



RT58x SOC Platform

Getting Started

V1.4

About this Document

This document supports at least “**Rafael RT58x SDK v0.3.1**”. For any version older than v0.3.1, there is no guaranteed it can work properly.

Table of Contents

About this Document	1
1. Introduction	2
2. Development Environment Setup	3
2.1 Debugger	3
2.2 Development Board	3
2.3 KEIL MDK-ARM	4
2.4 Visual Studio Code	4
3. Software Development Kit	5
3.1 SDK Software Architecture	5
3.2 SDK Directory Structure	6
4. Develop	7
4.1 Creating Applications	7
4.2 Debugging	16
4.3 Flash Programming	26
4.4 Developing Applications	32
4.4.1 Bootloader	32
4.4.2 Keil Projects	32
4.4.3 GCC Projects	35
Revision History	39

1. Introduction

The purpose of this document is to teach users about the installation and configuration of Rafael RT58x SDK, Keil MDK-ARM, and Visual Studio Code. The SDK provides examples and projects to develop applications on development kit.

2. Development Environment Setup



Rafael RT58x EVK



ARM J-Link Adapter



IDE/CMSIS

2.1 Debugger

- Install the ARM J-Link driver.
- Connect J-Link Adapter to RT58x EVK board via JTAG/SWD bus.
- Connect J-Link Adapter to PC via USB.

2.2 Development Board

The RT58x EVK provides an SWD interface with connector for use with an ICE debugger (J-Link Adapter) via 20pin IDE cable.



For detailed description of RT58x EVK board, please refer to RT58X EVK User Guide.

2.3 KEIL MDK-ARM

The MDK-ARM is a complete software development environment for Cortex-M, Cortex-R4, ARM7, and ARM9 processor-based devices. MDK-ARM is specifically designed for microcontroller applications, it is easy to learn and use, yet powerful enough for the most demanding embedded applications.

For detailed introduction and download, please click the following link:

<http://www.keil.com/arm/mdk.asp>

Note: MDK-Lite Edition is available for download. It does not require a serial number or license key. It features software development for microcontrollers based on ARM Cortex-M processors. It is intended for product evaluation, small projects, and the educational market. It is restricted to 32Kbyte code size.

2.4 Visual Studio Code

Visual Studio Code combines the simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging. Visual Studio Code features a lightning fast source code editor, perfect for day-to-day use. Visual Studio Code includes an interactive debugger, so you can step through source code, inspect variables, view call stacks, and execute command in the console. Visual Studio Code also integrates with build and scripting tools to perform common tasks making everyday workflows faster.

For detailed introduction and download, please click the following link:

<https://code.visualstudio.com>

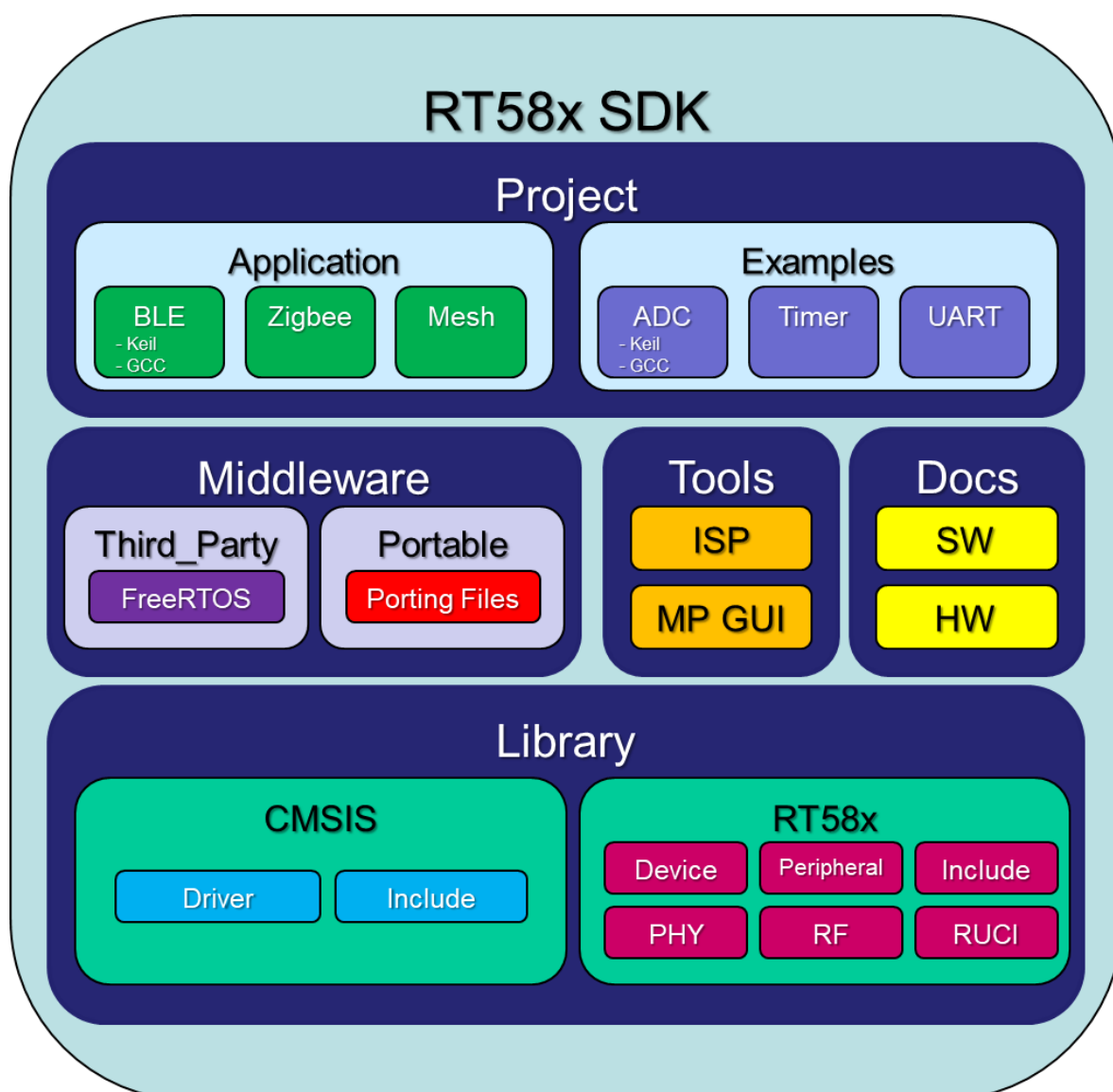
Note: VS Code is available and free for download. It does not require a serial number or license key. For detailed installation of VS Code, please refer to VS Code User Setup Readme.

3. Software Development Kit

Rafael RT58x SDK is a complete software development kit for application development such as BLE, Zigbee, Mesh, and Sub-G. Rafael RT58x SDK is specifically designed for Rafael's SoC with the ability to perform high-performance Cortex-M microcontroller and support to driver powerful RF and peripheral features. It is easy to install the Rafael RT58x SDK package by unzipping the RT58x_SDK_version.zip to the specified development directory.

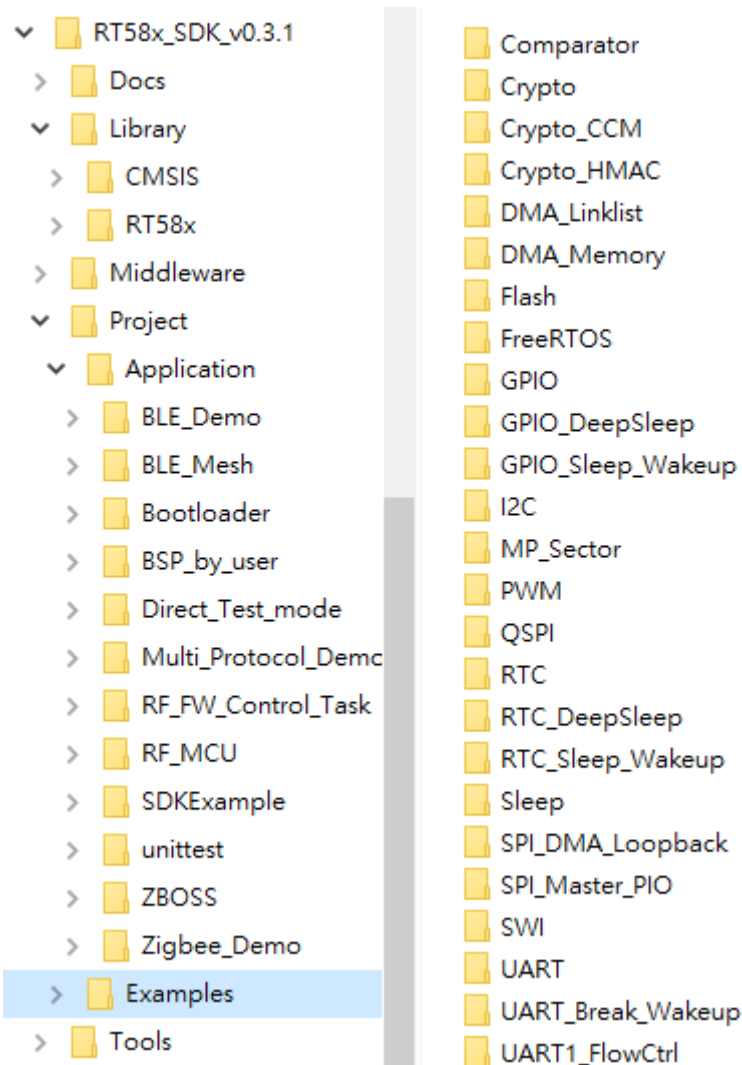
3.1 SDK Software Architecture

The following figure shows the software architecture in the Rafael RT58x SDK. This illustration will guide users to use RT58x to quickly develop structured applications.



3.2 SDK Directory Structure

The following figure shows the related files and directories in the Rafael RT58x SDK. These files and directories contain project, middleware, library, tools, and documents. Rafael RT58x SDK also provides complete examples for user reference. Users can add and modify related files to the corresponding directory to develop the required applications.



4. Develop

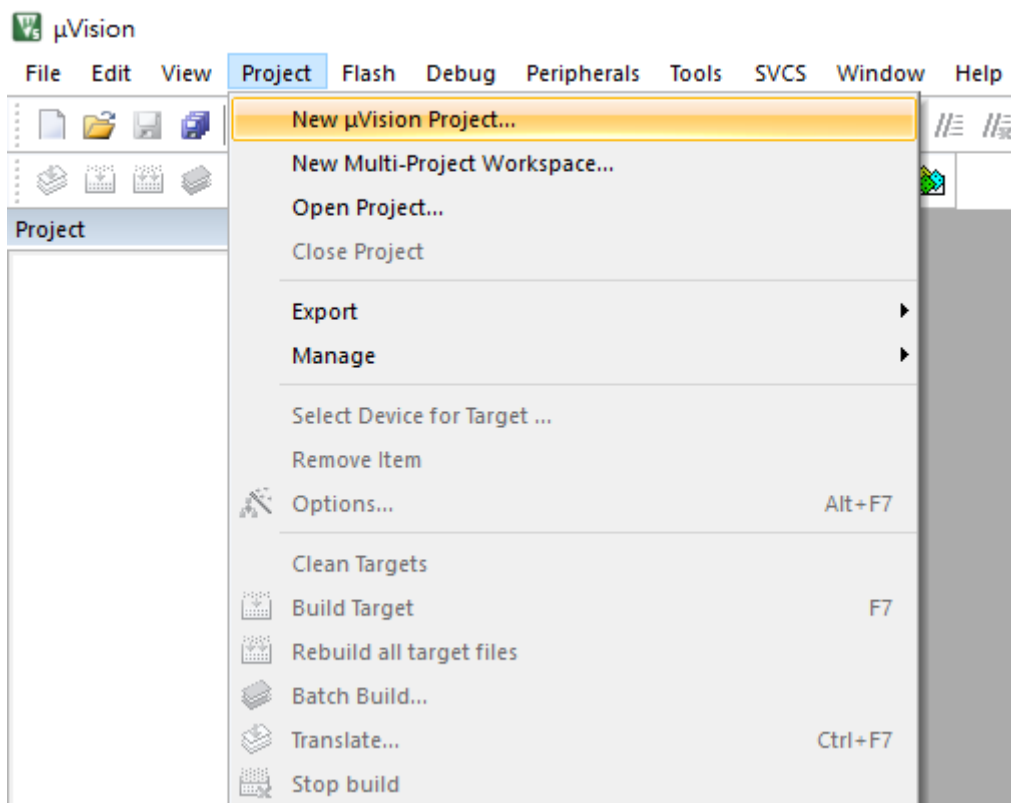
The uVision IDE and Debugger is the central part of the Keil development toolchain and has numerous features that help the programmer to develop embedded applications quickly and successfully. uVision offer a Build Mode for creating applications and a Debug Mode for debugging applications.

4.1 Creating Applications

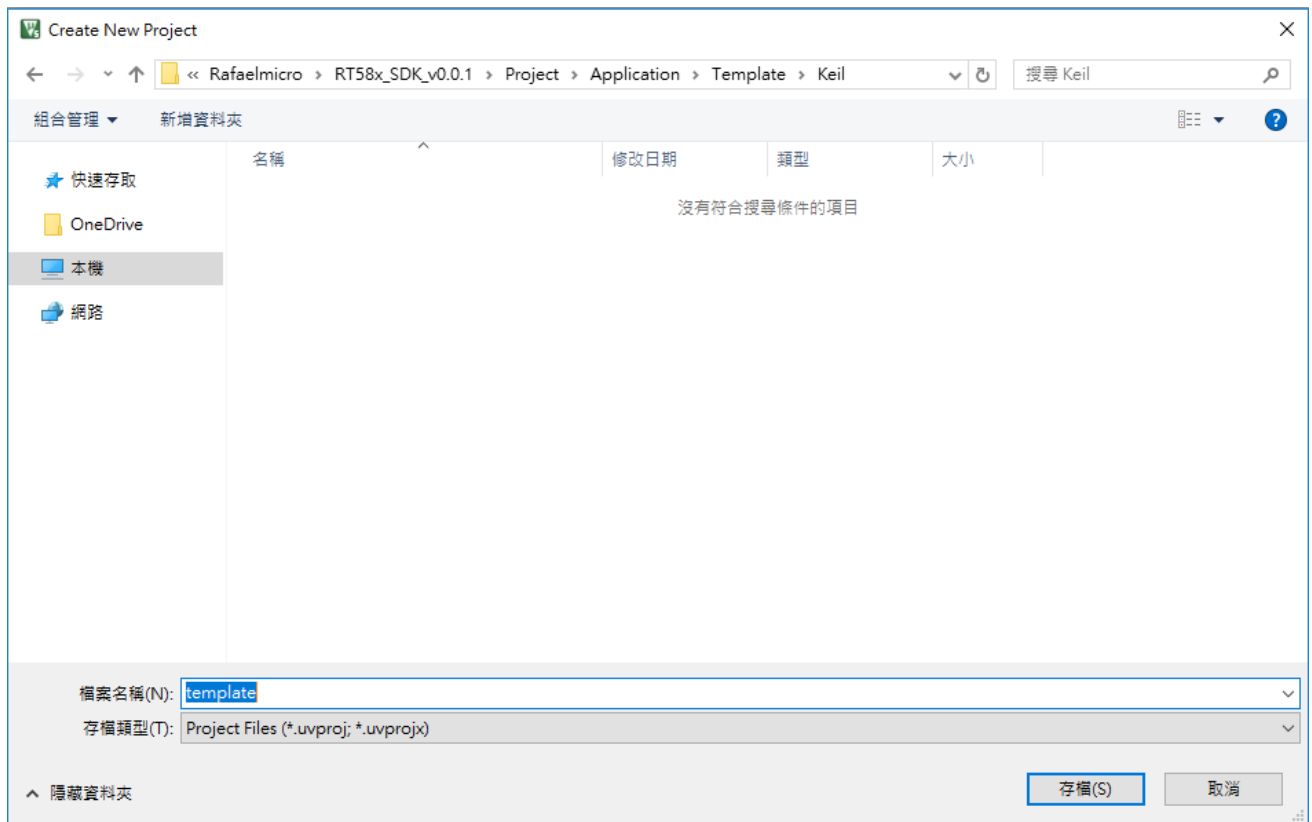
This chapter describes the creation of projects using Software Components, shows features to edit and compile source files, fix errors and warnings, and generate executable code.

The following will provide a step-by-step tutorial that creates a simple project. The Project Manager makes it easy to create a new project and to design the embedded application.

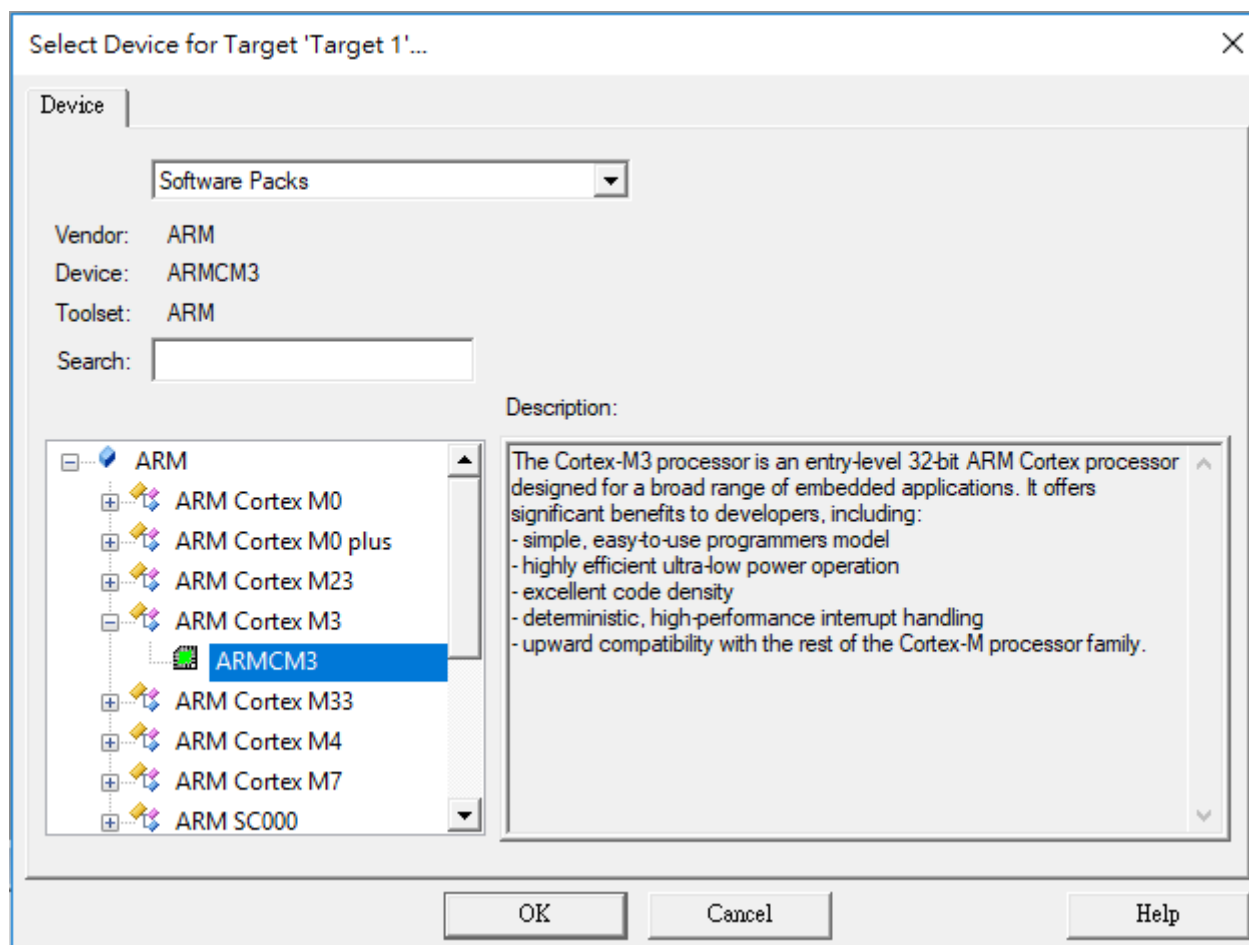
Step 1: Setup the Project. The menu Project → New uVision Project creates a new project.



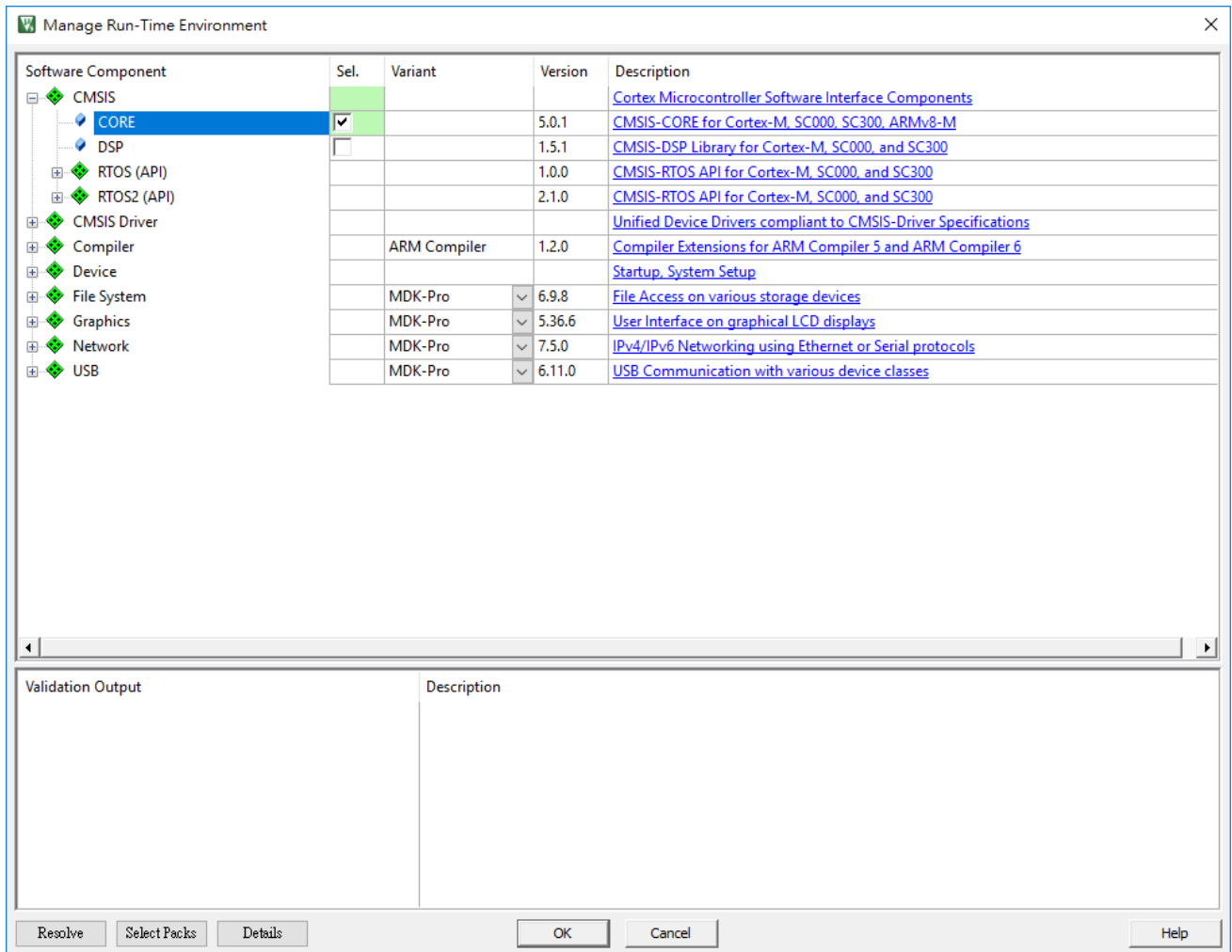
Step 2: Select an empty folder and enter the project name, for example *template*.



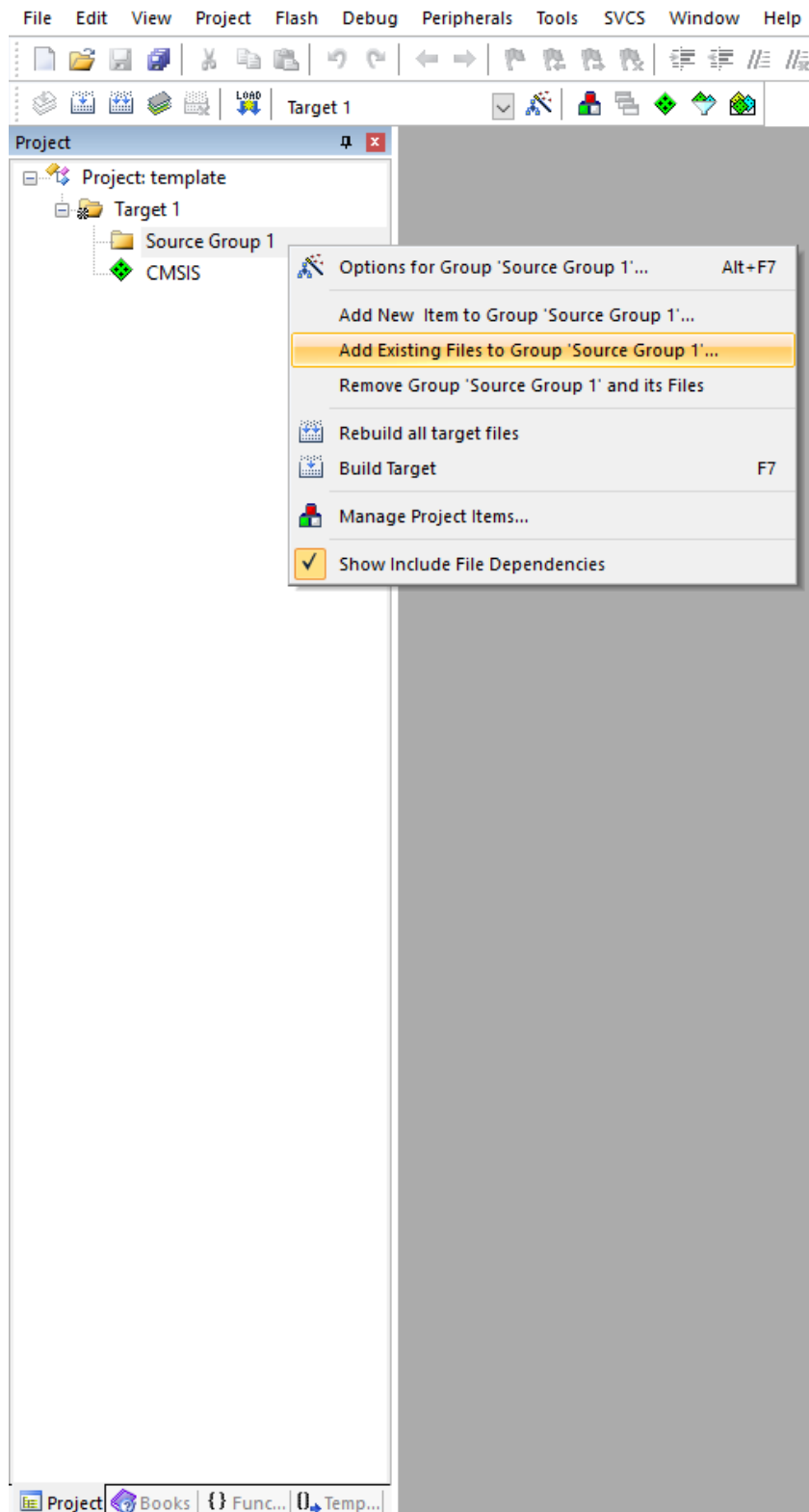
Step 3: Select ARM Cortex M3 → ARMCM3 as the device database.



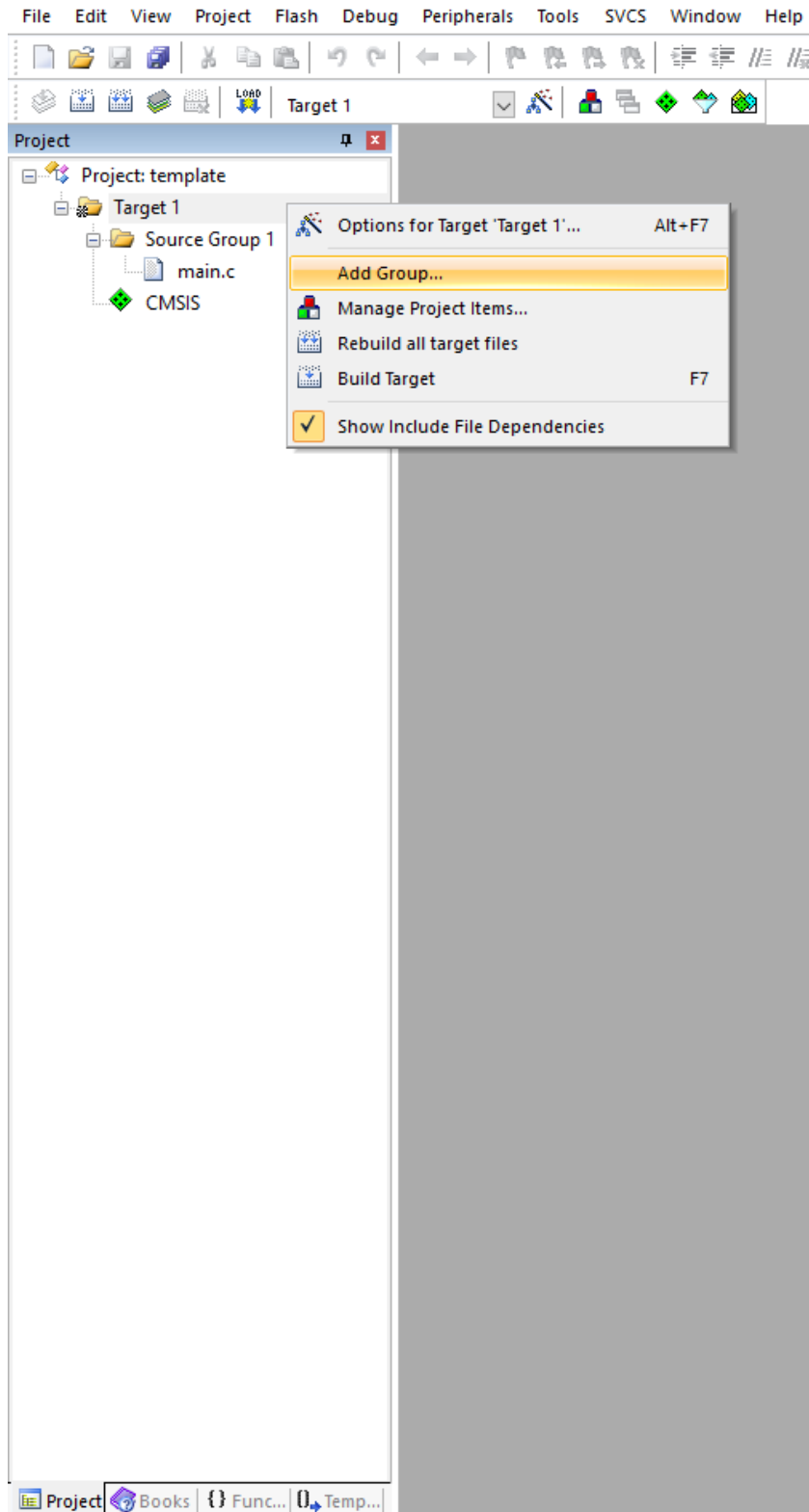
Step 4: Select Software Components. Add CMSIS → CORE to the project.

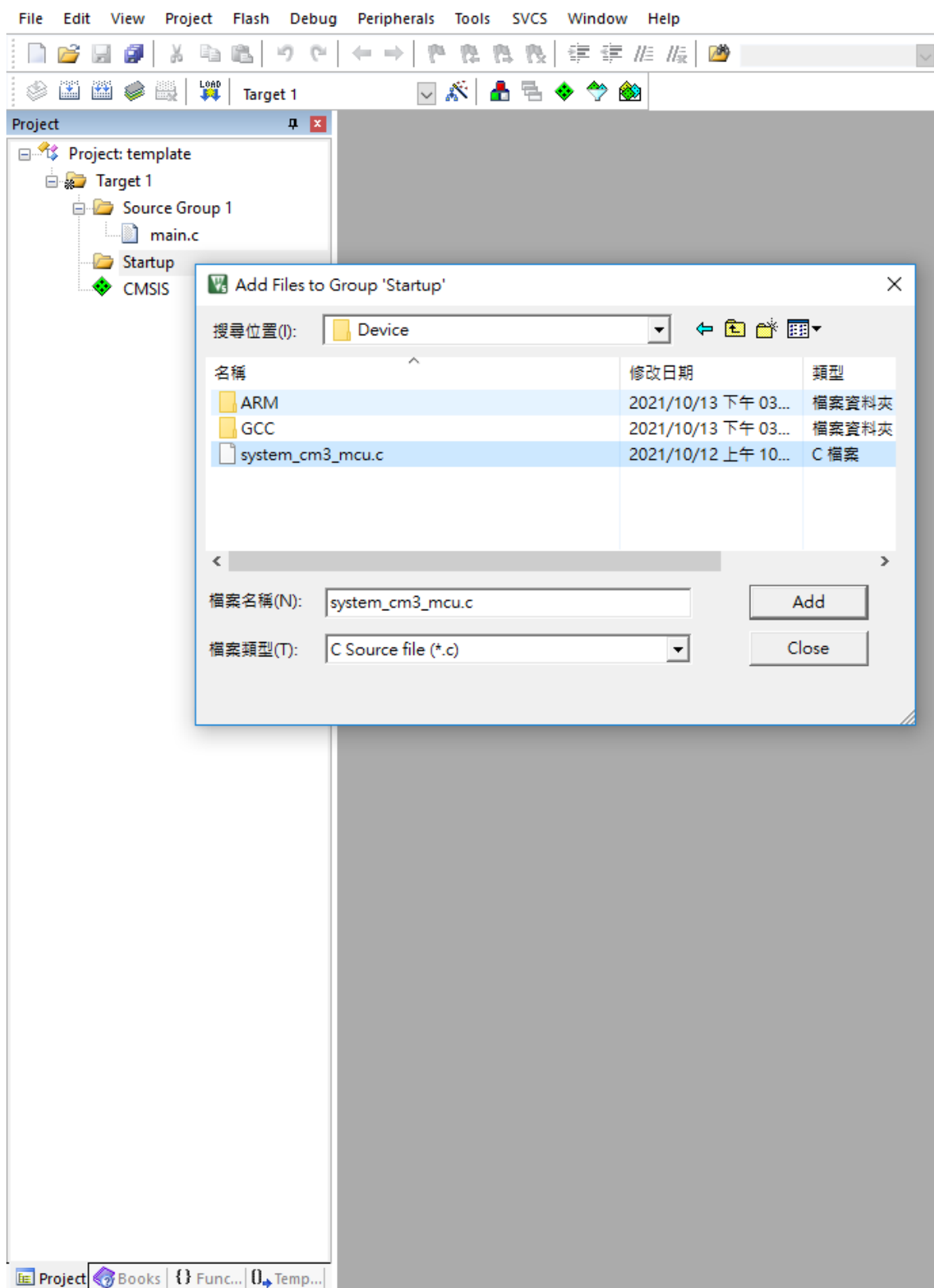


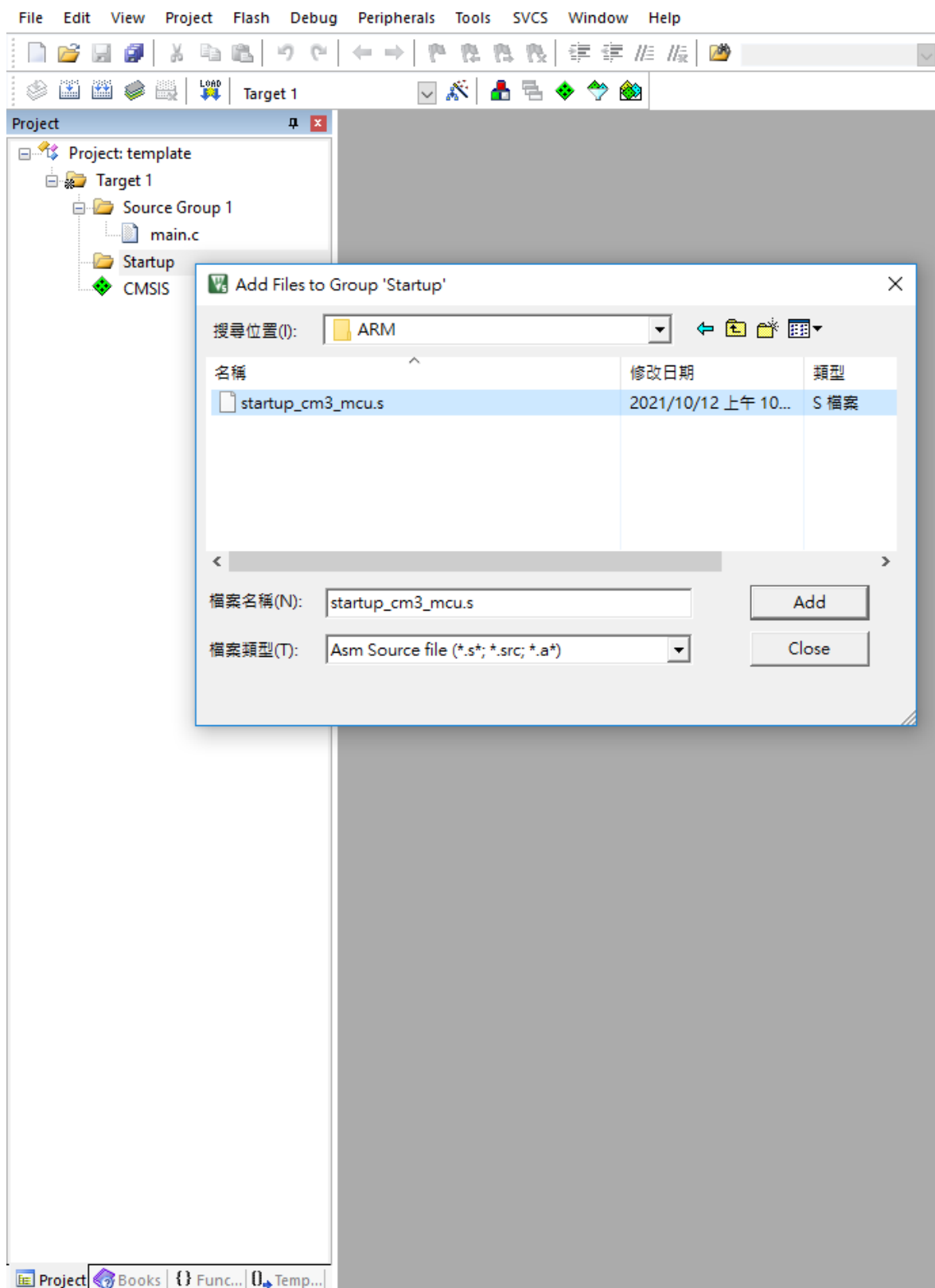
Step 5: Add Source Files to Project. Click on a file group in the window Project and use the context menu “Add New Item to Group” or “Add Existing Files to Group”.

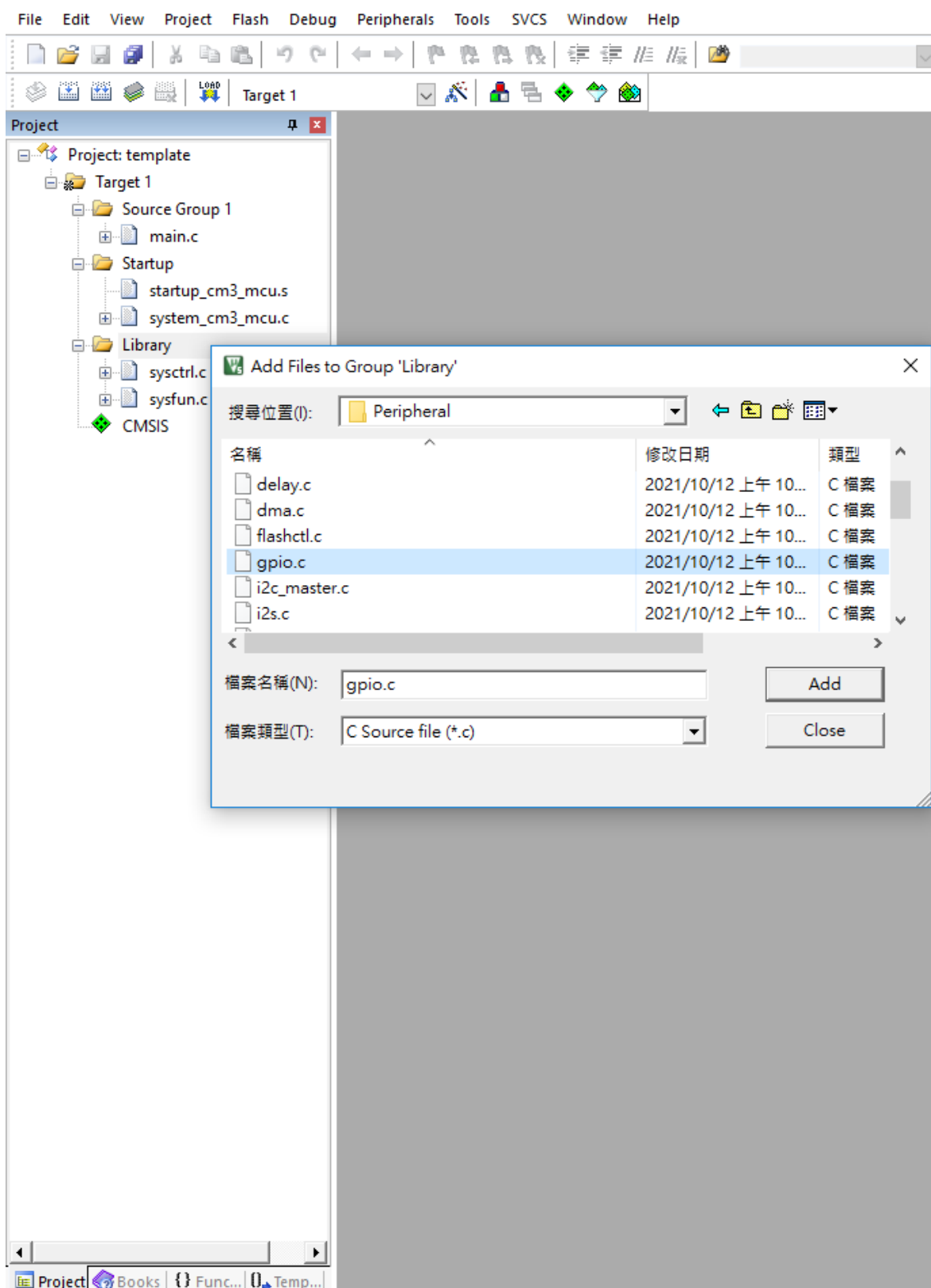


Step 6: Create file groups to simplify the project maintenance. Right-click the target name and select “Add Group” to add “Startup” group and “Library” group, and add startup related files and library related files to the corresponding groups.







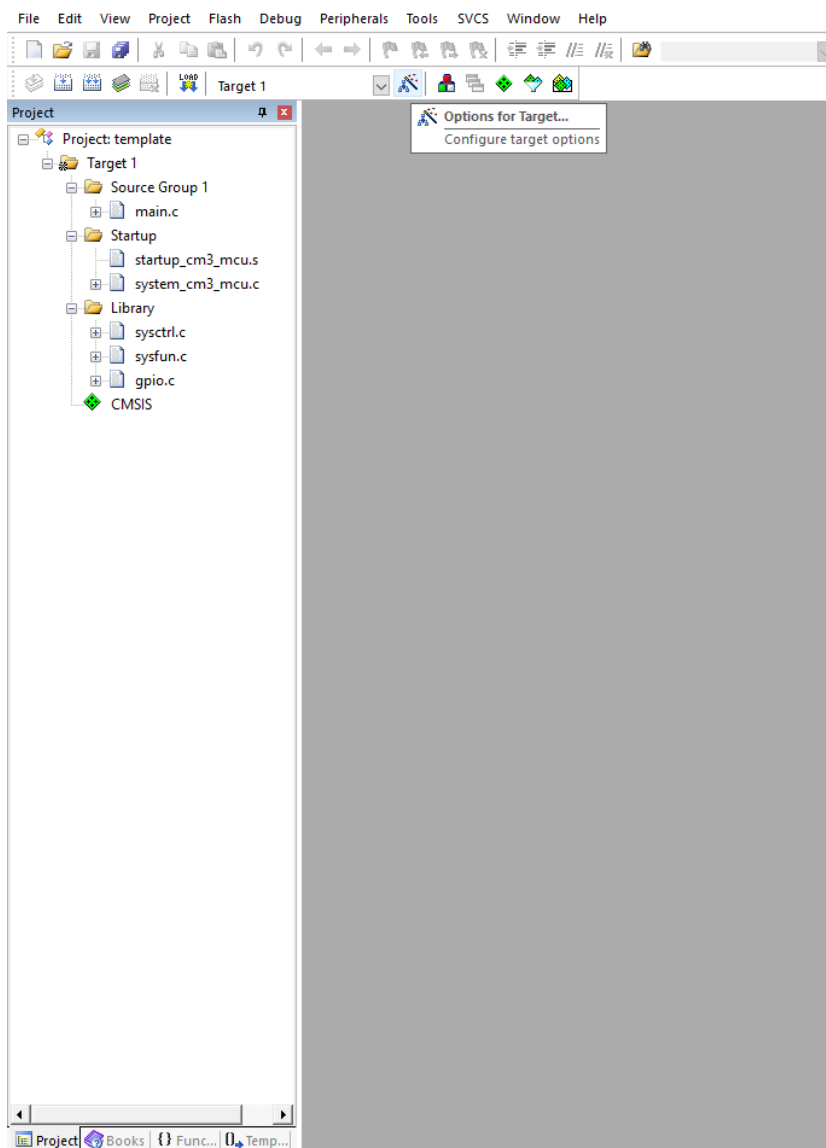


4.2 Debugging

This chapter describes the Debugger for controlling the target device using a debug adapter. While debugging, developers have full access to the source code and can control as well as analyze program execution on C or Assembly level.

The following will provide a step-by-step tutorial that uses Target Debugger connects to a debug adapter.

Step 1: Set the Options for Target. The options configure the target hardware and the development environment.



Step 2: Set Xtal, IROM, and IRAM options depend on the RT58x specifications.

a. RT58x (1MB Flash size) IROM and IRAM setting

Options for Target 'Target 1'

Device Target Output Listing User C/C++ Asm Linker Debug Utilities

ARM ARMCM3

Xtal (MHz): <undefined>

Operating system: None

System Viewer File: ARMCM3.svd

☐ Use Custom File

Code Generation
ARM Compiler: Use default compiler version 5

☐ Use Cross-Module Optimization
☒ Use MicroLIB ☐ Big Endian

Read/Only Memory Areas

default	off-chip	Start	Size	Startup
<input type="checkbox"/>	ROM1:			<input type="radio"/>
<input type="checkbox"/>	ROM2:			<input type="radio"/>
<input type="checkbox"/>	ROM3:			<input type="radio"/>
	on-chip			
<input checked="" type="checkbox"/>	IROM1:	0x8000	0x74000	<input checked="" type="radio"/>
<input type="checkbox"/>	IROM2:			<input type="radio"/>

Read/Write Memory Areas

default	off-chip	Start	Size	NoInit
<input type="checkbox"/>	RAM1:			<input type="checkbox"/>
<input type="checkbox"/>	RAM2:			<input type="checkbox"/>
<input type="checkbox"/>	RAM3:			<input type="checkbox"/>
	on-chip			
<input checked="" type="checkbox"/>	IRAM1:	0x20000000	0x20000	<input type="checkbox"/>
<input type="checkbox"/>	IRAM2:			<input type="checkbox"/>

OK Cancel Defaults Help

b. RT58x (512KB Flash size) IROM and IRAM setting

Options for Target 'RT58x'

Device Target Output Listing User C/C++ Asm Linker Debug Utilities

ARM ARMCM3

Xtal (MHz): <undefined>

Operating system: None

System Viewer File: ARMCM3.svd

☐ Use Custom File

Code Generation
ARM Compiler: Use default compiler version 5

☐ Use Cross-Module Optimization
☒ Use MicroLIB ☐ Big Endian

Read/Only Memory Areas

default	off-chip	Start	Size	Startup
<input type="checkbox"/>	ROM1:			<input type="radio"/>
<input type="checkbox"/>	ROM2:			<input type="radio"/>
<input type="checkbox"/>	ROM3:			<input type="radio"/>
	on-chip			
<input checked="" type="checkbox"/>	IROM1:	0x8000	0x39000	<input checked="" type="radio"/>
<input type="checkbox"/>	IROM2:			<input type="radio"/>

Read/Write Memory Areas

default	off-chip	Start	Size	NoInit
<input type="checkbox"/>	RAM1:			<input type="checkbox"/>
<input type="checkbox"/>	RAM2:			<input type="checkbox"/>
<input type="checkbox"/>	RAM3:			<input type="checkbox"/>
	on-chip			
<input checked="" type="checkbox"/>	IRAM1:	0x20000000	0x20000	<input type="checkbox"/>
<input type="checkbox"/>	IRAM2:			<input type="checkbox"/>

OK Cancel Defaults Help

c. RT58x 2MBB Flash size) IROM and IRAM setting

Options for Target 'Target 1'

Device Target Output Listing User C/C++ Asm Linker Debug Utilities

ARM ARMCM3

Xtal (MHz): <undefined>

Operating system: None

System Viewer File: ARMCM3.svd

☐ Use Custom File

Code Generation

ARM Compiler: Use default compiler version 5

☐ Use Cross-Module Optimization

☒ Use MicroLIB ☐ Big Endian

Read/Only Memory Areas

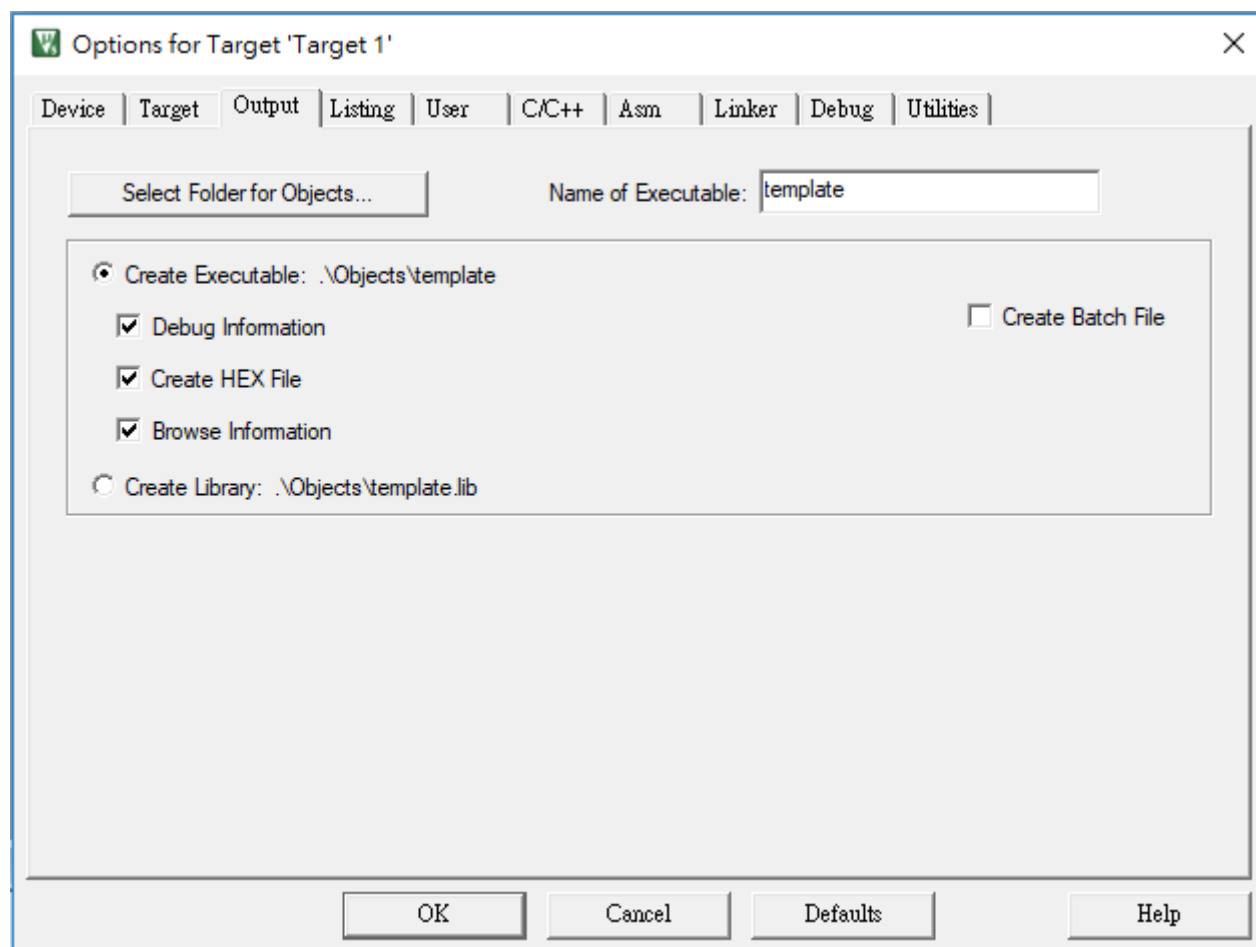
default	off-chip	Start	Size	Startup
<input type="checkbox"/>	ROM1:			<input type="radio"/>
<input type="checkbox"/>	ROM2:			<input type="radio"/>
<input type="checkbox"/>	ROM3:			<input type="radio"/>
	on-chip			
<input checked="" type="checkbox"/>	IROM1:	0x8000	0xF4000	<input checked="" type="radio"/>
<input type="checkbox"/>	IROM2:			<input type="radio"/>

Read/Write Memory Areas

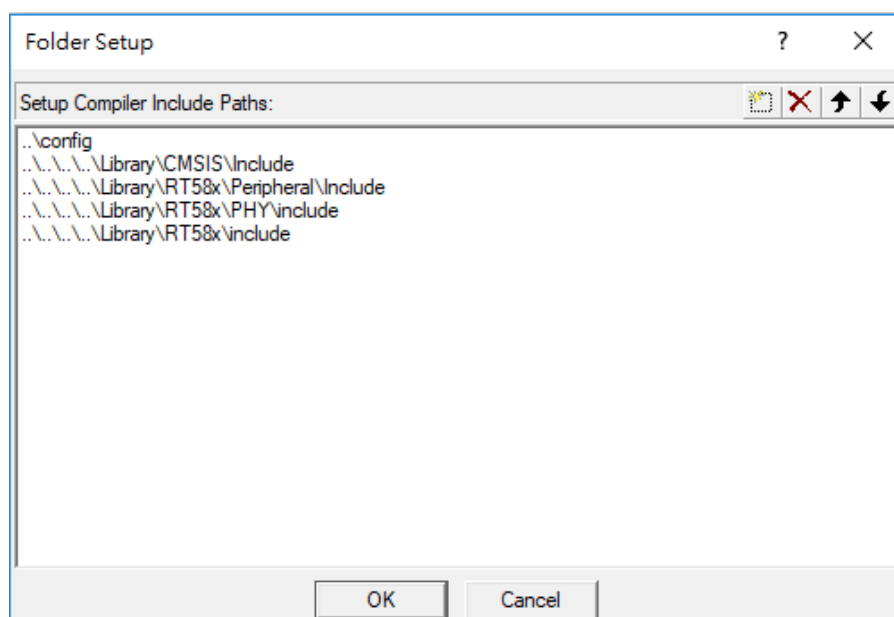
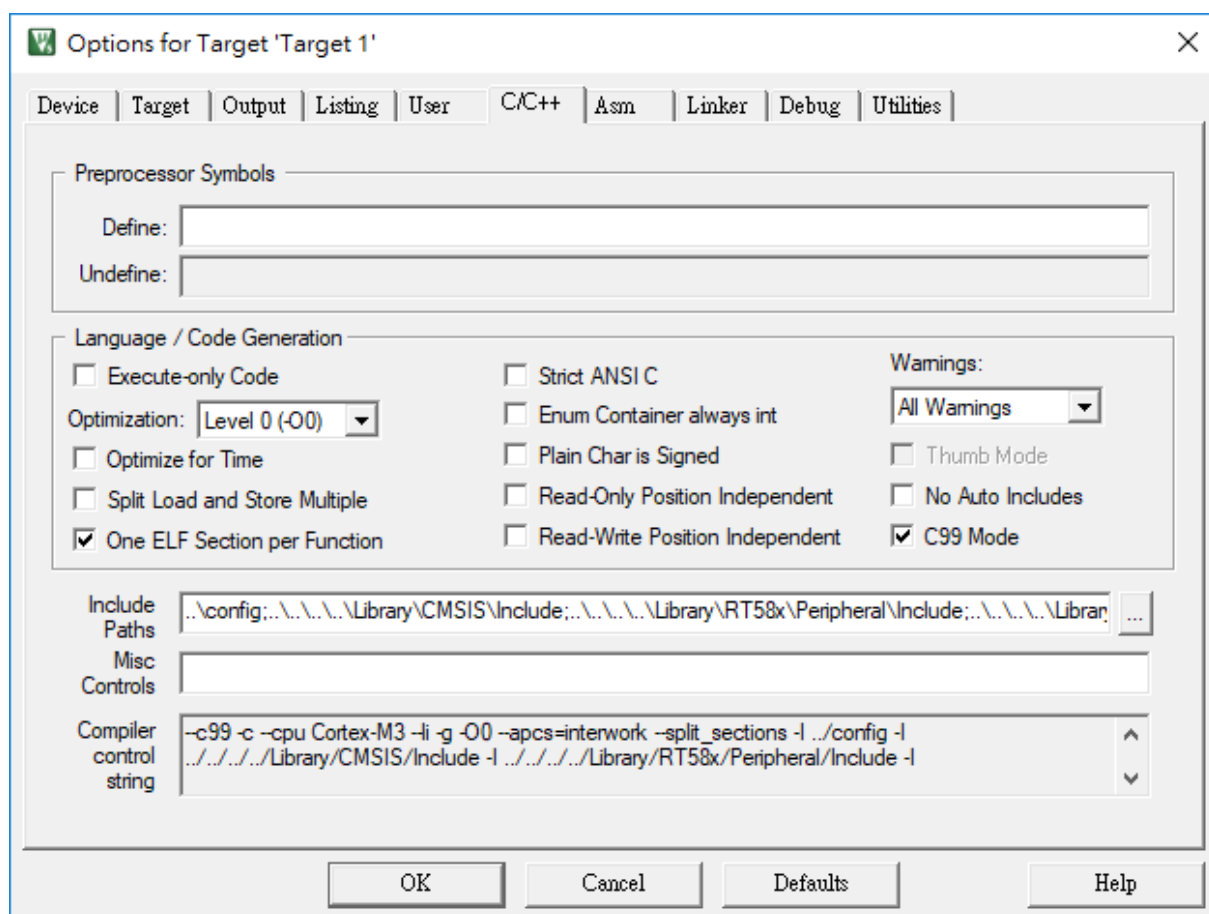
default	off-chip	Start	Size	NoInit
<input type="checkbox"/>	RAM1:			<input type="checkbox"/>
<input type="checkbox"/>	RAM2:			<input type="checkbox"/>
<input type="checkbox"/>	RAM3:			<input type="checkbox"/>
	on-chip			
<input checked="" type="checkbox"/>	IRAM1:	0x20000000	0x20000	<input type="checkbox"/>
<input type="checkbox"/>	IRAM2:			<input type="checkbox"/>

OK Cancel Defaults Help

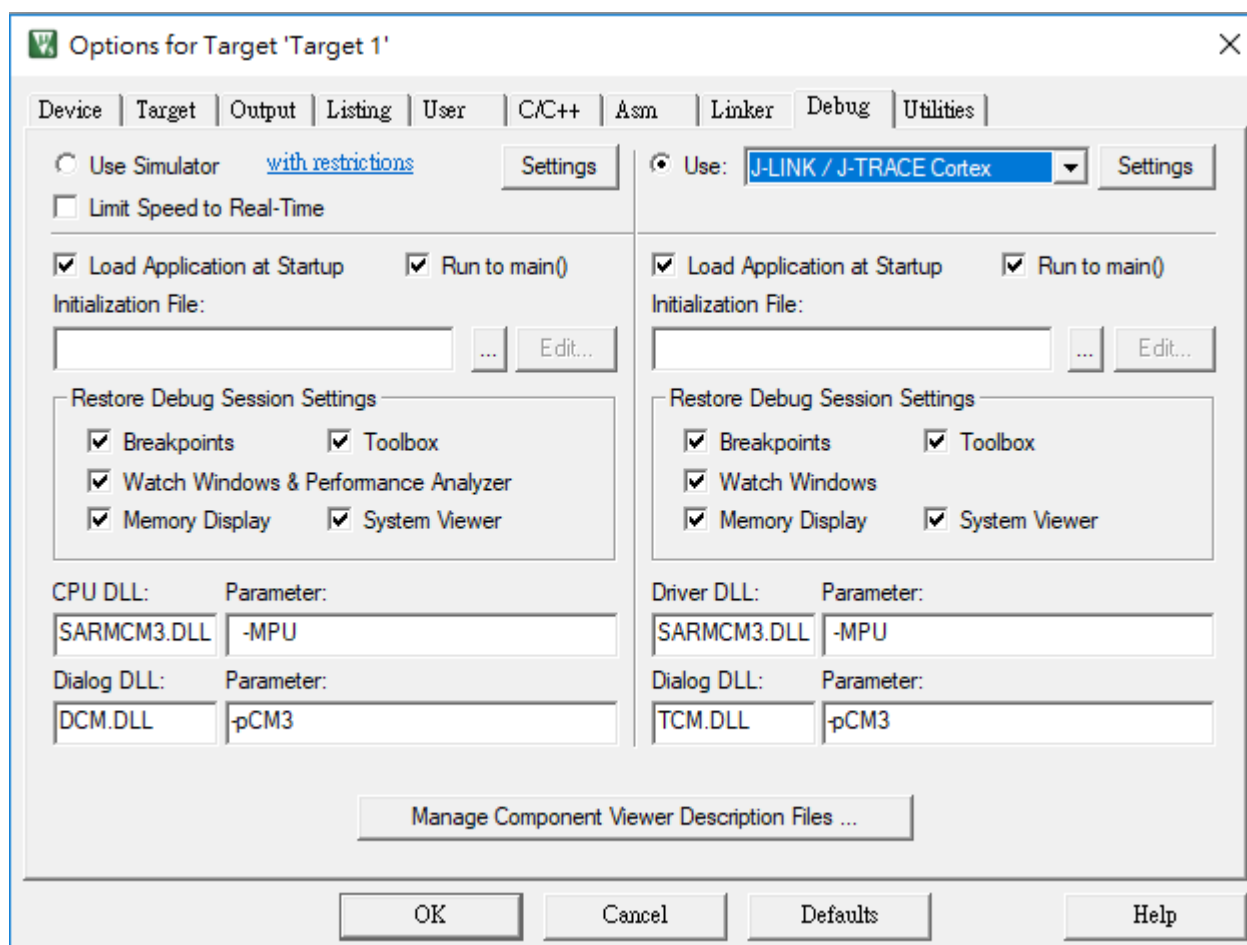
Step 3: Set the Create HEX File. HEX files are used to download the software to programmer memory.



Step 4: Set the Include Paths. Supply one or more paths to search for header files. Add CMSIS file path to Include Paths.



Step5: Set to use debug driver. When J-Link Adapter is available, select J-Link driver for debugging the target hardware.



Click the Settings button to display the dialog Target Driver Setup, and select the port as SW will display IDCODE and Device Name when J-Link Adapter is available.

Cortex JLink/JTrace Target Driver Setup

Debug | Trace | Flash Download

J-Link / J-Trace Adapter

SN: 59410059

Device: J-Link

HW: V9.40 dll: V6.16c

FW: J-Link V9 compiled Jun 2 22:22

Port: SW Max Clock: 5 MHz

Auto Clk

SW Device

IDCODE	Device Name	Move
SWDI 0x2BA01477	ARM CoreSight SW-DP	Up Down

☒ Automatic Detection ID CODE:
☐ Manual Configuration Device Name:
 IR len:

Add Delete Update

Connect & Reset Options

Connect: Normal Reset: Normal

☒ Reset after Connect

Cache Options

☒ Cache Code ☒ Cache Memory

Download Options

☐ Verify Code Download ☐ Download to Flash

Interface

☒ USB ☐ TCP/IP

Scan

State: ready

TCP/IP

Network Settings

IP-Address: 127 . 0 . 0 . 1 Port (Auto: 0): 0

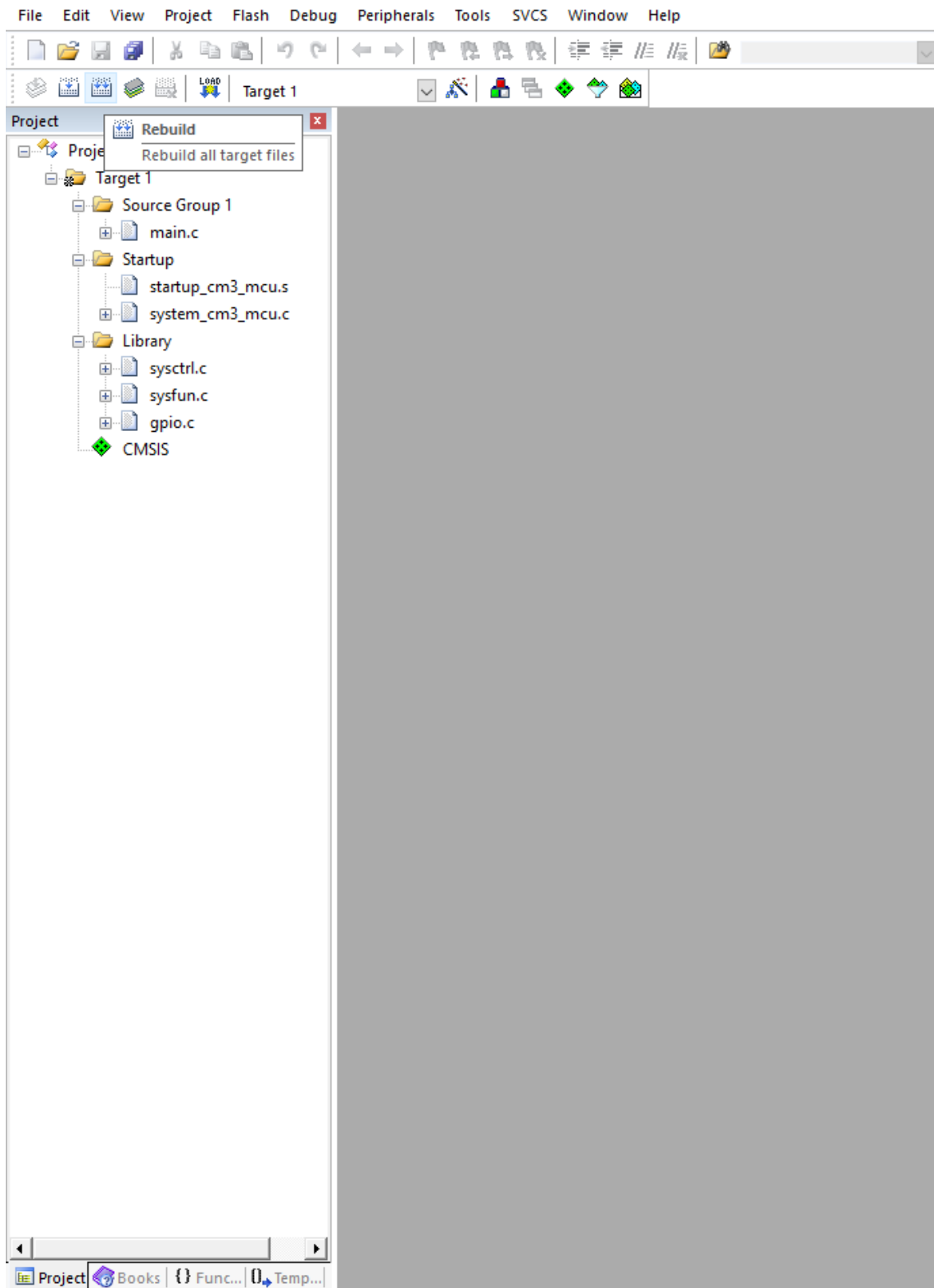
Autodetect Ping

Misc

JLink Info JLink Cmd

確定 取消 套用(A)

Step 6: Build the Project. Click the Build or Rebuild button to translate all source files and link the application, and generates the executable file.



Step 7: The Build Output window will display errors, warnings, and build messages during the build process.

The screenshot shows the Keil uVision IDE interface. The 'Project' window on the left displays the project structure for 'Target 1', including 'Source Group 1' with files 'main.c', 'startup_cm3_mcu.s', and 'system_cm3_mcu.c', and a 'Library' with 'sysctrl.c', 'sysfun.c', and 'gpio.c'. The 'main.c' file is open in the editor, showing C code for GPIO initialization and a toggle loop. The 'Build Output' window at the bottom shows the compilation and linking process, indicating that the build was successful with no errors or warnings.

```

82 int32_t main(void)
83 {
84     init_default_pin_mux();
85
86     gpio_pin_clear(GPIO0);
87     gpio_cfg_output(GPIO0);
88     pin_set_pulldown(GPIO0, PULL_NONE);
89
90     gpio_pin_clear(GPIO1);
91     gpio_cfg_output(GPIO1);
92     pin_set_pulldown(GPIO1, PULL_NONE);
93
94     pin_set_mode(GPIO2, MODE_GPIO);
95     pin_set_pulldown(GPIO2, PULL_UP_100K);
96     gpio_cfg_input(GPIO2, GPIO_PIN_INT_EDGE_FALLING);
97     gpio_register_isr(GPIO2, gpio2_isr, NULL);
98     gpio_int_enable(GPIO2);
99     gpio_set_debounce_time(DEBOUNCE_SLOWCLOCKS_1024);
100    gpio_debounce_enable(GPIO2);
101
102    while (1)
103    {
104        gpio_pin_toggle(GPIO0);
105
106        if (gpio_pin_get(GPIO2) == 1)
107        {
108            gpio_pin_set(GPIO1);
109        }
110        else
111        {
112            gpio_pin_clear(GPIO1);
113        }
114    }
115 }
116 }
117
118

```

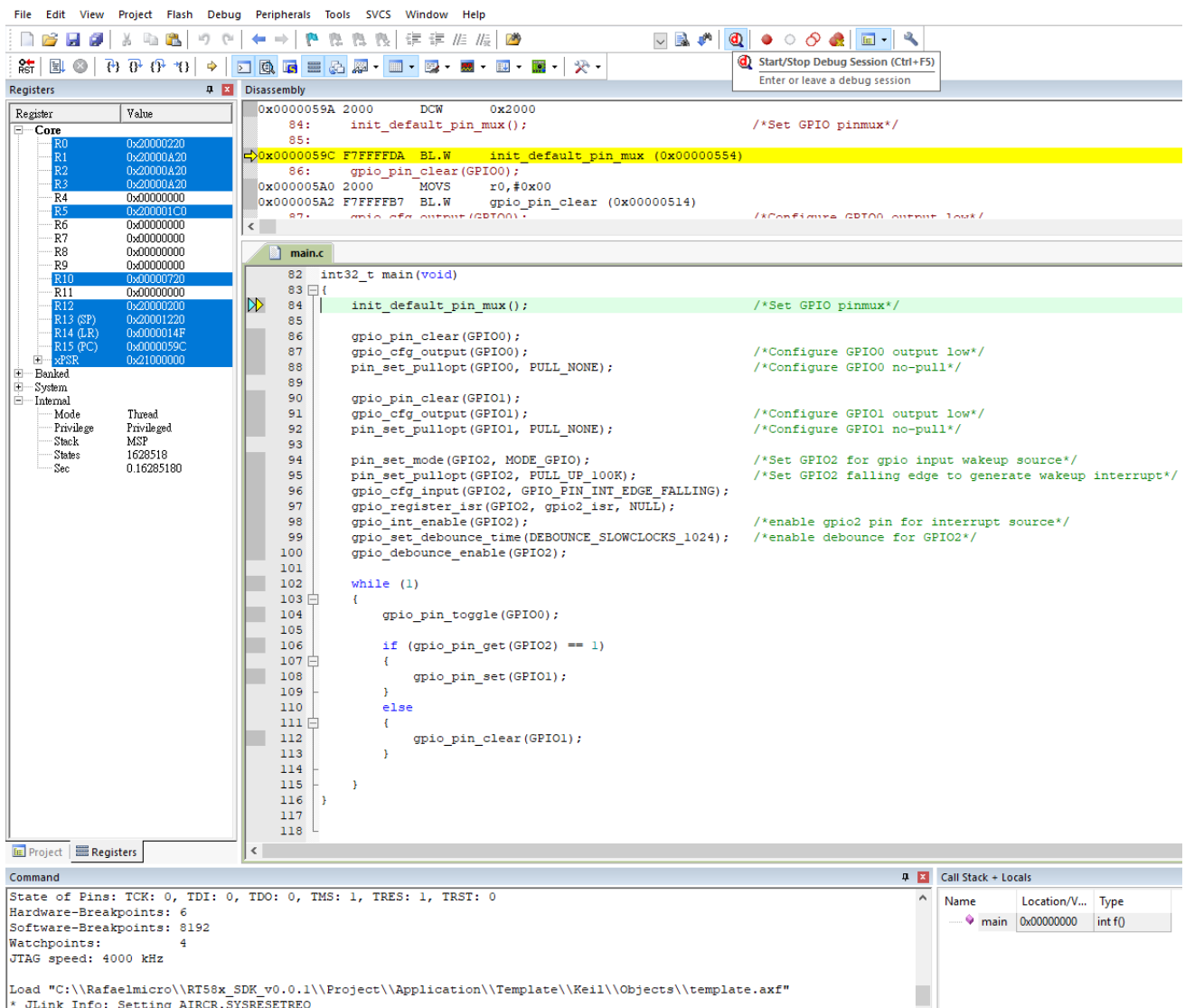
Build Output

```

*** Using Compiler 'V5.06 update 5 (build 528)', folder: 'C:\Keil_v5_ARM\ARM\ARMCC\Bin'
Rebuild target 'Target 1'
assembling startup_cm3_mcu.s...
compiling sysfun.c...
compiling system_cm3_mcu.c...
compiling main.c...
compiling sysctrl.c...
compiling gpio.c...
linking...
Program Size: Code=1596 RO-data=228 RW-data=192 ZI-data=4448
FromELF: creating hex file...
".\Objects\template.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed: 00:00:01

```


Step 8: Start Debugging. Click the Start/Stop Debug Session button to load the application program and executes the startup code, and Debug Menu and Commands are available, such as Reset CPU, Run, Stop, Step, Step Over, Step Out, Breakpoint, etc.



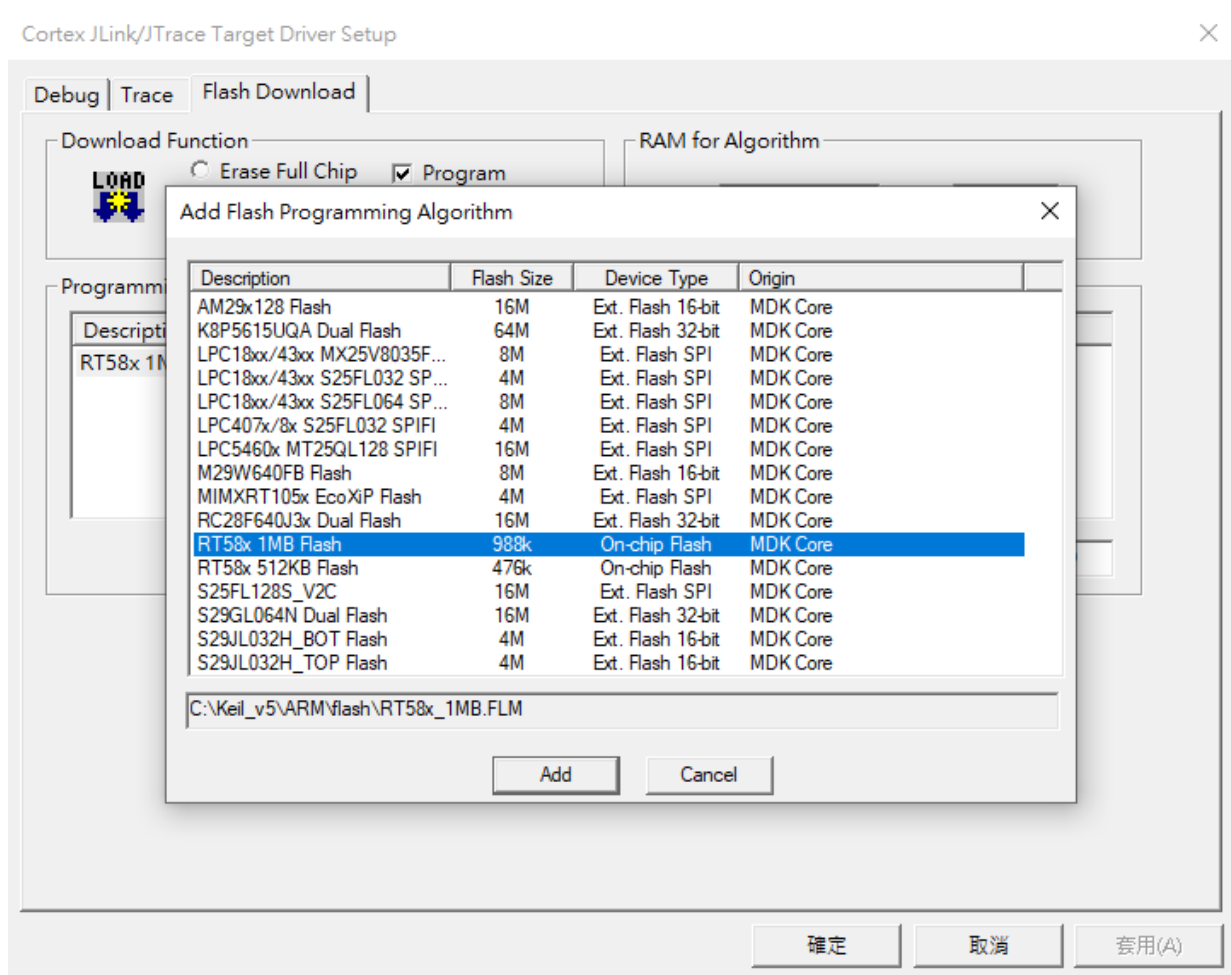
Note: The Reset command in the Debug Menu only resets the Cortex-M microcontroller, not for the RF and peripheral modules in the RT58x.

4.3 Flash Programming

This chapter describes the configuration options for downloading applications to Flash, provides information about Flash programming algorithms.

The following will provide a step-by-step tutorial that configures a debug adapter for Flash programming.

Step 1: Flash Download Configuration. Select the Options for Target → Debug → Settings → Flash Download. Click the Add button in the Programming Algorithm section to include a new device and programming algorithm (*RT58x 1MB Flash/RT58x 512K Flash*) from a list of supplied algorithm.



Note: RT58x Flash programming algorithm (RT58x_1MB.FLM/RT58x_512K.FLM) must be copied to the folder [MDK_Install_Path]\ARM\Flash\ to support RT58x in the list of supplied algorithm.

Windows (C:)	名稱	修改日期	類型	大小
Intel	S29JL032H_TOP.FLM	2021/2/4 下午 04:01	FLM 檔案	12 KB
Keil_v5	S29JL032H_BOT.FLM	2021/2/4 下午 04:01	FLM 檔案	12 KB
ARM	S29GL064Nx2.FLM	2021/2/4 下午 04:02	FLM 檔案	12 KB
ARMCC	S25FL128S_V2C.FLM	2021/2/4 下午 04:01	FLM 檔案	18 KB
ARMCLANG	RT58x_512KB.FLM	2022/5/11 下午 03:13	FLM 檔案	13 KB
BIN	RT58x_1MB.FLM	2022/5/11 下午 03:13	FLM 檔案	13 KB
Flash	RC28F640J3x_x2.FLM	2021/2/4 下午 04:01	FLM 檔案	13 KB
_Template	RC28F320J.FLX	2021/2/4 下午 04:01	FLX 檔案	13 KB
AM29F160DB	MIMXRT105x_ECOXIP_4MB_SEC.FLM	2021/2/4 下午 04:01	FLM 檔案	1,617 KB
AM29F160DT	M29W640F.FLM	2021/2/4 下午 04:01	FLM 檔案	12 KB
AM29F320DB	LPC5460x_MT25QL128.FLM	2021/2/4 下午 04:01	FLM 檔案	591 KB
AM29F320DBx2	LPC407x_8x_S25FL032.FLM	2021/2/4 下午 04:01	FLM 檔案	34 KB
AM29F320DT	LPC18xx43xx_S25FL064.FLM	2021/2/4 下午 04:01	FLM 檔案	77 KB
AM29F320DTx2	LPC18xx43xx_S25FL032.FLM	2021/2/4 下午 04:01	FLM 檔案	77 KB
AM29x033	LPC18xx43xx_MX25V8035F.FLM	2021/2/4 下午 04:01	FLM 檔案	112 KB
AM29x128	K8P5615UQA_x2.FLM	2021/2/4 下午 04:01	FLM 檔案	11 KB
AM29x800BB	FlashOS.h	2021/2/4 下午 04:01	C Header 來源檔案	4 KB
AM29x800BBx2	AM29x800DBx2.FLX	2021/2/4 下午 04:01	FLX 檔案	14 KB
AM29x800BT	AM29x800DB.FLX	2021/2/4 下午 04:01	FLX 檔案	14 KB
AM29x800BTx2	AM29x800BTx2.FLX	2021/2/4 下午 04:01	FLX 檔案	14 KB
K8P5615UQA_x2	AM29x800BT.FLX	2021/2/4 下午 04:01	FLX 檔案	14 KB
LPC18xx43xx_S25FL064	AM29x800BBx2.FLX	2021/2/4 下午 04:01	FLX 檔案	14 KB
LPC407x_8x_S25FL032	AM29x800BB.FLX	2021/2/4 下午 04:01	FLX 檔案	14 KB
LPC5460x_MT25QL128	AM29x128.FLX	2021/2/4 下午 04:02	FLX 檔案	13 KB
M29W640F	AM29x128.FLM	2021/2/4 下午 04:01	FLM 檔案	13 KB
MiMXRT105x_AT91SAM7S256	AM29x033.FLX	2021/2/4 下午 04:01	FLX 檔案	13 KB
RC28F640J3x_x2	AM29F320DTx2.FLX	2021/2/4 下午 04:01	FLX 檔案	14 KB
S25FL128S	AM29F320DT.FLX	2021/2/4 下午 04:01	FLX 檔案	14 KB
S29GL064Nx2	AM29F320DBx2.FLX	2021/2/4 下午 04:01	FLX 檔案	14 KB
S29JL032H	AM29F320DB.FLX	2021/2/4 下午 04:01	FLX 檔案	14 KB
	AM29F160DT.FLX	2021/2/4 下午 04:01	FLX 檔案	14 KB
	AM29F160DB.FLX	2021/2/4 下午 04:01	FLX 檔案	14 KB
	SP29JL032H	2022/4/27 下午 03:51	檔案資料夾	
	S29GL064Nx2	2022/4/27 下午 03:51	檔案資料夾	
	S25FL128S	2022/4/27 下午 03:51	檔案資料夾	
	RC28F640J3x_x2	2022/4/27 下午 03:51	檔案資料夾	

Step 2: Select the Erase Sectors, Program, Verify, Reset and Run options in the Download Function section.

a. RT58x (1MB Flash size) Flash Programming Algorithm select

Cortex JLink/JTrace Target Driver Setup

Debug | Trace | Flash Download

Download Function

LOAD

☐ Erase Full Chip
☒ Erase Sectors
☐ Do not Erase

☒ Program
☒ Verify
☒ Reset and Run

RAM for Algorithm

Start: 0x20000000 Size: 0x1000

Programming Algorithm

Description	Device Size	Device Type	Address Range
RT58x 1MB Flash	988k	On-chip Flash	00007000H - 000FDFFFH

Start: 0x00007000 Size: 0x000F7000

Add Remove

確定 取消 套用(A)

b. RT58x (512K Flash size) Flash Programming Algorithm select

Cortex JLink/JTrace Target Driver Setup

Debug | Trace | Flash Download

Download Function

LOAD

☐ Erase Full Chip
☒ Erase Sectors
☐ Do not Erase

☒ Program
☒ Verify
☒ Reset and Run

RAM for Algorithm

Start: 0x20000000 Size: 0x1000

Programming Algorithm

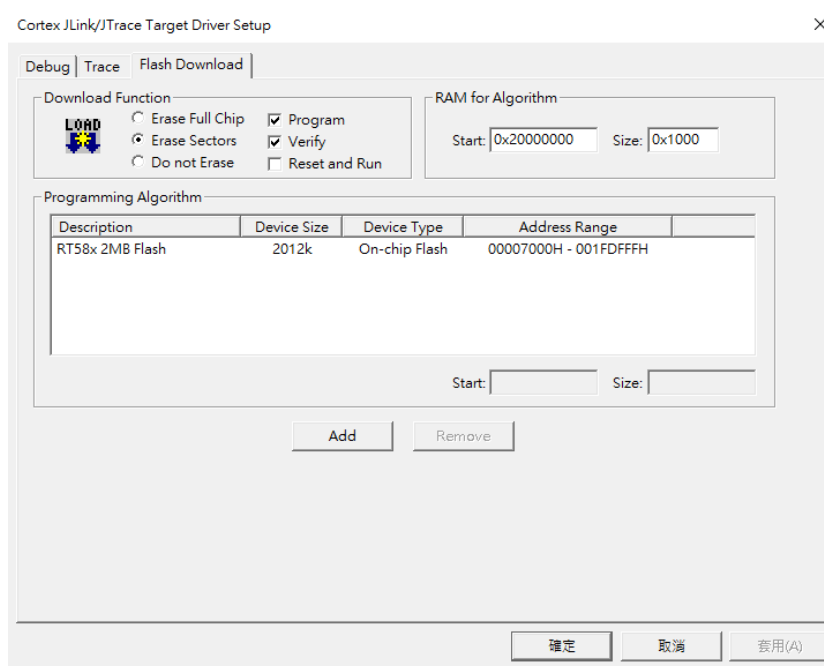
Description	Device Size	Device Type	Address Range
RT58x 512KB Flash	476k	On-chip Flash	00007000H - 0007DFFFH

Start: Size:

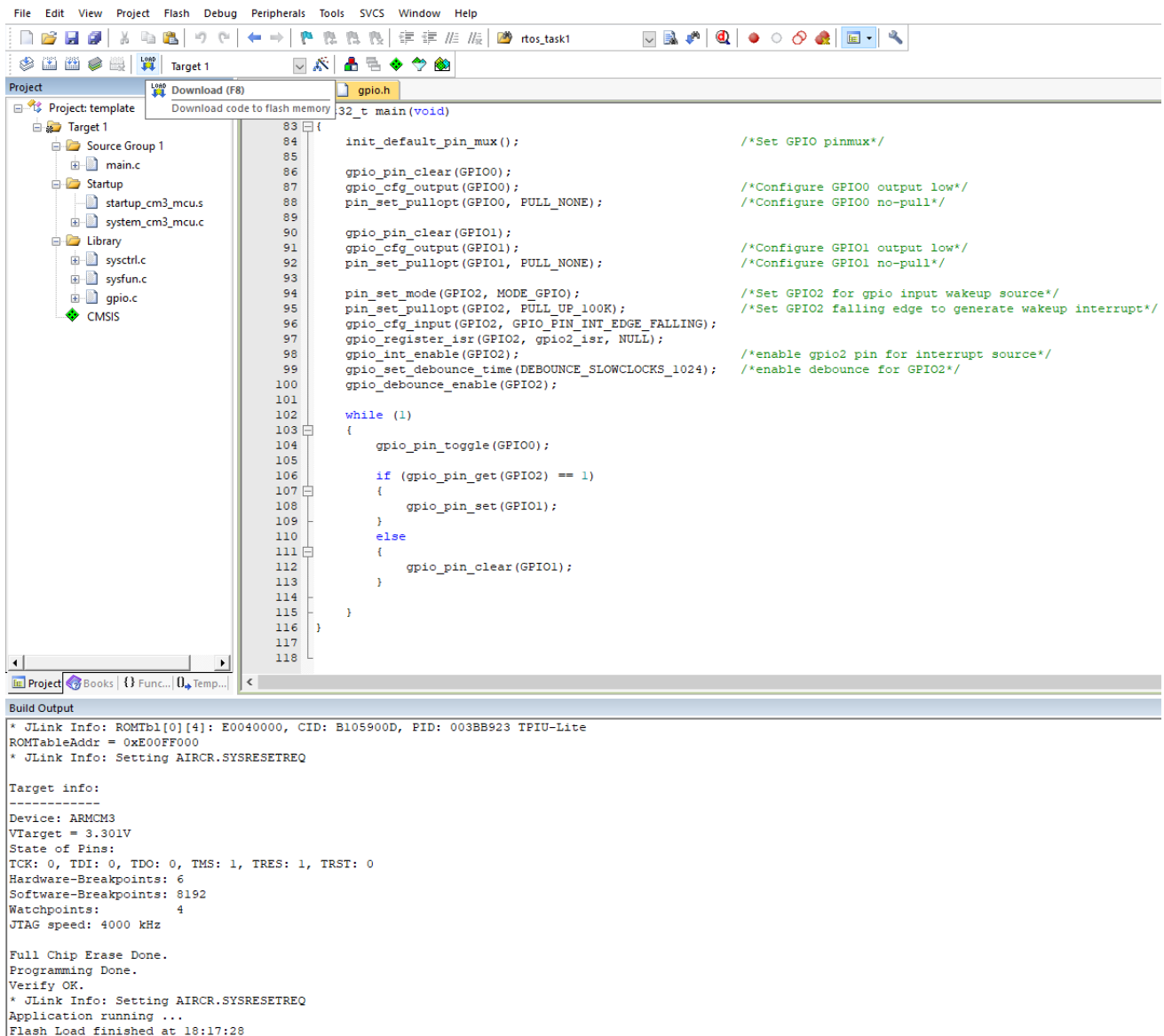
Add Remove

確定 取消 套用(A)

c. RT58x (2MB Flash size) Flash Programming Algorithm select



Step 3: Click the Download button to download code to Flash memory.



The screenshot displays the RT58x IDE interface. The main editor shows the `gpio.h` file with the following code:

```
32_t main(void)
{
    init_default_pin_mux(); /*Set GPIO pinmux*/

    gpio_pin_clear(GPIO0);
    gpio_cfg_output(GPIO0); /*Configure GPIO0 output low*/
    pin_set_pulldown(GPIO0, PULL_NONE); /*Configure GPIO0 no-pull*/

    gpio_pin_clear(GPIO1);
    gpio_cfg_output(GPIO1); /*Configure GPIO1 output low*/
    pin_set_pulldown(GPIO1, PULL_NONE); /*Configure GPIO1 no-pull*/

    pin_set_mode(GPIO2, MODE_GPIO); /*Set GPIO2 for gpio input wakeup source*/
    pin_set_pulldown(GPIO2, PULL_UP_100K); /*Set GPIO2 falling edge to generate wakeup interrupt*/
    gpio_cfg_input(GPIO2, GPIO_PIN_INT_EDGE_FALLING);
    gpio_register_isr(GPIO2, gpio2_isr, NULL);
    gpio_int_enable(GPIO2); /*enable gpio2 pin for interrupt source*/
    gpio_set_debounce_time(DEBOUNCE_SLOWCLOCKS_1024); /*enable debounce for GPIO2*/
    gpio_debounce_enable(GPIO2);

    while (1)
    {
        gpio_pin_toggle(GPIO0);

        if (gpio_pin_get(GPIO2) == 1)
        {
            gpio_pin_set(GPIO1);
        }
        else
        {
            gpio_pin_clear(GPIO1);
        }
    }
}
```

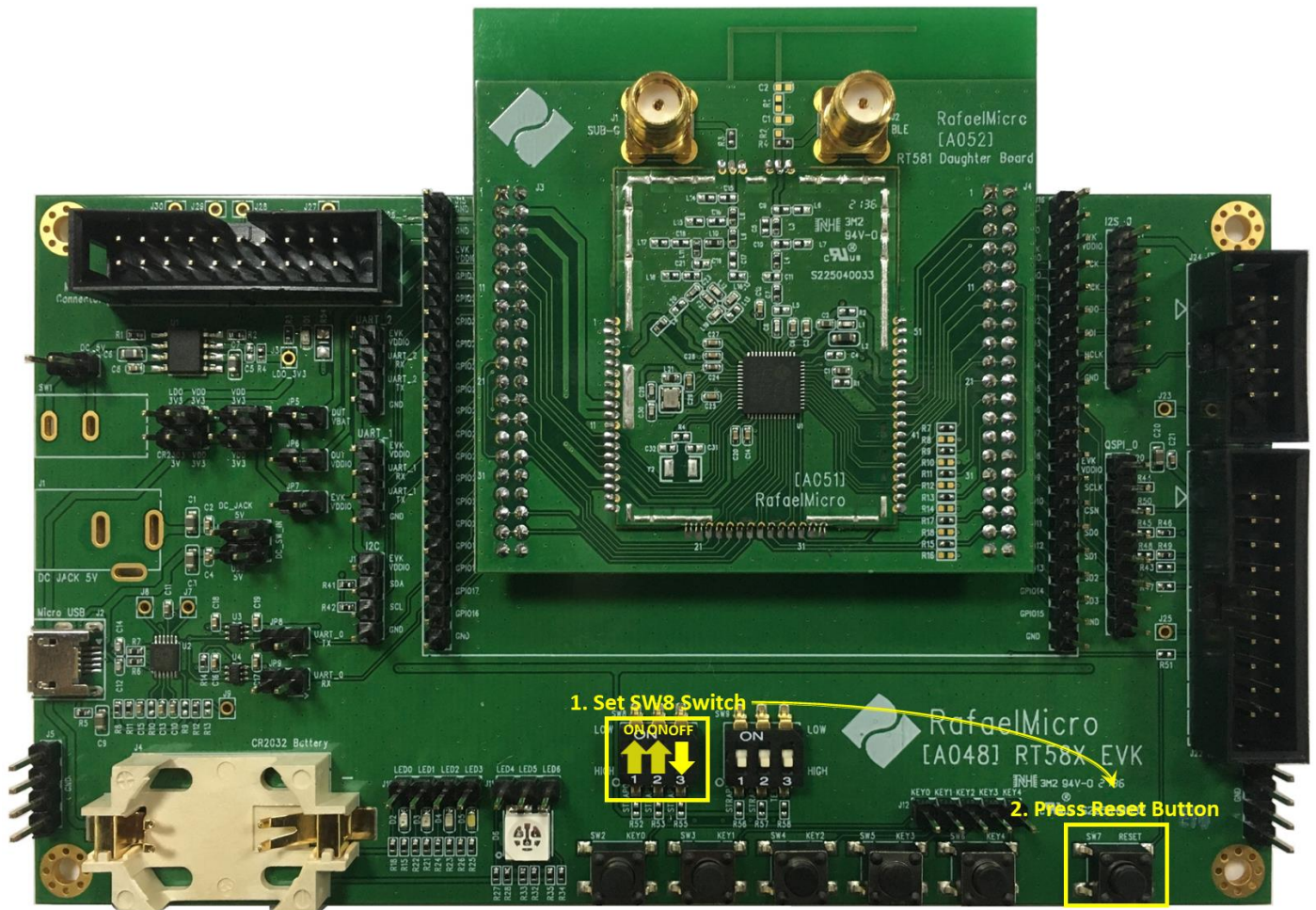
The Build Output window at the bottom shows the following messages:

```
Build Output
* JLink Info: ROMTbl[0][4]: E0040000, CID: B105900D, PID: 003BB923 TPIU-Lite
ROMTableAddr = 0xE00FF000
* JLink Info: Setting AIRCR.SYSRESETREQ

Target info:
-----
Device: ARMCM3
VTarget = 3.301V
State of Pins:
TCK: 0, TDI: 0, IDO: 0, TMS: 1, TRES: 1, TRST: 0
Hardware-Breakpoints: 6
Software-Breakpoints: 8192
Watchpoints: 4
JTAG speed: 4000 kHz

Full Chip Erase Done.
Programming Done.
Verify OK.
* JLink Info: Setting AIRCR.SYSRESETREQ
Application running ...
Flash Load finished at 18:17:28
```

Note: RT58x Flash download will fail when RT58x has entered low power mode (Sleep mode or Deep Sleep mode). User can download code to Flash memory in low power mode according to the following RT58x EVK settings. RT58x EVK settings must be restored after downloading, and reset the RT58x EVK to execute the downloaded code.



Rafael Microelectronics RT58x SOC Platform Getting Started

The information contained herein is the exclusive property of Rafael Microelectronics, Inc. and shall not be distributed, reproduced or disclosed in whole or in part without prior written permission of Rafael Microelectronics, Inc.

4.4 Developing Applications

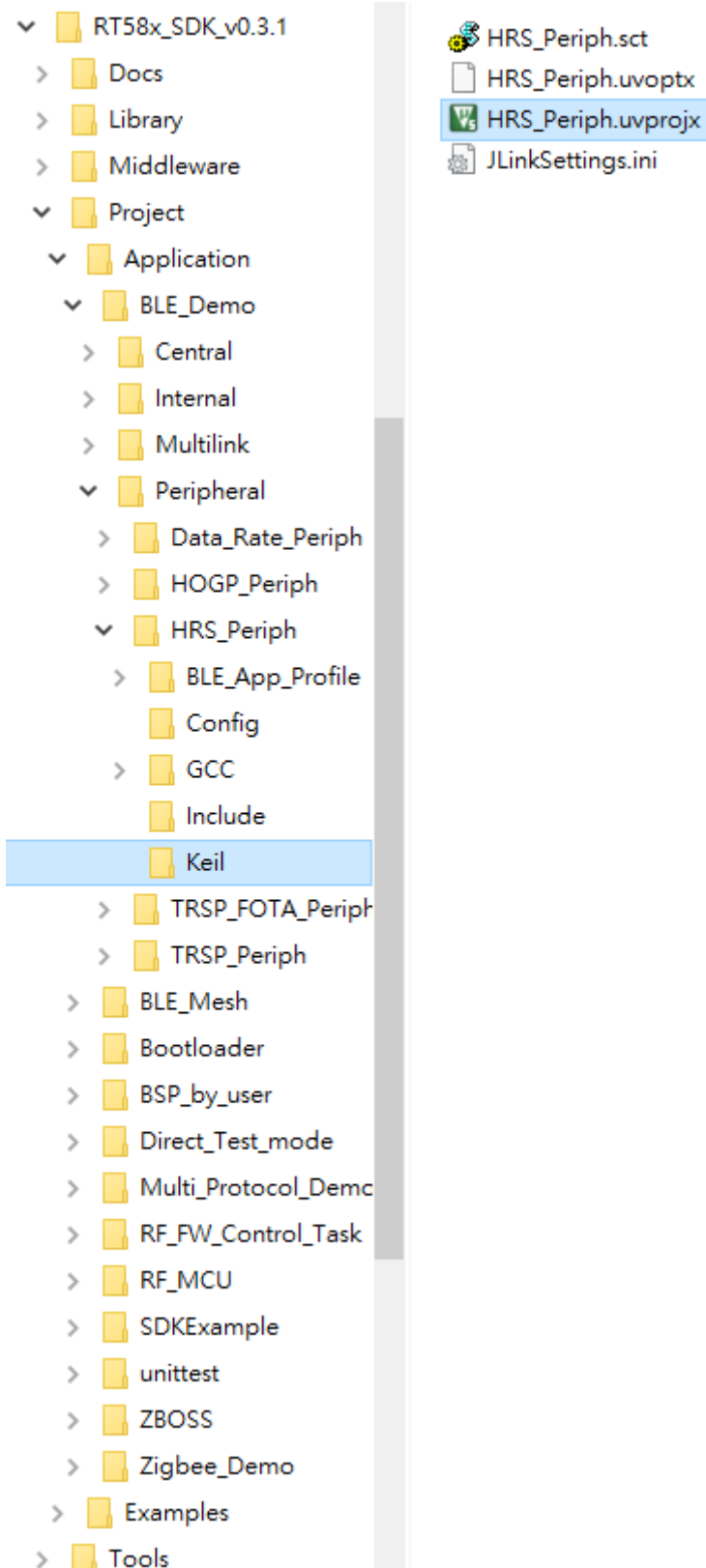
This chapter describes the development of applications using project provided by Rafael as a reference design. In addition to the introduction in the previous Creating Applications chapter, this chapter shows how to easily duplicate existing applications and examples in the Project directory of the RT58x SDK to add and modify related files to the corresponding directory to develop the required applications.

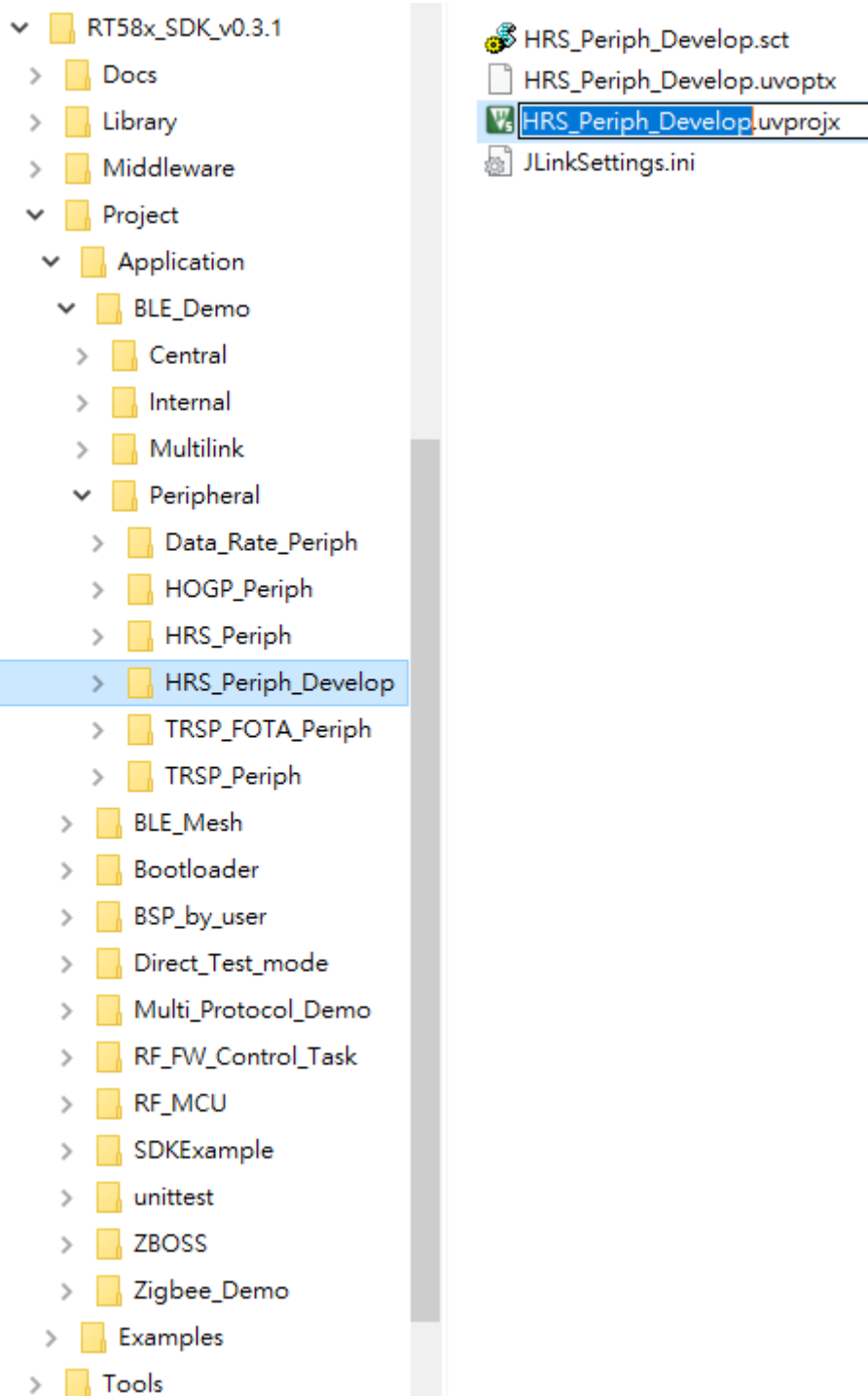
4.4.1 Bootloader

All applications and examples in the Rafael RT58x SDK support OTA feature. Before developing and executing applications and examples, the Bootloader in the \Middleware\Bootloader\ directory of RT58x SDK must be downloaded to RT58x via ISP tool or MP tool to make the applications and examples work properly.

4.4.2 Keil Projects

Rafael RT58x SDK provides rich and complete applications and examples to meet various development requirements. According to the application requirements, select the appropriate project of application or example in the Project directory and open it directly to edit the program (for example, open a HRS_Periph application in the Project directory as shown below), or duplicate it to the corresponding directory and rename the appropriate project name to develop a practical application program (for example, duplicate a HRS_Periph_Develop application from the HRS_Periph application to the same Project directory as shown below).





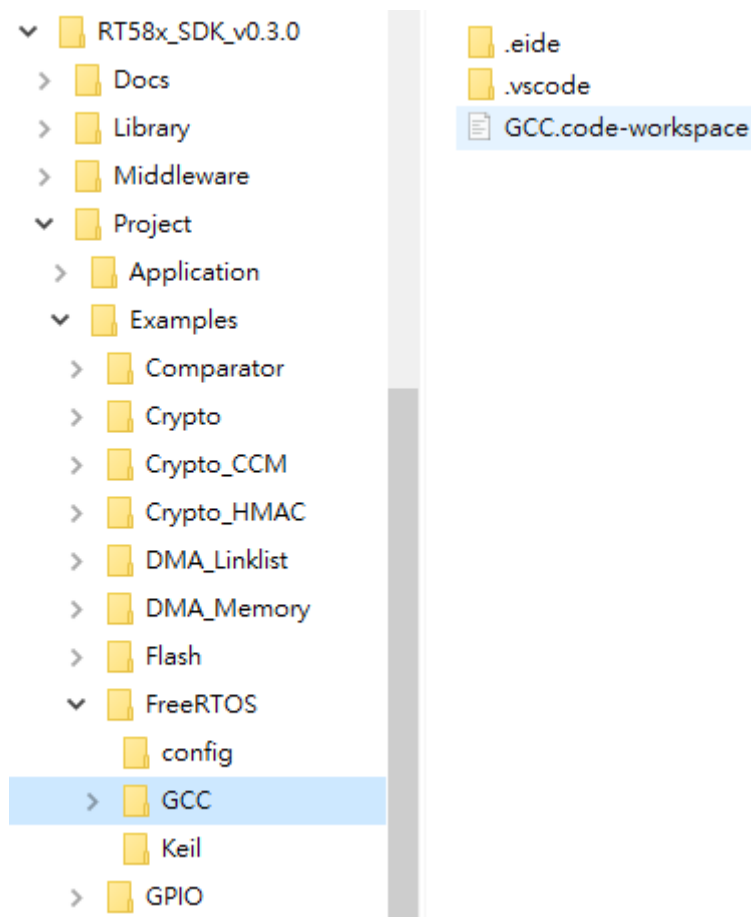
Note: If the path of the application or example in the Project directory has been moved, it may have to modify the Include Paths in the Options for Target window as described in the previous Debugging chapter.

4.4.3 GCC Projects

Rafael RT58x SDK also provides [GCC projects by the Visual Studio Code](#) corresponding to the Keil projects for applications and examples to meet various development requirements.

Before opening the GCC project, users must setup the Visual Studio Code installer for Windows. For detailed installation of Visual Studio Code, please refer to VS Code User Setup Readme.

According to the application requirements, select the appropriate project of application or example in the Project directory and open it directly to edit the program (for example, open a FreeRTOS example in the Project directory as shown below), or duplicate it to the corresponding directory and rename the appropriate project name to develop a practical application program.

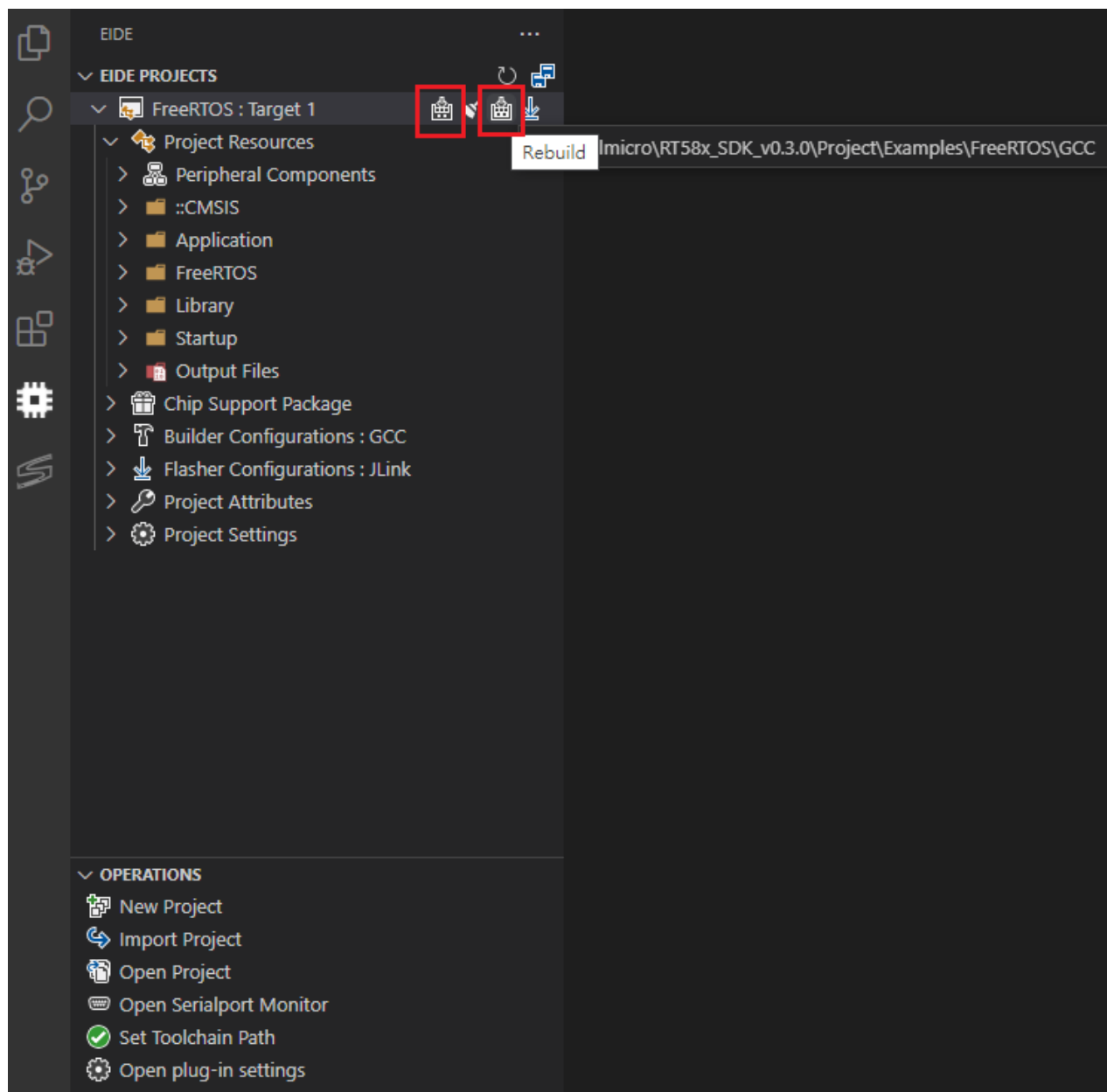


Note: If the path of the application or example in the Project directory has been moved, it may have to modify the Include Paths, please refer to VS Code User Setup Readme.

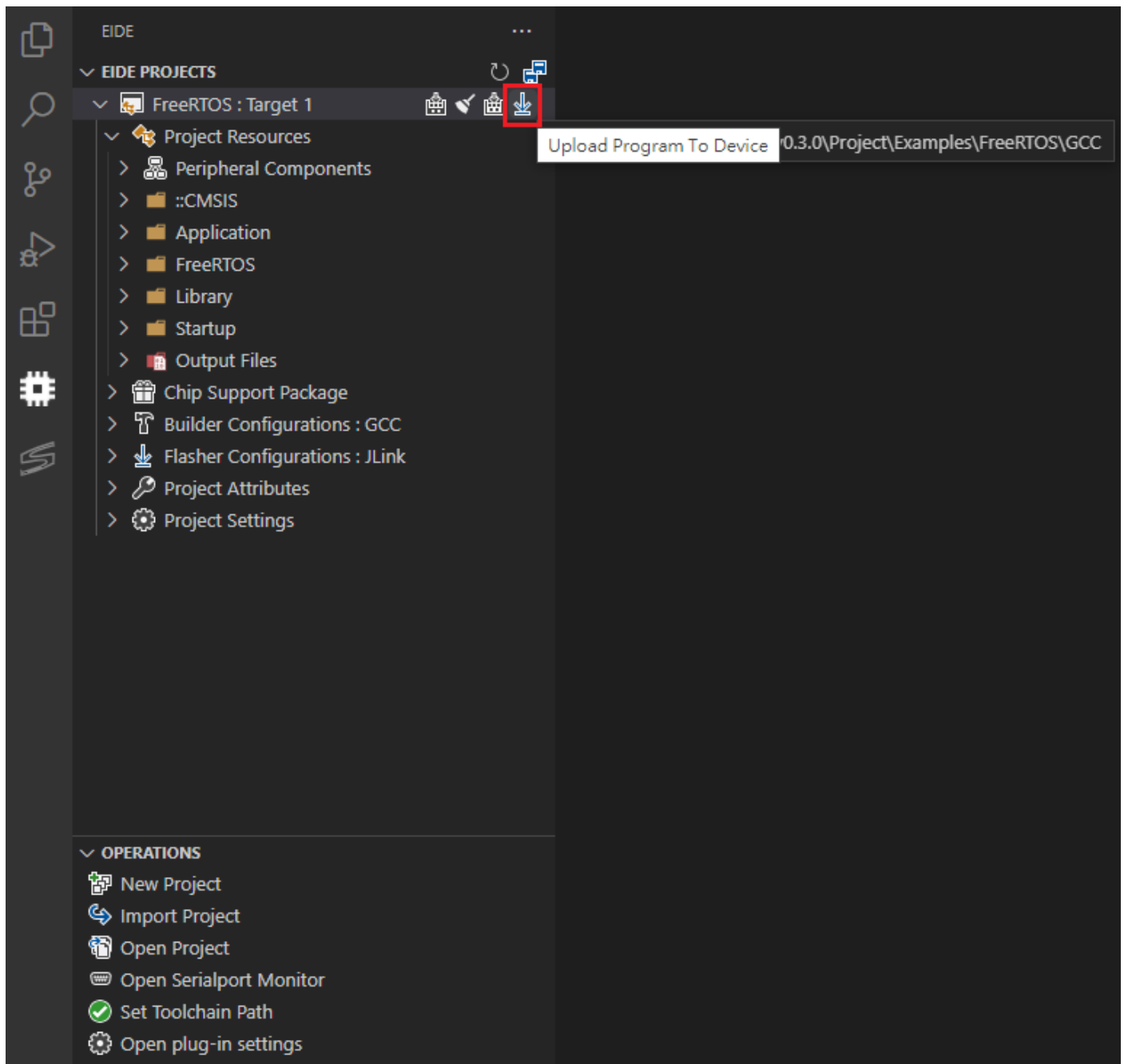
An embedded development environment (EIDE) for Cortex-M microcontroller on VS Code will

provide project development, compilation, download, and ICE debugger functions, the following will provide a step-by-step tutorial for using these functions on EIDE.

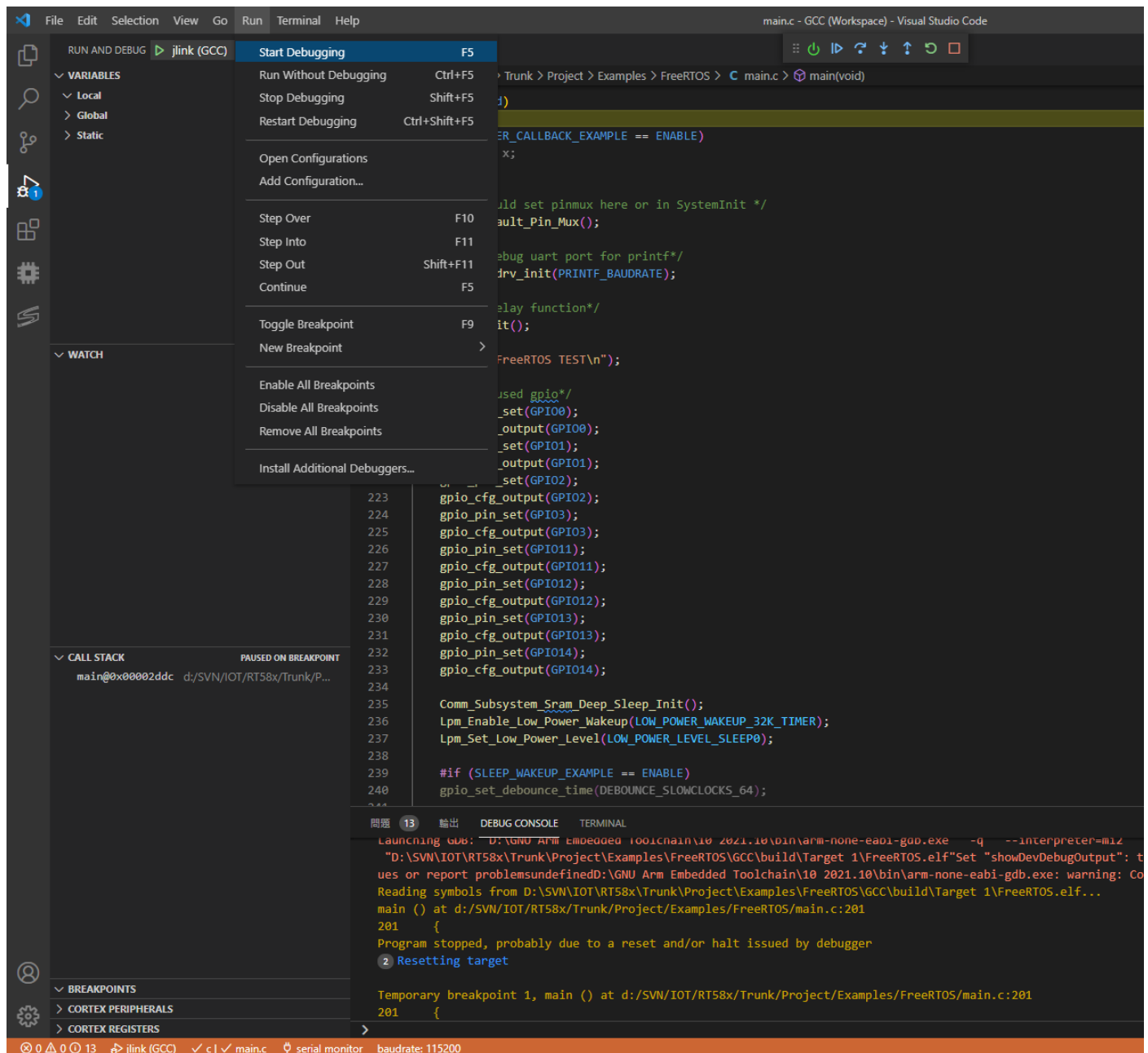
Step 1: Build the Project. Click the Build or Rebuild button to translate all source files and link the application, and generates the executable file.



Step 2: Download the Project. Click the Upload Program To Device button to download code to Flash memory.



Step 3: Start Debugging. Click the Run ➔ Start Debugging (F5) button to executes the startup code and application program, and Debug Menu and Commands are available, such as Reset device, Continue, Step Over, Step Into, Step Out, Restart, Stop, and Breakpoint, etc.



Note: The Reset command in the Debug Menu only resets the Cortex-M microcontroller, not for the RF and peripheral modules in the RT58x.

Revision History

Revision	Description	Owner	Date
1.0	Initial version.	Chiaho Hu	2021/11/05
1.1	1. Added GCC project compilation instructions. 2. Added description of Flash download in low power mode.	Chiaho Hu	2022/02/15
1.2	Revised VS Code setup note	Chiaho Hu	2022/03/28
1.3	Add Keil project select flash program algorithm	Ives Lee	2022/07/15
1.4	Add Keil project select flash program algorithm	Ives Lee	2022/05/02

© 2021 by Rafael Microelectronics, Inc.

All Rights Reserved.

Information in this document is provided in connection with **Rafael Microelectronics, Inc.** ("Rafael Micro") products. These materials are provided by **Rafael Micro** as a service to its customers and may be used for informational purposes only. **Rafael Micro** assumes no responsibility for errors or omissions in these materials. **Rafael Micro** may make changes to this document at any time, without notice. **Rafael Micro** advises all customers to ensure that they have the latest version of this document and to verify, before placing orders, that information being relied on is current and complete. **Rafael Micro** makes no commitment to update the information and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to its specifications and product descriptions.

THESE MATERIALS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, RELATING TO SALE AND/OR USE OF **RAFAEL MICRO** PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, CONSEQUENTIAL OR INCIDENTAL DAMAGES, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. **RAFAEL MICRO** FURTHER DOES NOT WARRANT THE ACCURACY OR COMPLETENESS OF THE INFORMATION, TEXT, GRAPHICS OR OTHER ITEMS CONTAINED WITHIN THESE MATERIALS. **RAFAEL MICRO** SHALL NOT BE LIABLE FOR ANY SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING WITHOUT LIMITATION, LOST REVENUES OR LOST PROFITS, WHICH MAY RESULT FROM THE USE OF THESE MATERIALS.

Rafael Micro products are not intended for use in medical, lifesaving or life sustaining applications. **Rafael Micro** customers using or selling **Rafael Micro** products for use in such applications do so at their own risk and agree to fully indemnify **Rafael Micro** for any damages resulting from such improper use or sale. **Rafael Micro**, logos and **RT568** are **Trademarks** of **Rafael Microelectronics, Inc.** Product names or services listed in this publication are for identification purposes only, and may be trademarks of third parties. Third-party brands and names are the property of their respective owners.