3 20 08 00 08	Addre 4 01 5 A3 5 00 0	ss range 6 6 0 00 7 00 0 00 0 00	Flash :[0x0800 7 08 08 00 00	CRC 5000 0x 8 9 11 58 10 00 57	Fil 0800684 00 00 00 00	e CRC IF] ch 08 08 08 00 00	C C C C C C C C C C C C C C C C C C C	m: 0x00 D E 57 0 00 0 57 0 57 0	Down 00A4D7 F 0 08 0 00 0 08 0 08 0 08	Load 7 8回 鼓	Current SN 0x 00000001 Increase step 0x 00000001 Write user system data User system data file path

Figure 47. 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 as an encrypted slib.benc file 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.



	Delete Creat	
Project name slib_project	Device AT32F421 V AT32F421C8T7 V	
No File name	File size Address range(0x) Storage loca Add	
1 ER_SLIB_READ_ONLY.bin	3480 08001000-08001D97 Delete	
2 ER_SLIB_INST.bin	268 08003000-08003108	
<		
Erase option Erase the sectors of	file size 🔹	
Download times	Download interface SWD -	
Encryption transmit	Verify 🔲 Reset and run	
 Encryption transmit Write user system data 	Verify Reset and run	
Encryption transmit Write user system data Enable FAP after download	Verify Reset and run	
Encryption transmit Write user system data Enable FAP after download Basic access protection	Verify Reset and run System memory AP mode Key:(0x) (0xA35F6D24)	
Encryption transmit Write user system data Enable FAP after download Basic access protection	Verify Reset and run System memory AP mode Key:(0x) (0xA35F6D24)	
 Encryption transmit Write user system data Enable FAP after download Basic access protection Software serial number(SN) SPI 	Verify Reset and run System memory AP mode Key:(0x) (0xA35F6D24) A settings sLib settings	
 Encryption transmit Write user system data Enable FAP after download Basic access protection • Software serial number(SN) SPII ✓ Enable sLib 	Verify Reset and run System memory AP mode Key:(0x) (0xA35F6D24) settings sLib settings sLib position Main Flash	
 Encryption transmit Write user system data Enable FAP after download Basic access protection • Software serial number(SN) SPII Enable sLib sLib enable password 0x 55560 	Verify Reset and run System memory AP mode Key:(0x) stib settings stib settings stib position Main Flash Start sector Start sector	
 Encryption transmit Write user system data Enable FAP after download Basic access protection • Software serial number(SN) SPII Enable sLib sLib enable password 0x 5566 Disable sLib before download 	Verify Reset and run System memory AP mode Keys(0x) A settings SLib settings Stofe Start sector INSTR sta	AT-Link project file settings
 Encryption transmit Write user system data Enable FAP after download Basic access protection • Software serial number(SN) SPII Enable slib slib enable password 0x 5566 Disable slib before download slib disable password 0x 	Verify Reset and run System memory AP mode Keys(0x) A settings SLib settings Stib position Main Flash Start sector INSTR start sector Sector120x08001000 End sector Sector190x08004000	AT-Link project file settings
 Encryption transmit Write user system data Enable FAP after download Basic access protection • Software serial number(SN) SPII Enable slib slib enable password 0x 5566 Disable slib before download slib disable password 0x 	✓ Verify Reset and run System memory AP mode Keys(0x) (0xA35F6D24) A settings sLib settings stib position Main Flash S566 Start sector Sector40x08001000 ▼ INSTR start sector Sector120x08003000 ▼ End sector Sector190x08004C00 ▼ Intervention Interventin </th <th>AT-Link project file settings</th>	AT-Link project file settings
Encryption transmit Write user system data Enable FAP after download Basic access protection Software serial number(SN) SPII Enable sLib sLib enable password 0x 5566 Disable sLib before download sLib disable password 0x	✓ Verify Reset and run System memory AP mode Keys(0x) (0xA35F6D24) A settings sLib settings stib position Main Flash S5566 Start sector Start sector Sector120x08001000 ▼ INSTR start sector Sector120x08003000 ▼ End sector Sector190x08004C00 ▼ Load parameters Save parameters	AT-Link project file settings
 Encryption transmit Write user system data Enable FAP after download Basic access protection Software serial number(SN) SPII Enable slib slib enable password 0x Disable slib before download slib disable password 0x 	Verify Reset and run System memory AP mode Keys(0x) (0xA35F6D24) A settings sLib settings stib position Main Flash S566 Start sector Start sector Sector120x08001000 End sector Sector120x08004000 Load parameters Save parameters	AT-Link project file settings Image: This project is only used at the specified AT-Link. AT-Link SN : D4C354230040B56D01973402
 Encryption transmit Write user system data Enable FAP after download Basic access protection Software serial number(SN) SPII Enable slib slib enable password 0x Disable slib before download slib disable password 0x 	Verify Reset and run System memory AP mode Keys(0x) (0xA35F6D24) A settings sLib settings sLib position Main Flash 5566 Start sector Start sector Sectorial0x08001000 End sector Sectorial0x0800000 Load parameters Save parameters	AT-Link project file settings

Figure 48. 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 49 shows how to add the project file.

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



Offline project		• Delete	Crea	at
Project name slib_project	D	evice AT32F421	▼ AT32F421C8T7	
No. File name 1 ER_SLIB_READ_ONLY.bin 2 ER_SLIB_INST.bin 3 project 11 box	File size 3480 268 776 III	Address range(0x 08001000-08001D 08003000-080031) Storage loc: A /	Add
 Download times Encryption transmit Write user system data Enable FAP after download Basic access protection Software serial number(SN) SPIM] Verify settings SI	Download interfa Reset and rur System memory Key:(0x)	ory AP mode (0xA35F	 F6D24)
 ✓ Enable sLib sLib enable password 0x ✓ Disable sLib before download sLib disable password 0x 	t**	sLib position Start sector INSTR start sector End sector	Main Flash	
Open project Save project file This project is only used once.]	Load parar	project to AT-Link	lose

Figure 49. Add project file



5 Revision history

Table 2. Document revision history

Date	Version	Revision note
2021.11.2	2.0.0	Initial release.
2023.03.21	2.0.1	Modified screenshots.

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.

© 2023 Artery Technology -All rights reserved