

Application Note

Document No.: AN1187

Quick Start Guide to

APM32F402_403_DAL_SDK

Version: V1.0

1. Introduction

APM32F402_403 SDK (Software Development Kit) is a comprehensive set of development drivers and documents designed specifically for the APM32F402_403 series MCUs. It includes a low-level peripheral driver library, a collection of examples for different development boards, and a rich set of middleware. The SDK aims to significantly improve developer productivity by reducing development workload, time, and cost.

The materials referenced in this application note are:

[AN1080 APM32 Series Toolchain User Guide](#)

[Eclipse Development Tutorial under AN1093 APM32 Arm MCU Windows System](#)

Note: These materials can be obtained from the official website www.geehy.com.

Contents

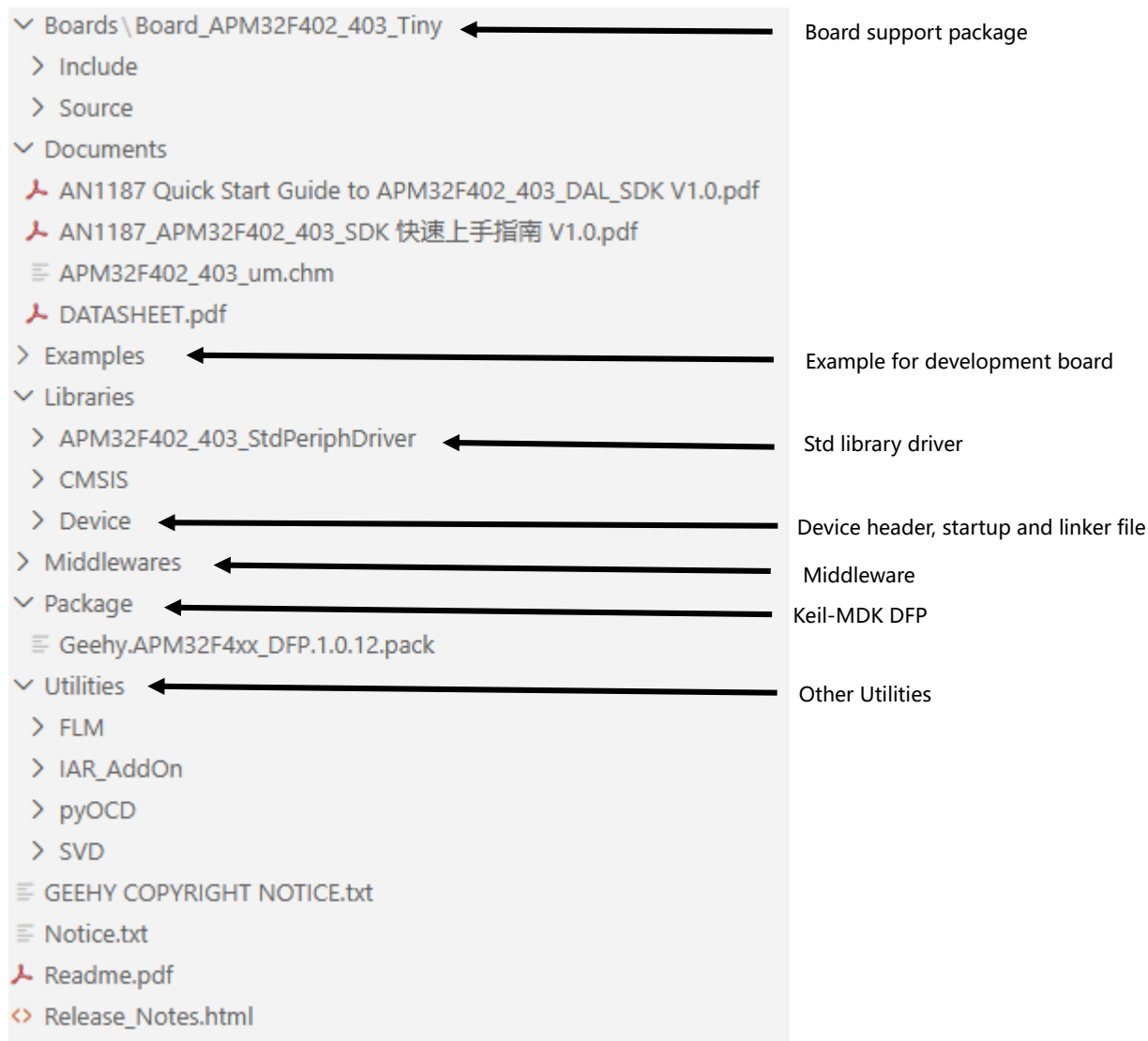
1.	Introduction	1
2.	Content about SDK	4
3.	Content about Boards	5
4.	Content about Documents	6
5.	Content about Examples	7
6.	Content about Libraries.....	11
6.1.	APM32F402_403_StdPeriphDriver.....	11
6.2.	CMSIS.....	13
6.3.	Device	13
7.	Content about Middlewares	14
7.1.	CMSIS-FreeRTOS.....	14
7.2.	FatFs.....	14
7.3.	USB_Library	14
8.	Content about Package	15
8.1.	SVD	15
8.2.	Pack Package.....	15
9.	Content about Utilities	16
9.1.	FLM.....	16
9.2.	IAR_AddOn	16
9.3.	pyOCD.....	16
9.4.	SVD	16
10.	Quick Start	17
10.1.	Template Project.....	18
10.2.	SDK-related Macro Definitions	19
10.3.	Device Initialization	19
10.4.	Compilation and Download.....	20
11.	Revision History	25

12.	Annex	26
12.1.	Uninstalling and Installing Keil pack	26

2. Content about SDK

The complete "SDK" directory is as follows:

Figure 1 "SDK" Directory



3. Content about Boards

The "Boards" folder contains the BSP (Board Support Package) for the APM32F402_403 development board. It can help to quickly drive the peripheral circuits or components on the board. The BSP can be found in the [~/Boards](#) directory.

The provided BSP is built for the APM32F402_403 mainboard. For other user-developed boards, some minor modifications may be required.

The directory structure of the " APM32F402_403 SDK Boards" folder is as follows:

- "Board_ APM32F402_403_Tiny" folder

4. Content about Documents

The "Documents" folder contains a link file that can redirect to Geehy Semiconductor's Technical Support Center. The library support documentation can be found in the [/Documents](#) directory. This file can be used to view all supported function explanations, parameter functions, and return values.

The directory structure of the "APM32F402_403 SDK Documents" folder is as follows:

- DATASHEET.pdf
- APM32F402_403_um.chm
- 快速上手指南.pdf
- Quick Start Guide.pdf

5. Content about Examples

Example applications can be found in the [~/Examples](#) directory.

The provided examples are built for the APM32F402 Tiny、APM32F403 Tiny development board. For use on other user-developed boards, appropriate adjustments are required.

The directory structure of the "APM32F402 SDK Example" folder is as follows:

- Include
- Project
 - Eclipse
 - IAR
 - MDK
- Source
- Clean Project.bat
- readme.txt

All example applications are tested using APM32F402 SDK V1.0.0, including the following examples:

Table 1 List of Example Programs Supported by the Development Board

IP / Module	Example	APM32F402 Tiny	APM32F403 Tiny	NA	NA
ADC	ADC_AnalogWindowWatchdog	√	√		
	ADC_ContinuousConversion	√	√		
	ADC_DMA	√	√		
	ADC_DualRegulSimulMode	√	√		
	ADC_InJectTimerTrig	√	√		
	ADC_MultiChannelScan	√	√		
	ADC_TemperatureSensor	√	√		
BAKPR	BAKPR_Tamper	√	√		
CAN	CAN_LoopBack	√	√		
CRC	CRC_Calculation	√	√		
	CRC_Calculation	√	√		

IP / Module	Example	APM32F402 Tiny	APM32F403 Tiny	NA	NA
DMA	DMA_MemoryToMemory	√	√		
EINT	EINT_Config	√	√		
FMC	FMC_Protection	√	√		
	FMC_Read_Write	√	√		
	FMC_Write	√	√		
GPIO	GPIO_Toggle	√	√		
I2C	I2C_TwoBoardsDMA	√	√		
	I2C_TwoBoardsInterrupt	√	√		
	I2C_TwoBoardsPolling	√	√		
I2S	I2S_TwoBoardsInterrupt	√	√		
IAP	IAP_Application1	√	√		
	IAP_Application2	√	√		
	IAP_BootLoader	√	√		
IWDT	IWDT_Reset	√	√		
NVIC	NVIC_Priority	√	√		
	NVIC_WFI	√	√		
PMU	PMU_Consumption	√	√		
	PMU_Standby	√	√		
	PMU_Stop	√	√		
RCM	RCM_ClockConfig	√	√		
RTC	RTC_Alarm	√	√		
	RTC_Second	√	√		
RTOS	CMSIS_FreeRTOS	√	√		
	CMSIS_RTX	√	√		
SPI	SPI_FullDuplex	√	√		
SysTick	SysTick_TimeBase	√	√		

IP / Module	Example	APM32F402 Tiny	APM32F403 Tiny	NA	NA
Template	Template	√	√		
TMR	TMR_6Steps	√	√		
	TMR_32bitCount	√	√		
	TMR_CascadeSynchro	√	√		
	TMR_EncoderInterface	√	√		
	TMR_ExtTriggerSynchro	√	√		
	TMR_InputCapture	√	√		
	TMR_OCAActive	√	√		
	TMR_OCInactive	√	√		
	TMR_OCToggle	√	√		
	TMR_ParallelSynchro	√	√		
	TMR_PWMInput	√	√		
	TMR_PWMOutput	√	√		
	TMR_SinglePulse	√	√		
	TMR_TimeBase	√	√		
	TMR_TMR1DMABurst	√	√		
	TMR_TMR1PWMOutput	√	√		
	TMR_TMR1Synchro	√	√		
	TMR_TMR2PWMOutput	√	√		
	TMR_TMR3PWMOutput	√	√		
	TMR_TMR8DMA	√	√		
USART	USART_IrDA	√	√		
	USART_LIN	√	√		
	USART_Printf	√	√		
	USART_Smartcard	√	√		
	USART_TwoBoardsDMA	√	√		

IP / Module	Example	APM32F402 Tiny	APM32F403 Tiny	NA	NA
	USART_TwoBoardsInterrupt	√	√		
	USART_TwoBoardsPolling	√	√		
USB_OTG	OTGD_CDC	√	√		
	OTGD_Composite_CDC	√	√		
	OTGD_Composite_CDC_WINUSB	√	√		
	OTGD_Custom_HID	√	√		
	OTGD_Custom_HID_Keyboard	√	√		
	OTGD_HID	√	√		
	OTGD_HID_Keyboard	√	√		
	OTGD_MSC	√	√		
	OTGD_MSC_IAP	√	√		
	OTGD_WINUSB	√	√		
	OTGH_CDC	√	√		
	OTGH_DynamicSwitch	√	√		
	OTGH_HID	√	√		
	OTGH_MSC	√	√		
	OTGH_MSC_FWUpgrade	√	√		
WWDT	WWDT_OverTime	√	√		

6. Content about Libraries

The "Libraries" folder contains a series of library files. It can provide support for APM32F402_403 MCU, such as device support, startup files, linker files, and Std peripheral support. The library files can be found in the [~/Libraries](#) directory.

The "APM32F402_403 SDK Libraries" folder includes the following materials:

- APM32F402_403_Driver
- CMSIS
- Device

6.1. APM32F402_403_StdPeriphDriver

This folder contains all Std peripheral library driver files.

Table 2 List of Std Drivers Supported by the Device

IP / Module	APM32F402	APM32F403	NA	NA
apm32f402_403_adc	√	√		
apm32f402_403_bakpr	√	√		
apm32f402_403_can	√	√		
apm32f402_403_crc	√	√		
apm32f402_403_dbgmcu	√	√		
apm32f402_403_dma	√	√		
apm32f402_403_eint	√	√		
apm32f402_403_fmc	√	√		
apm32f402_403_gpio	√	√		
apm32f402_403_i2c	√	√		
apm32f402_403_iwdt	√	√		
apm32f402_403_misc	√	√		
apm32f402_403_pmu	√	√		
apm32f402_403_rcm	√	√		
apm32f402_403_rtc	√	√		
apm32f402_403_spi	√	√		

IP / Module	APM32F402	APM32F403	NA	NA
apm32f402_403_tmr	√	√		
apm32f402_403_usart	√	√		
apm32f402_403_usb	√	√		
apm32f402_403_usb_device	√	√		
apm32f402_403_usb_host	√	√		
apm32f402_403_wwdt	√	√		

6.1.1. Std peripheral library

The Std peripheral library provides peripheral development interfaces in the form of regular functions.

Regular functions are declared in the header files of the Std peripheral library and implemented in the source files. They are responsible for operations such as peripheral parameter configuration, function control, data transmission and reception, status reading, as well as interrupt and flag management.

The common naming characteristics of regular functions are shown in the following table:

Table 3 Naming Characteristics of Std Inline Functions

Scenario	Function	Naming Characteristic
General configuration	Peripheral/module configuration	PPP_Config
	Subfunction configuration	PPP_ConfigXXX
	Structure initialization	PPP_ConfigStructInit
	Default initialization of a configuration	PPP_ConfigXXXStructInit
Reset	Peripheral/module reset	PPP_Reset
	Subfunction reset/clear	PPP_ResetXXX / PPP_XXXReset
Enable control	Enable a peripheral/module	PPP_Enable
	Disable a peripheral/module	PPP_Disable
	Enable a subfunction	PPP_EnableXXX
	Disable a subfunction	PPP_DisableXXX
Event generation	Generate an event directly	PPP_GenerateXXX
	Enable generation of a condition/signal	PPP_EnableGenerateXXX
	Disable generation of a condition/signal	PPP_DisableGenerateXXX
Flag	Read a status flag	PPP_ReadStatusFlag

	Clear a status flag	PPP_ClearStatusFlag
Interrupt	Enable an interrupt	PPP_EnableInterrupt
	Disable an interrupt	PPP_DisableInterrupt
	Read an interrupt flag	PPP_ReadIntFlag
	Clear an interrupt flag	PPP_ClearIntFlag
DMA	Enable DMA	PPP_EnableDMA
	Disable DMA	PPP_DisableDMA
	Configure DMA	PPP_ConfigDMA
	Enable a DMA subfunction	PPP_EnableDMAXXX
	Disable a DMA subfunction	PPP_DisableDMAXXX

6.2. CMSIS

This folder contains header files related to the Arm® Cortex®-M4F core, which define the registers and functions of the core.

6.3. Device

This folder contains device-specific files related to the APM32F402_403. Wherein:

- The "Geehy\APM32F4xx\Include" folder contains all APM32F402_403 device header files, which define the device registers and interrupt vectors.
- The "Geehy\APM32F4xx\Source" folder contains startup files and linker files for the MDK, IAR, and GCC platforms, including the reset vector table and startup code.

7. Content about Middlewares

The "Middlewares" folder contains a series of third-party middleware. Such middleware can be found in the [~/Middlewares](#) directory.

The middleware used in the "APM32F402_403 SDK Middlewares" folder includes the following materials:

- CMSIS-FreeRTOS

7.1. CMSIS-FreeRTOS

The "CMSIS-FreeRTOS folder" contains the FreeRTOS operating system compliant with the CMSIS-RTOS2 standard.

7.2. FatFs

The "FatFs folder" contains the FatFs file operation system.

7.3. USB_Library

The "USB_Library folder" contains the USB development framework.

8. Content about Package

The "Package" folder includes the Geehy APM32F402_403 DFP Package. This package can be found in the [~/Package](#) directory.

The "APM32F402_403 DAL SDK Libraries" folder includes the following materials:

- Geehy. APM32F402_403_DFP.1.0.12.pack

8.1. SVD

This folder contains SVD files, which allow users to simulate and view register usage.

8.2. Pack Package

This folder stores the latest version of the Keil pack package, for use in the MDK-ARM environment to support the APM32F402_403 series chips. The latest version of the Keil pack package can also be obtained from the following link:

<https://www.keil.arm.com/packs/>

<https://www.geehy.com/design/software>

9. Content about Utilities

The "Utilities" folder includes some utilities. These utilities can be found in the [~/Utilities](#) directory.

The "APM32F402_403 SDK Utilities" folder includes the following materials:

- FLM
- IAR_AddOn
- pyOCD
- SVD

9.1. FLM

This folder stores the FLM files.

9.2. IAR_AddOn

This folder stores the IAR AddOn exe files

9.3. pyOCD

This folder stores the python script files for pyOCD.

9.4. SVD

This folder contains SVD files, which allow users to simulate and view register usage.

10. Quick Start

To quickly EVBuate the APM32F402_403 SDK, you may need to prepare the following environment or content:

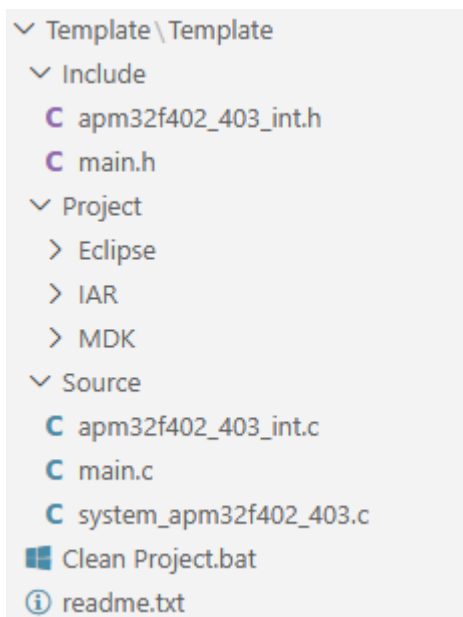
- Window 10/11
- MDK-ARM v5.40 or updated version
- IAR EW for ARM 8.50.5 or updated version
- Eclipse 4.24 or updated version
- arm-gnu-toolchain 10.3.1 or updated version
- Emulate debugger (e.g. Geehy-Link or J-Link)
- Any of the following development boards (depending on the chip used in the project):
 - APM32F402 Tiny Development Board
 - APM32F403 Tiny Development Board

Note: The content of this section takes the APM32F402 Tiny Development Board as an example.

10.1. Template Project

The template project is stored in `~/Examples/Board_APM32F402_Tiny/Template`, and its contents are shown in Figure 2. It includes template projects for MDK, IAR, and Eclipse. The "Config" folder contains the initialization code for the devices used, while the "Include" and "Source" folders store the header files and source files used by the application code in the template project, respectively.

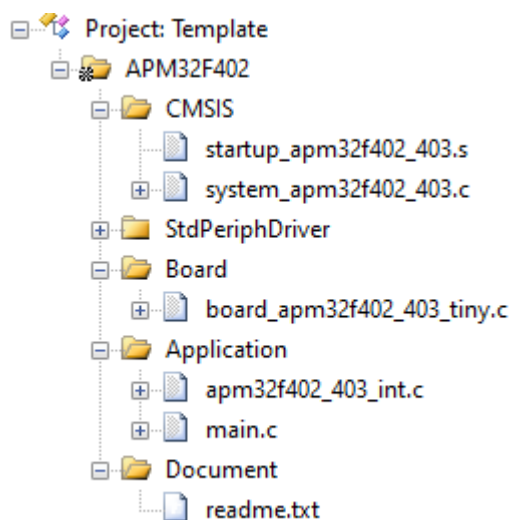
Figure 2 Contents of the Template Folder



Open the folder of the corresponding project and click the project file to open the IDE project. Shown in Figure 3 is the MDK template project. The files in the project are described as follows:

1. `system_apm32f402_403.c` – System initialization configuration file.
2. `startup_apm32f402_403.s` – Startup file.
3. `apm32f402_403_ppp.c` in the "StdPeriphDriver" folder: Low-level driver files for various peripherals.
4. `board_apm32f402_403_tiny.c` – Onboard support file.
5. `apm32f402_403_int.c` – Interrupt file, containing kernel and peripheral interrupt service functions.
6. `main.c` – Main source code file of the template project.
7. `readme.txt` – Project description file.

Figure 3 MDK Template Project



10.2. SDK-related Macro Definitions

When creating a new project, in addition to importing the startup file (startup_apm32f402_403.s) into the project, it is also necessary to enable the relevant macro definitions according to the MCU model. The relationships of the macro definitions used in the SDK are shown in the table below.

Table 4 Macro Definition Relationships

Macro definition	Description
APM32F402xx、APM32F403xx	Enable support for the APM32F402 and APM32F403 models.
BOARD_APM32F402_403_TINY	Enable BSP support for the apm32f402_403 tiny development board

10.3. Device Initialization

When initializing peripherals with the standard peripheral library, the peripheral initialization configuration is usually completed directly in `main.c`. In general, the standard peripheral library defines peripheral initialization structure variables in the application. After completing the system clock and basic hardware initialization, each member of the structure is assigned one by one, and then the corresponding initialization configuration interface is called to complete the peripheral initialization.

For example, when initializing USART, a configuration structure variable of type `USART\_Config\_T` is usually defined first, and then parameters such as baud rate, hardware flow control, operating mode, parity, stop bits, and word length are assigned.

After the parameter configuration is completed, the peripheral initialization interface is called to initialize the USART so that the configuration takes effect.

The USART initialization process should usually include system clock configuration, related GPIO configuration, USART parameter configuration, and peripheral enabling.

When the transmit function is required, the transmit buffer status flag should also be checked before sending data, and the data transmit interface should be called to complete data writing..

10.4. Compilation and Download

For installation and usage of IDEs and simulation/download toolchains, refer to the following documents:

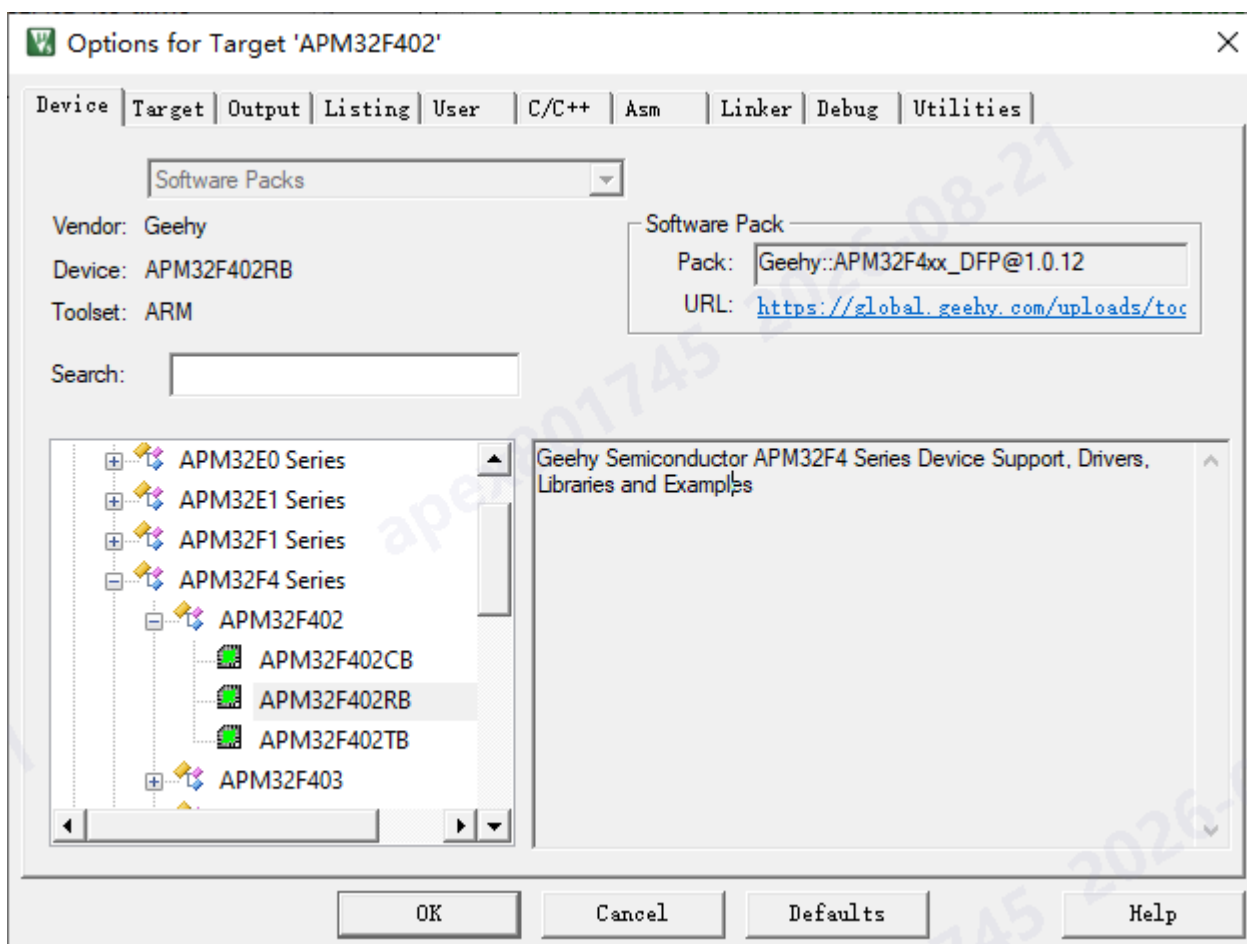
[AN1080_APM32 Series Toolchain User Guide](#)

[Eclipse Development Tutorial under AN1093_APM32 Arm MCU Windows System](#)

10.4.1. Configuration Project

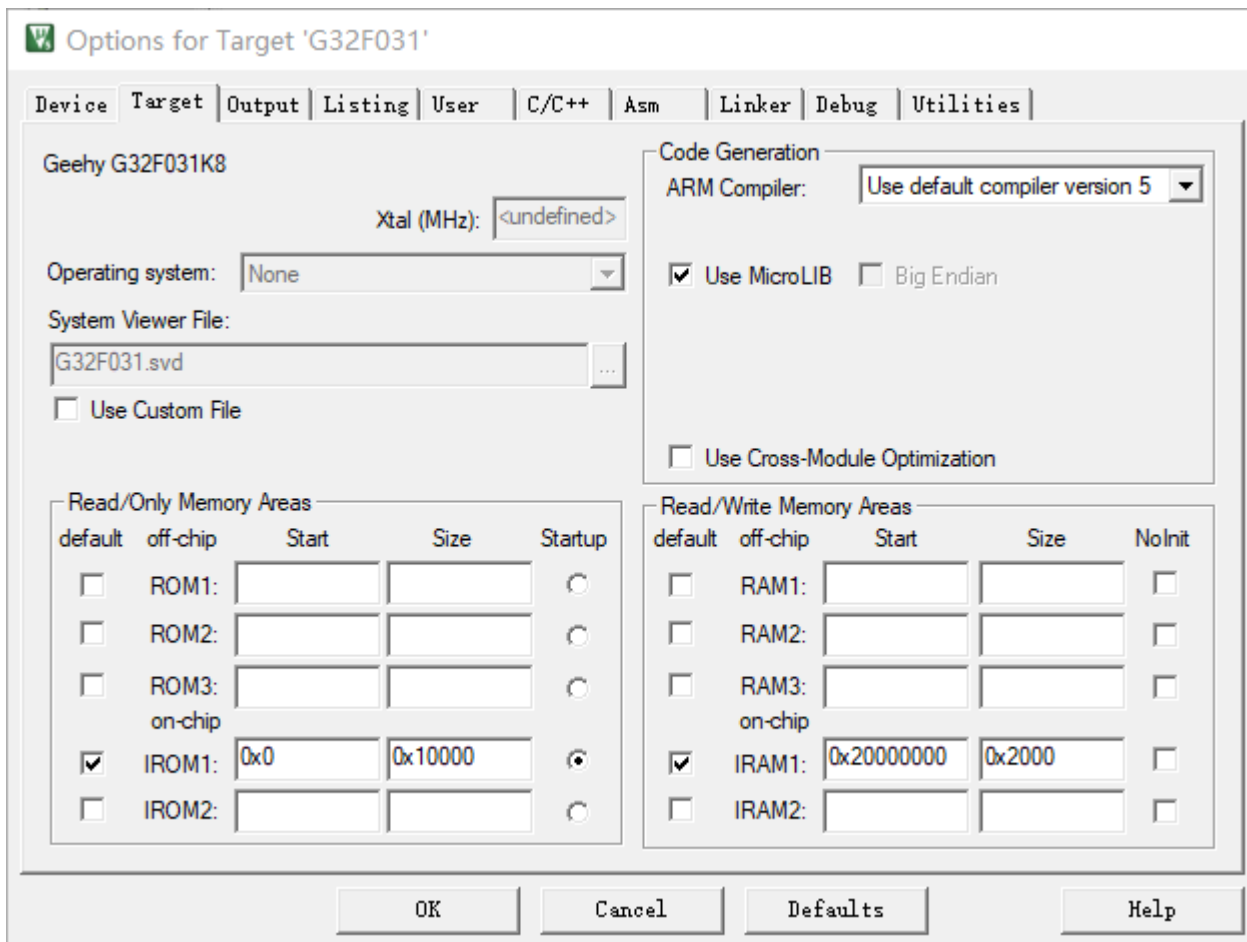
After installing the Keil pack toolchain as described in the above documents, open the Template MDK project and click the “Magic Wand” to confirm the chip selection. As shown in Figure 4, select APM32F402RB for development.

Figure 4 Viewing Chip Selection



By switching to the "Target" tab, you can view the address and size of the ROM and RAM configured for the K8.

Figure 5 ROM and RAM



The dialog box "Options for Target 'G32F031'" has several tabs: Device, Target, Output, Listing, User, C/C++, Asm, Linker, Debug, and Utilities. The "Target" tab is selected.

Device Information:

- Device: Geehy G32F031K8
- Xtal (MHz): <undefined>
- Operating system: None
- System Viewer File: G32F031.svd
- ☐ Use Custom File

Code Generation:

- ARM Compiler: Use default compiler version 5
- ☒ Use MicroLIB ☐ Big Endian
- ☐ Use Cross-Module Optimization

Read/Only Memory Areas:

default	off-chip	Start	Size	Startup
☐	ROM1:			☐
☐	ROM2:			☐
☐	ROM3:			☐
	on-chip			
☒	IROM1:	0x0	0x10000	☒
☐	IROM2:			☐

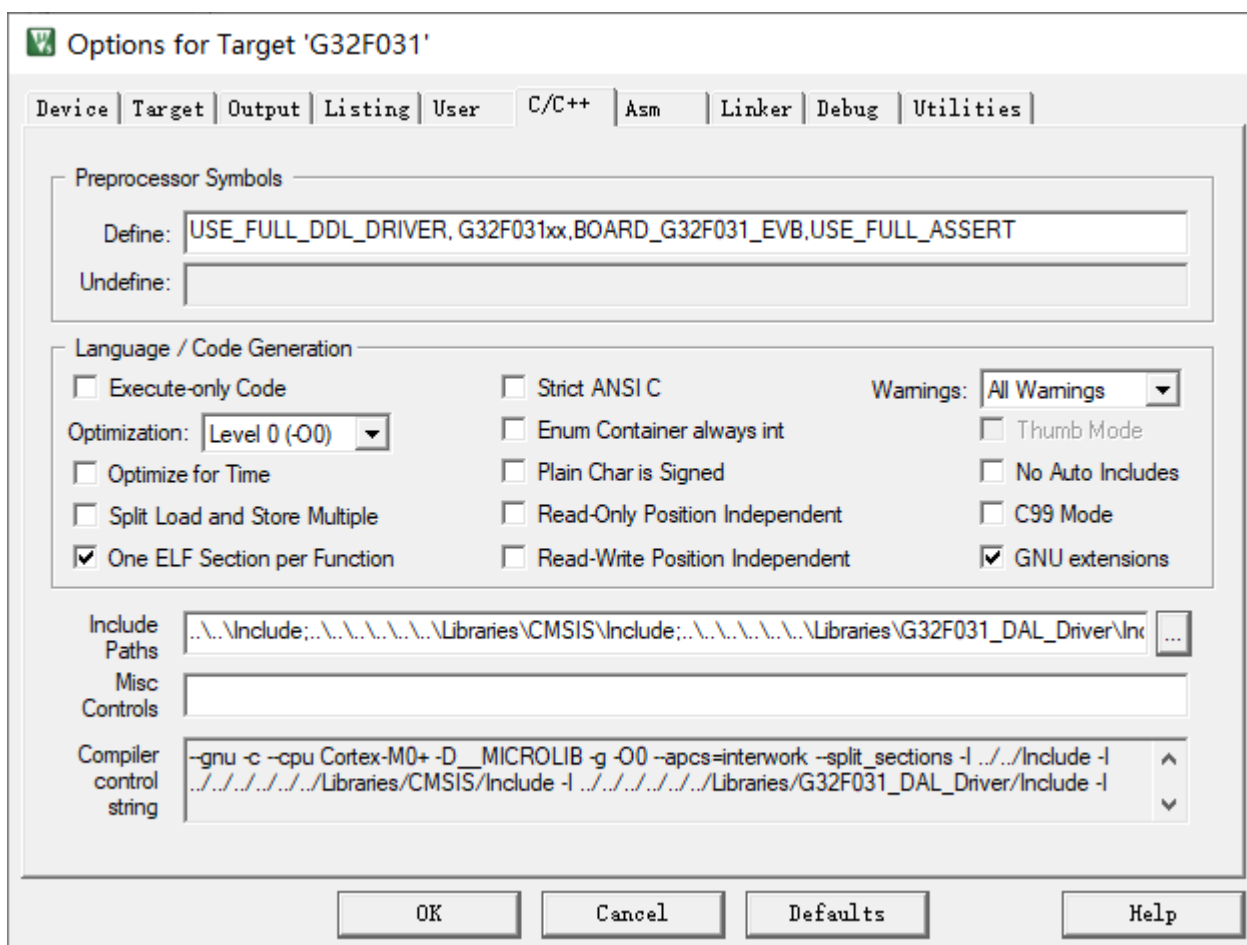
Read/Write Memory Areas:

default	off-chip	Start	Size	NoInit
☐	RAM1:			☐
☐	RAM2:			☐
☐	RAM3:			☐
	on-chip			
☒	IRAM1:	0x20000000	0x2000	☐
☐	IRAM2:			☐

Buttons: OK, Cancel, Defaults, Help

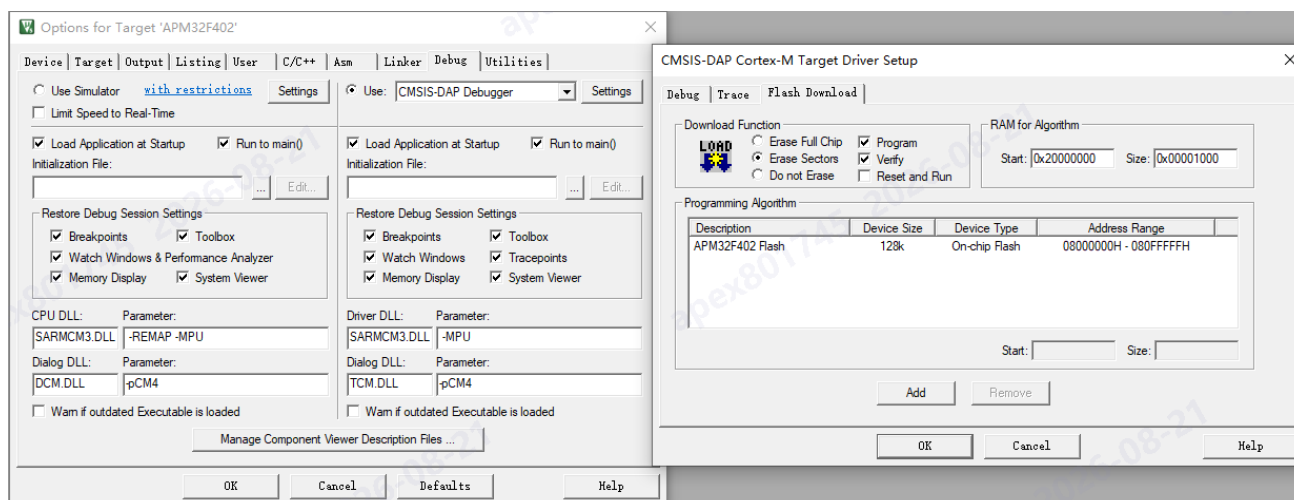
Switch to the C/C++ tab and confirm that the APM32F402xx macro definition is enabled to support the development of APM32F402RB.

Figure 6 C/C++ Configuration



Switch to the "Debug" tab, select the *CMSIS-DAP* Debugger emulator, and click the "Settings" button to enter the "Flash Download" tab. Confirm that the APM32F402 128KB Flash download algorithm is selected. Note that the algorithm usage range is 0x80000000H ~ 80FFFFFF H, as shown in Figure 7.

Figure 7 Debug



10.4.2. Compilation and Download

Click the "Build" button on the MDK interface for compilation. After confirming that the compilation results are correct, click the "Download" button to perform the download. The download result is as shown in Figure 10.

Figure 8 Compilation Project

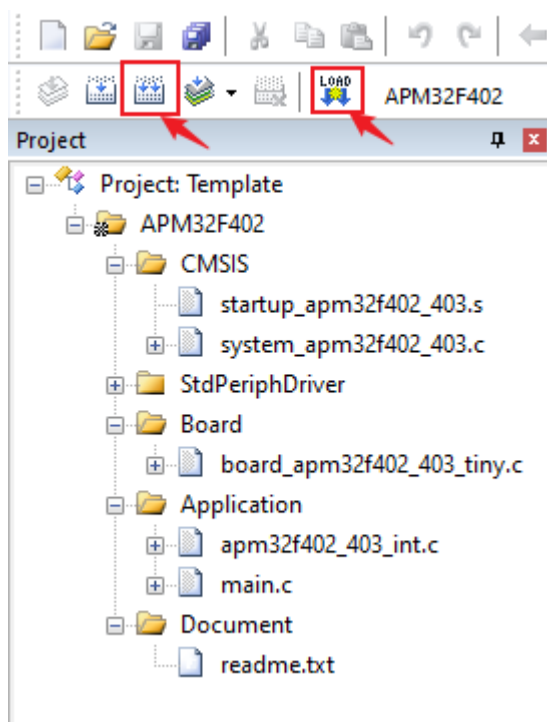


Figure 9 Compilation Results

```
Build Output
compiling apm32f402_403_usb_device.c...
compiling apm32f402_403_usb_host.c...
linking...
Program Size: Code=704 RO-data=360 RW-data=4 ZI-data=1028
".\Objects\APM32F402\Template.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed: 00:00:03
```

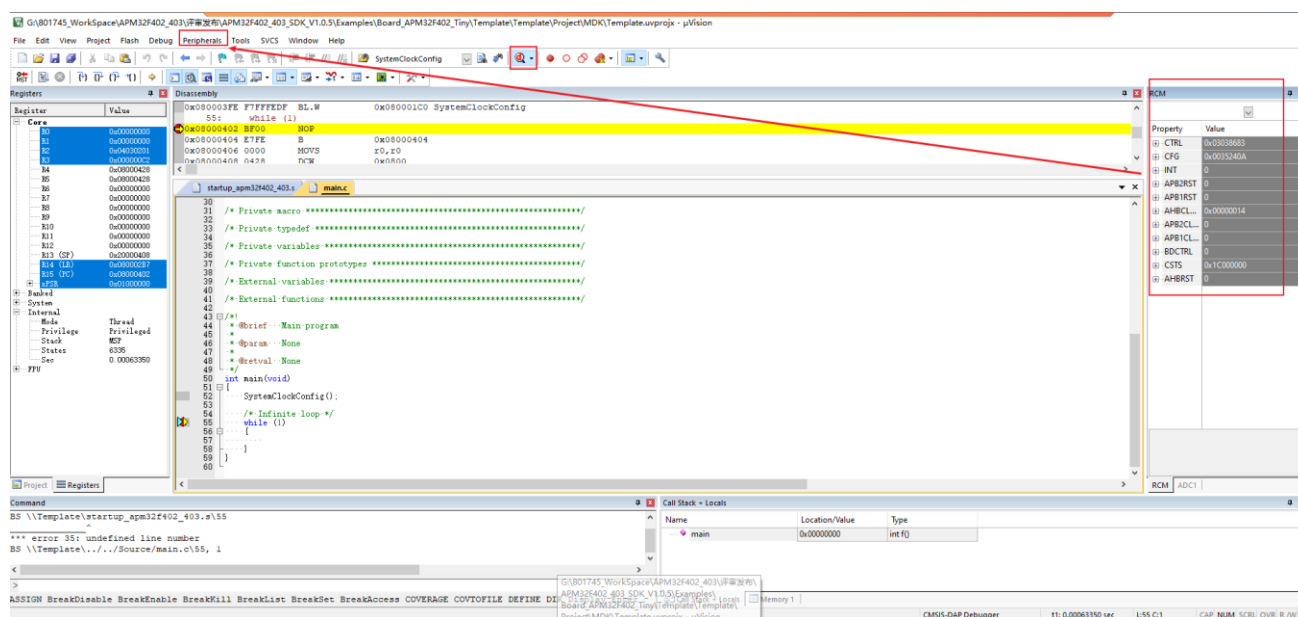
Figure 10 Download Results

```
Build Output
Erase Done.
Programming Done.
Verify OK.
Flash Load finished at 16:50:49
```

10.4.3. Simulation Project

Click the "Start/Stop Debug Session" button on the interface to start simulation. On the simulation interface, you can use the "Peripherals", "Memory", and "Registers" tabs to view information such as the MCU core, peripheral registers, and memory data.

Figure 11 Simulation Project



11. Revision History

Table 5 Revision History

Date	Version	Revision History
September,2026	V1.0	New

12. Annex

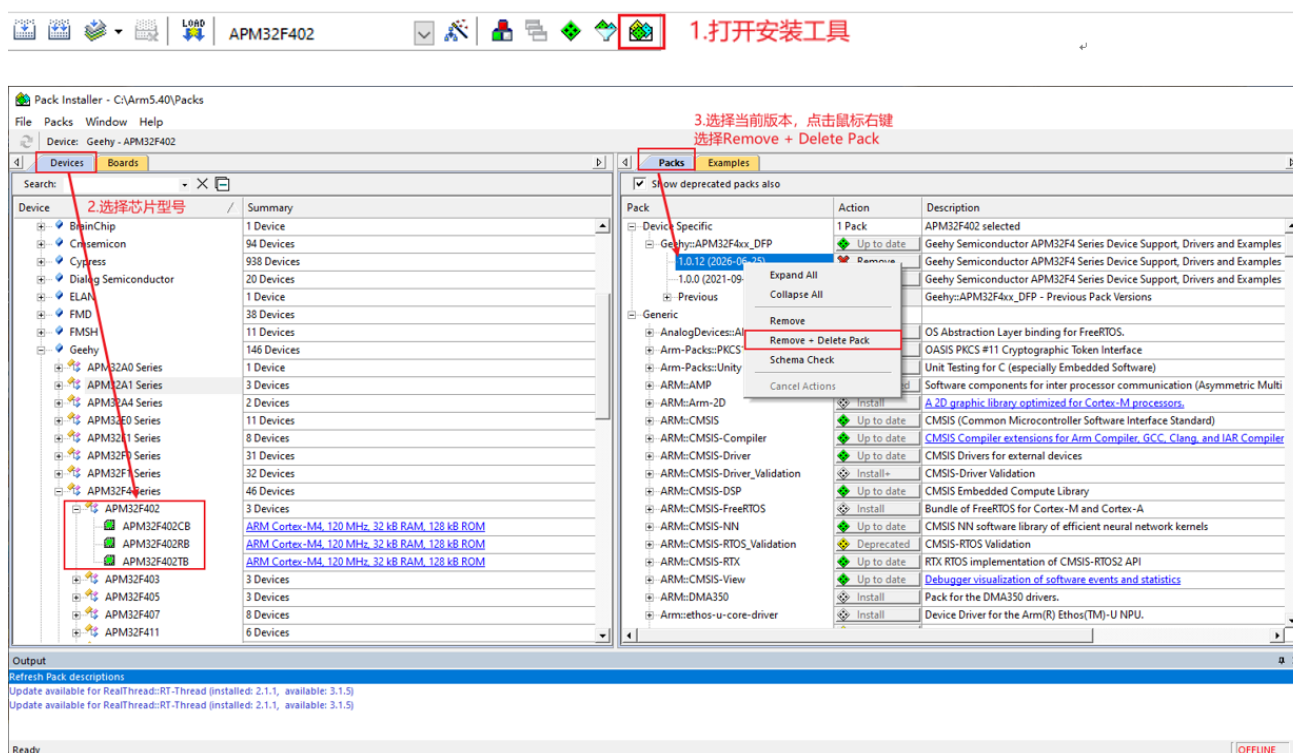
12.1. Uninstalling and Installing Keil pack

The following shows how to uninstall all versions of the pack and install the latest version of the pack.

Steps:

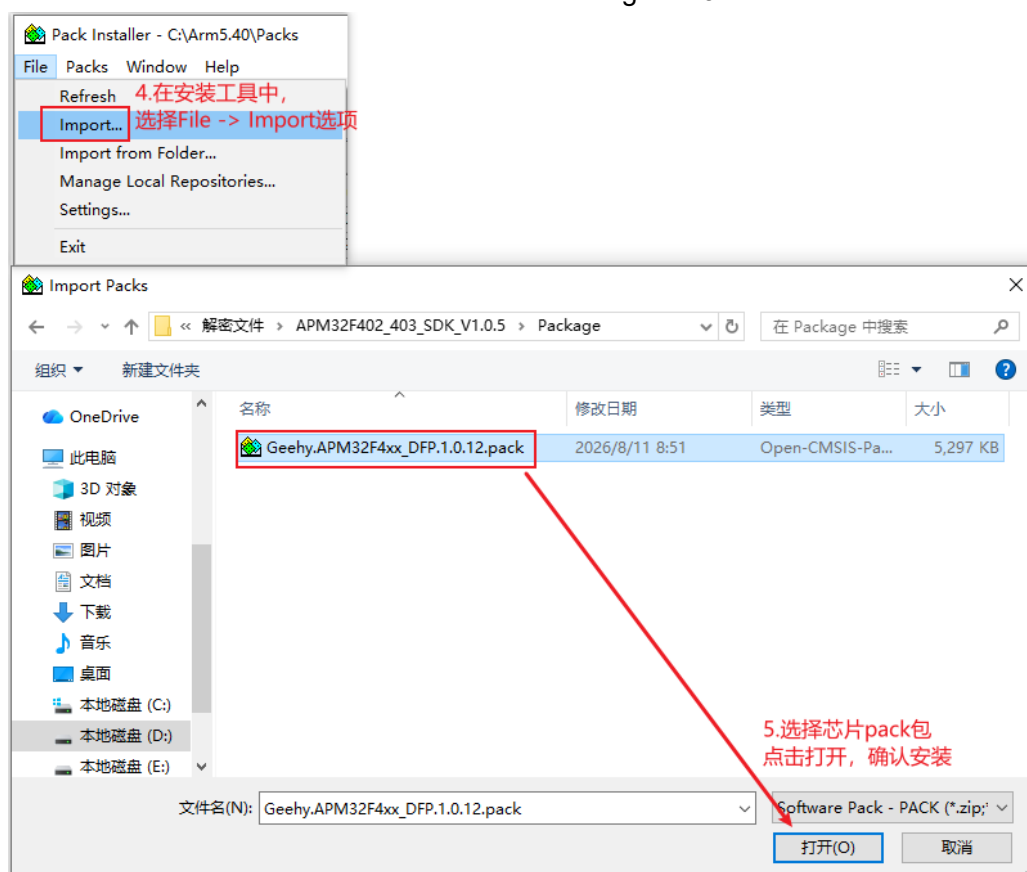
1. Open the pack installer.
2. In the Devices tab, select the corresponding chip model.
3. In the Packs tab, right-click and perform the delete operation.
4. In the pack installer, click File → Import to import and install the pack.
5. Select the pack and click Open to start the installation.
6. Check the installation directory of the pack to see if the pack contents have been updated.

Figure 12



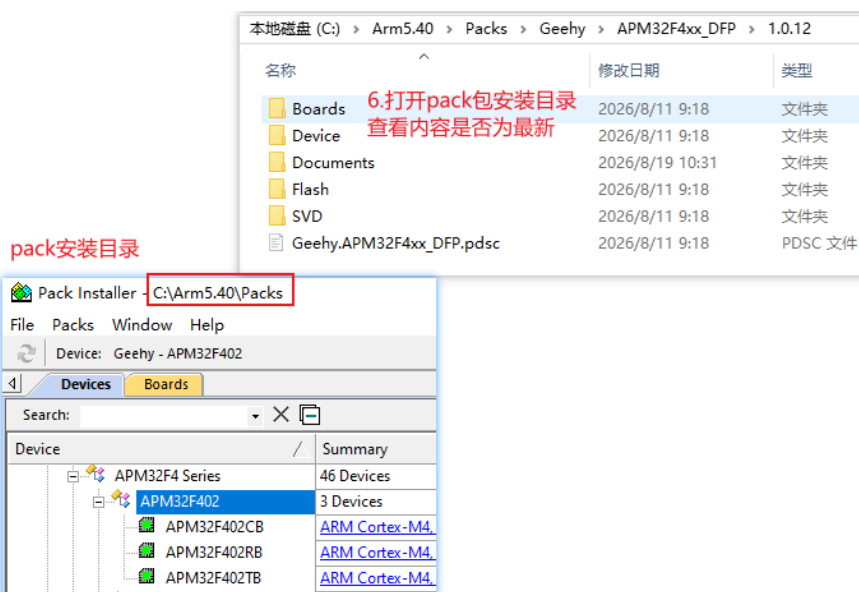
1. Open the pack installer
2. Select the chip model
3. Select the current version, right-click the mouse, and select "Remove + Delete Pack"

Figure 13



4. In the pack installer, select File → Import.
5. Select the pack and click Open for installation.

Figure 14



6. Check the pack to see if it is the newest version.

Statement

This manual is formulated and published by Zhuhai Geehy Semiconductor Co., Ltd. (hereinafter referred to as "Geehy"). The contents in this manual are protected by laws and regulations of trademark, copyright and software copyright. Geehy reserves the right to correct and modify this manual at any time. Please read this manual carefully before using the product. Once you use the product, it means that you (hereinafter referred to as the "users") have known and accepted all the contents of this manual. Users shall use the product in accordance with relevant laws and regulations and the requirements of this manual.

1. Ownership of rights

This manual can only be used in combination with chip products and software products of corresponding models provided by Geehy. Without the prior permission of Geehy, no unit or individual may copy, transcribe, modify, edit or disseminate all or part of the contents of this manual for any reason or in any form.

The "Geehy" or "Geehy" words or graphics with "®" or "TM" in this manual are trademarks of Geehy. Other product or service names displayed on Geehy products are the property of their respective owners.

2. No intellectual property license

Geehy owns all rights, ownership and intellectual property rights involved in this manual.

Geehy shall not be deemed to grant the license or right of any intellectual property to users explicitly or implicitly due to the sale and distribution of Geehy products and this manual.

If any third party's products, services or intellectual property are involved in this manual, Geehy shall not be deemed to authorize users to use the aforesaid third party's products, services or intellectual property, nor shall it be deemed to provide any form of guarantee for third-party products, services, or intellectual property, including but not limited to any non-infringement guarantee for third-party intellectual property, unless otherwise agreed in sales order or sales contract of Geehy.

3. Version update

Users can obtain the latest manual of the corresponding products when ordering Geehy products.

If the contents in this manual are inconsistent with Geehy products, the agreement in Geehy sales order or sales contract shall prevail.

4. Information reliability

The relevant data in this manual are obtained from batch test by Geehy Laboratory or cooperative third-party testing organization. However, clerical errors in correction or errors caused by differences in testing environment are unavoidable. Therefore, users should understand that Geehy does not bear any responsibility for such errors that may occur in this manual. The relevant data in this manual are only used to guide users as performance parameter reference and do not constitute Geehy's guarantee for any product performance.

Users shall select appropriate Geehy products according to their own needs, and effectively verify and test the applicability of Geehy products to confirm that Geehy products meet their own needs, corresponding standards, safety or other reliability requirements. If losses are caused to users due to the user's failure to fully verify and test Geehy products, Geehy will not bear any responsibility.

5. Compliance requirements

Users shall abide by all applicable local laws and regulations when using this manual and

the matching Geehy products. Users shall understand that the products may be restricted by the export, re-export or other laws of the countries of the product suppliers, Geehy, Geehy distributors and users. Users (on behalf of itself, subsidiaries and affiliated enterprises) shall agree and undertake to abide by all applicable laws and regulations on the export and re-export of Geehy products and/or technologies and direct products.

6. Disclaimer

This manual is provided by Geehy on an "as is" basis. To the extent permitted by applicable laws, Geehy does not provide any form of express or implied warranty, including without limitation the warranty of product merchantability and applicability of specific purposes.

Geehy products are not designed, authorized, or guaranteed to be suitable for use as critical components in military, life support, pollution control, or hazardous substance management systems, nor are they designed, authorized, or guaranteed to be suitable for applications that may cause injury, death, property, or environmental damage in case of product failure or malfunction.

If the product is not labeled as "Automotive grade", it means it is not suitable for automotive applications. If the user's application of the product is beyond the specifications, application fields, and standards provided by Geehy, Geehy will assume no responsibility.

Users shall ensure that their application of the product complies with relevant standards, and the requirements of functional safety, information security, and environmental standards. Users are fully responsible for their selection and use of Geehy products. Geehy will bear no responsibility for any disputes arising from the subsequent design and use of Geehy products by users.

7. Limitation of liability

In any case, unless required by applicable laws or agreed in writing, Geehy and/or any third party providing this manual and the products on an "as is" basis shall not be liable for damages, including any general or special direct, indirect or collateral damages arising from the

use or no use of this manual and the products (including without limitation data loss or inaccuracy, or losses suffered by users or third parties), which cover damage to personal safety, property, or environment, for which Geehy will not be responsible.

8. Scope of application

The information in this manual replaces the information provided in all previous versions of the manual.

©2025 Zhuhai Geehy Semiconductor Co., Ltd. All Rights Reserved