应用手册

基于 EWARM 新建 M0+工程指南

适用范围：

SYM32 系列 Cortex-M0+内核所有 MCU

1 文档说明

本应用手册将详细介绍如何使用 EWARM 新建工程的流程，以 SYM32F030 为例。

2 开发环境准备

2.1 IAR 下载

请从 IAR 官网下载并安装 IAR 软件，建议版本不低于 IAR 8。

2.2 文档准备

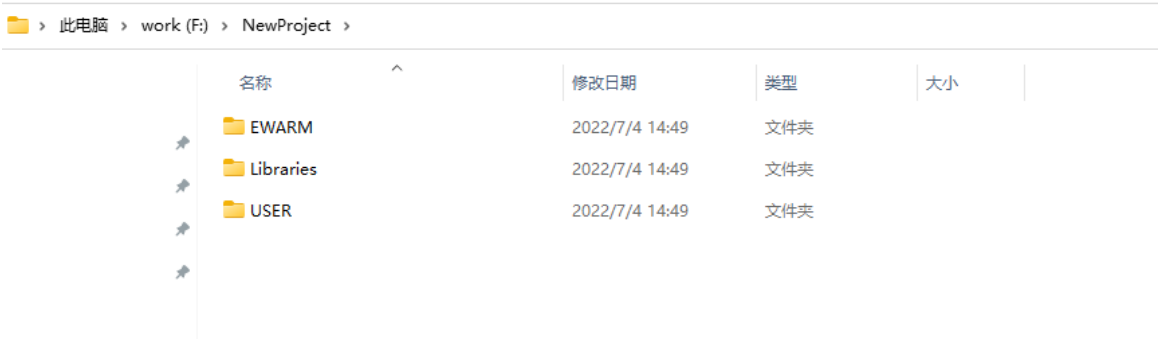
Step1. 例如在 F 盘新建夹 NewProject，用于存放新建 EWARM 工程所有文件

Step2. 将 SYM32xxxx_Firmware_Library\IdeSupport\EWARM 文件夹拷贝到 Step1 新建的 NewProject 文件夹下；

Step3. 将 SYM32xxxx_Firmware_Library\Libraries 文件夹拷贝到 Step1 新建的 NewProject 文件夹下；

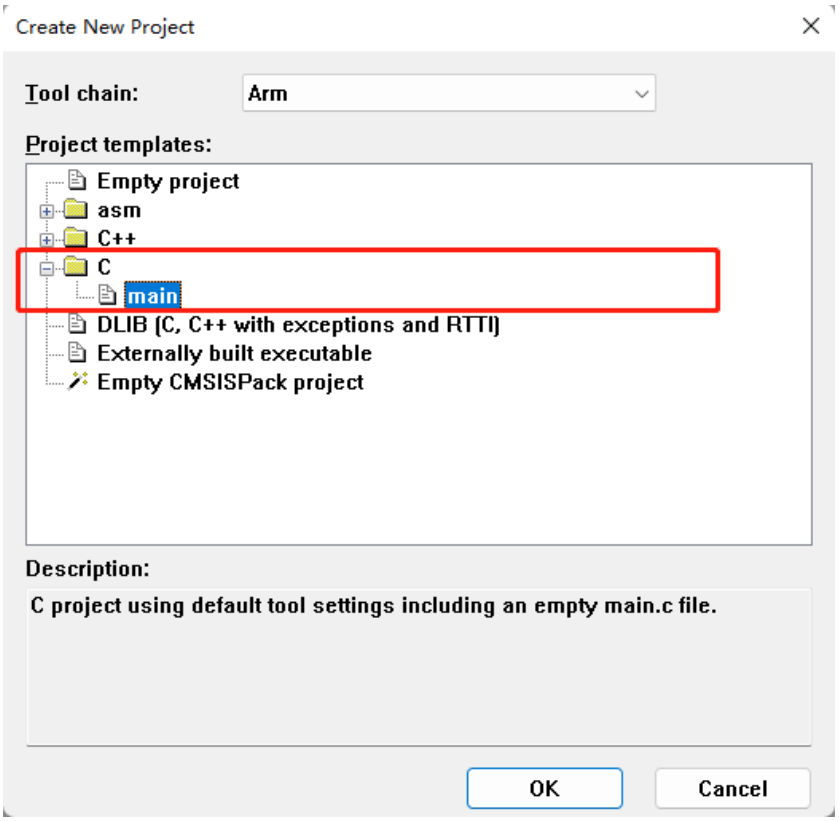
Step4. 将 SYM32xxxx_Firmware_Library\Examples\GPIO\GPIO_Blink\USER 文件夹拷贝到 Step1 新建的 NewProject 文件夹下；

NewProject 的文件目录如下：



3 建立工程

打开 EWARM 软件，点击菜单栏 Project＞Create New Project 弹出如下对话框，选择如下：



点击【OK】，弹出另存为对话框，选择保存路径为 F: \NewProject\EWARM，输入工程名 Sym32Project，点击【保存】。

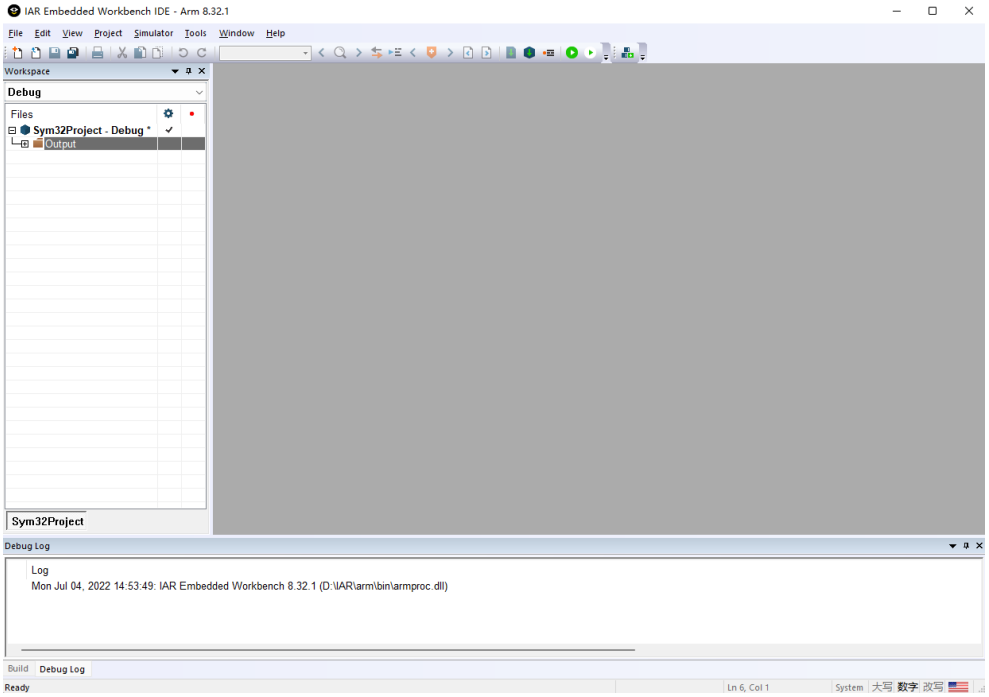
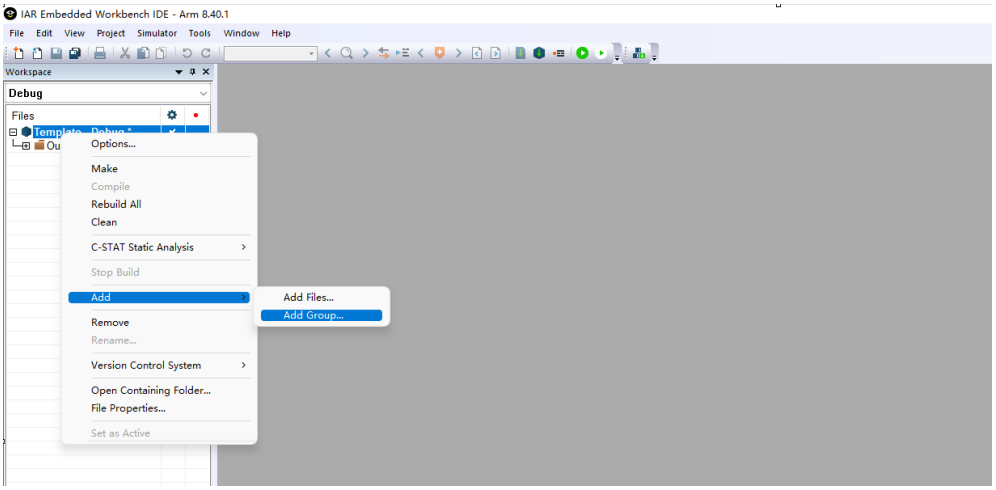
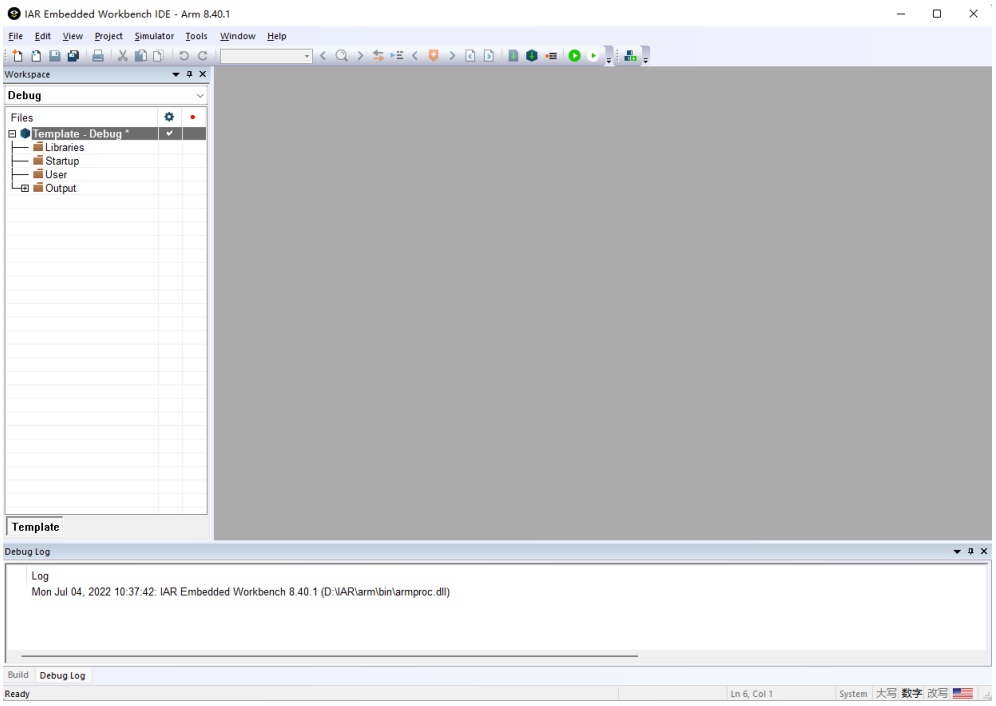


图 1 新建 Sym32Project 工程

这时工程会自动生成一个 main.c 文件，存放在 F: \NewProject\EWARM 文件夹里， 由于 USER 文件夹里的 src 文件夹内已经有 main.c 文件了，所以这里删除自动生成的 main.c 文件，同时移除 EWARM 工程里的 main.c

Step5. 添加文件组，在左侧工程名上右键 ＞Add＞Add Group 依次添加 Librares、Startup、User 三个文件组。





Libraries:用于存放驱动库文件；

Startup：用于存放启动文件；

User：用于存放用户 C 语言文件；

Step6. 给各组添加文件

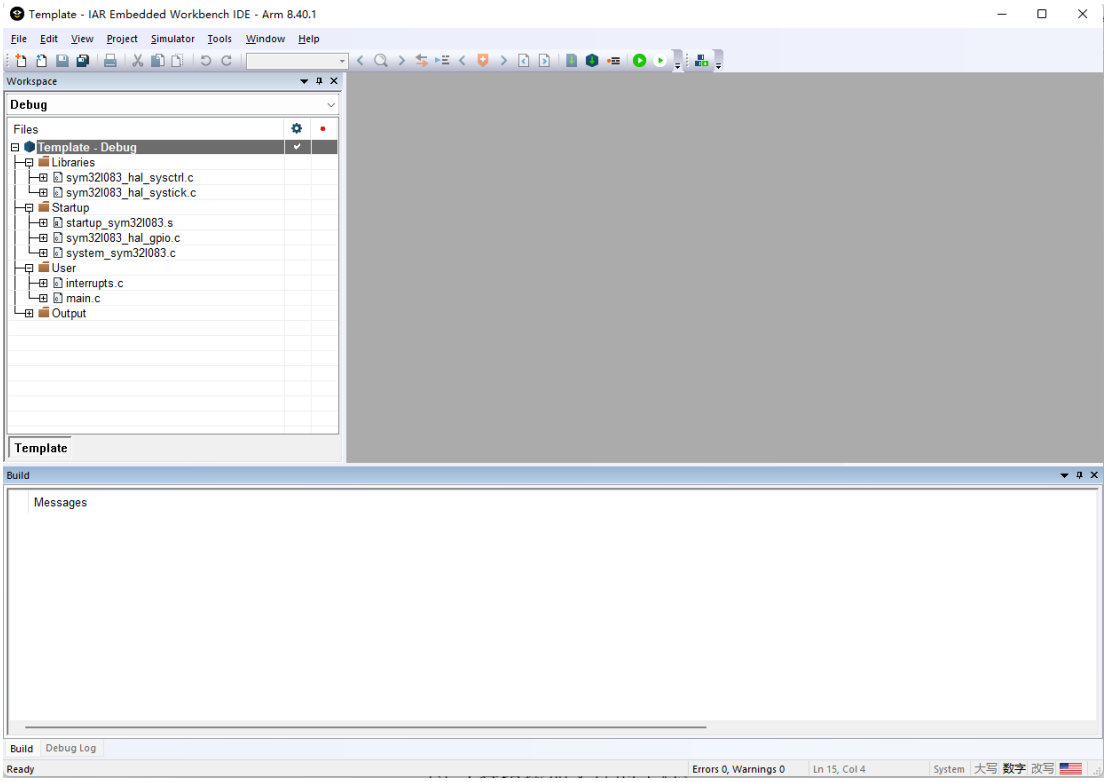
在 Libraries 组名上右键 ＞Add＞Add Files，选择路径 F: \NewProject\Libraries\src，添加需要的库文件，其中 sym32f030_hal_sysctrl.c 和 sym32f030_hal_systick.c 是系统必须添加的驱动文件,此外，该工程用到 GPIO 驱动 LED，所以还需要添加 sym32f030_hal_gpio.c

在 Startup 组名上右键 > Add > Add Files；

添加 F: \NewProject\Libraries\src|system_sym32f030.c 文件；

添加 F: \NewProject \EWARM\startup_sym32f030.s 文件；

在 User 组名上右键 > Add > Add Files,添加 F: \NewProject\USER\src\interrupts.c 文件和 main.c 文件。

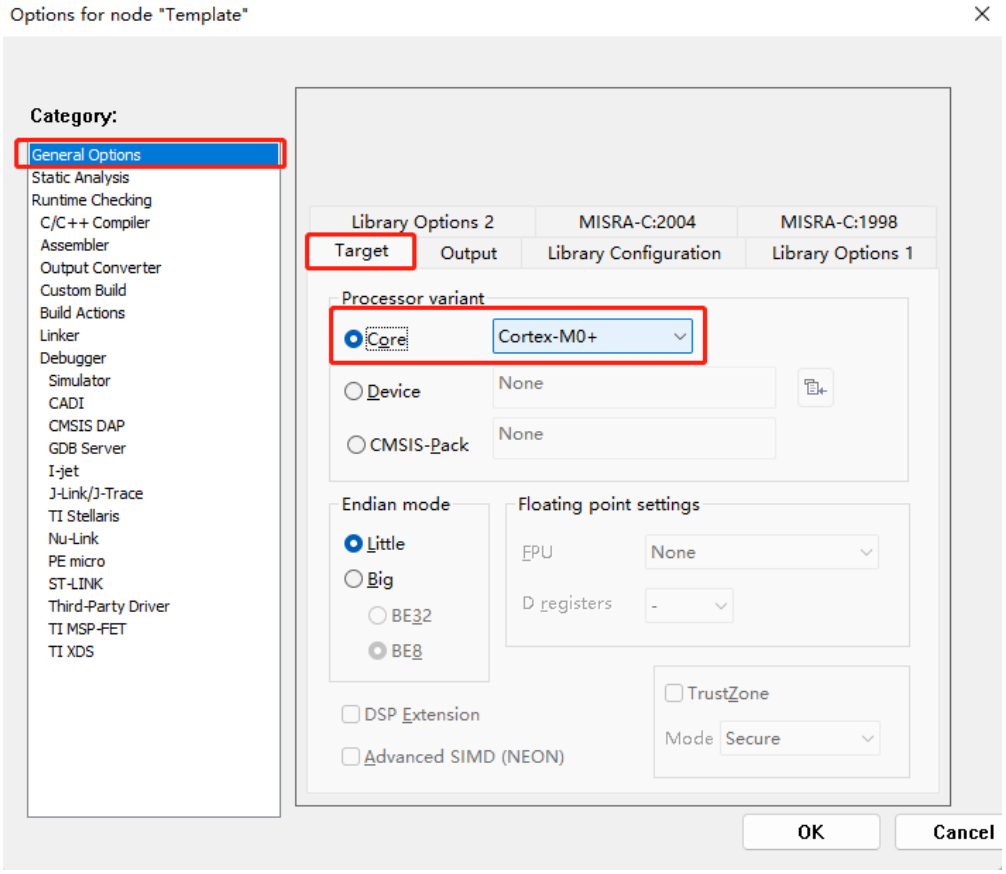


经过上面操作，基础文件已经添加完毕，在工程名上右键 > Rebuild All,编译时弹出对话框 Save Workspace As，输入工程名 Sym32Project，保存路径为 F: \NewProject\EWARM。点击【保存】，但此时仍然不能编译通过，还需要对工程进行设置。

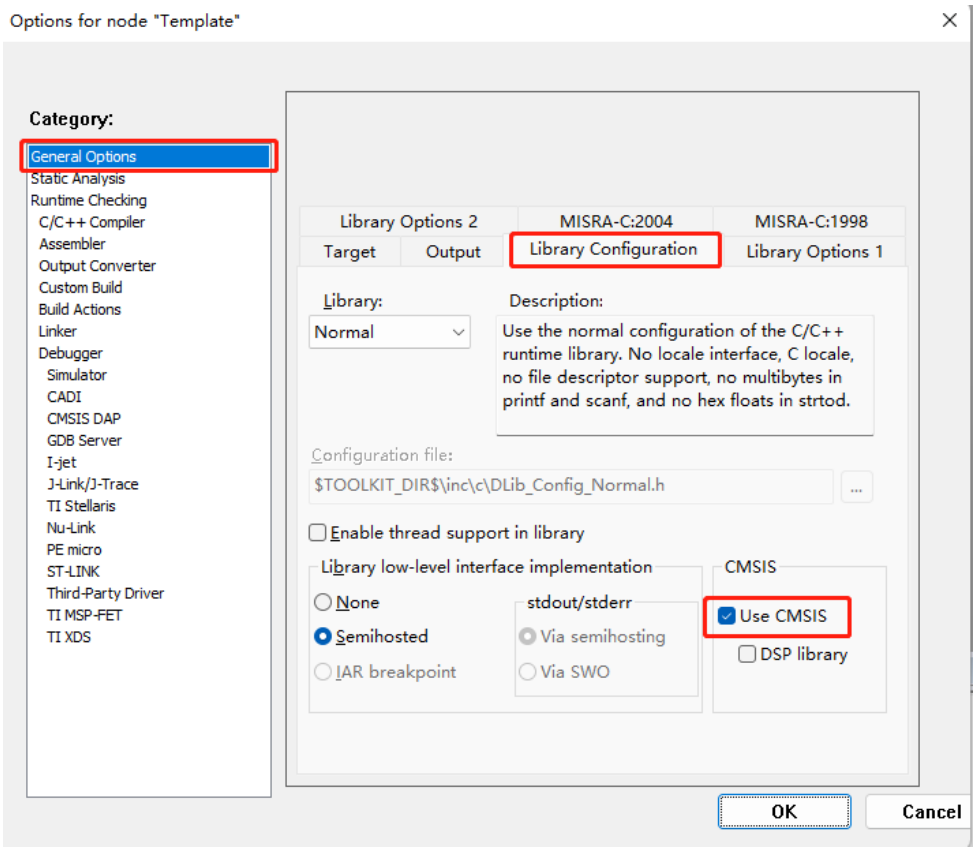
4 工程设置

Step1. 在工程名上右键 > Options > General Options 选项卡下:

在 Target > Processor variant > Core 选项下选择 Cortex-M0+

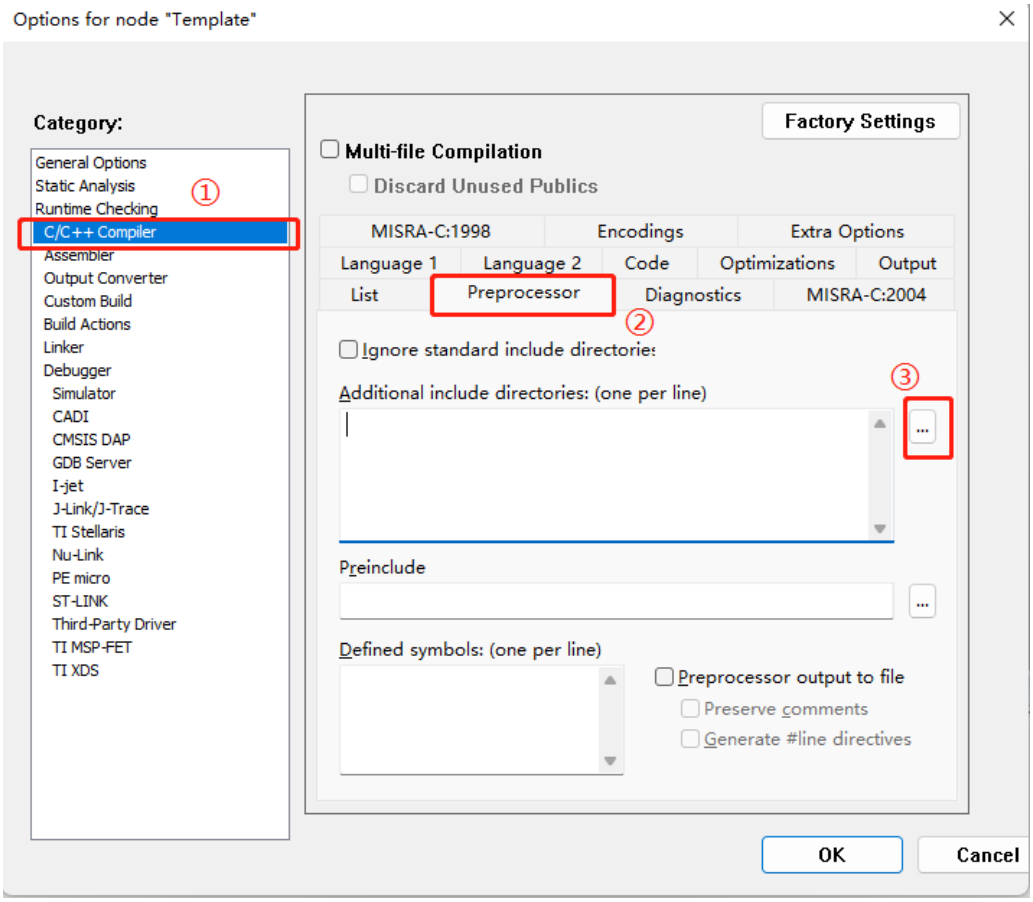


在 Libraries Configuration > CMSIS 选项卡下勾选 Use CMSIS



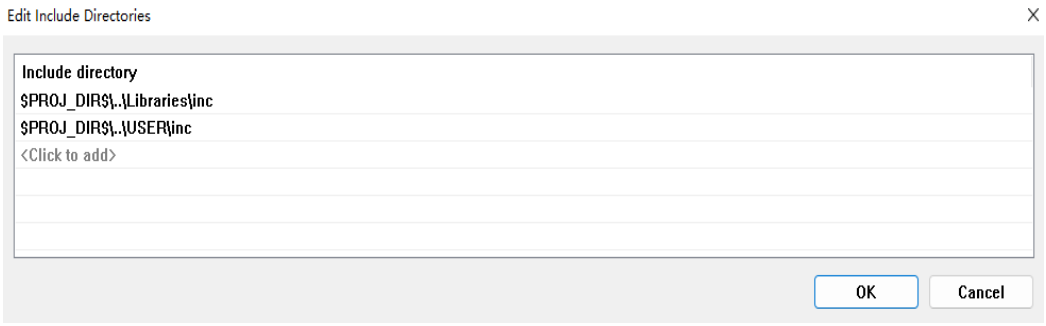
点击【OK】，保存以上设置。

在工程名上右键 > Options > C/C++Compiler > Preprocessor 选项卡下添加头文件路径，按照下图标注顺序进入：

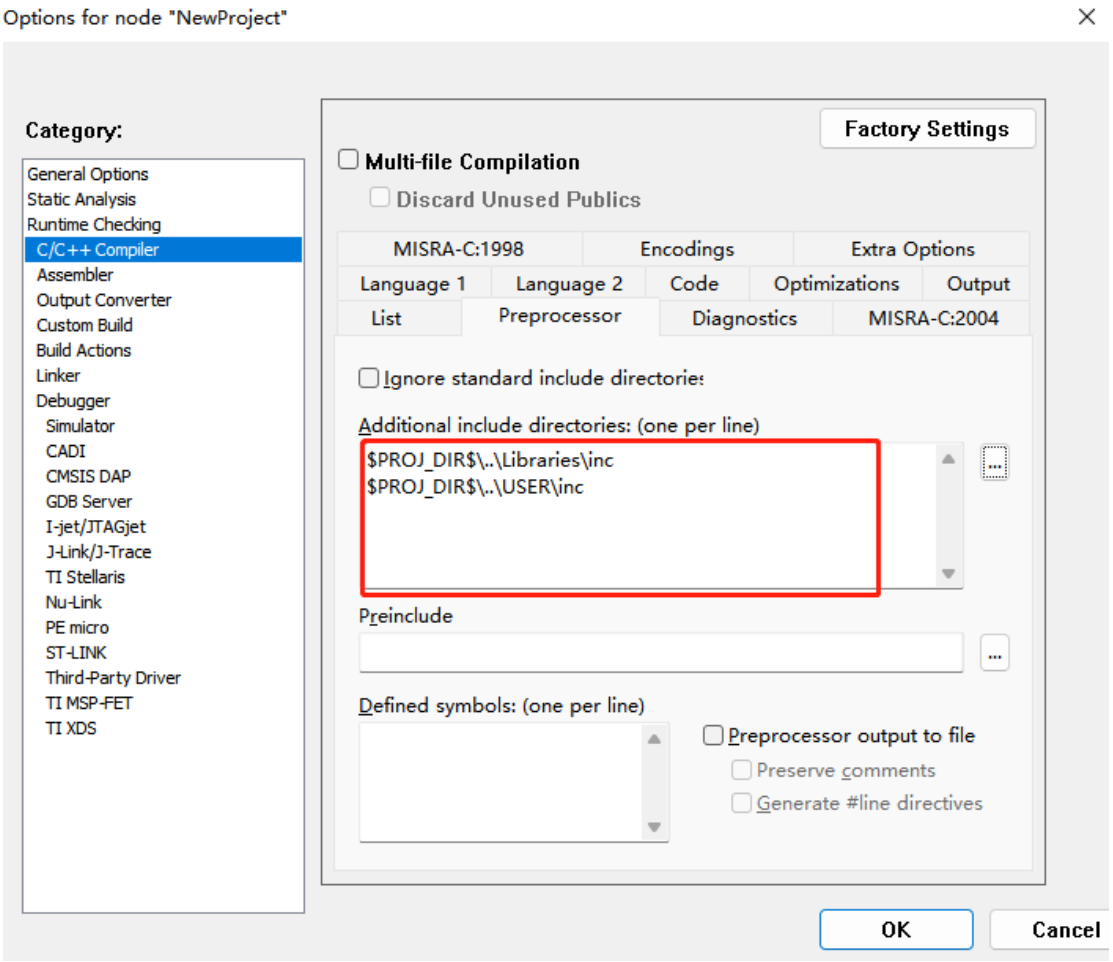


添加驱动库头文件路径 F:\ NewProject\Libraries\inc

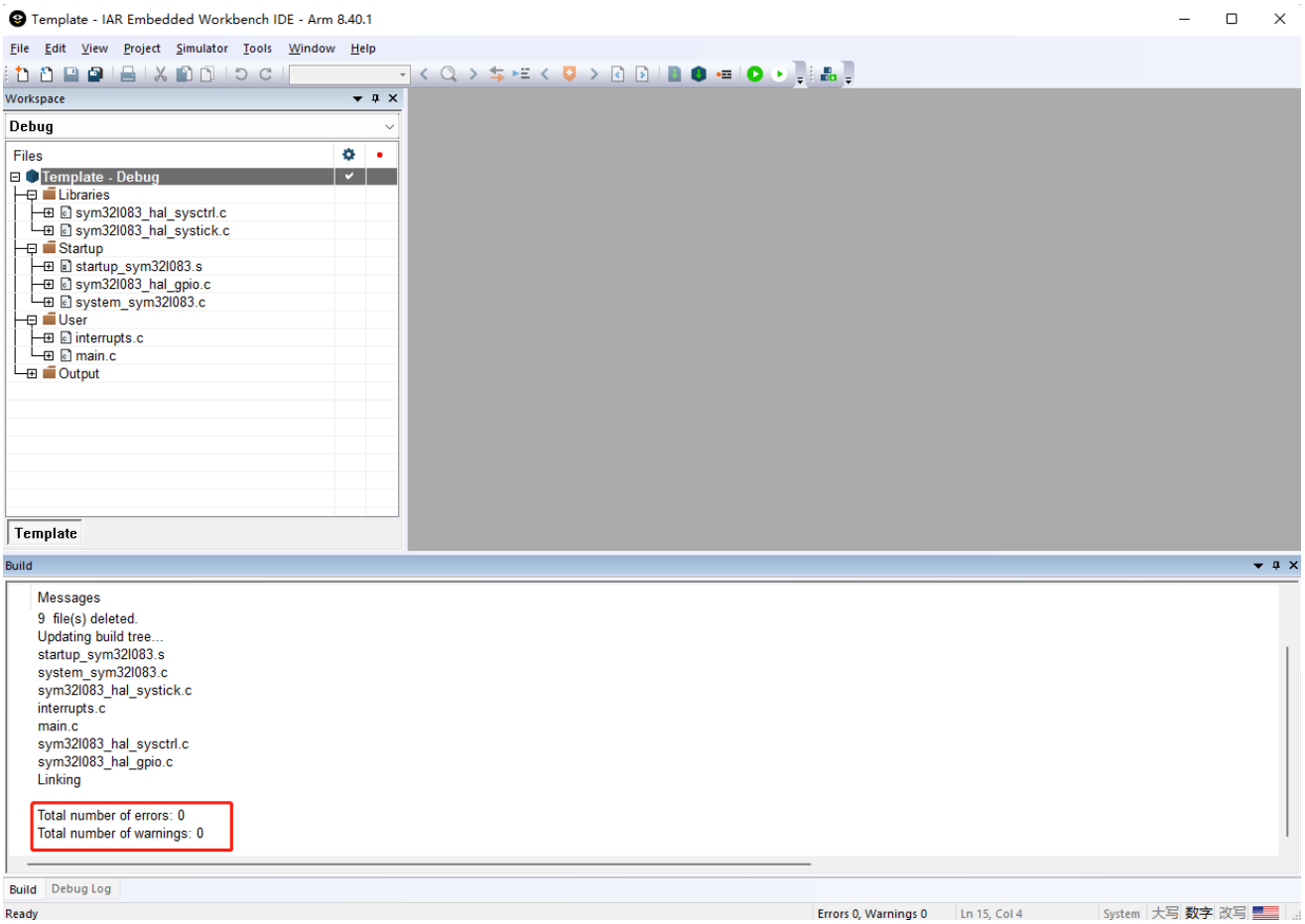
添加用户头文件路径 F:\ NewProject\USER\inc



添加完成后如下图所示：



点击【OK】，保存设置，这时在工程名上右键 ＞Rebuild All，可以编译通过：



5 在线仿真设置

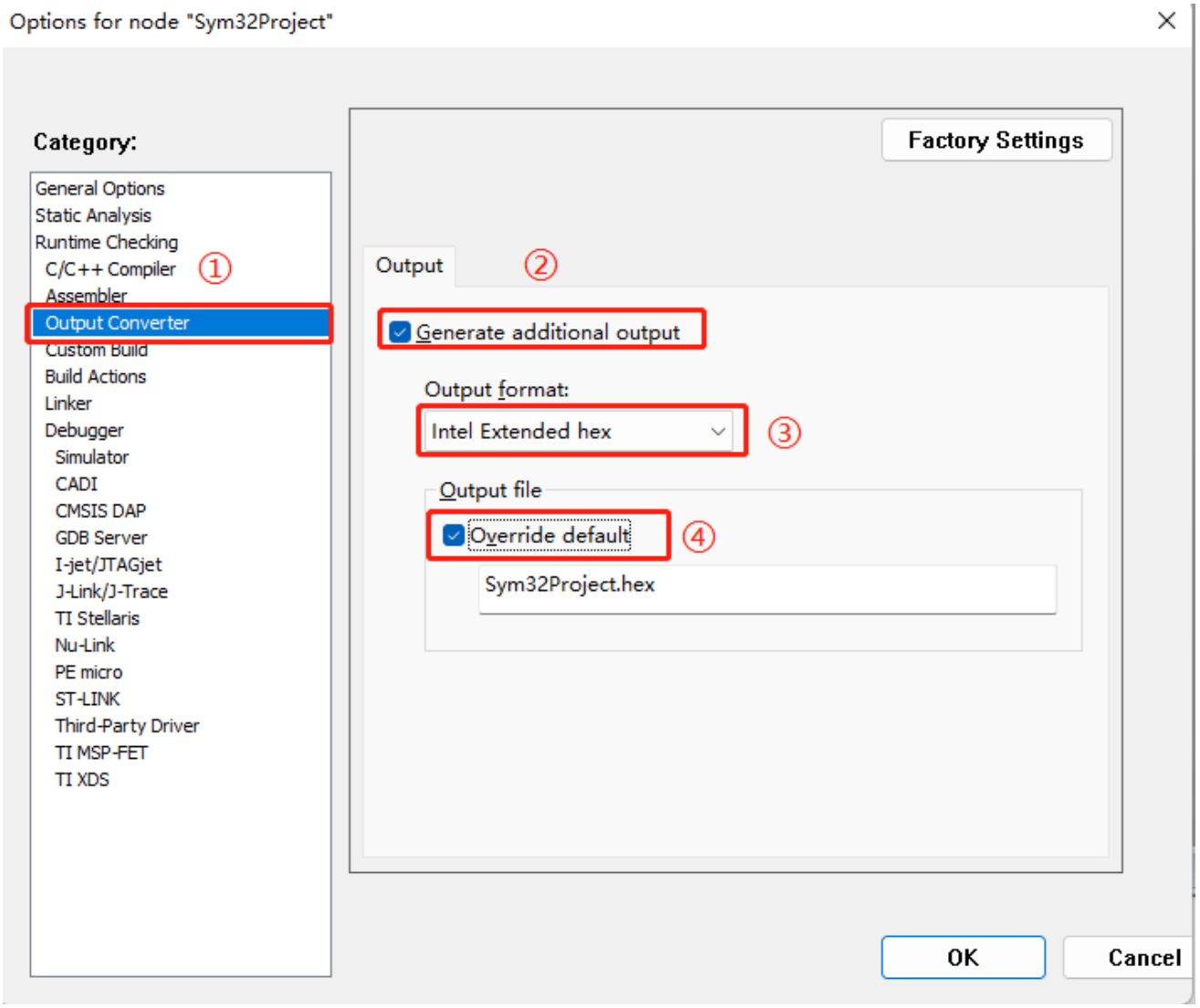
Step1. 在工程名上右键 > Options > Output Converter > Output 选项卡下:

勾选 Gennerate additional output.

Output format 选项选择 intel Extended hex.

Output file 选项勾选 Override default.

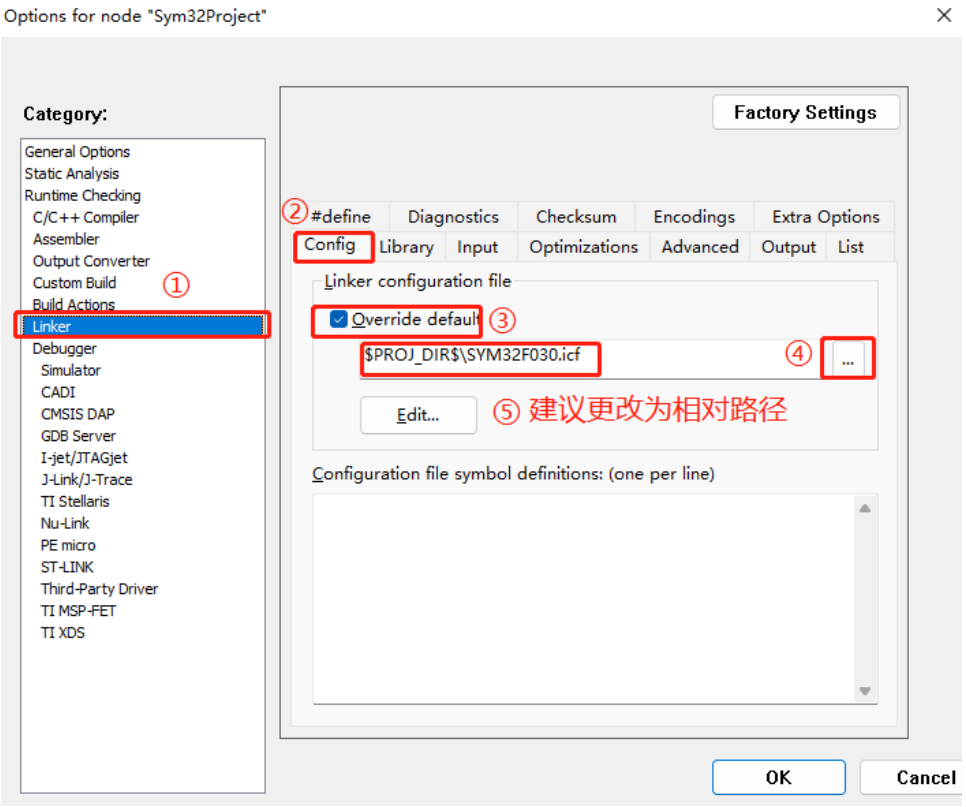
设置之后，在 EWARM 编译后可生成.hex 执行文件。点击【OK】后保存生效。



Step2. 在工程名上右键 > Options > Linker > Config 选项卡下:

勾选 Override default.

下图中④路径选择 F:\NewProject\EWARM\SYM32F030.icf. 建议把绝对路径更改为相对路径，以免工程文件移动时需重新配置路径。把路径更改为$PROJ_DIR$\SYM32F030.icf. 如下图所示，点击【OK】后保存生效。



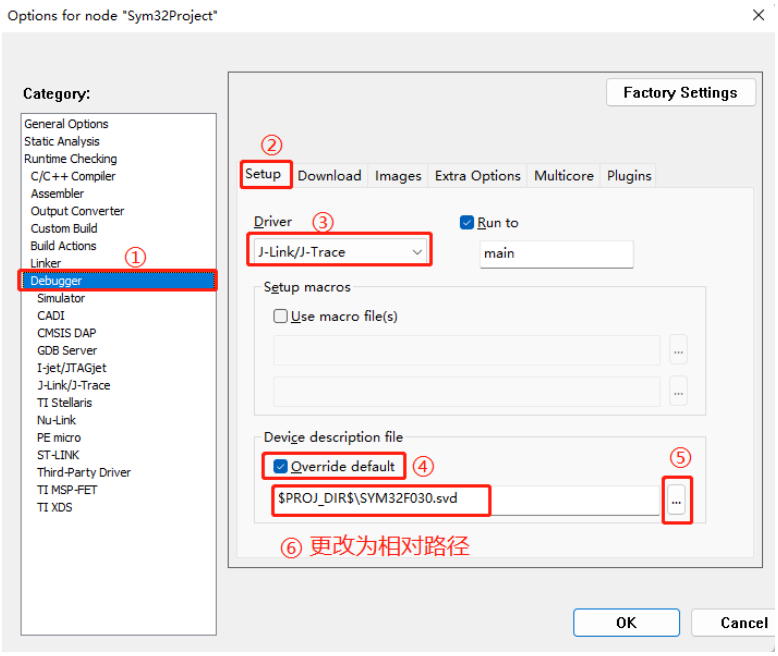
Step3. 在工程名上右键 > Options > Debugger > Setup 选项卡下：

Driver 根据实际使用的仿真器选择，例如 J-Link，则选择 J-Link/J-Trace.

勾选 Override default.

下图中⑤路径选择 F:\NewProject\EWARM\SYM32F030_PV.svd.

建议把绝对路径更改为相对路径，以免工程文件移动时需重新配置路径。把路径更改为 $PROJ_DIR$\SYM32F030_PV.svd.如下图所以，点击【OK】后保存生效。



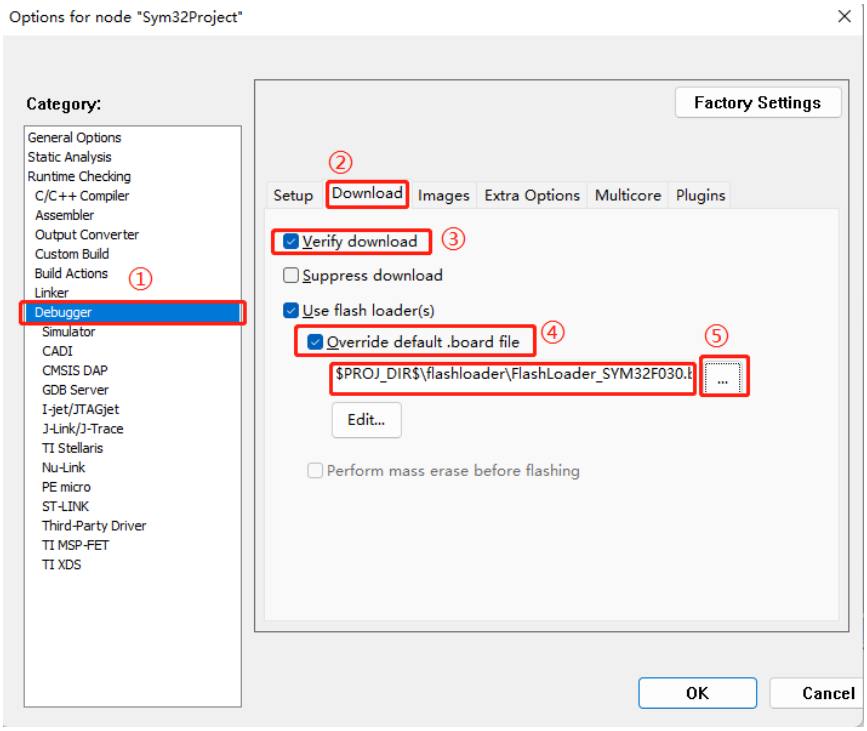
Step4. 在工程名上右键 > Options > Debugger > Download 选项卡下:

勾选 Verify download.

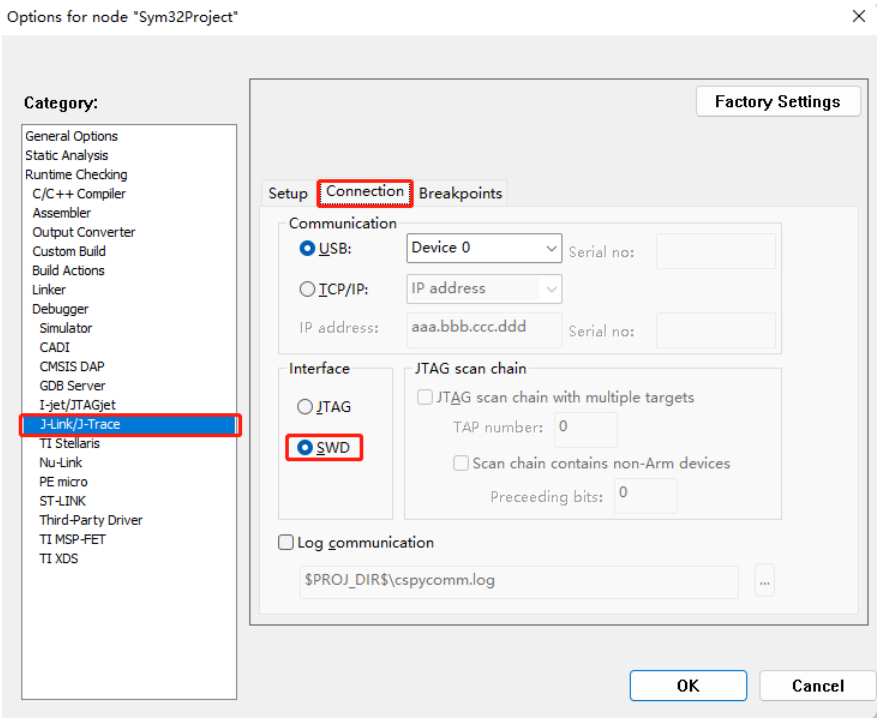
勾选 Override default .board file.

下图中⑤路径选择 F:\NewProject\EWARM\flashloader\FlashLoader_SYM32F030.board.

使用相对路径，更改为$PROJ_DIR$\flashloader\FlashLoader_SYM32F030.board

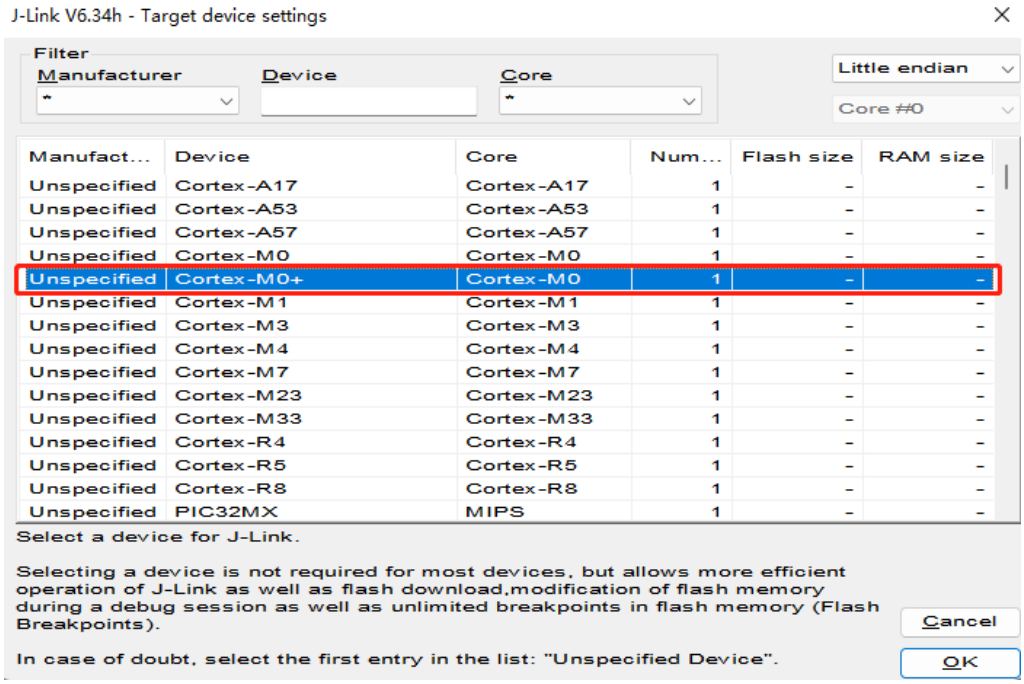


在工程名上右键 > Options > J-Link/J-Trace > Connection 选项卡下勾选 SWD.

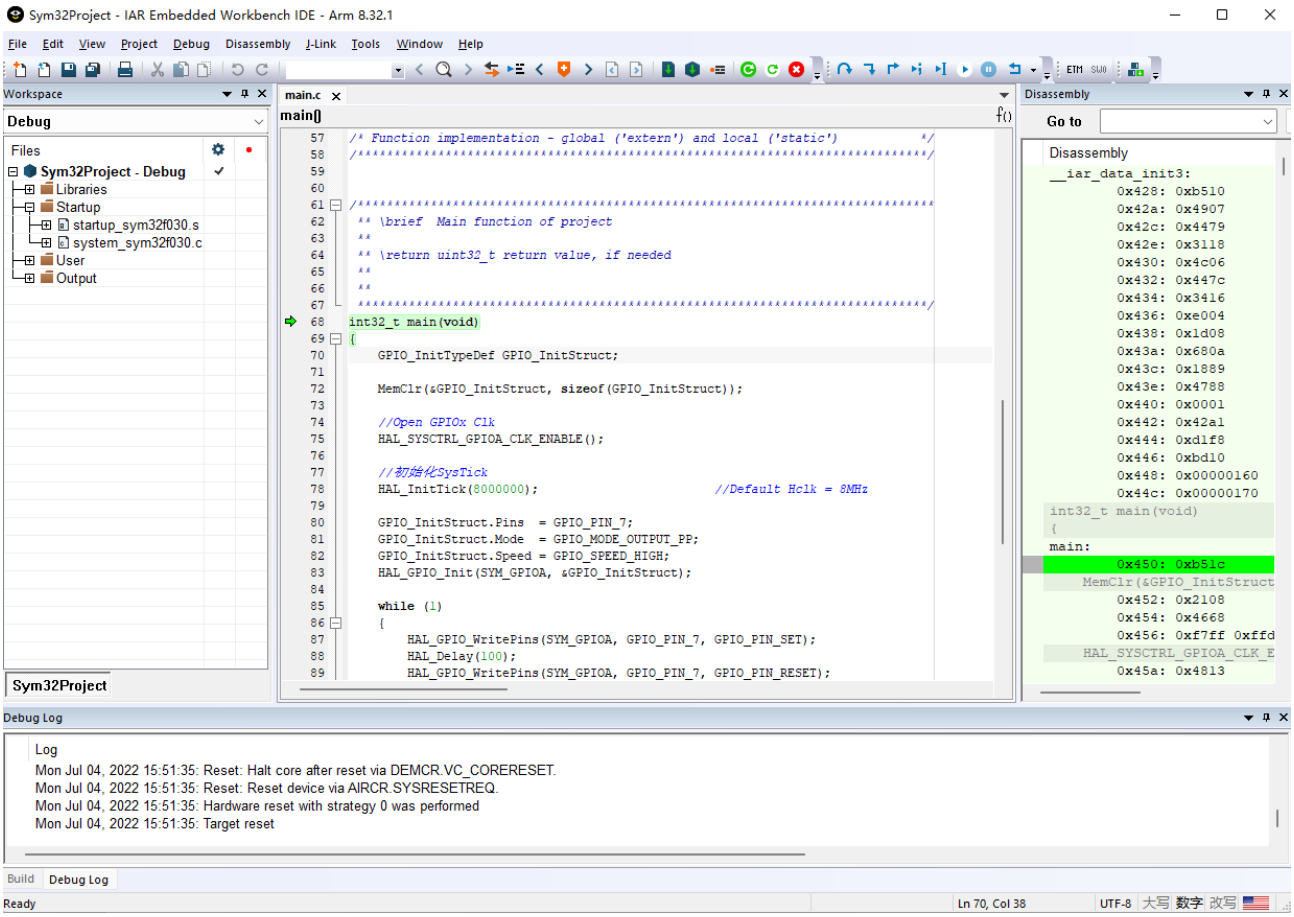


Step5. 在线仿真

以上设置都需要点击【OK】确认，保存生效，连接好电脑-仿真器-目标板，并且给目标板供电，在工程名上右键 >Rebuild All 编译整个工程，编译通过后，使用快捷键 Ctrl+D 进入 Download and Debug. 第一次下载仿真会弹出对话框，选择 Cortex-M0+ 如下图所示：



点击【OK】，进入仿真调试。


Sym32Project - IAR Embedded Workbench IDE - Arm 8.32.1
File Edit View Project Debug Disassembly J-Link Tools Window Help
Workspace
Debug
Files
Sym32Project - Debug
Libraries
Startup
startup_sym32f030.s
system_sym32f030.c
User
Output
Sym32Project
main.c
main()
/* Function implementation - global ('extern') and local ('static') */
** \brief Main function of project
** \return uint32_t return value, if needed
int32_t main(void)
{
GPIO_InitTypeDef GPIO_InitStruct;
MemClr(&GPIO_InitStruct, sizeof(GPIO_InitStruct));
//Open GPIOx Clk
HAL_SYSCTRL_GPIOA_CLK_ENABLE();
//初始化SysTick
HAL_InitTick(8000000); //Default Hclk = 8MHz
GPIO_InitStruct.Pins = GPIO_PIN_7;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Speed = GPIO_SPEED_HIGH;
HAL_GPIO_Init(SYM_GPIOA, &GPIO_InitStruct);
while (1)
{
HAL_GPIO_WritePins(SYM_GPIOA, GPIO_PIN_7, GPIO_PIN_SET);
HAL_Delay(100);
HAL_GPIO_WritePins(SYM_GPIOA, GPIO_PIN_7, GPIO_PIN_RESET);
Disassembly
Go to
Disassembly
__iar_data_init3:
0x428: 0xb510
0x42a: 0x4907
0x42c: 0x4479
0x42e: 0x3118
0x430: 0x4c06
0x432: 0x447c
0x434: 0x3416
0x436: 0xe004
0x438: 0x1d08
0x43a: 0x680a
0x43c: 0x1889
0x43e: 0x4788
0x440: 0x0001
0x442: 0x42a1
0x444: 0xd1f8
0x446: 0xbd10
0x448: 0x00000160
0x44c: 0x00000170
int32_t main(void)
{
main:
0x450: 0xb51c
MemClr(&GPIO_InitStruct
0x452: 0x2108
0x454: 0x4668
0x456: 0xf7ff 0xffd
HAL_SYSCTRL_GPIOA_CLK_E
0x45a: 0x4813
Debug Log
Log
Mon Jul 04, 2022 15:51:35: Reset: Halt core after reset via DEMCR.VC_CORERESET.
Mon Jul 04, 2022 15:51:35: Reset: Reset device via AIRCR.SYSRESETREQ.
Mon Jul 04, 2022 15:51:35: Hardware reset with strategy 0 was performed
Mon Jul 04, 2022 15:51:35: Target reset
Build
Debug Log
Ready
Ln 70, Col 38
UTF-8

6 版本记录

版本	修订日期	修订说明
Rev1.0	2022-07-05	初始版本