
AT32 Work Bench User Manual

Introduction

This user manual gives an overview of AT32 Work Bench. The AT32 Work Bench can generate initialization C code and corresponding IDE project through MCU graphical configuration, so as to reduce the development workload, time and cost.

AT32 Work Bench has the following features:

1. Support peripheral initialization configuration
2. Support PIN MUX configuration and customized PIN label
3. Support automatic system clock configuration
4. Support online code view
5. Support “add user code” function (existing code is not overwritten by the new project)
6. Support automatic project generation in Keil, IAR and AT32 IDE
7. Record recent designs
8. Generate configuration report (.pdf)
9. Support simplified Chinese & English menu
10. Support Windows and Linux
11. Support middleware such as FREERTOS, USB_DEVICE and USB_HOST
12. Support online software upgrade

Contents

1	Introduction	5
1.1	Environmental requirements.....	5
2	Installation	6
2.1	Set up AT32 Work Bench on Windows	6
2.2	Set up AT32 Work Bench on Linux	6
3	Getting started	7
4	Project configuration.....	8
4.1	Menu bar and toolbar	8
4.1.1	Menu bar	8
4.1.2	Toolbar.....	9
4.2	Pin out configuration.....	9
4.2.1	Peripherals	9
4.2.2	Mode and configuration.....	10
4.2.3	Pin layout.....	13
4.2.4	GPIO mode and configuration	16
4.2.5	DMA mode and configuration.....	17
4.2.6	NVIC mode and configuration.....	18
4.3	Clock configuration	20
4.4	Code view.....	21
4.5	Generate code.....	22
4.5.1	Project Manager	22
4.5.2	Advance Settings	25
4.5.3	Keep user code when re-generating.....	26
4.6	Package manager	27
5	Revision history.....	29

List of tables

Table 1. Peripheral status	9
Table 2. Status of mode and parameters.....	11
Table 3. Status of pins	14
Table 4. Document revision history.....	29

List of figures

Figure 1. dpkg command in Linux	6
Figure 2. Graphical installation	6
Figure 3. Getting started page.....	7
Figure 4. Project configuration.....	8
Figure 5. Menu bar and toolbar	8
Figure 6. Peripheral mode and configuration	10
Figure 7. USART1 mode and configuration.....	10
Figure 8. Mode error prompt.....	11
Figure 9. Parameters settings.....	11
Figure 10. GPIO settings	12
Figure 11. Interrupts settings	12
Figure 12. DMA settings	13
Figure 13. NVIC settings.....	13
Figure 14. PIN layout	14
Figure 15. Pin configuration menu.....	15
Figure 16. Custom label.....	15
Figure 17. GPIO mode and configuration.....	16
Figure 18. DMA mode and configuration	17
Figure 19. NVIC mode and configuration	18
Figure 20. Clock configuration	20
Figure 21. Code view	21
Figure 22. Project Manager	22
Figure 23. Project file directory	23
Figure 24. MDK_V5 project files	24
Figure 25. Advance Settings.....	25
Figure 26. Package Manager -- Firmware.....	27
Figure 27. Package Manager -- Components	28

1 Introduction

1.1 Environmental requirements

■ Software

Windows

Windows 7 and above is required.

Linux

Ubuntu or Fedora that supports AMD x86_64.

■ Hardware

At least 2GB RAM.

At least 4GB hard drive space.

2 Installation

2.1 Set up AT32 Work Bench on Windows

Run the executable AT32_Work_Bench.exe directly, without the need for installation.

2.2 Set up AT32 Work Bench on Linux

This software supports Ubuntu 16.4 and above.

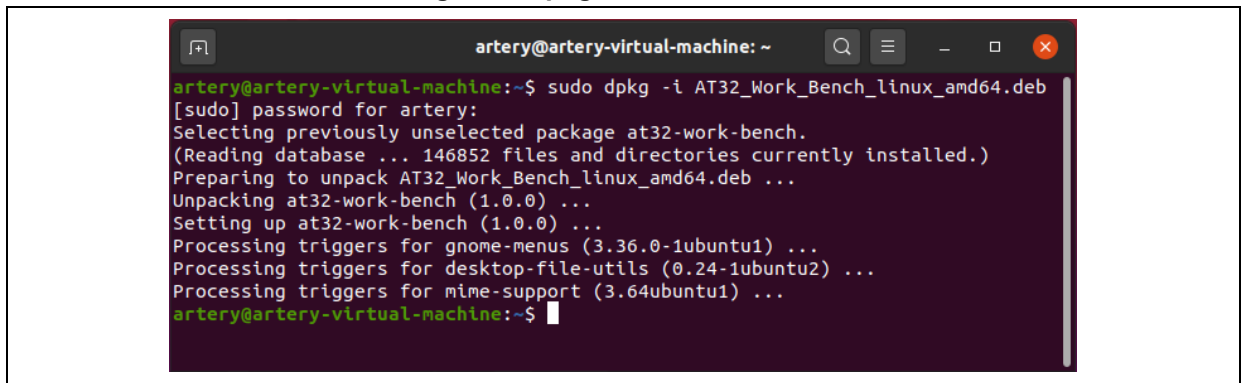
Installation methods on Linux: dpkg command and graphical installation.

■ dpkg command

As shown in Figure 1, enter the following command in the terminal for setup:

```
sudo dpkg -i AT32_Work_Bench_linux_amd64_V1.0.0.deb
```

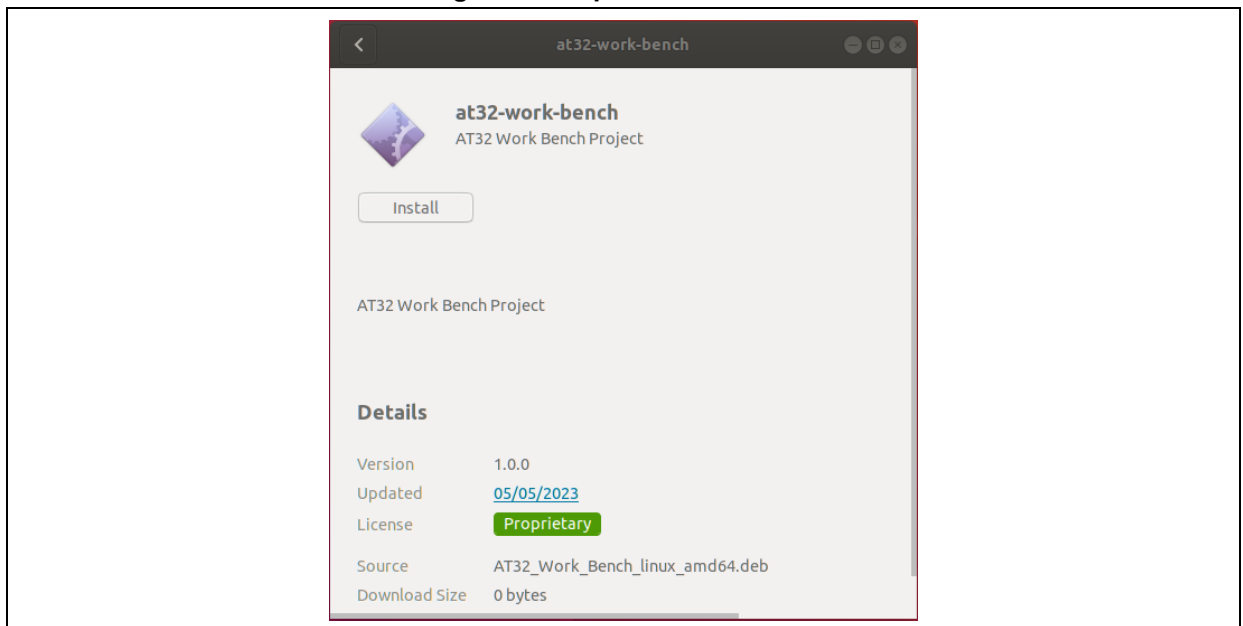
Figure 1. dpkg command in Linux



■ Graphical installation

Copy the AT32_Work_Bench_linux_amd64_V1.0.0.deb to Linux and double click. Then, click on the "Install", as shown in Figure 2.

Figure 2. Graphical installation



After the installation is complete, click the "All Programs" button at the bottom of the left taskbar, find and click on the "AT32 Work Bench" in the program list to start AT32 Work Bench.

3 Getting started

The getting started page is the first window that opens when AT32 Work Bench is started. It includes three options, i.e., “Start a New Design”, “Open an Existing Design” and “Recent Designs”.

■ Start a new design

Select the MCU serials and device, Click on “New” or double click the device to create a new project.

■ Open an existing design

Users can find a saved project through File Explorer and then open it (*.ATWP file).

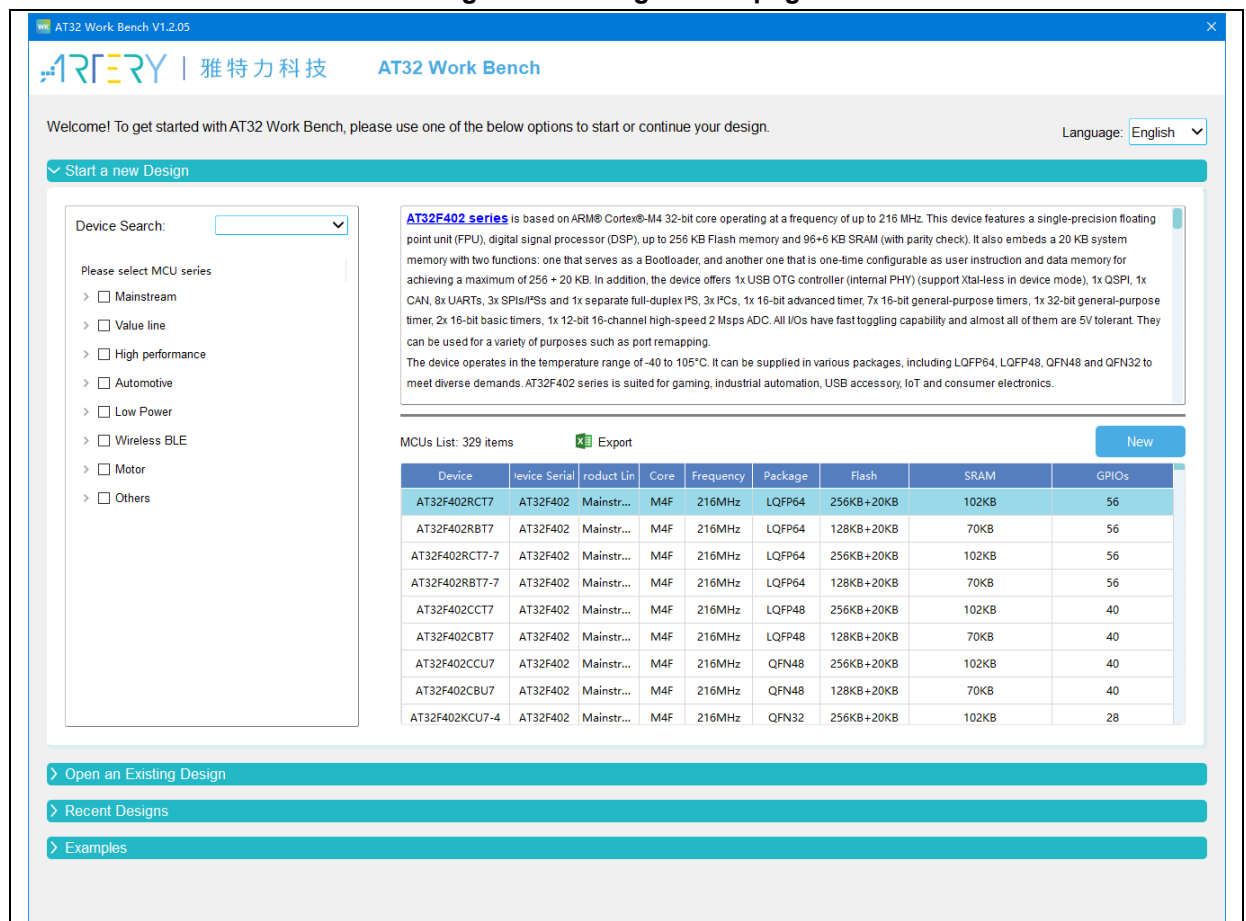
■ Recent designs

It displays a list of recently created projects. Users can select one to open.

■ Examples

Displays a list of examples. You can select an example to install and open it.

Figure 3. Getting started page

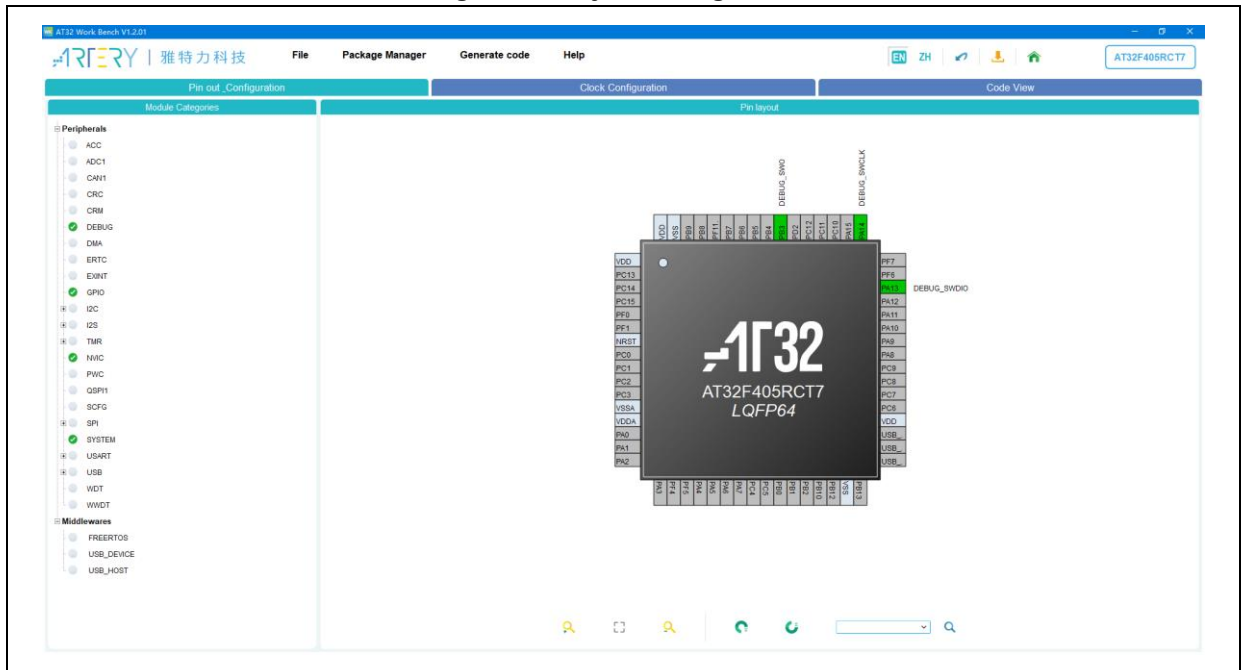


4 Project configuration

After a new project is created or a saved file is loaded, the project configuration page opens. It contains a menu bar, a toolbar, and the following set of views:

- Pin out configuration
- Clock configuration
- Code view

Figure 4. Project configuration



Pin out configuration: Users can configure peripheral pins and relevant parameters.

Clock configuration: Users can configure MCU clock as needed.

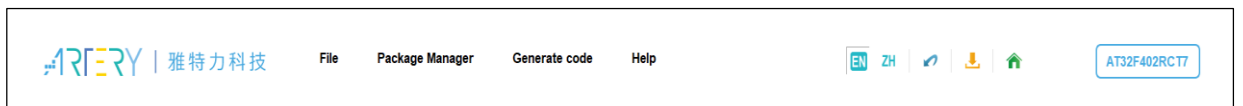
Code view: Users can view the code automatically generated according to the current configuration.

Click on the “Generate code” in the menu bar or the  icon in the toolbar to generate the corresponding user code and project.

4.1 Menu bar and toolbar

Figure 5 shows the menu bar and toolbar.

Figure 5. Menu bar and toolbar



4.1.1 Menu bar

- File
 - New Design Create a new project.
 - Open Design Open a saved project.
 - Save Design Save the current project.
 - Design Save As Save the current project as...

Open Example	Select an example and open it.
Generate Report	Generate a project report (.pdf).
Generate Pin Configuration Excel	Generate a pin configuration report (.xlsx).
Recent Design	Display recent projects.
■ Package Manager	
Package Manager	Install and manage BSP.
■ Generate code	
After the peripheral parameters and clock are configured as needed, click on “Generate code” to select the project location and generate the corresponding project file.	
■ Help	
Open Manual	Open the user manual.
New version download	Download the new version online.
Doc & Resources	Open browser to visit the doc and resources website of MCU.
User Guide	Open browser to visit the user guide website of Work Bench.
Version	View the current version.

4.1.2 Toolbar

EN: Select English as the display language.

ZH: Select simplified Chinese as the display language.

: Reset. Reset all peripheral parameters, pins and clock.

: Generate code. Open “Project Manager” window and generate the corresponding code. Refer to [Section 4.5](#).

: Visit the official website of Artery.

: Display the current MCU model.

4.2 Pin out configuration

Pin out configuration: Configure pins and parameters of MCU peripherals.

It contains the “Peripherals”, “Mode and Configuration” and “Pin layout” windows. Users can adjust the window size as follows: hover the mouse over the window border and then a two-way arrow appears; then, hold down the left mouse button and move to expand or shrink the window size.

4.2.1 Peripherals

The “Peripherals” window displays a list of available peripherals and middleware for the selected MCU model. The icon in front of peripheral name indicates the peripheral configuration status.

Table 1. Peripheral status

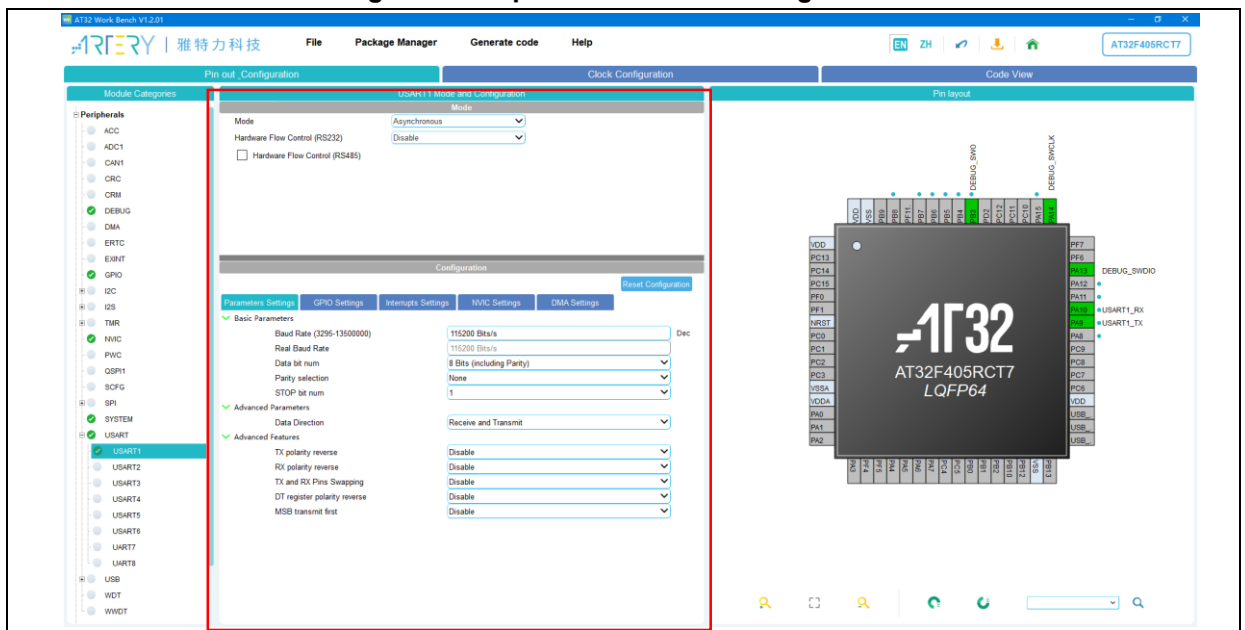
Icon	Status	Description
------	--------	-------------

	Not configured	Mode and parameters are not configured.
	Configured successfully	Mode and parameters are configured successfully.
	Configuration parameter error	The mode or relevant parameters are incorrect. Please confirm and re-configure as needed.

4.2.2 Mode and configuration

Select a peripheral from the list of peripherals, and then the corresponding “Mode and Configuration” window opens. Users can select peripheral mode and the corresponding MCU pins in the “Mode”. For AT32 MCUs, same pins can be used for different peripherals and multiple features; therefore, once the mode is selected, the optimum pin layout will be configured automatically.

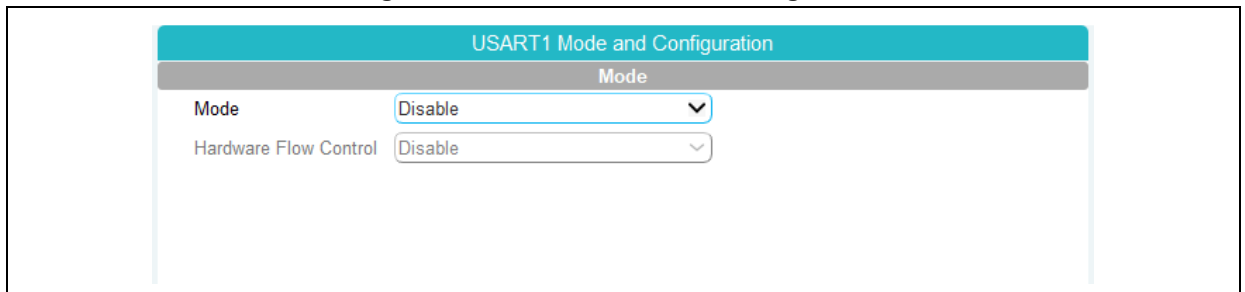
Figure 6. Peripheral mode and configuration



Take the USART1 as an example:

■ USART1 mode

Figure 7. USART1 mode and configuration



As shown in Figure 7, when the USART1 is set to asynchronous mode, PA9 and PA10 are mapped to “USART1_TX” and “USART1_RX” automatically; when “CTS/RTS” is selected for hardware flow control, PA11 and PA12 are mapped to “USART1_CTS” and “USART1_RTS” automatically. When the condition is not satisfied or in case of signal conflict, the corresponding content is displayed in a different color or background color, as shown in Table 2. Move the mouse over the corresponding content, and a message appears to prompt users to modify or re-configure, as

shown in Figure 8.

Figure 8. Mode error prompt

Examples of status of mode and parameters:

Table 2. Status of mode and parameters

Icon	Status	Description
Hardware Flow Control CTS/RTS	Configurable	
☐ Clock Output	Condition not satisfied	Configure other parameters as prompted.
Please select a Inverting input selection (Input [-])	Configure the corresponding mode	Select "Inverting input selection (Input[-])".
☐ IN5	Signal conflict	There is a conflict with other function signals.
Parameters Settings GPIO Settings I2C Settings DMA Settings Basic Parameters Baud Rate (3-14648) 115200 Bauds	Parameter error	The parameter is beyond the required value range.

Parameters settings

Figure 9. Parameters settings

This window displays configurable USART1 parameters, including the “baud rate” and “data bit number” in the “Basic Parameters” and “data direction” and “Tx and Rx pins swapping” in the “Advanced Parameters”.

If an invalid setting is detected, the corresponding parameter will be optimized automatically according to the value range. For example, when the user-defined value is lower than the minimum

value (or greater than the maximum value), it is automatically reset to the minimum (or maximum) value.

■ GPIO settings

Figure 10. GPIO settings

Parameters Settings

GPIO Settings

Interrupts Settings

NVIC Settings

DMA Settings

Pin Name	Signal on Pin	Output level	GPIO type	Pull type	GPIO mode	Driver capability	Label	Modified
PA9	USART1_TX	n/a	Push Pull	Pull-none	Mux function mode	Moderate		N
PA10	USART1_RX	n/a	Push Pull	Pull-none	Mux function mode	Moderate		N

This window displays configurable GPIOs, for example, GPIOs for “USART1_TX” and “USART1_RX” in Figure 10.

For details about GPIO settings, refer to [Section 4.2.4](#).

■ Interrupts settings

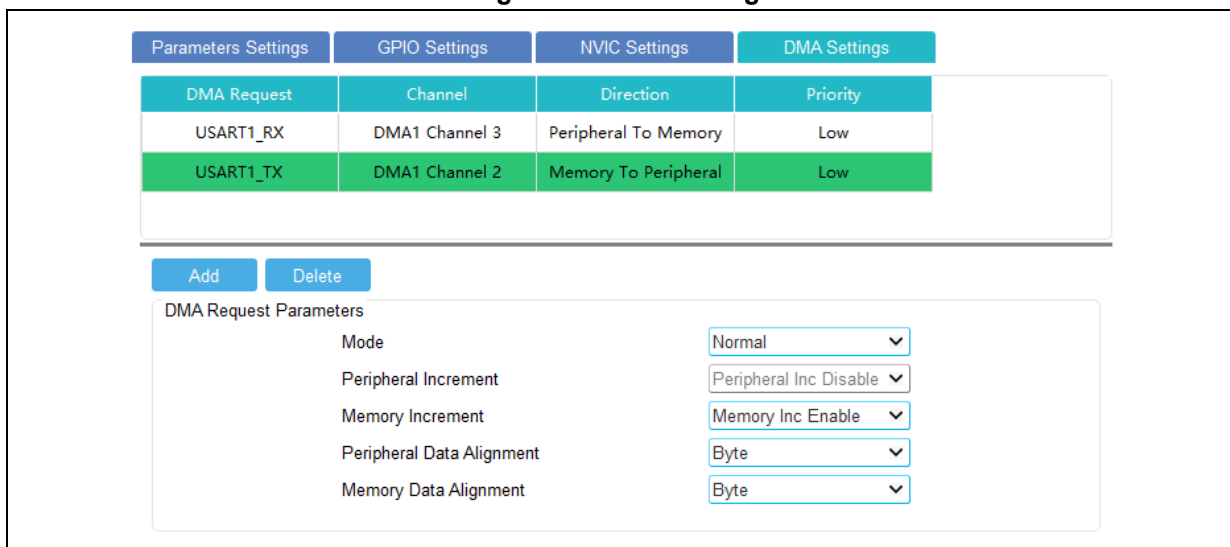
Figure 11. Interrupts settings

Parameters Settings		GPIO Settings		Interrupts Settings		NVIC Settings		DMA Settings	
Interrupt event				Enabled					
Receive data buffer full(RDBF)				☐					
Transmit data register empty(TDBE)				☐					
Transmit data complete(TDC)				☐					
Idle flag(IDLE)				☐					
Noise error/overflow error/framing error(ERR)				☐					
Parity error(PERR)				☐					
Break frame flag(BF)				☐					
Receiver time out detection flag(RTODF)				☐					
Character match detection flag(CMDF)				☐					
CTS flag(CTSCF)				☐					

This window displays configurable interrupts, for example, interrupts for “USART1” in Figure 11.

■ DMA settings

Figure 12. DMA settings



DMA Request	Channel	Direction	Priority
USART1_RX	DMA1 Channel 3	Peripheral To Memory	Low
USART1_TX	DMA1 Channel 2	Memory To Peripheral	Low

Add Delete

DMA Request Parameters

Mode: Normal

Peripheral Increment: Peripheral Inc Disable

Memory Increment: Memory Inc Enable

Peripheral Data Alignment: Byte

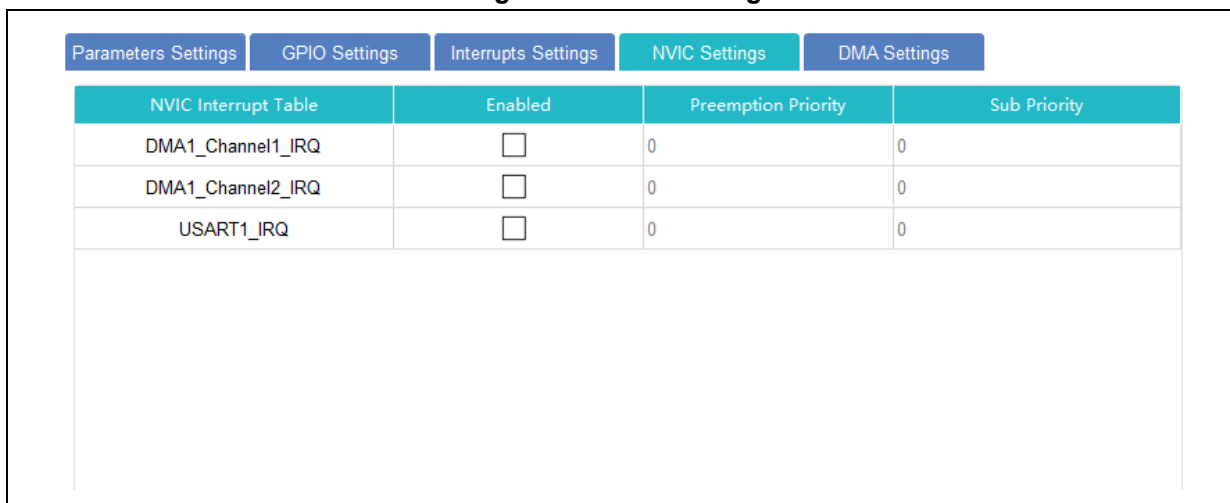
Memory Data Alignment: Byte

This window allows users to add configurable DMA requests for the current peripheral, for example, the corresponding DMA channels and DMA requests for “USART1_TX” and “USART1_RX” in Figure 12.

For details about DMA settings, refer to [Section 4.2.5](#).

■ NVIC settings

Figure 13. NVIC settings



NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
DMA1_Channel1_IRQ	☐	0	0
DMA1_Channel2_IRQ	☐	0	0
USART1_IRQ	☐	0	0

This window displays configurable interrupts for the current peripheral. When the corresponding DMA channel is enabled, the corresponding DMA channel interrupt can be configured, for example, the “USART1_IRQ” and “DAMA1_Channel3_2_IRQ” in Figure 13.

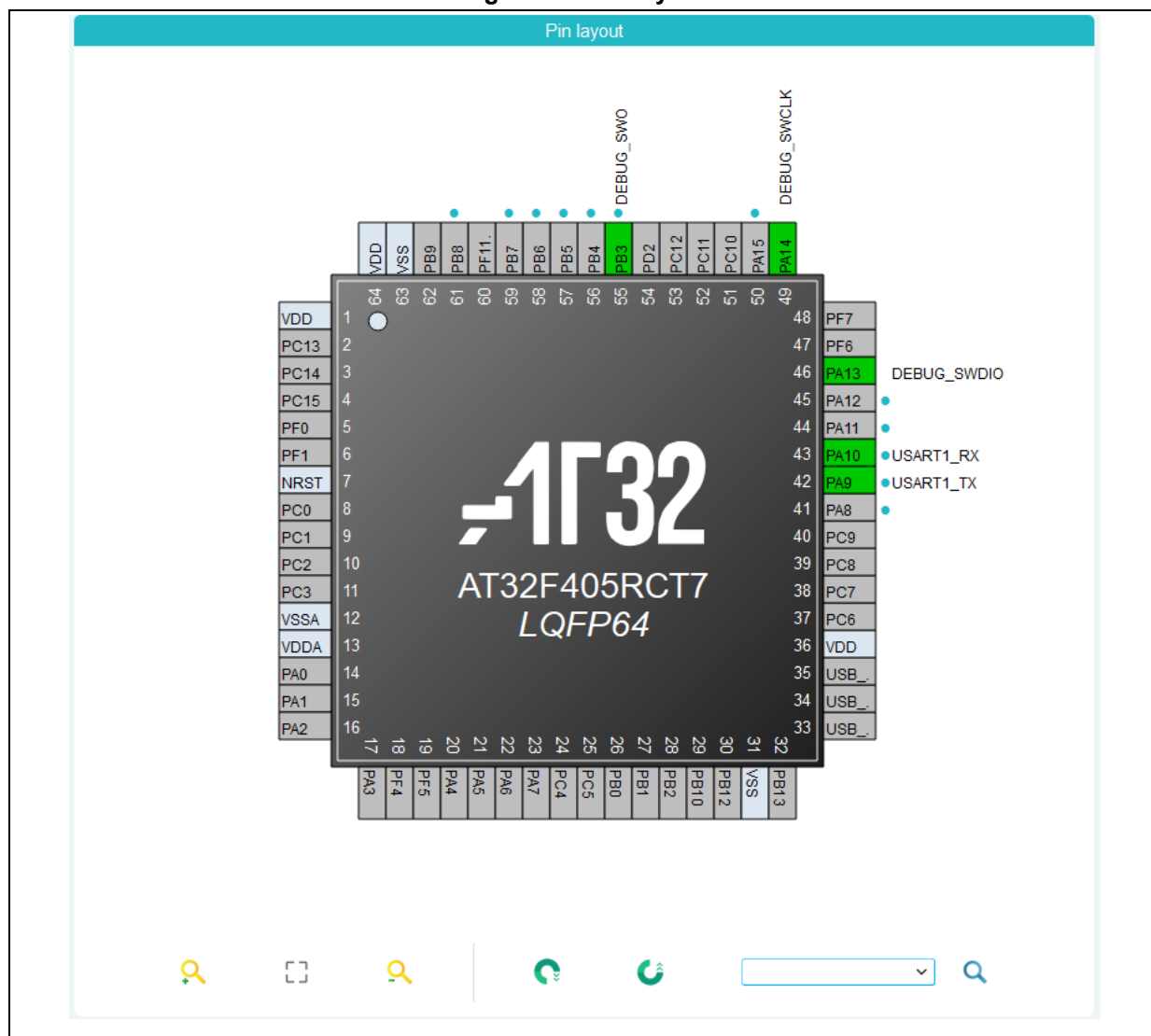
The “preemption priority” and “sub priority” are configured in the “NVIC Mode and configuration” window. Refer to [Section 4.2.6](#).

4.2.3 Pin layout

The pin layout of the selected package (such as LQFP48, QFN32 and TSSOP20) is displayed in graphic. Each pin is represented by its name (such as PA9), configuration status, and current signal

distribution.

Figure 14. PIN layout



Pins are displayed in different colors and have different functions according to their status, as shown in Table 3.

Table 3. Status of pins

Icon	Status	Description
	Pin not in use	Users can configure signals for this pin.
	Pin in use	A valid signal is configured, and it has a corresponding peripheral mode.
	Pin no mode	A signal is configured, but it has no corresponding peripheral mode.
	Fixed mode	Fixed configuration, not support other functions.
	Pin searched	Click on the "Search" icon, and the searched pins are displayed in blinking blue.
	Peripheral pin label	Click on a peripheral in the peripheral list on the left to label all pin positions for that peripheral.

Users can press the Ctrl + scroll mouse wheel to adjust the pin layout window size, and move the mouse to relocate its position.

Click on the chip internal area to show or hide the pin number.

In addition, users can zoom in/out the pin layout by using the below toolbar to view the global configuration or local details. Click on the “Reset” icon to restore the pin layout to its initial size.

Select the corresponding pin, signal or label in the search box and press Enter; then, the searched pins are displayed in blinking blue.

 : Zoom into the pin layout.

 : Restore to the initial size and position.

 : Zoom out the pin layout.

 : Search. Find the positions of pins, signals and labels in the pin layout.

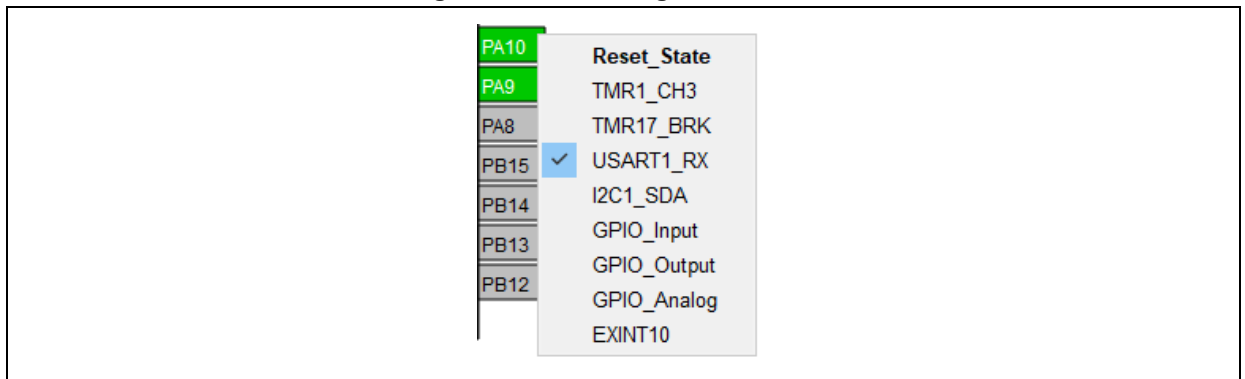
 : Clockwise rotate the pin layout.

 : Anti-clockwise rotate the pin layout.

User can press the Ctrl + click on the signal to search for the pin where the signal is located.

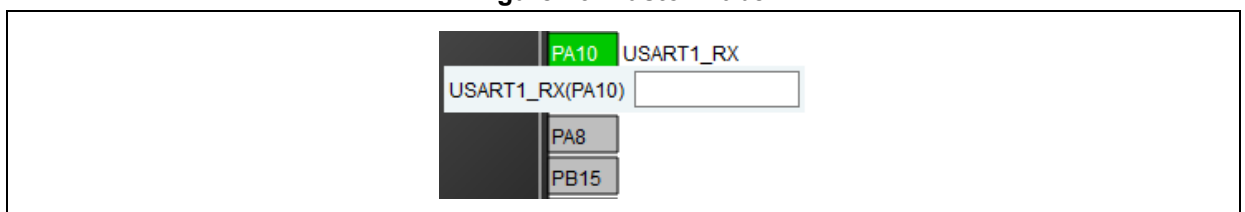
In the pin layout, left click on the pin (except for pins in fixed mode), and then a right-click menu pops up, showing configurable signals for the pin. Select “Reset_State” to reset this pin to the unused status, as shown in Figure 15.

Figure 15. Pin configuration menu



Right click on the configured pin, and then an “Enter label” button pops up. Users can click on the button to specify a custom label for this signal. The new label replaces the signal name set in the pin layout.

Figure 16. Custom label



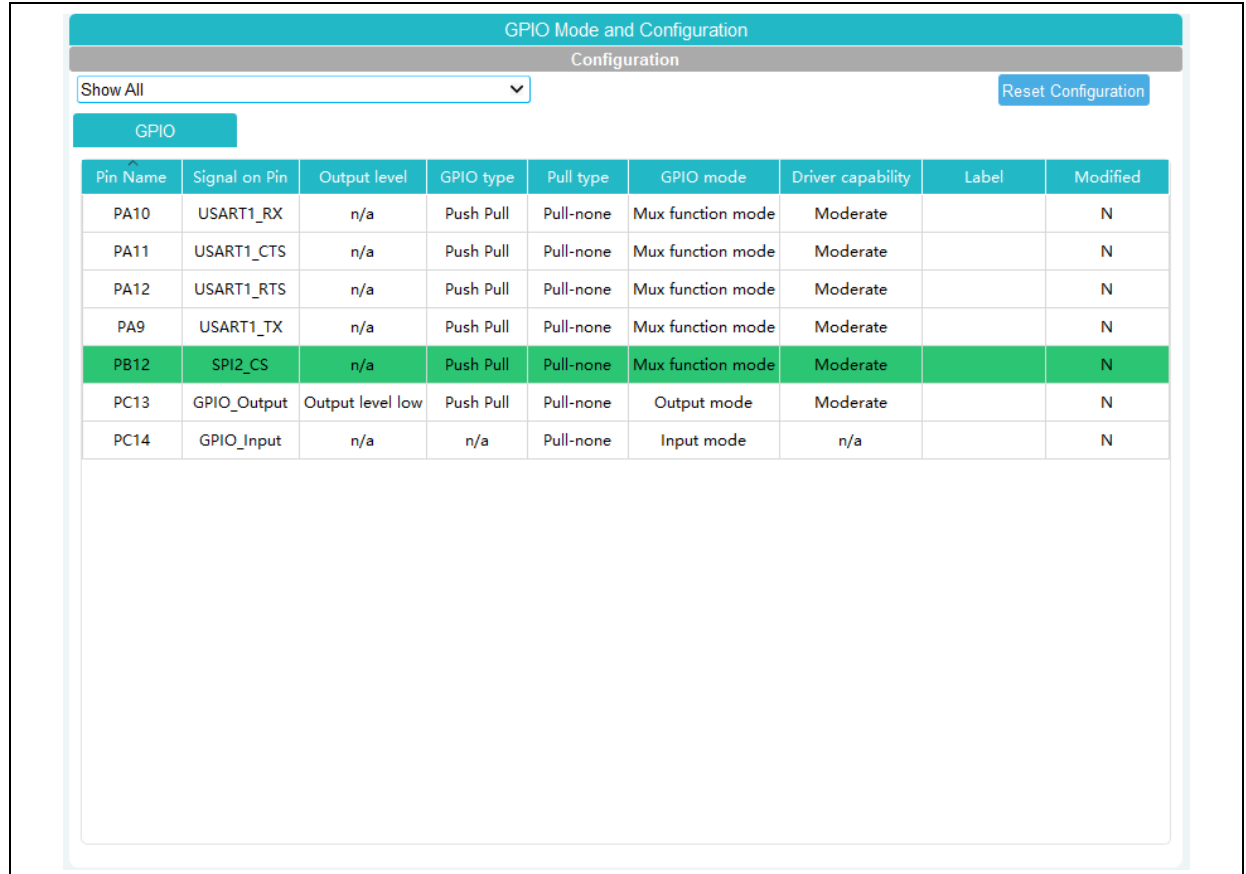
4.2.4 GPIO mode and configuration

Click on “GPIO” under “Peripherals” to open the GPIO Mode and Configuration window. Users can also configure GPIO for specific peripherals in a dedicated window for GPIO configuration.

Click on the drop-down box to select “Show All” or “Groups by Peripherals”.

Click on “Reset Configuration” button to reset all GPIO parameters.

Figure 17. GPIO mode and configuration



Pin Name	Signal on Pin	Output level	GPIO type	Pull type	GPIO mode	Driver capability	Label	Modified
PA10	USART1_RX	n/a	Push Pull	Pull-none	Mux function mode	Moderate		N
PA11	USART1_CTS	n/a	Push Pull	Pull-none	Mux function mode	Moderate		N
PA12	USART1_RTS	n/a	Push Pull	Pull-none	Mux function mode	Moderate		N
PA9	USART1_TX	n/a	Push Pull	Pull-none	Mux function mode	Moderate		N
PB12	SPI2_CS	n/a	Push Pull	Pull-none	Mux function mode	Moderate		N
PC13	GPIO_Output	Output level low	Push Pull	Pull-none	Output mode	Moderate		N
PC14	GPIO_Input	n/a	n/a	Pull-none	Input mode	n/a		N

Double click to modify GPIO parameters for each signal. The configurable GPIO parameters include:

- Pin Name

The name of the pin where the signal is located. If several pins have the same signal and the pin is not used, double click to switch signal to this pin.

(This function is not supported on AT32F403A/F407/A403A/F413/F415 devices)

- Output level

When the current signal is “GPIO_Output”, it can be configured as “Output level low” or “Output level high”.

- GPIO type

Configure the GPIO type (push-pull/open-drain).

- Pull type

It is set to a default value, which can be configured accordingly.

- GPIO mode (analog, input, output and MUX function)

Select a mode in the pin layout, and the corresponding pin is configured according to the MUX function and GPIO mode.

- Driver capability

Configure the I/O port drive capability (moderate sourcing/sinking strength and stronger sourcing/sinking strength).

■ Label

Change the default name to a user-defined name, and the pin layout changes accordingly.

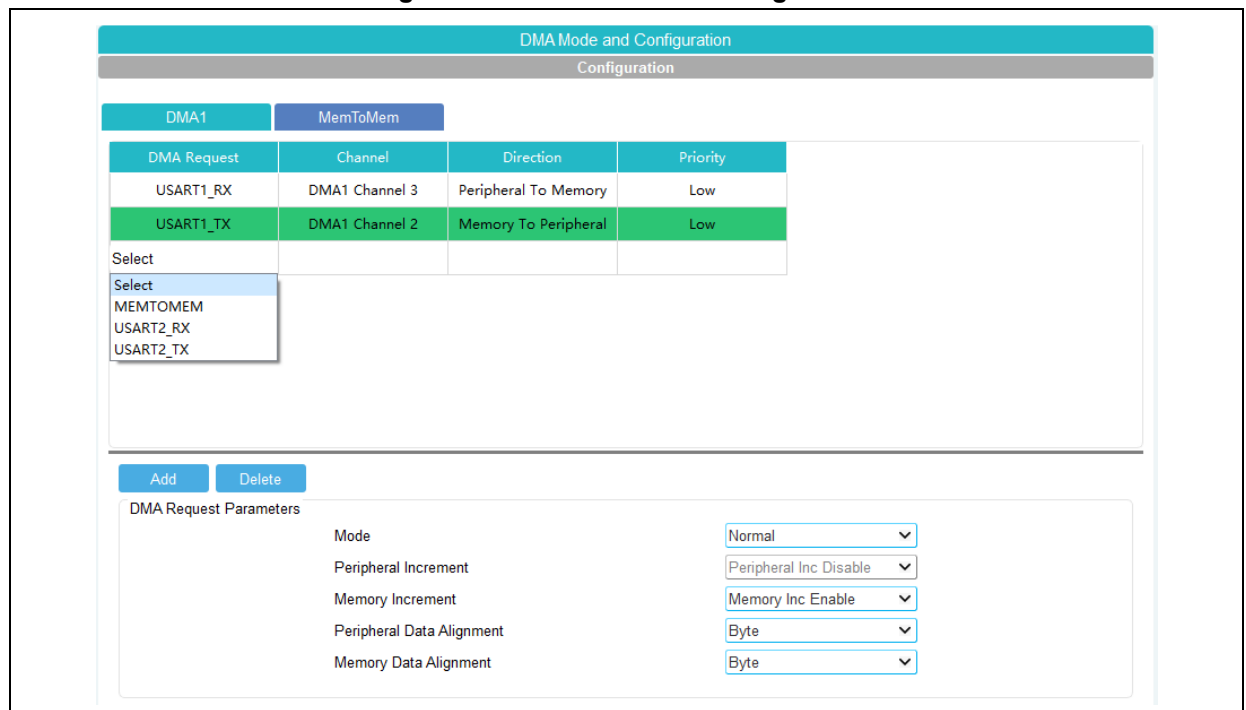
4.2.5 DMA mode and configuration

Click on “DMA” under “Peripherals” to open the DMA Mode and Configuration window to configure available general-purpose DMA controller. DMA interface supports data transfer between memory and peripherals and memory-to-memory data transfer when CPU is running.

Some peripherals have a dedicated DMA controller, which by default can be configured in the “DMA Mode and Configuration” window.

Click on “Add” to add a new line in the last row of the table in “Configuration” window. Optional DMA requests are listed in the combo box.

Figure 18. DMA mode and configuration



DMA request is used to reserve a data flow for data transfer between peripherals and memory. The priority determines which flow to select for the next DMA transfer.

Users can configure DMA in the “DMA Mode and Configuration” window.

■ Direction

Peripheral-to-memory, memory-to-peripheral, and memory-to-memory transfers.

■ Priority

There are four levels, including very high priority, high priority, medium priority and low priority. If the two channels have the same priority level, then the channel with lower number will get priority over the one with higher number. For example, channel 1 has priority over channel 2

■ Mode

There are normal mode, circular mode and peripheral flow control mode.

■ Peripheral increment

Configure peripheral increment type and enable peripheral increment. Once enabled, the peripheral address increments after each transfer.

■ Memory increment

Configure memory increment type and enable memory increment. Once enabled, the memory address increments after each transfer.

■ Peripheral data alignment

Configure peripheral data bit width (byte: 8-bit, half-word: 16-bit, word: 32-bit).

■ Memory data alignment

Configure memory data bit width (byte: 8-bit, half-word: 16-bit, word: 32-bit).

4.2.6 NVIC mode and configuration

Click on “NVIC” under “Peripherals” to open the NVIC Mode and Configuration window.

Click on the “Show” drop-down box to select to display all interrupts, available interrupts and enabled interrupts.

Figure 19. NVIC mode and configuration

NVIC Mode and Configuration				
Configuration				
Priority Group 4 bits for pre-emption priority, 0 bits for subp			Show All interrupts	
NVIC Interrupt Table	Enabled	Generate IRQ Handler	Preemption Priority	Sub Priority
NonMaskableInt_IRQ	☒	☒	0	0
HardFault_IRQ	☒	☒	0	0
MemoryManagement_IRQ	☒	☒	0	0
BusFault_IRQ	☒	☒	0	0
UsageFault_IRQ	☒	☒	0	0
SVCALL_IRQ	☒	☒	0	0
DebugMonitor_IRQ	☒	☒	0	0
PendSV_IRQ	☒	☒	0	0
SysTick_IRQ	☒	☒	15	0
WWDTC_IRQ	☐	☐	0	0
PVM_IRQ	☐	☐	0	0
TAMP_STAMP_IRQ	☐	☐	0	0
ERTC_WKUP_IRQ	☐	☐	0	0
FLASH_IRQ	☐	☐	0	0
CRM_IRQ	☐	☐	0	0
EXINT0_IRQ	☐	☐	0	0
EXINT1_IRQ	☐	☐	0	0
EXINT2_IRQ	☐	☐	0	0
EXINT3_IRQ	☐	☐	0	0
EXINT4_IRQ	☐	☐	0	0
DMA1_Channel1_IRQ	☐	☐	0	0
DMA1_Channel2_IRQ	☐	☐	0	0
DMA1_Channel3_IRQ	☐	☐	0	0
DMA1_Channel4_IRQ	☐	☐	0	0
DMA1_Channel5_IRQ	☐	☐	0	0

As shown in Figure 19, interrupts in gray are not configurable and the corresponding peripheral mode needs to be enabled; interrupts with the “Enabled” check box ticked and cannot be modified are system interrupts, and these interrupts cannot be disabled.

NVIC configuration includes “Priority Group”, “Enabled” (check box) and interrupt priority

(preemption priority and sub priority).

1. Priority Group

The priority group has multiple bits that can be used to define the NVIC priority. These bits are divided into two priority groups, corresponding to two priority types, i.e., preemption priority and sub priority. Priority bits indicate the number of priorities that can be configured, for example, 0: only one priority, 4: 16 priorities (0-15).

2. Click on an interrupt in the “NVIC Interrupt Table” to configure:

- Enabled: tick/untick to enable/disable this interrupt.
- Generate IRQ Handler: Controls whether to generate the IRQ Handler function.
- Preemption priority: Select a preemption priority. It defines the ability of one interrupt to interrupt another.
- Sub priority: Select a sub priority. It defines the interrupt priority.

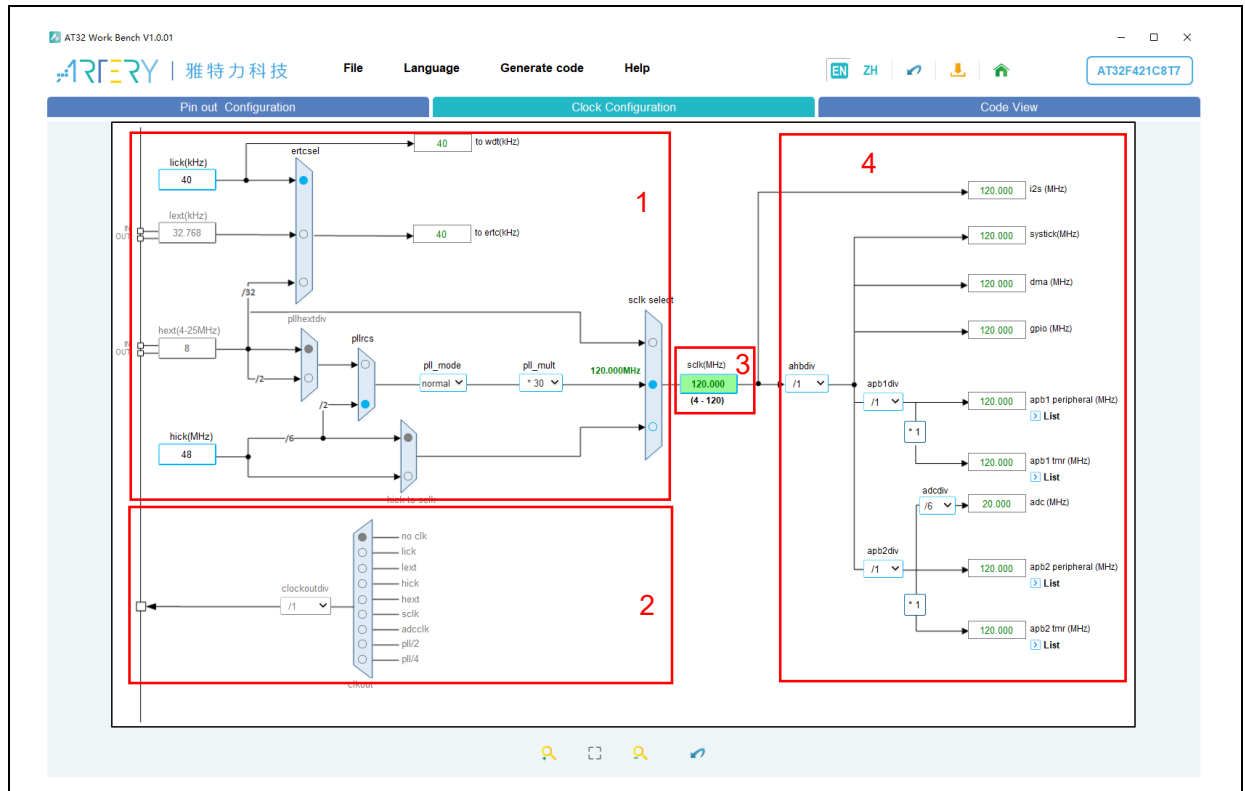
Dedicated peripheral interrupts also can be configured in the NVIC Mode and Configuration window.

4.3 Clock configuration

Users can configure the clock path and parameters in the Clock Configuration window, and use drop-down menus and input boxes to modify the actual clock tree configuration to meet application requirements.

As shown in Figure 20, the Clock Configuration window mainly includes four blocks.

Figure 20. Clock configuration



1. Configuration: Select and configure the clock path and parameters as needed.
2. Output: Configure the clock output (CLKOUT).
3. SCLK: It is an input box when PLL is used as the system clock. Users can input the desired system clock frequency to automatically configure the frequency multiplication factor.
4. Result: Display the clock frequency of the current peripheral, and peripherals on bus.

Users can press the Ctrl + scroll mouse wheel to adjust the MCU clock configuration window size. The toolbar of the Clock Configuration window has the following functions:

: Zoom into the clock configuration view.

: Restore the clock configuration view to the initial size and position.

: Zoom out the clock configuration view.

: Reset clock configuration.

Note 1: Set the “LEXT” mode in the “CRM Mode and Configuration” window to enable LEXT.

Note 2: Set the “HEXT” mode in the “CRM Mode and Configuration” window to enable HEXT.

Note 3: Tick “Clock Output” in the “CRM Mode and Configuration” window to enable clockout.

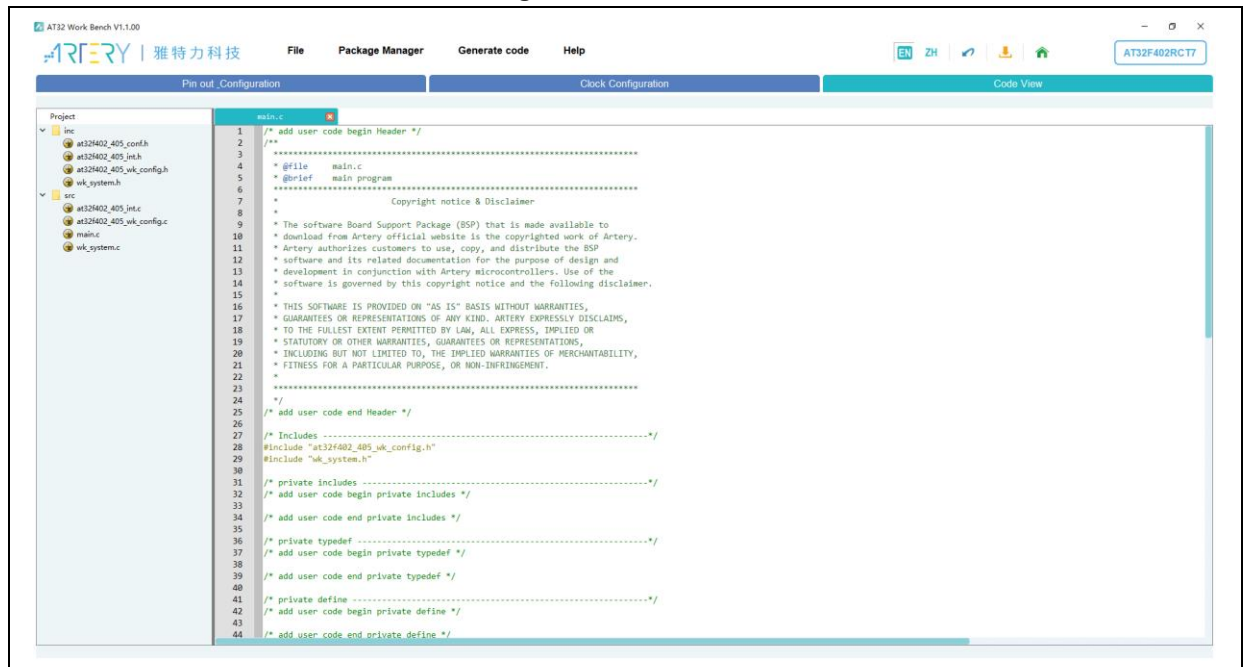
4.4 Code view

Click on the “Code View” to generate code automatically. Code files are listed in the left, and the corresponding code is shown in the right window.

- main.c: main source file, which is used to call peripheral initialization functions.
- AT32xxxx_wk_config.h: header file for peripheral configuration, which is used for declaration of each peripheral initialization function.
- AT32xxxx_wk_config.c: source code for peripheral configuration, which is used to define each peripheral initialization function.
- AT32xxxx_int.h: header file of interrupt functions.
- AT32xxxx_int.c: source file of interrupt functions.
- AT32xxxx_conf.h: header file for library configuration.

Users can view the code generated automatically according to the current configuration.

Figure 21. Code view



Users can press the Ctrl + scroll mouse wheel to adjust the code view size, and use the right click menu to:

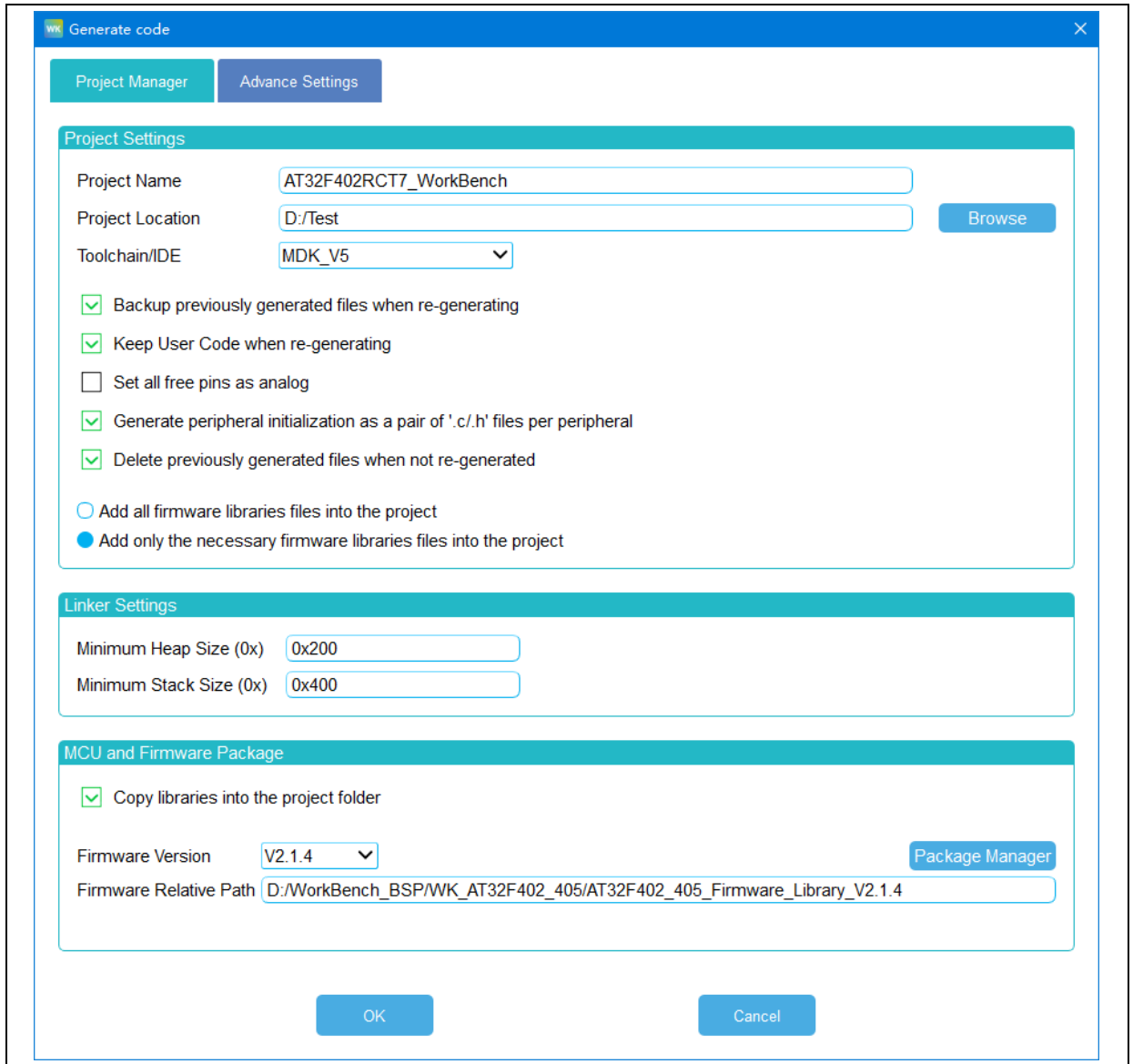
- Fold all: fold code to view its structure;
- Expand all: expand all folded code;
- Copy: cope the selected code;
- Select all: select all of the code.
- Find: Search keywords in the preview code;
- EnCoding: Switch the encoding format to display code.
- Theme: Switch the “Light theme” and “Dark theme”.

4.5 Generate code

Click on the “Generate code” in the menu bar or the  icon in the toolbar, and then the “Project Manager” window pops up.

4.5.1 Project Manager

Figure 22. Project Manager



The Project Manager dialog box is titled "Generate code" and contains the following settings:

- Project Settings:**
 - Project Name: AT32F402RCT7_WorkBench
 - Project Location: D:/Test (with a Browse button)
 - Toolchain/IDE: MDK_V5 (dropdown menu)
 - Backup previously generated files when re-generating: ☒
 - Keep User Code when re-generating: ☒
 - Set all free pins as analog: ☐
 - Generate peripheral initialization as a pair of '.c/.h' files per peripheral: ☒
 - Delete previously generated files when not re-generated: ☒
 - Add all firmware libraries files into the project: ☐
 - Add only the necessary firmware libraries files into the project: ☒
- Linker Settings:**
 - Minimum Heap Size (0x): 0x200
 - Minimum Stack Size (0x): 0x400
- MCU and Firmware Package:**
 - Copy libraries into the project folder: ☒
 - Firmware Version: V2.1.4 (dropdown menu, with a Package Manager button)
 - Firmware Relative Path: D:/WorkBench_BSP/WK_AT32F402_405/AT32F402_405_Firmware_Library_V2.1.4

Buttons: OK, Cancel

Users can configure required parameters as follows:

■ Project Settings

- Project name: create the project name.
- Project location: storage directory for the project folder.
- Toolchain/IDE: Generate a project of the selected toolchain/IDE.
- Backup previously generated files when re-generating: Backup the .c/.h files.
- Keep User Code when re-generating code. Refer to [Section 4.5.2](#) for details.
- Set all free pins as analog (to optimize the power consumption).
- Generate peripheral initialization as a pair of ".c/.h" files per peripheral: The peripheral initialization code is configured to a separate ".c/.h" file when the code is generated. For example, "usart.c" and "usart.h" are generated for USART.

- Delete previously generated files when not re-generated.
- Add all firmware libraries files into the project.
- Add only the necessary firmware libraries files into the project.

■ Linker Settings

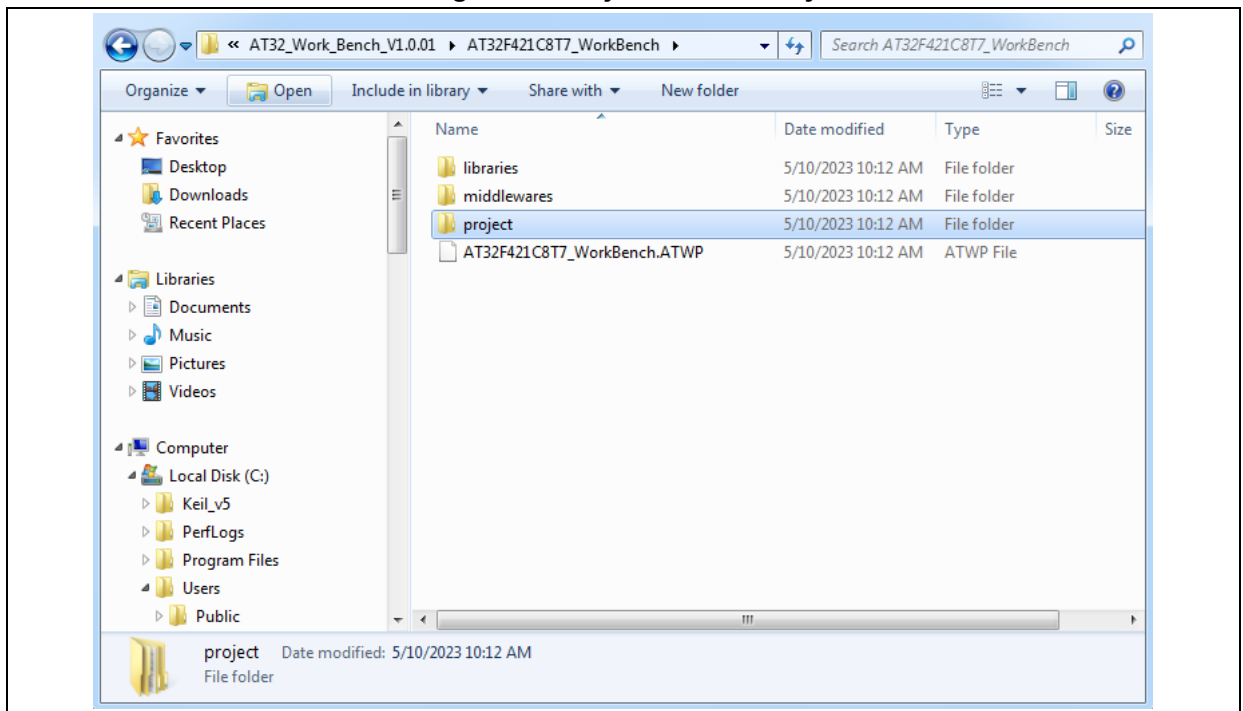
- Minimum heap size and minimum stack size. Default values are 0x200 and 0x400, respectively. Users can modify the value when middleware stack is used.

■ MCU and Firmware Package

- Copy libraries into the project folder: Tick this check box, and libraries in the firmware package are copied automatically into the project folder when generating code.
- Package Version: Select the firmware package version. If the package is not setup, click on “Package Manager” to install.
- Package Manager: Install and manage firmware package. Refer to [Section 4.6](#) for details.
- Firmware Relative File: Display the path of the selected firmware package.

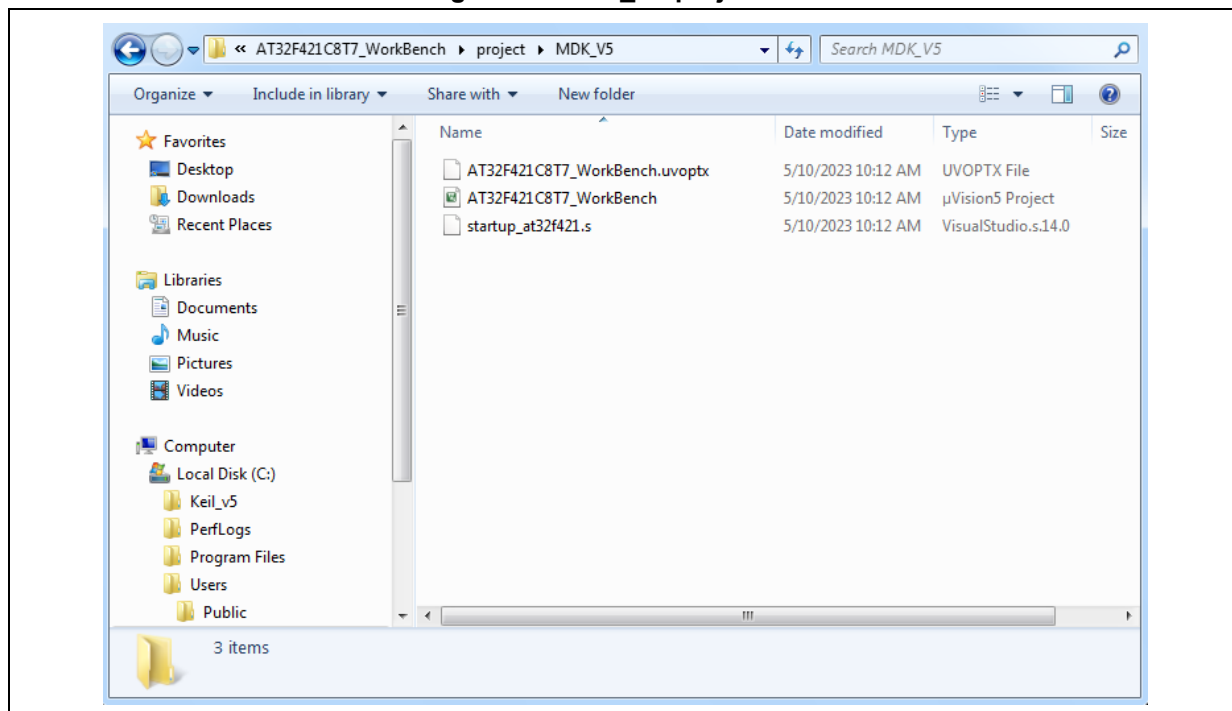
Finally, click on “OK” to generate user code and project of the selected IDE automatically. The generated project file structure is shown in Figure 23.

Figure 23. Project file directory



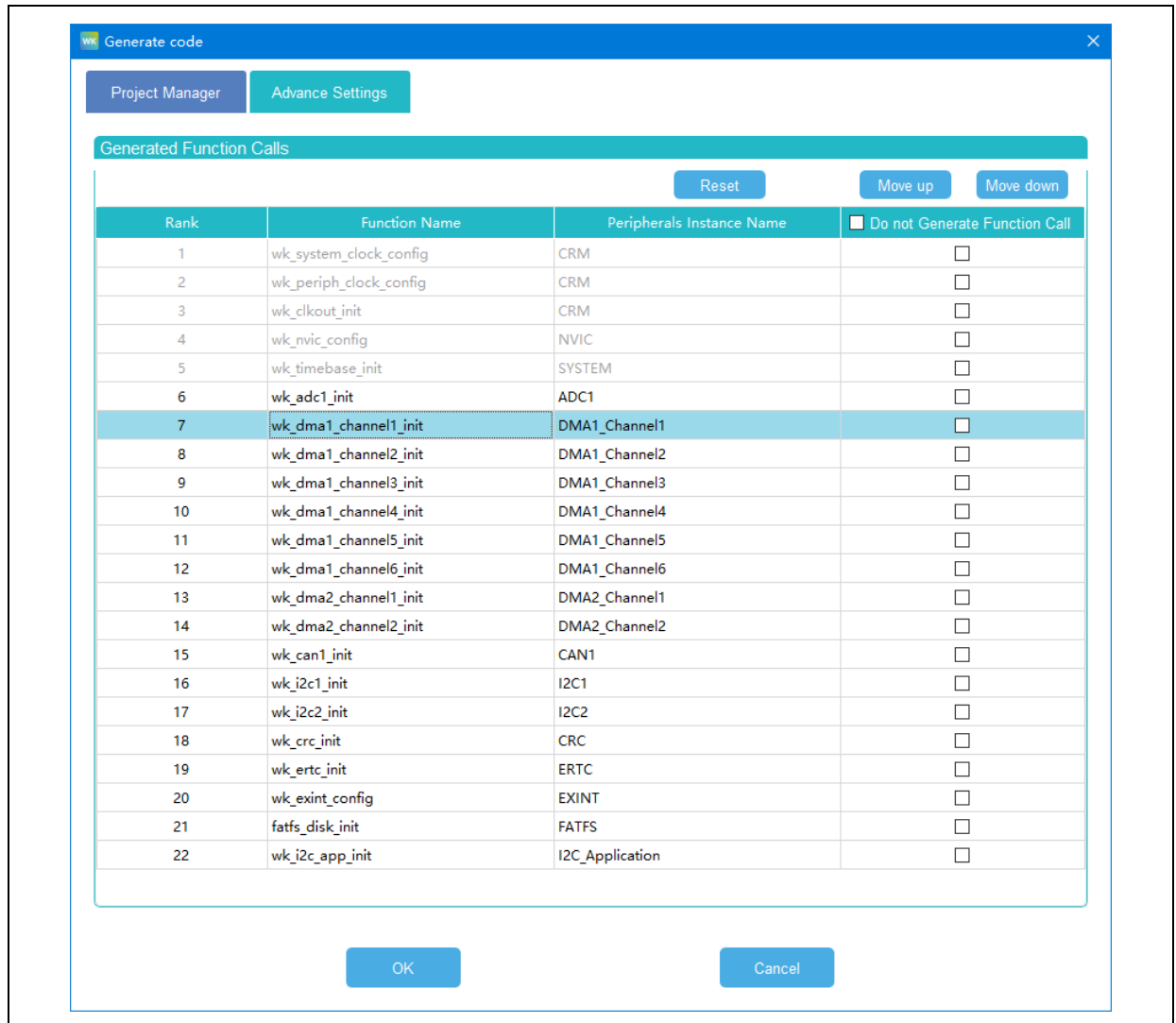
The file directory contains the “*libraries*” and “*middleware*” folders copied from the firmware package, as well as the generated project folder “*Project*”. The “*Project*” contains a header file folder “*inc*”, source code folder “*src*” and “*MDK_V5*” (generated according to the selected toolchain/IDE). The *MDK_V5* folder contains MDK project files that can be opened in Keil.

Figure 24. MDK_V5 project files



4.5.2 Advance Settings

Figure 25. Advance Settings



■ Generated Function Calls

This feature allows you to adjust the call order of peripheral and middleware initialization functions within the main() function, and to control whether these functions are called.

- Reset: Reset the call order to restore the default sequence.
- Move up: Move this function to an earlier position in the call sequence.
- Move down: Move this function to a later position in the call sequence.
- Do not Generate Function Call: Do not call this function in main().

4.5.3 Keep user code when re-generating

The AT32 Work Bench generated C code allows users to add custom code. The custom code needs to be inserted into the software-defined location and can be reserved for the next generation of C code. The software-defined location is as follows:

```
/* add user code begin ..... */
```

```
/* add user code end ..... */
```

When re-generating code, user codes in the software-defined position are reserved and not moved or renamed, while user codes not in the software-defined location are ignored and discarded.

Note 1: User-defined “add user code” tag is not supported. Custom code can be added to software-defined tags only.

Note 2: Code annotation cannot be the same as the software-defined “add user code” tag.

Take the main function in *main.c* as an example. User code can be added to the software-defined positions (in bold).

```
int main(void)
{
    /* add user code begin 1 */

    /* add user code end 1 */

    /* system clock config. */
    wk_system_clock_config();

    /* nvic config. */
    wk_nvic_config();

    /* add user code begin 2 */

    /* add user code end 2 */

    while(1)
    {
        /* add user code begin 3 */

        /* add user code end 3 */
    }
}
```

4.6 Package manager

Click on “File” – “Package Manager”, and the Package Manager window pops up, as shown below.

Figure 26. Package Manager -- Firmware

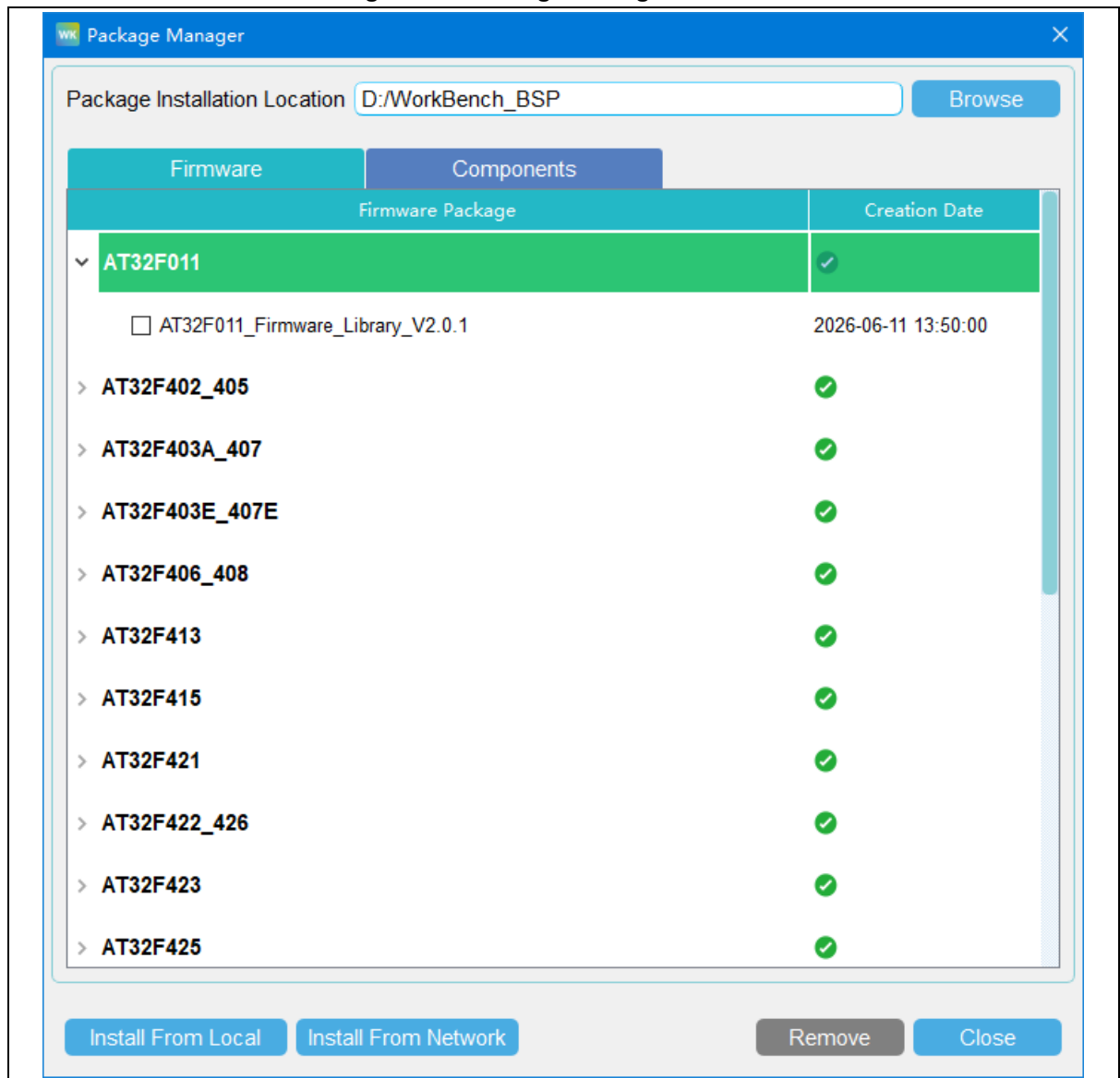
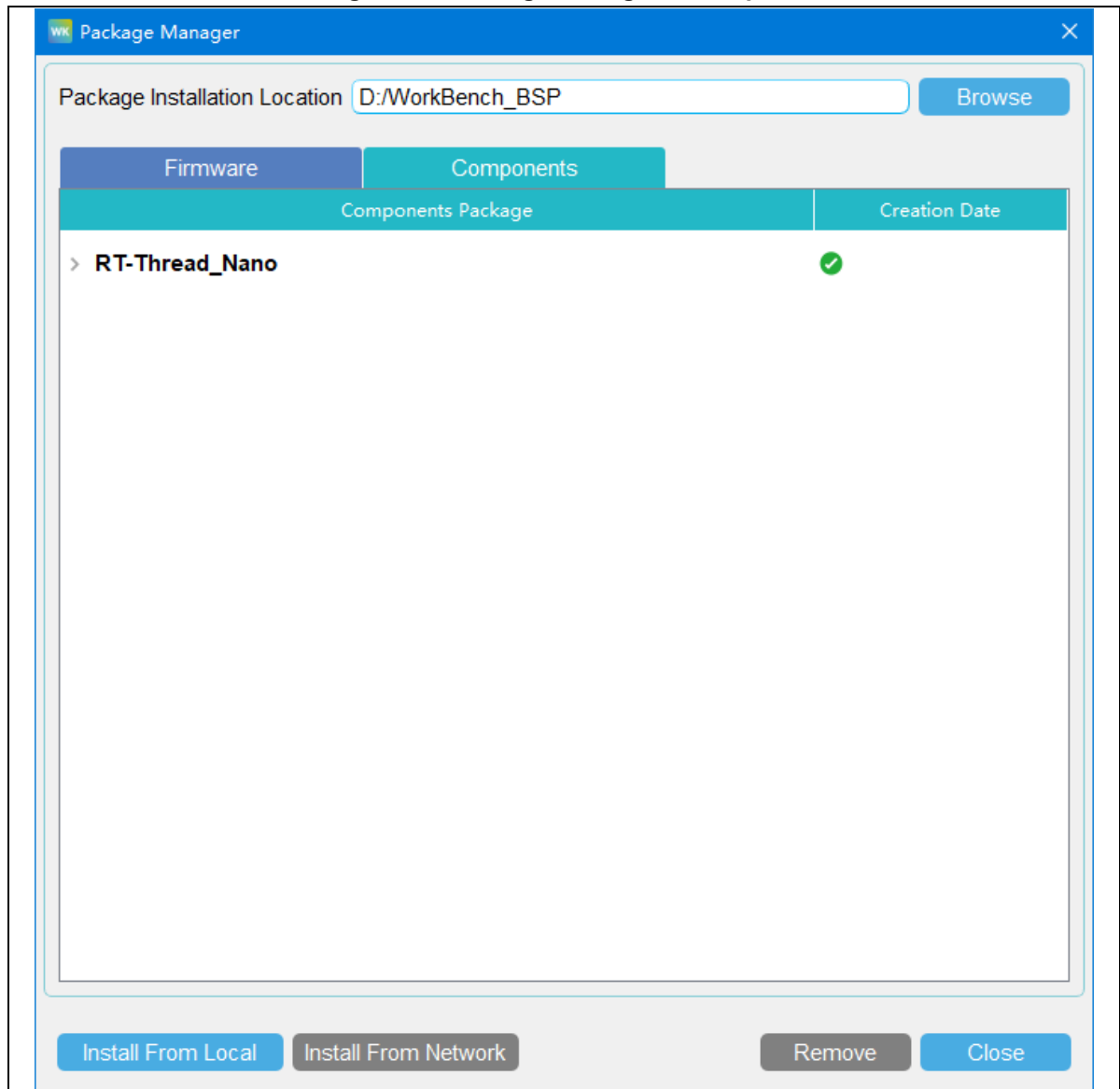


Figure 27. Package Manager -- Components



■ Package installation location

Select and confirm the package installation path. The software scans for the installed package at this location each time it starts up.

The installed package is stored in the "WK_AT32xxx" directory, such as WK_AT32F425.

Note: Users need to select the installation location after software update.

■ Install from local

Install the package that has been downloaded. This local package (*.ZIP) should be selected manually.

■ Install from Network

Download package from Network and then set up automatically.

■ Remove

Tick the installed package and then remove it.

5 Revision history

Table 4. Document revision history

Date	Version	Revision note
2026/07/30	V1.12	Updated some descriptions.
2026/04/28	V1.11	Updated some descriptions.
2026/01/23	V1.10	Updated some descriptions.
2025/11/07	V1.09	Updated some descriptions.
2024/12/30	V1.08	Updated some descriptions.
2024/12/25	V1.07	Updated some descriptions.
2024/10/28	V1.06	Updated some descriptions and figures.
2024/08/15	V1.05	Updated some descriptions and figures.
2024/03/05	V1.04	Updated some descriptions.
2024/01/26	V1.03	Added Section 4.6 "Package manager".
2023/11/28	V1.02	Updated some figures.
2023/09/08	V1.01	Updated some descriptions.
2023/05/09	V1.00	Initial release.

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.