

北京中科昊芯科技有限公司  
Haawking-IDE 用户使用手册

V1.6.0

北京中科昊芯科技有限公司

北京中科昊芯科技有限公司

2021 年 12 月

## 版本更新记录

版本号	修改日期	修改人	修改内容
V0.0.1	2020.09.09	齐如士	初稿
V0.0.6	2020.11.2	吴军宁	IDE 升级
V0.1.0	2021.02.01	吴军宁	IDE 升级
V0.6.0	2021.06.01	吴军宁	IDE 升级
V1.2.0	2021.08.02	吴军宁	重大版本升级
V1.2.1	2021.08.07	吴军宁	增强稳定性
V1.3.0	2021.09.01	王思腾	重大版本升级
V1.4.0	2021.09.16	王思腾	重大版本升级
V1.5.0	2021.11.12	王思腾	增强稳定性
V1.5.1	2021.11.19	王思腾	增强稳定性
V1.6.0	2021.12.25	王思腾	增强稳定性

注 1：版本号注意区分重大更新和部分修改，版本更新需审批后决定。

注 2：前三位为 IDE 版本号，\_ 分割为文档更新版本号。

---

---

## V1.2.0 版本（仅支持量产版芯片）

更新 IQMath 函数库，补全乘法以及类型转换的相关函数

更新驱动库

默认使用 stdlib 库文件

更新 027 和 034 外设寄存器 json 文件

---

---

## V1.2.1 版本

IDE 升级，包括：1. 提升 28027 擦除及烧写的主频为 12MHz，及修改 Flash 对应主频的配置参数；2. 修复 Erase 函数 Bug；增加时钟校准功能；28027 增加 ENPIE 的定义，将系统时钟设为 120M，修改 delay 函数名字，labs 警告；3. 28034 增加 ENPIE 的定义，将系统时钟设为 120M，修改 IQmath 的表格，在 bootrom 中的定义，修改 delay 函数名字，labs 警告；4. 修改 28335 的 1d 文件的 fPU 表格

---

---

## V1.3.0 版本

1. 增加了 segger 实时运行库
2. 驱动库更新记录：
  - (1) 修正文件名规范为 DSP2802x
  - (2) 增加对 \_\_interrupt 的支持
3. IQMath 函数库更新记录：
  - (1) 增加了 intof 函数
  - (2) 增加对 IQcosTable 的支持
  - (3) 提高 mpy 函数的运行速度
4. OpenOCD 版本升级为 V0.1.6, 更新内容为：
  - (1) 支持 DSC28027 主频为 3MHz、12MHz 下的擦写（在配置 DivSel 成功的情况下，以 12MHz 主频的速度擦写 Flash，否则以 3MHz 主频擦写 Flash）
  - (2) 缩短了读 FLASH 状态标志位超时时间为 10 秒

(3) 增加芯片加密状态弹窗

---

---

V1.4.0 版本

1. Driver 更新记录:

(1) 支持调试时对内部时钟晶振校准

2. IDE 功能更新记录:

(1) 提供更便捷的新建工程方式

(2) 实时刷新模块支持全局变量的实时刷新

---

---

V1.5.0 版本

1. 更新驱动库

2. IDE 功能更新记录:

(1) 提供新版界面的实时刷新视图, 增加稳定性

(2) 增强了 Haawking Project 创建工程方式的稳定性

3. OpenOCD 版本升级为 0.2.0

---

---

V1.5.1 版本

1. 更新驱动库

---

---

V1.6.0 版本

1. IDE 功能更新记录:

(1) 增加 Debug Without Download 功能

(2) 增加直接烧入程序功能

(3) 支持生成静态库文件

(4) 支持切换 RAM/FLASH 后, 无需执行 Clean Project

2. Openocd 版本升级为 0.2.1

3. 更新驱动库

## 目 录

1 Haawking IDE 介绍.....	8
1.1 Haawking IDE 目录结构.....	8
1.2 Haawking IDE 常见问题.....	11
1.2.1 Haawking IDE 不支持导入友商工程.....	11
1.2.2 调试器.....	11
1.2.3 调试时遇到的问题及解决方法.....	11
1.2.4 如何添加源文件和头文件目录.....	14
1.2.5 怎样获得反汇编文件.....	17
1.2.6 如何切换已有工程 FLASH 和 RAM.....	19
1.2.7 头文件、函数、变量无法跳转到声明位置.....	20
1.2.8 Haawking IDE 不支持在任务栏中启动.....	21
1.2.9 IDE 路径包含中文字符或者空格.....	22
1.3 HaawkingIDE 工程目录介绍.....	23
1.4 Map 文件简要说明.....	24
1.5 HaawkingIDE 暂未支持的功能.....	25
1.5.1 工程重命名.....	25
1.5.2 实时波形显示.....	25
2 Haawking IDE 使用.....	26
2.1 打开 Haawking IDE.....	26
2.2 新建工程.....	27
2.3 编译.....	32
2.3.1 整个工程编译.....	32
2.3.2 单个文件编译.....	34
2.4 如何生成库文件.....	35
2.4.1 前置工作.....	35
2.4.2 生成静态库.....	37
2.4.3 使用静态库.....	39

2.5	编译优化介绍.....	40
2.5.1	-O1 级别优化.....	40
2.5.2	-O2 级别优化.....	41
2.6	生成 bin 格式文件.....	41
2.7	下载功能.....	43
2.7.1	前置工作.....	43
2.7.2	使用 IDE 下载程序.....	45
2.8	导入工程.....	48
2.9	导出工程.....	50
2.9.1	导出压缩包格式工程.....	50
2.9.2	导出文件夹格式工程.....	51
2.10	使用 Rebuild Project 功能.....	53
2.11	Debug 目录下的文件介绍.....	54
2.12	用 Spike 模拟器仿真程序.....	54
3	调试.....	57
3.1	启动调试.....	57
3.2	调试模式工具栏介绍.....	59
3.3	断点的使用及断点视图介绍.....	60
3.3.1	如何添加断点.....	60
3.3.2	如何删除断点.....	60
3.3.3	使用 Breakpoints 视图管理断点.....	60
3.3.4	Breakpoints 视图功能按钮介绍.....	61
3.3.5	FLASH 工程使用断点的注意事项.....	61
3.4	如何使用 Restart 功能复位芯片.....	62
3.4.1	FLASH 工程使用 Restart 功能.....	62
3.5	如何查看变量值.....	64
3.6	如何查看外设寄存器的值.....	65
3.7	使用 Memort 视图查看存储器中的值.....	67

3.7.1	如何查看指定地址处的数据.....	67
3.7.2	如何导出 Memory 视图中的数据.....	67
3.8	如何进入汇编调试.....	68
3.9	使用 Debug Without Download 功能.....	69
3.9.1	前置工作.....	69
3.9.2	启动 Debug Without Download.....	71
3.9.3	Debug Without Download 功能注意事项.....	72
3.10	使用实时刷新功能.....	72
3.10.1	实时刷新视图使用注意事项.....	72
3.10.2	实时刷新视图按钮介绍.....	73
3.10.3	向实时刷新视图添加变量.....	73
3.10.4	删除实时刷新视图中的变量.....	75
3.10.5	切换变量输出格式.....	75
3.10.6	切换刷新间隔.....	76
3.10.7	数据存在变化时的显示效果.....	76
3.10.8	导出实时刷新视图中的数据.....	77
3.10.9	实时刷新视图存在的问题.....	78
3.11	如何加密解密.....	79
3.11.1	加密操作.....	79
3.11.2	解密操作.....	80
4	审核.....	83

# 1 Haawking IDE 介绍

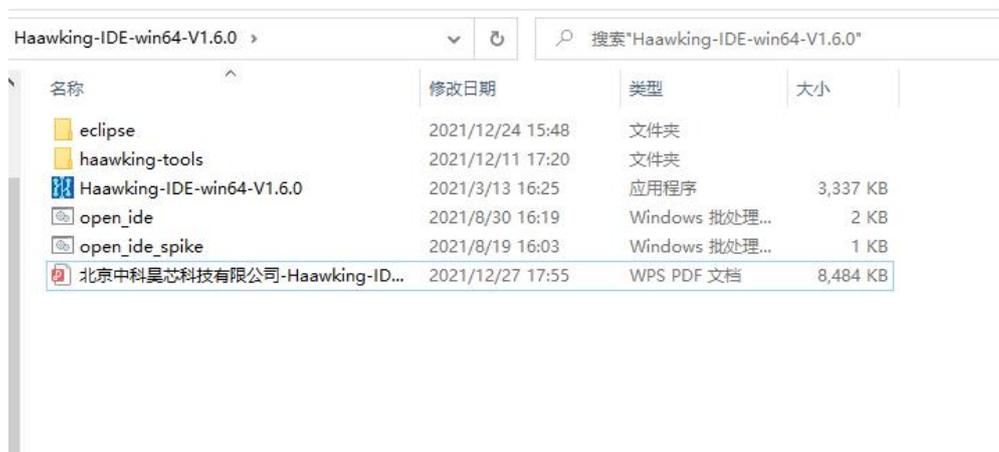
核心功能	IDE 1.2.0	IDE 1.3.1	IDE 1.4.0	IDE 1.5.1	IDE 1.6.0
实时刷新变量功能 (手册 3.10 小节)	-	-	✓	✓	✓
调试时复位芯片功能 (手册 3.4 小节)	-	-	✓	✓	✓
新版的创建工程方式 (手册 2.2 小节)	-	-	✓	✓	✓
Rebuild Project 功能 (手册 2.10 小节)	-	-	-	✓	✓
Debug Without Download 功能 (手册 3.9 小节)	-	-	-	-	✓
生成静态库功能 (手册 2.4 小节)	-	-	-	-	✓
直接烧入程序功能 (手册 2.7 小节)	-	-	-	-	✓

## 1.1 Haawking IDE 目录结构

用户可以从昊芯官方网站 (<http://haawking.com/zyxz>) 下载到最新版本的 Haawking IDE 软件和相关的驱动库，首次使用，请下载完整包 ([Haawking-IDE-win64-V1.6.0.exe](#))，包括编译器、JRE 环境、HX-Link 以及 OpenOCD 等工具。



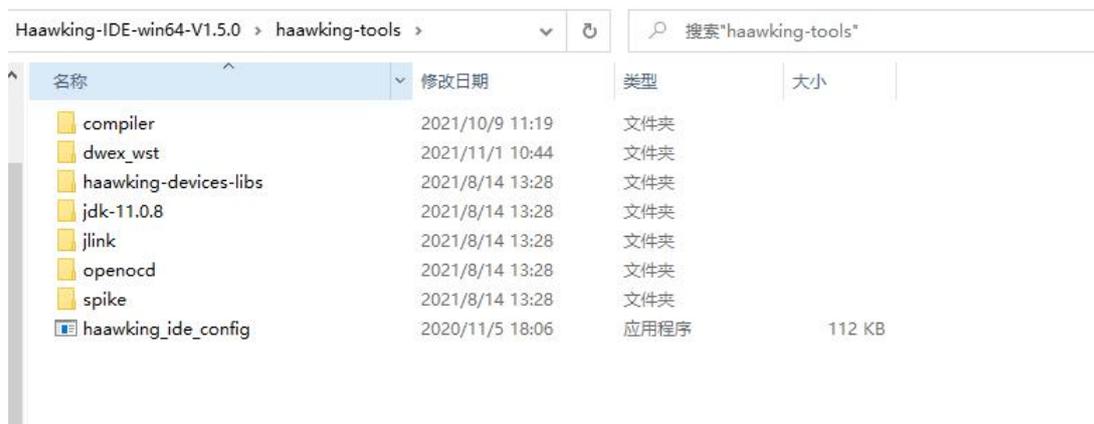
解压之后，将会得到下图所示的文件，包括 eclipse 目录、开发工具 haawking-tools 目录、Haawking-IDE 用户手册、Haawking-IDE-win64-V1.6.0.exe 文件。



**注意：**用户需记住 Haawking IDE 的解压路径，说明示例的解压路径为 D:\Haawking-IDE-win64-V1.6.0；后面很多操作均依赖该路径。

**☆：Haawking IDE 的解压路径中不能包含中文字符及空格，否则无法正常工作。建议解压在根目录！**

Haawking IDE 的开发工具都集中在 haawking-tools 目录，包括编译器、链接器、调试器等。该目录的文件内容如下图：



Haawking IDE 的编译器基于 LLVM 12.0.0 开发，链接器 ld 和调试器 gdb 则使用了 RISC-V GCC 10.2.0 官方发布版本。OpenOCD 是根据最新源码编译的官方版本。

开发工具后续存在升级的可能，使用者可以根据需要选择对某一开发工具单独进行升级。在升级的时候，只需替换相关路径下的文件即可。

**☆☆☆：使用 Haawking IDE 之前，请先阅读第一章的 2、3、4、5 小节。**

## 1.2 Haawking IDE 常见问题

使用过程中，如果遇到 FAQ 未列的情况，可以<https://gitee.com/haawking/haawking-tools/issues>下面**按照格式要求**，提交问题反馈，开发人员会尽快给予回复。

### Haawking IDE V0.0.6-IQmathLib-Bug

#I23S3B 待办的 JunningWu 拥有者 创建于 2天前

IDE名称: Haawking-IDE-Eclipse-CDT.win32.x86\_64\_V0.0.6

IDE版本: V0.0.6

运行环境: FPGA-ZYNQ-MZ7035FD;主频10MHz

问题描述: 如下

原因分析: 库文件中的函数调用存在bug

解决办法:

建议:

勾选Bootrom Table选项，需要这样写程序，才可以运行出正确结果。

```
a = _IQsin(998);
```

而写成下面的形式，则运行结果错误。

```
a = _IQ29sin(998);
```

在不勾选Bootrom Table选项时，a = \_IQ29sin(998);可以得到正确的结果。

### 1.2.1 Haawking IDE 不支持导入友商工程

Haawking IDE 不支持导入、编译及调试友商工程。

### 1.2.2 调试器

**我们始终建议客户选择 HX-Link 进行调试。**

### 1.2.3 调试时遇到的问题及解决方法

#### 1.2.3.1 设备连接错误之一

当出现下面错误信息，说明 HX-Link 调试器与开发板未正确连接。请检查 JTAG 线序或因为排线过长导致的信号不稳定。建议使用中科昊芯提供的 14-PIN 排线或连接线不长于 14cm。

```

<terminated> HX_DSC28027_FLASH_120MHZ_APPOINTED_ADDR_TEST_V_0_0_5-debug-ooecd [GDB Op
Error: JTAG scan chain interrogation failed: all ones
Error: Check JTAG interface, timings, target power, etc.
Error: Trying to use configured scan chain anyway...
Error: riscv.cpu: IR capture error; saw 0x1f not 0x01
Warn : Bypassing JTAG setup events due to errors
Error: Unsupported DTM version: 15
Error: Target not examined yet

Error: Unsupported DTM version: 15

```

### 1.2.3.2 设备连接错误之二

当出现下面错误信息，说明芯片未上电，需要检查供电或检查芯片是否按正确方向电装。

```

<terminated> HX_DSC28027_FLASH_120MHZ_APPOINTED_ADDR_TEST_V_0_0_5-debug-ooecd [GDB OpenOCD Debugging] open
Error: JTAG scan chain interrogation failed: all zeroes
Error: Check JTAG interface, timings, target power, etc.
Error: Trying to use configured scan chain anyway...
Error: riscv.cpu: IR capture error; saw 0x00 not 0x01
Warn : Bypassing JTAG setup events due to errors
assertion "info->abits != 0" failed: file "../src/target/riscv/riscv-013.c", line 506, function: dmi_scan

```

### 1.2.3.3 设备烧写错误

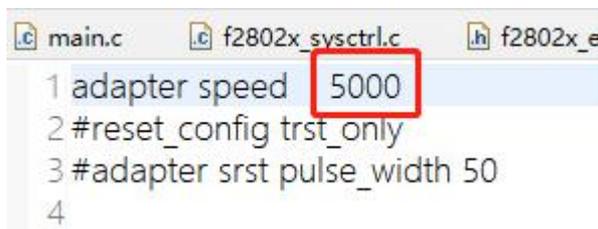
当出现下面错误信息，说明芯片在烧写 Flash 时，烧写校验失败。建议用户在烧写前检查芯片的供电或芯片周边是否有电磁干扰，或降低“.cfg”配置文件中的调试频率的参数，调试频率的参数最低为 500，默认单位为“KHz”。排查问题后，可复位芯片重新烧写。

```

Flash Sector 0/63 Loading...
Sector 0/63 Verify Succeed
Flash Sector 0/63 Loading...
Started by GNU MCU Eclipse
Error: Load Sector Failed...
Error: write data is 0x7d, read data is 0x0
Error: Load Sector Failed...
Error: write data is 0x15, read data is 0x0
Error: write data is 0xe3, read data is 0xe3
Error: write data is 0x4f, read data is 0x4f
Error: write data is 0xa0, read data is 0xa0
Error: write data is 0xfe, read data is 0xfe
Error: write data is 0x82, read data is 0x82

```

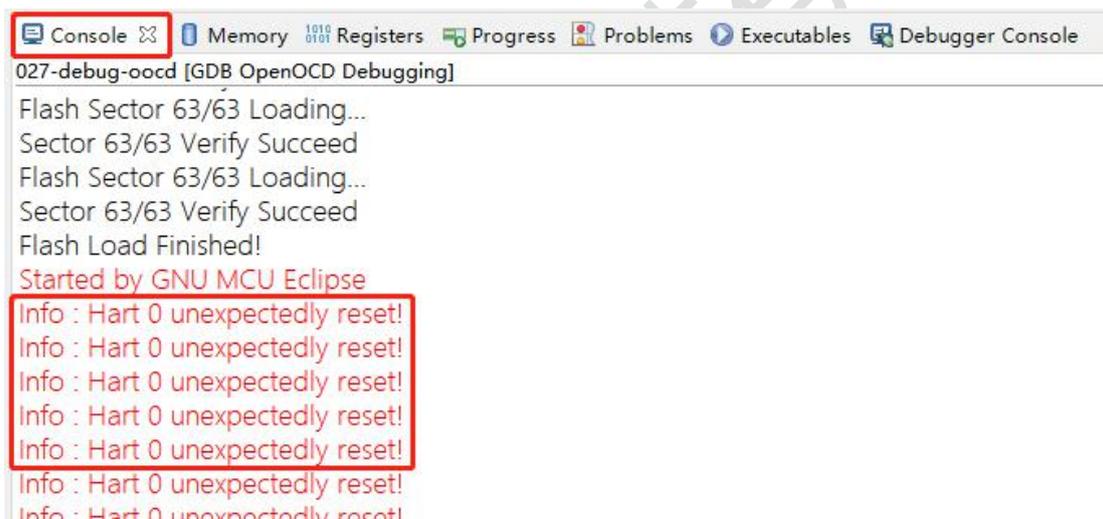
“.cfg”文件的位置在  
“/haawking-drivers/haawking-dsc28027-board/board/”下，修改配置参数见下图：



```
main.c f2802x_sysctrl.c f2802x_e
1 adapter speed 5000
2 #reset_config trst_only
3 #adapter srst pulse_width 50
4
```

### 1.2.3.4 芯片复位导致与仿真器断开连接

当出现下面错误信息，说明芯片复位导致与仿真器断开连接。是由于程序中未关闭看门狗导致，或因程序配置问题引起芯片复位。因此需要用户检查程序代码。



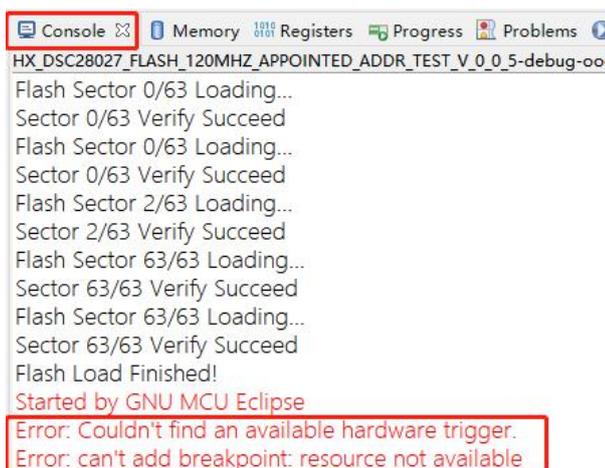
```
Console Memory Registers Progress Problems Executables Debugger Console
027-debug-occd [GDB OpenOCD Debugging]
Flash Sector 63/63 Loading...
Sector 63/63 Verify Succeed
Flash Sector 63/63 Loading...
Sector 63/63 Verify Succeed
Flash Load Finished!
Started by GNU MCU Eclipse
Info : Hart 0 unexpectedly reset!
```

### 1.2.3.5 断点设置过多

在RAM工程中，断点的数量是任意的。

在Flash工程中，用户最多添加两个断点。

**注意：**Debug模式下，Flash工程中再加入第三个断点，或进入Debug模式之前就断点数量就已经超过了三个，那么控制台就会报如下错误：



```

HX_DSC28027_FLASH_120MHZ_APPOINTED_ADDR_TEST_V_0_0_5-debug-oo
Flash Sector 0/63 Loading...
Sector 0/63 Verify Succeed
Flash Sector 0/63 Loading...
Sector 0/63 Verify Succeed
Flash Sector 2/63 Loading...
Sector 2/63 Verify Succeed
Flash Sector 63/63 Loading...
Sector 63/63 Verify Succeed
Flash Sector 63/63 Loading...
Sector 63/63 Verify Succeed
Flash Load Finished!
Started by GNU MCU Eclipse
Error: Couldn't find an available hardware trigger.
Error: can't add breakpoint: resource not available

```

导致芯片无法再继续运行，需要点击关闭按钮  关掉进程，再移除所有断点后重新进入 Debug 模式。

### 1.2.3.6 警告提示但不影响调试

以下的调试警告信息提示，只要不是一直在循环提示，是不影响芯片的正常调试的。



```

HX_DSC28027_FLASH_120MHZ_APPOINTED_ADDR_TEST_V_0_0_5-debug-ooocd [GDB OpenOCD Debugging]
Error: Failed read (NOP) at 0x11; value=0x1, status=f
Error: Failed read (NOP) at 0x11; value=0x1, status=f
Error: OpenOCD only supports Debug Module version 2 (0.13) and 3 (0.14), not 0 (dmstatus=0x80000000). This error might be caused by a JTAG signal issue. Try reducing the JTAG clock speed.
Error: Failed read (NOP) at 0x11; value=0x1, status=f

```

### 1.2.4 如何添加源文件和头文件目录

在实际的使用中，除了芯片的驱动库和“BSP”之外，可能需要添加自己的源文件目录和头文件目录，除了可以将源文件和头文件放在“src”目录之外，还可以手动添加。

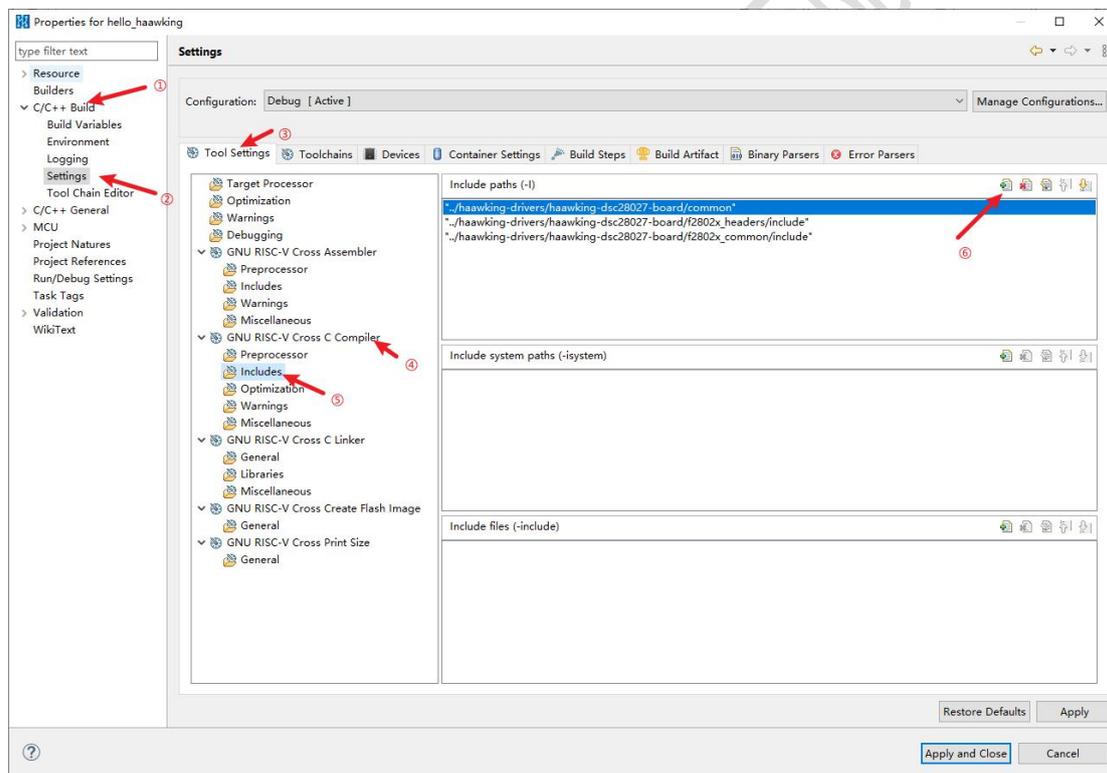
#### 1.2.4.1 添加头文件目录

下面以 hello\_hawking 工程下的 test\_include 目录作为演示添加头文件的例子，如下图所示。



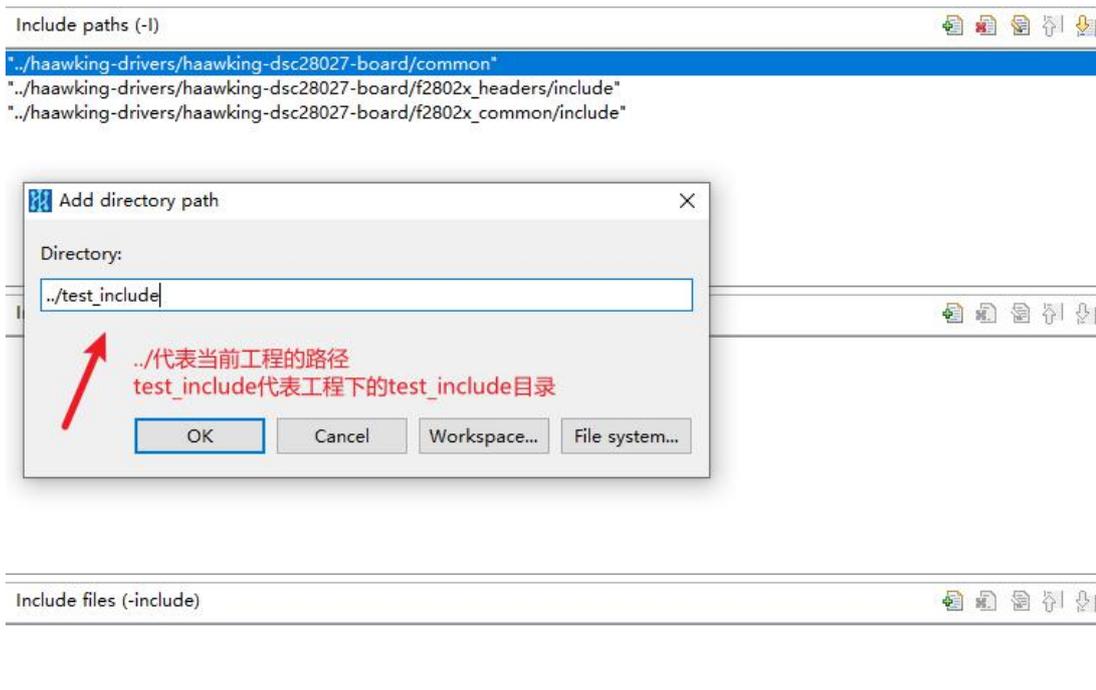
实例头文件目录及头文件

右键点击项目名称，再点击“Properties”，打开项目属性选项卡。“C/C++ Build→Settings→Tool Settings→GNU RISC-V Cross C Compiler→Includes”。

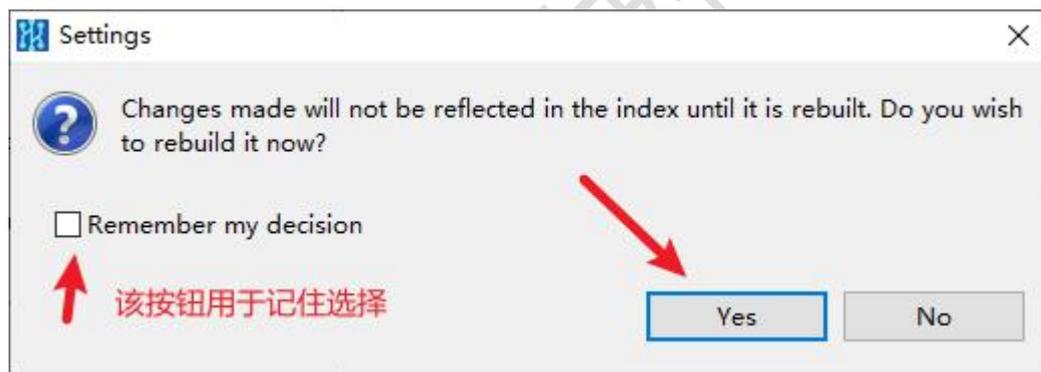


在“Include paths (-I)”中，点击“Add...”按钮，会弹出选择头文件目录的面板，由于例子中的头文件目录位于 hello\_haawking 项目中，因此可以使用相对路径的方式，即输入“../test\_include”，然后点击 OK 按钮。

注：添加头文件时，使用“../”路径就代表项目根路径。



用户添加完头文件目录后，会弹出下方的对话框，点击“Yes”按钮即可。



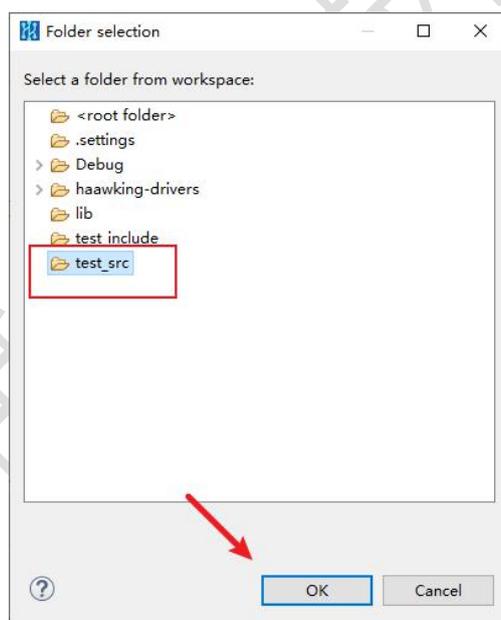
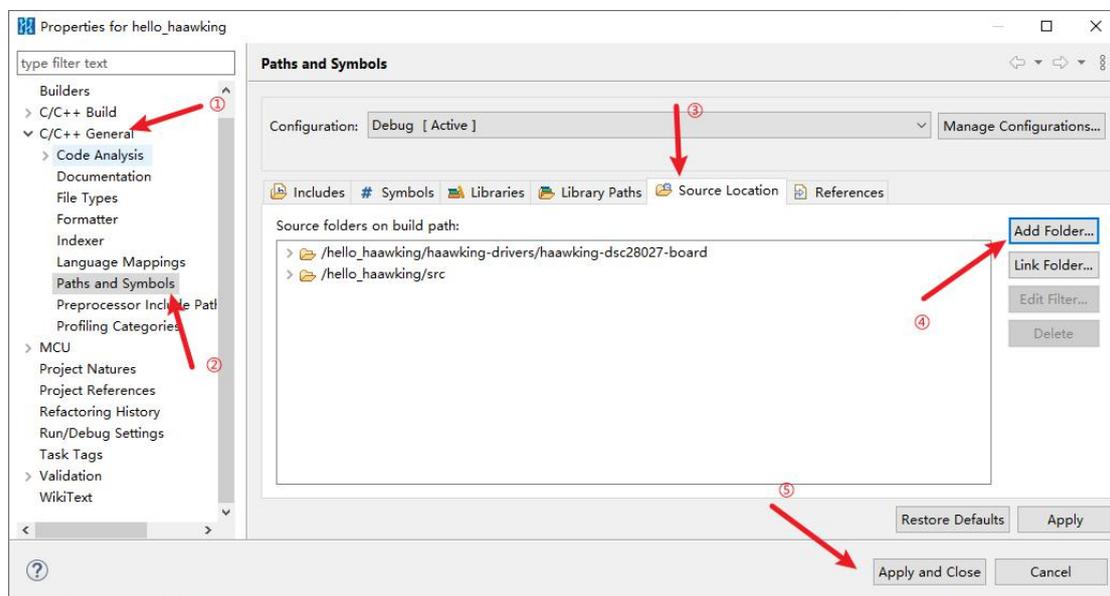
### 1.2.4.2 添加源文件目录

下面以 `hello_haawking` 工程下的 `test_src` 目录作为演示添加头文件的例子，如下图所示。



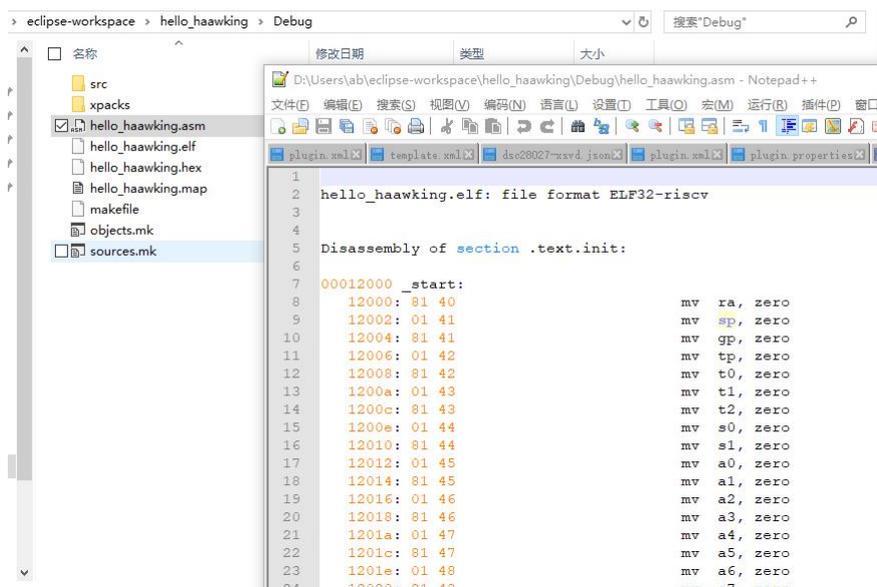
对于源文件目录，如果不想放在“`src`”目录下，可以通过项目属性选项卡，

“C/C++ General → Paths and Symbols → Source Location”，点击“Add Folder...”按钮，找到并添加自己的源文件目录后，点击 Apply and Close 按钮即可。

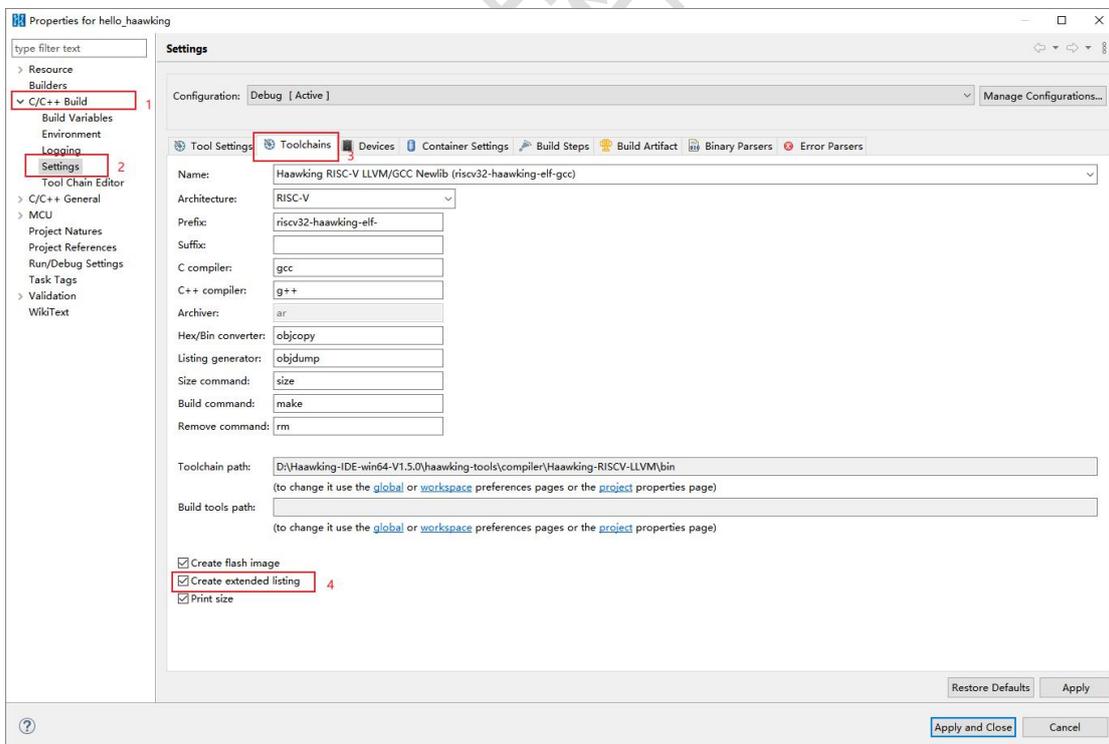


### 1.2.5 怎样获得反汇编文件

V1.5.1 版本的 IDE，默认会输出可执行文件的反汇编文件，在 Debug 目录下，文件以“.asm”和“.s”后缀结尾，分别由“LLVM”和“GCC”反汇编工具生成，其中“.asm”支持中科昊芯自定义指令助记符显示；二者指令编码一样。

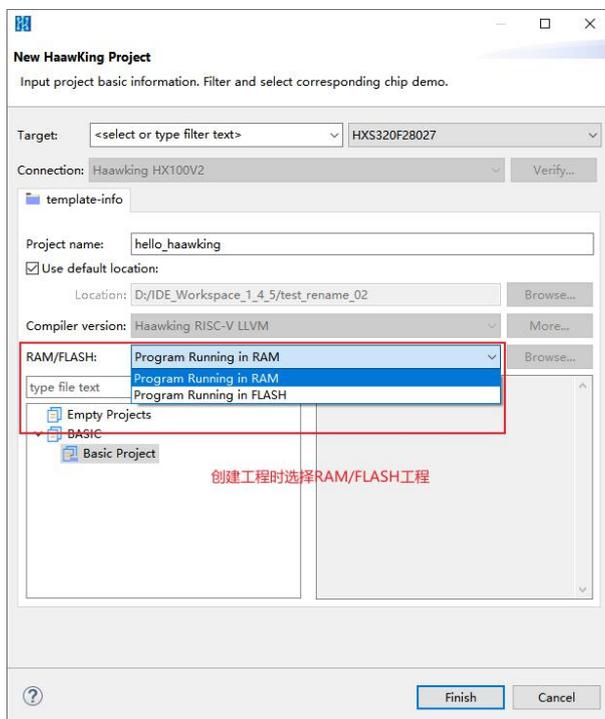


Haawking IDE 中也提供了另外一种方式获得反汇编文件，右键点击项目名称，在弹出的菜单中点击“Properties”按钮，打开项目属性选项卡。“C/C++ Build→Settings→Toolchains”，勾选“Create extended listing”。右键项目名称，在菜单中点击“Rebuild Project”按钮，待项目重新编译完成后，在项目的 Debug 目录下就能够看到反汇编文件，文件后缀为“.lst”。



## 1.2.6 如何切换已有工程 FLASH 和 RAM

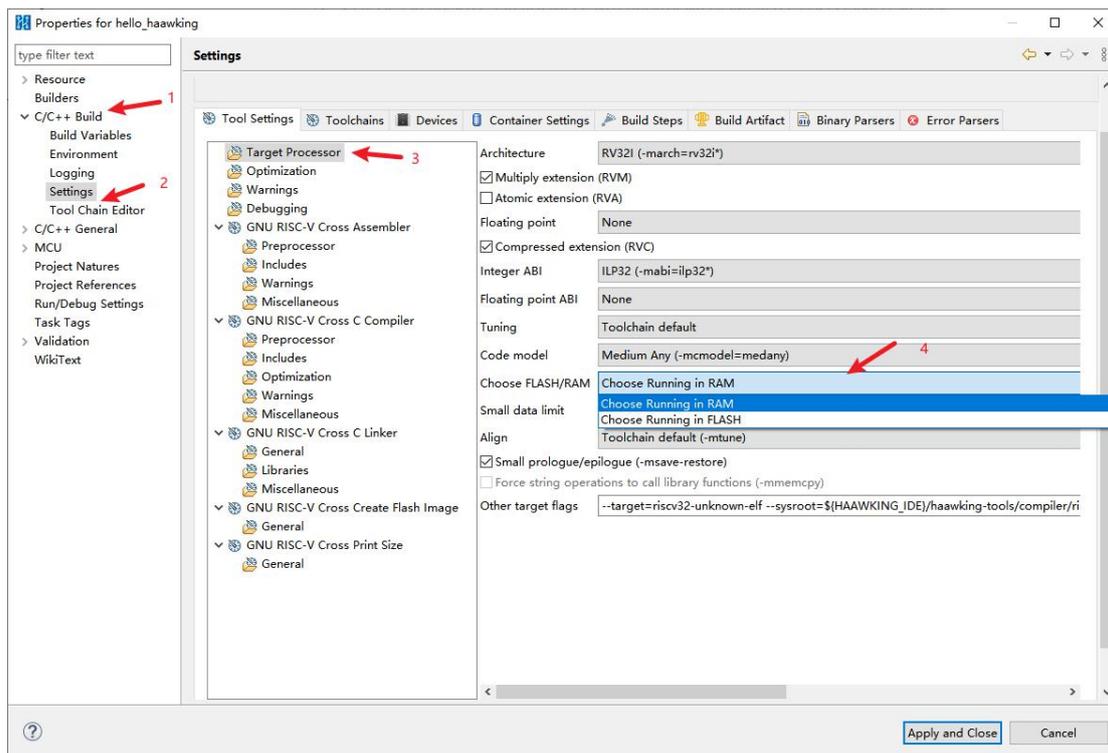
在新建工程的时候，一般都会选择了 RAM 或者 FLASH 执行。



对于已有工程，首先通过右键项目名称->“Properties”按钮，打开项目的 Properties 面板。

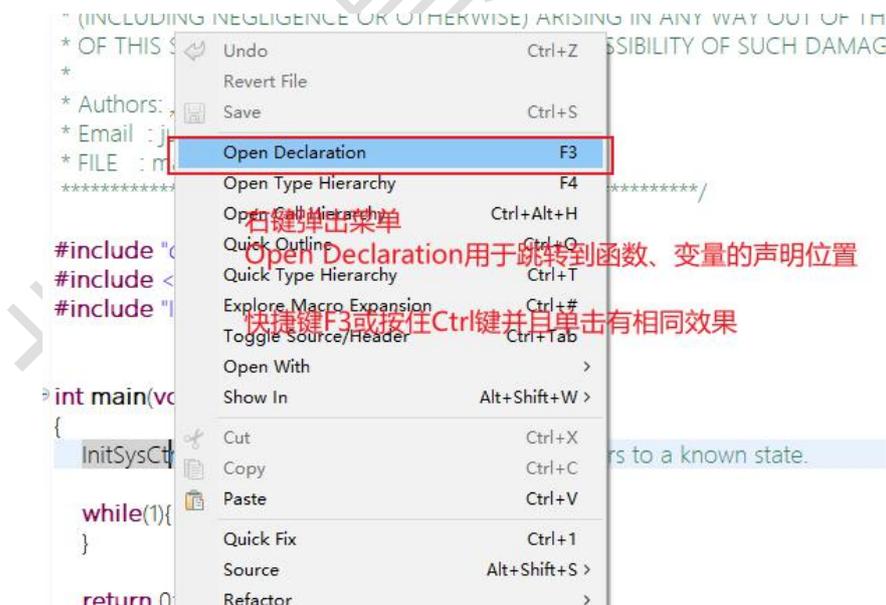
在“C/C++ Build”->“Settings”->“Tool Settings”->“Target Processor”->“Choose FLASH/RAM”选项中，通过下拉框来对工程的 RAM 和 FLASH 属性进行切换，切换 RAM/FALSH 工程选项之后，点击“Apply and Close”来使配置生效。切换配置后，重新编译工程即可。

注：操作步骤可以看下方图片



### 1.2.7 头文件、函数、变量无法跳转到声明位置

在 Haawking IDE 中，使用右键菜单->“Open Declaration”、快捷键 F3 或按住 Ctrl 键并且单击可以跳转到函数、变量声明的位置。

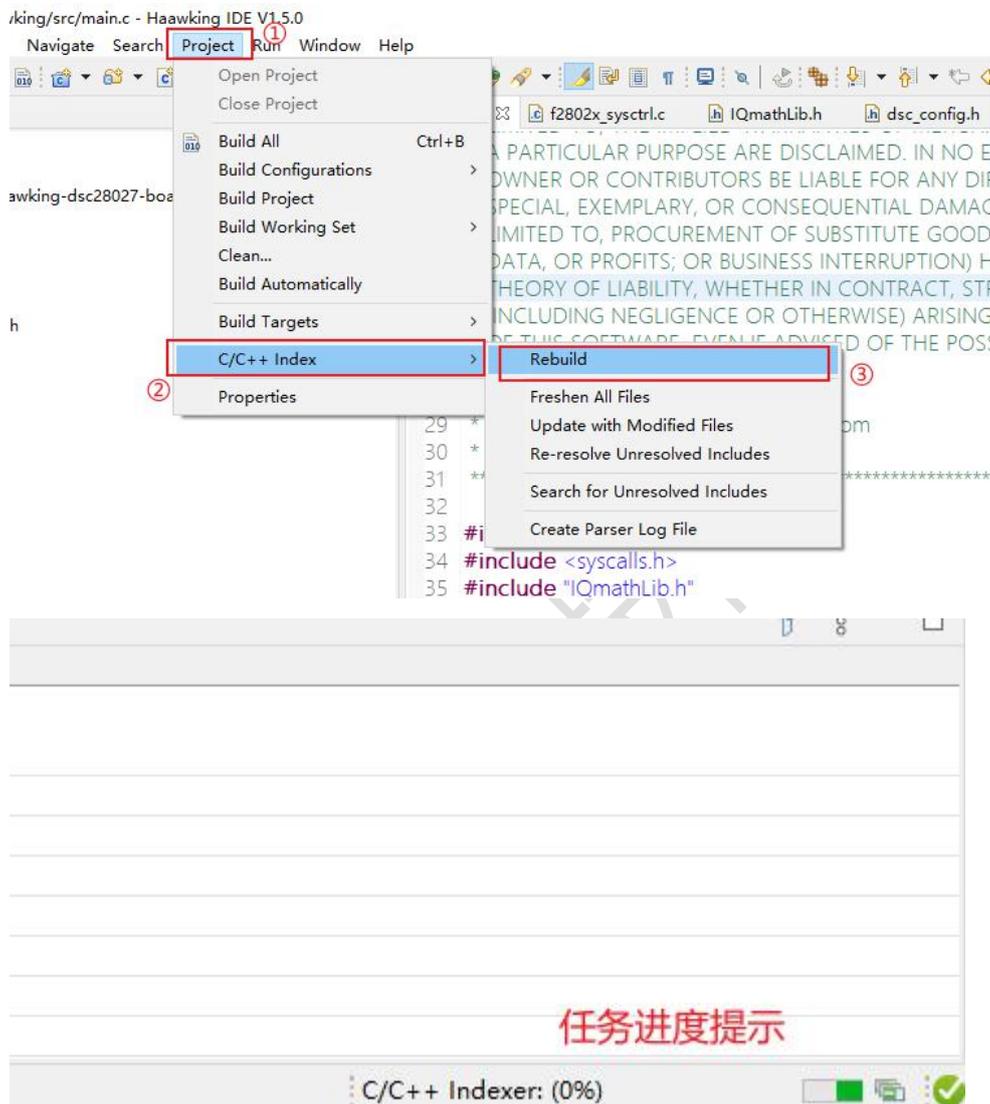


如果出现无法跳转或者跳转位置出错的问题，请按照下面步骤操作：

步骤一：如果用户使用 F3 快捷键方式，那么先尝试使用右键菜单->“Open Declaration”方式，如果仍然出错，按步骤二操作；如果能够正确跳转，请用

户检查系统的 F3 快捷键是否被其它软件占用。

步骤二：点击菜单栏中的“Project” -> “C/C++ Index” -> “Rebuild”按钮，待 IDE 右下方的进度条完成后，即可正常使用。



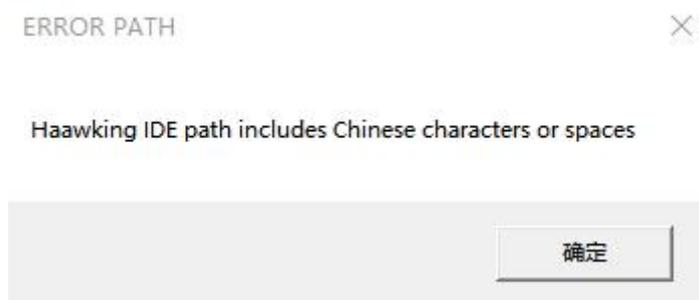
### 1.2.8 Haawking IDE 不支持在任务栏中启动

Haawking IDE 尚未支持在 Windows 10 任务栏中启动，请用户勿将 Haawking IDE 固定到任务栏。



### 1.2.9 IDE 路径包含中文字符或者空格

Haawking IDE 不支持路径中包含中文字符或者空格，如果解压路径包含中文字符或者空格，则将无法启动 Haawking IDE，同时会弹出如下错误信息：



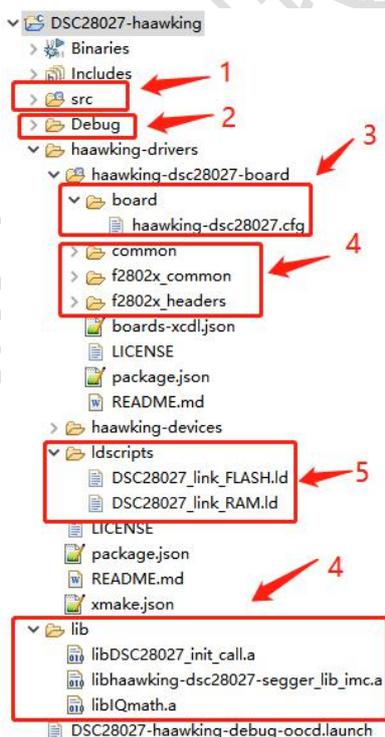
### 1.3 HaawkingIDE 工程目录介绍

Haawking IDE 的工程目录如下表：

目录结构	对应下图中得序号	Haawking IDE 工程目录
用户文件目录	1	src 文件夹下
生成文件目录	2	Debug 文件夹下
驱动库	4	common、f2802x_common、 f2802x_header 文件夹下
调试器配置文件	3	board 下的 .cfg 文件
链接文件	5	ldscripts 文件夹下的 .ld 文件

**注意：**HaawkingIDE 是把 FLash 和 RAM 的地址空间单独配置的，但不建议用户对 “.ld” 文件进行修改。

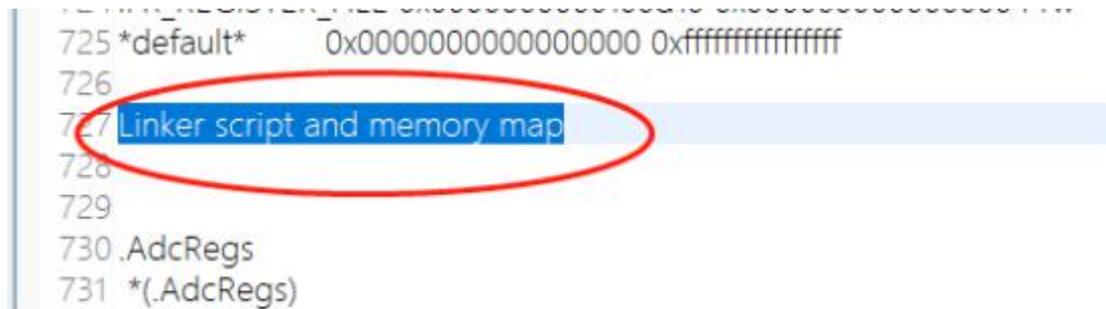
Haawking IDE 的工程目录如下图：



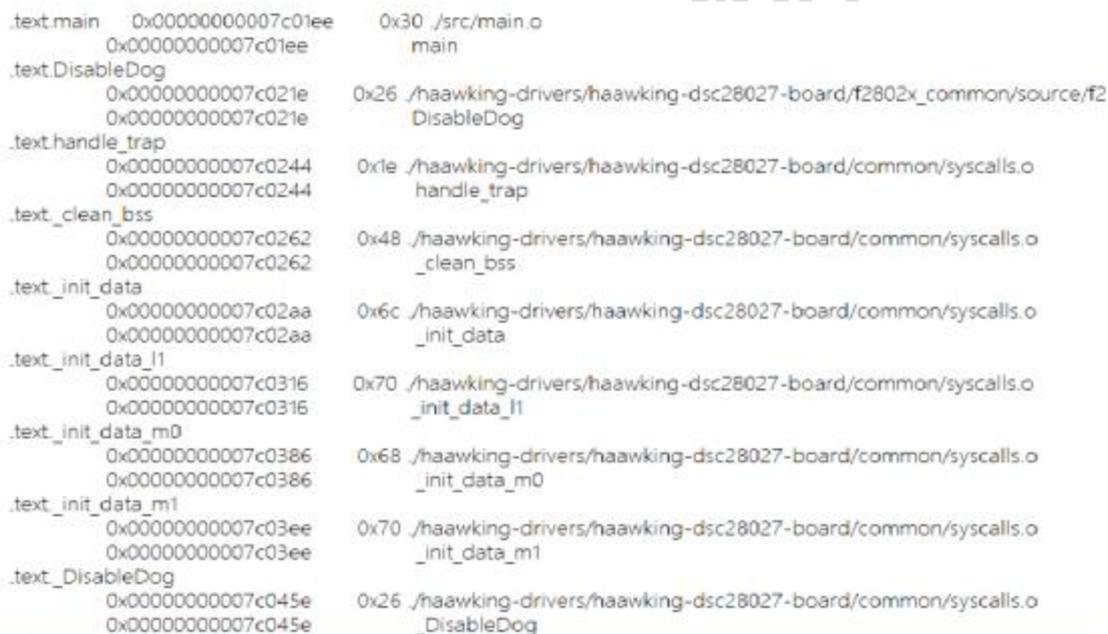
## 1.4 Map 文件简要说明

### 1、如何通过 map 文件查看程序分布

打开“map”文件，找到“Linker script and memory map”这一段，如下图：



1) 它下面的内容如图所示：



2) 其中的格式如表格中显示：

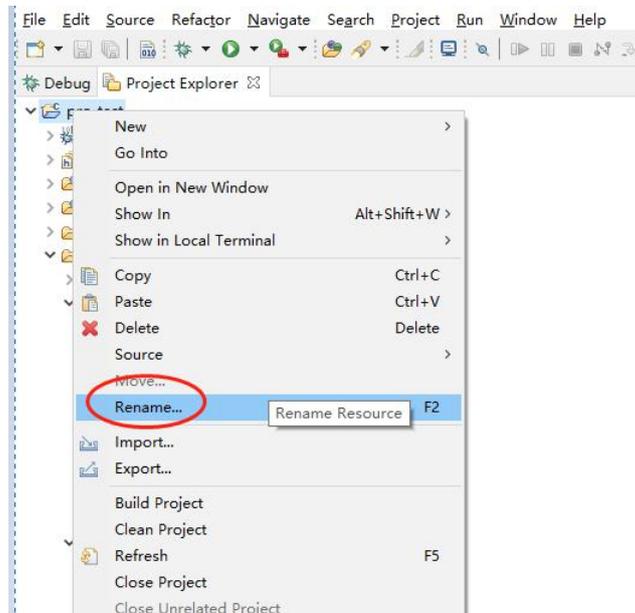
段名称	段运行地址	段大小、段所在文件
	函数运行地址	函数名称
	函数运行地址	函数名称

**注意：**当只有段名称，没有段运行地址的时候，该段已经被编译器删除了。

## 1.5 HaawkingIDE 暂未支持的功能

### 1.5.1 工程重命名

Haawking IDE 暂不支持工程重命名功能。



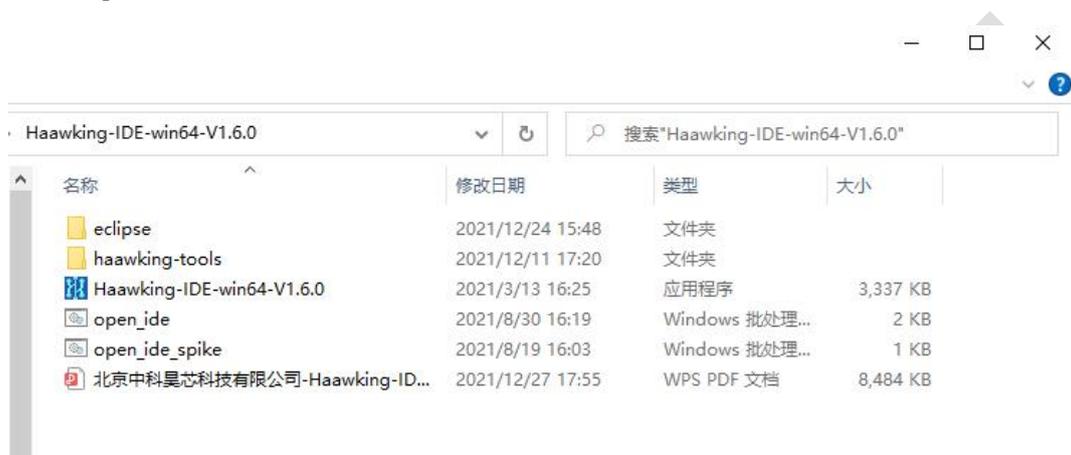
### 1.5.2 实时波形显示

Haawking IDE 暂不支持实时波形显示功能。

## 2 Haawking IDE 使用

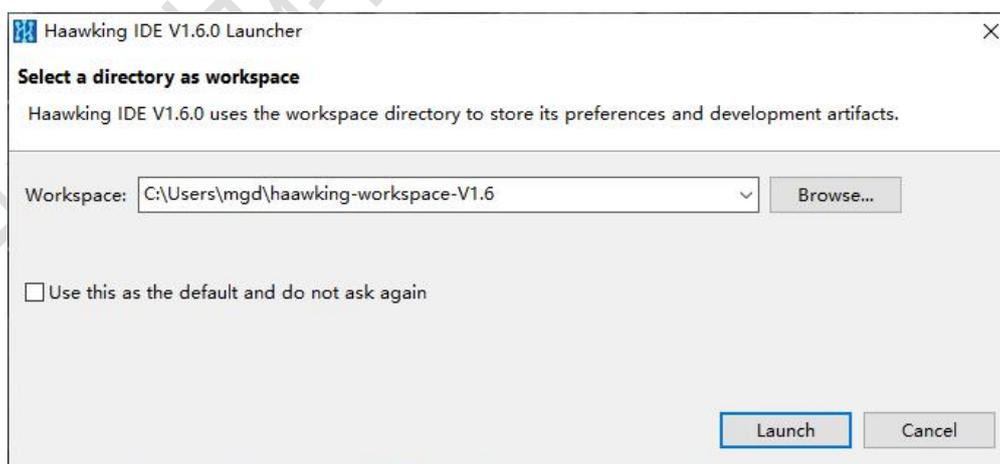
### 2.1 打开 Haawking IDE

双击根目录下 Haawking-IDE-win64-V1.5.1.exe 可执行文件，即可打开 Haawking IDE。



该执行文件会配置好需要的工具，如“JDK”、“OpenOCD”、编译器和链接器等。用户可以选择默认的工作空间 (C:\Users\username\haawking-workspace-V1.5)，也可以根据需要进行其他路径。

注意事项：工作空间所在的路径不允许包含中文字符！



在 IDE 启动成功后会弹出下方的欢迎界面。



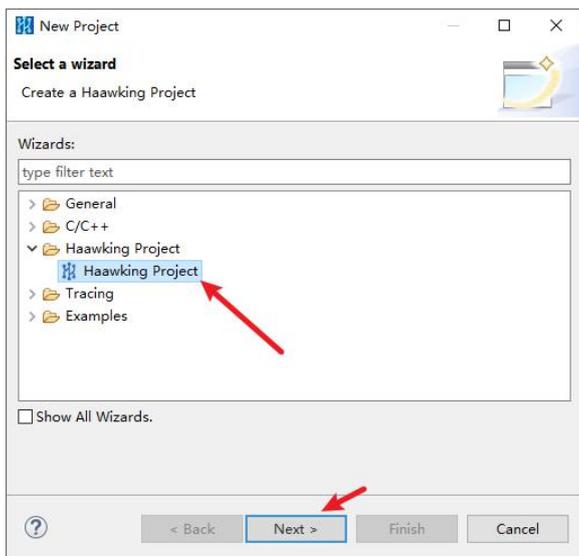
## 2.2 新建工程

Haawking IDE 不支持“&”、“|”等运算符号以及中文字符进行命名，推荐使用字母开头，数字+字母+下划线的组合进行命名。

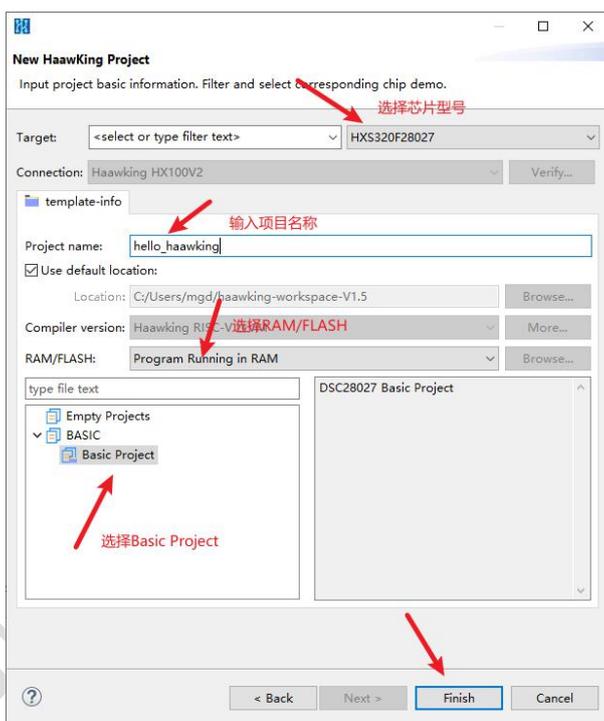
目前 Haawking IDE 支持两种创建工程的方式，下面对两种方式分别进行介绍。

第一种方式：使用 Haawking Project 创建工程

首先点击“File->New->Projects”，打开新建工程的窗口，选择“Haawking Project->Haawking Project”选项，点击“Next”进入下一个界面。

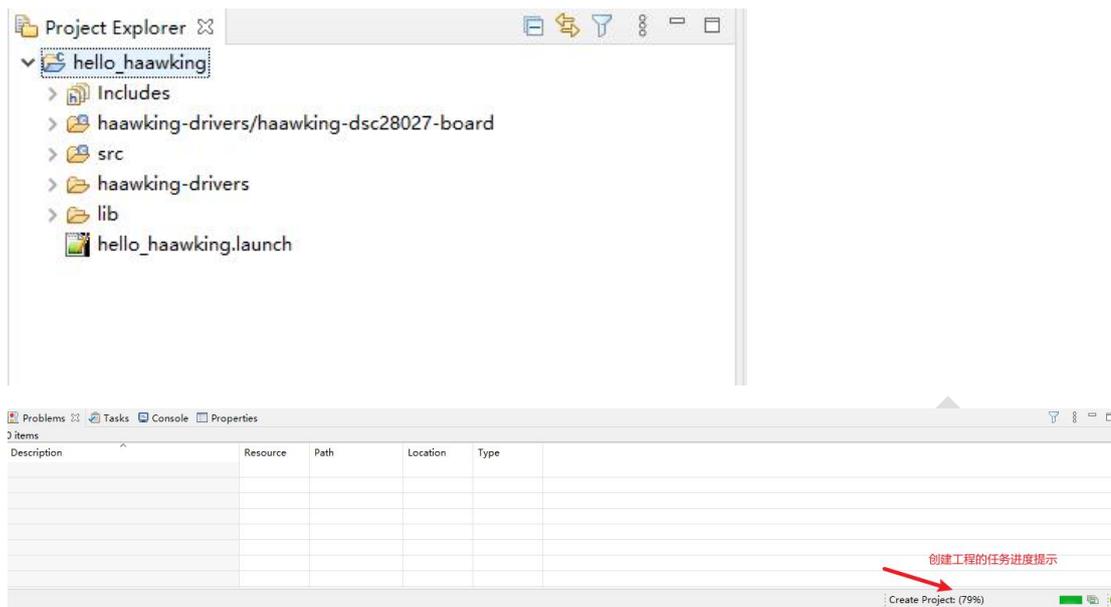


在新的界面中依次选择芯片型号、选择 RAM/FLASH、选择“Basic Project”作为模板、输入项目名称，点击 Finish 按钮即可。



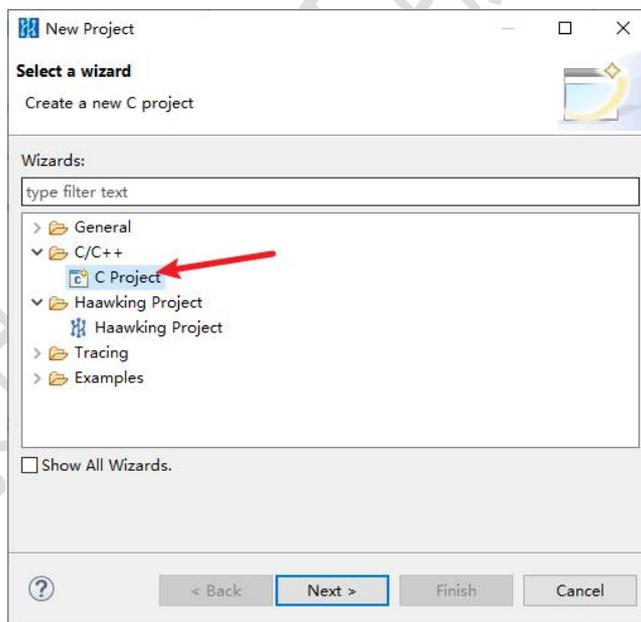
点击 Finish 后，等待一会后 IDE 左侧的“Project Explorer”面板中会出现之前新建的工程。

注：当一个新的空间中创建第一个工程时，可能会花费较长的时间，此时可以查看 IDE 右下角的任务进度提示，如下图 2 所示。

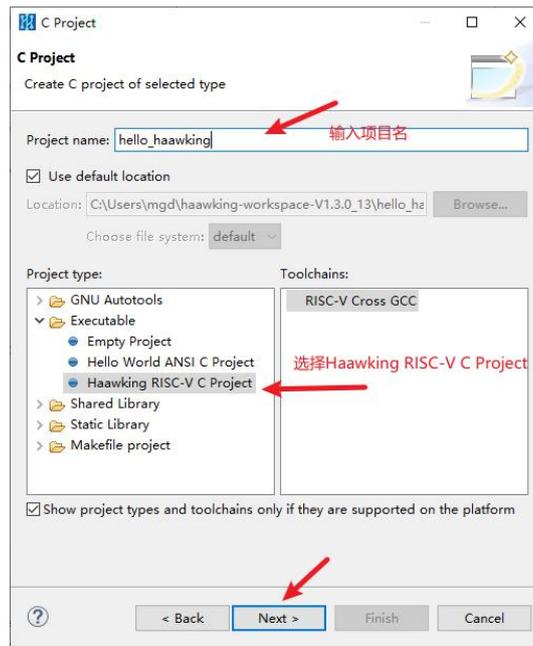


第二种方式：使用 C Project 创建工程

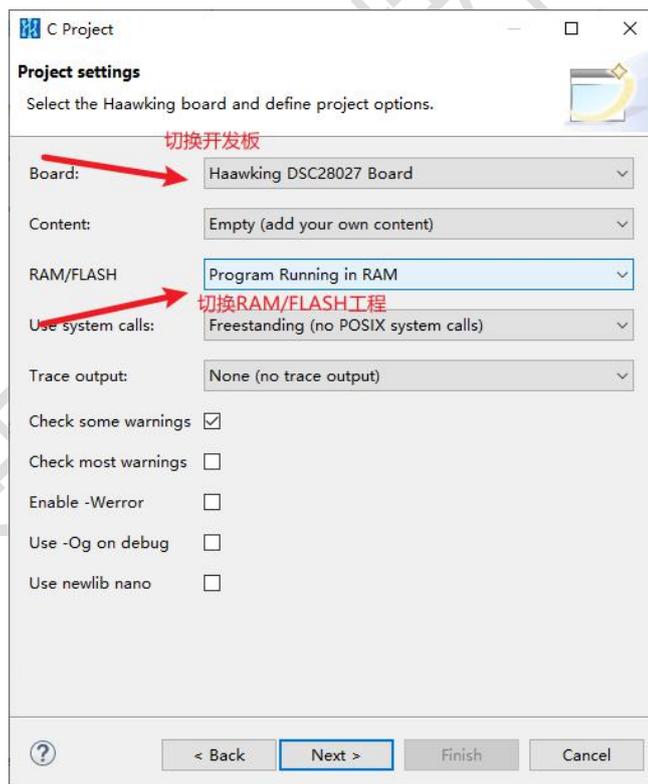
点击“File->New->Projects”，打开新建工程的窗口，选择“C/C++->C Project”选项，点击“Next”进入下一个界面。



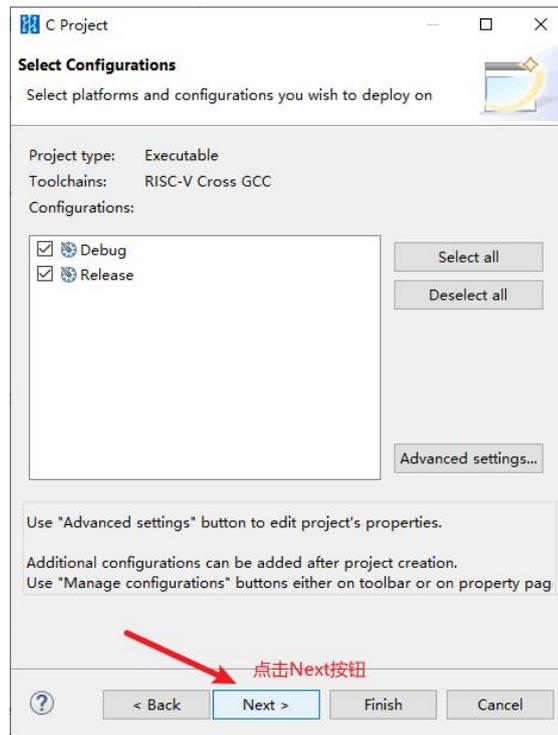
在打开的“C Project”面板中，选择“Haawking RISC-V C Project”，输入项目名称“hello\_haawking”，点击 next 按钮。



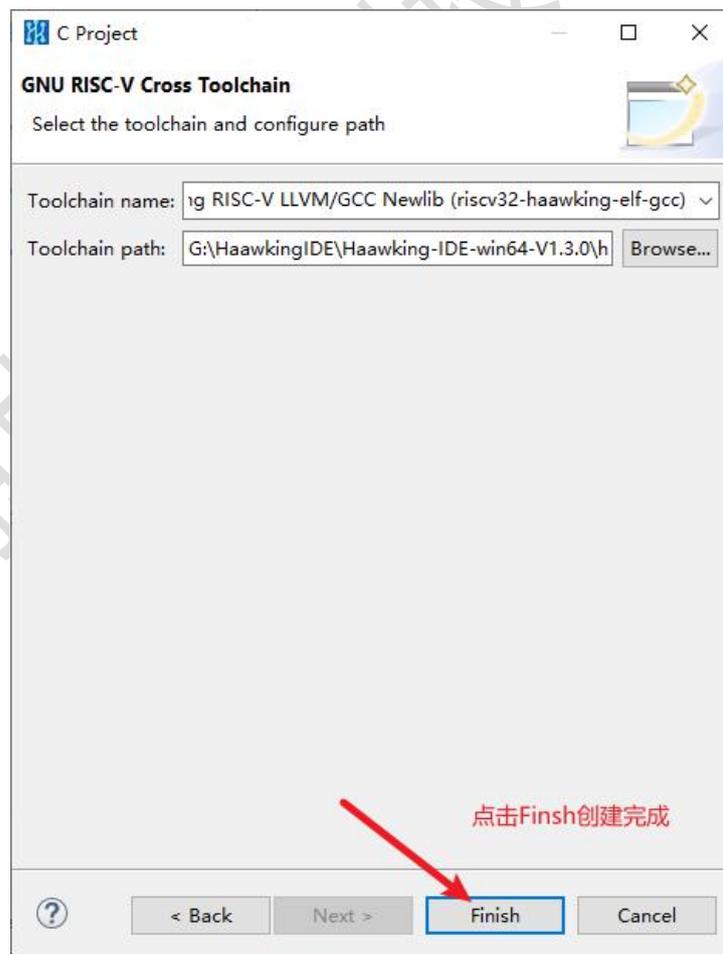
在新出现的面板中，选择芯片型号、RAM/FLASH 工程，点击“Next”按钮



在新出现的面板中，点击“Next”按钮。



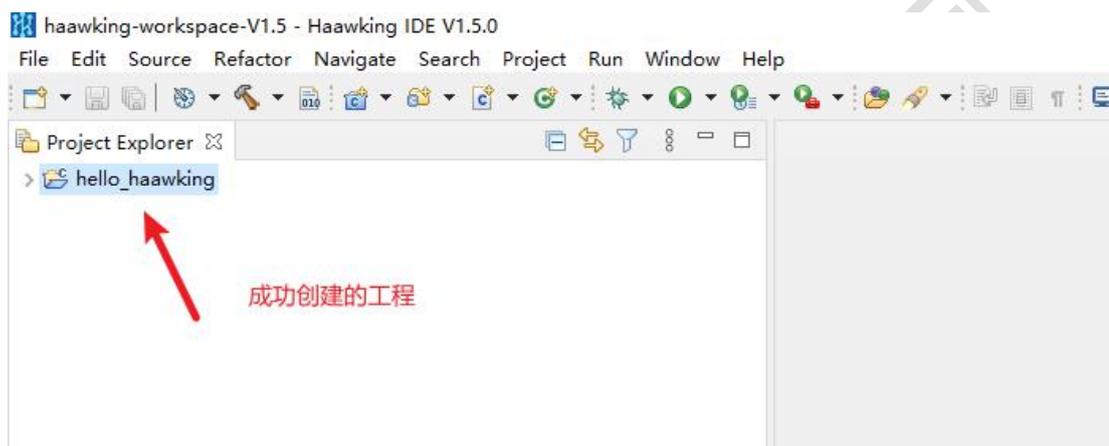
在新面板中点击“Finish”按钮



如果出现下图中的弹窗， 点击“Open Perspective”按钮。



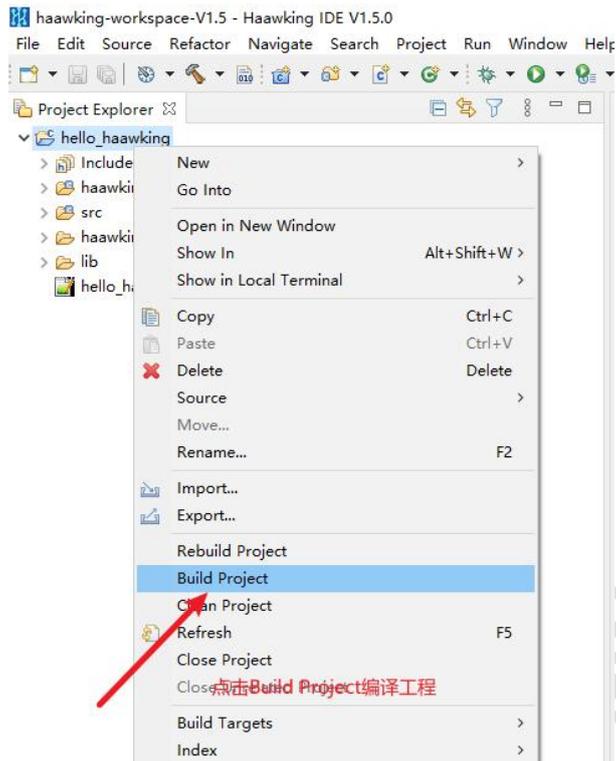
创建成功后效果如下图所示



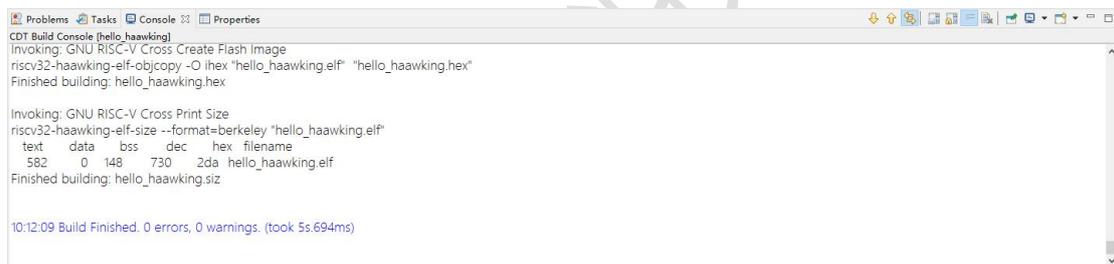
## 2.3 编译

### 2.3.1 整个工程编译

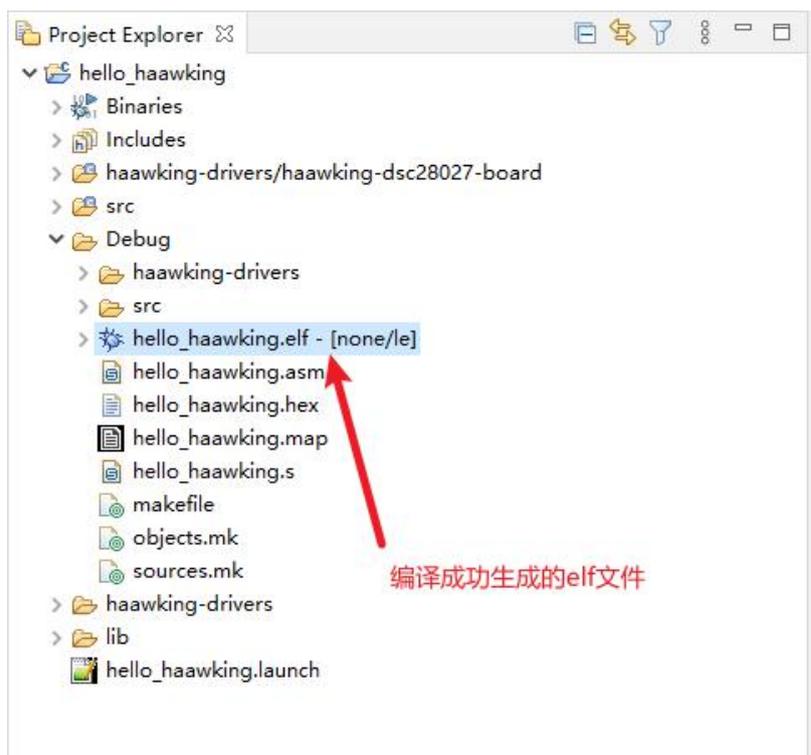
右键单击工程名，选择 Build Project 编译整个工程。如果编译出现错误，请检查环境变量是否设置正确。



编译完成后，如果没有错误，则会出现如下提示：

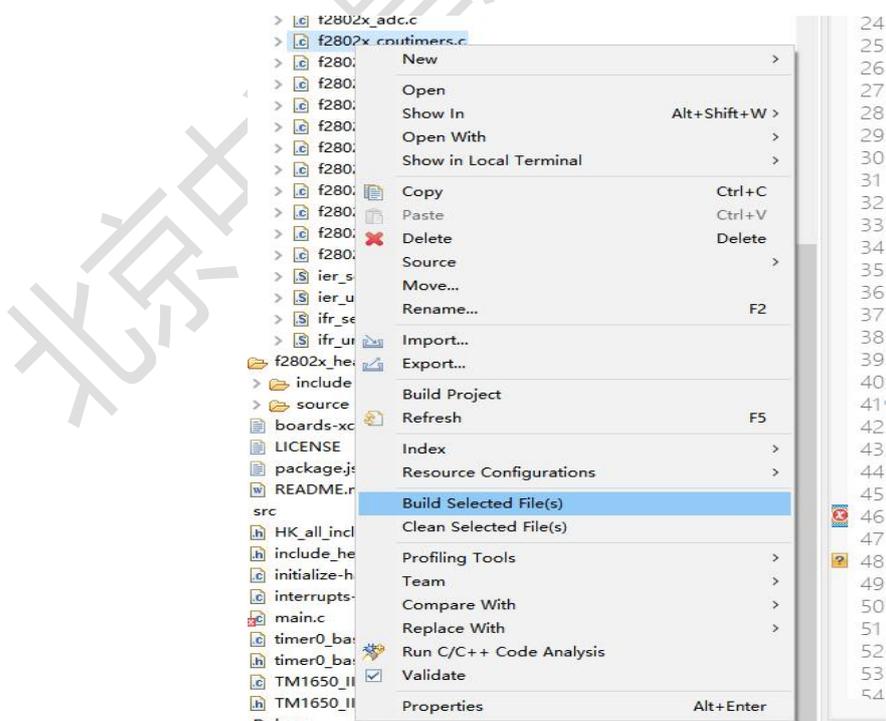


编译成功后，在“hello\_haawking->Debug”目录下，会出现可执行文件 hello\_haawking.elf。示例程序的路径为：



### 2.3.2 单个文件编译

在所需编译的文件处鼠标右击，然后左键点击“Build Selected File(s)”即可编译选中文件。



## 2.4 如何生成库文件

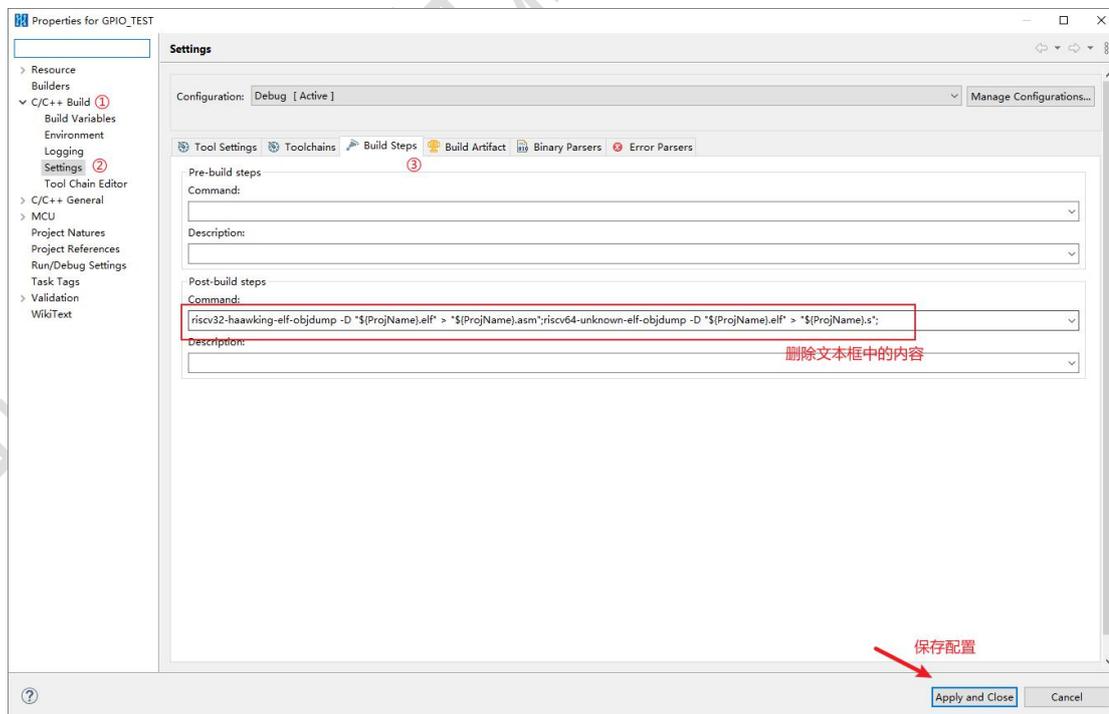
在 Haawking IDE，工程编译后，默认会在工程的 Debug 目录下生成 elf 格式的文件，该格式的文件可用于调试和烧入，如果用户需要生成静态库文件，可按照下方步骤操作。

### 2.4.1 前置工作

如果用户使用的是 Haawking IDE1.5.4 版本前创建的工程，首先需要按照下面的操作进行配置。

1. 选中项目，右键弹出菜单，点击最下方的“Properties”选项。
2. 依次选择 C/C++ Build->Settings->Build Steps 选项。
3. 删除 Post-build steps->Command 文本框中的内容。
4. 点击 Apply and Close 按钮保存配置。

注: 用户也可以按照下方的图片进行配置。

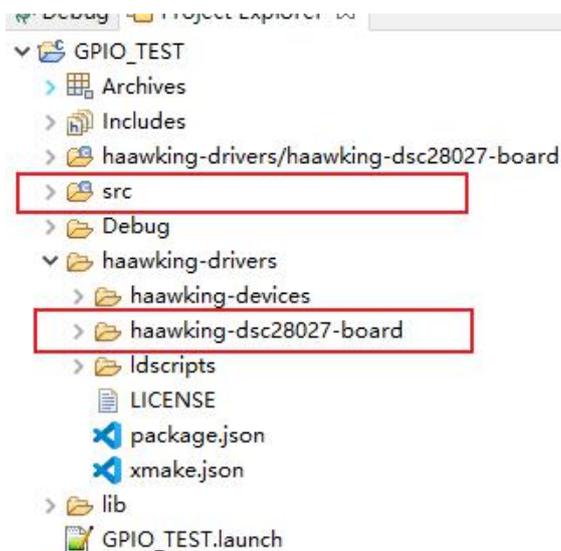


使用 IDE 生成静态库时，工程的 Source Location 中配置的目录中的所有 .c 和 .S 文件均会被包含在内。

而使用 Haawking IDE 建立的工程，默认会包含下方的两个 Source Location 目录，因此，如果用户生成的静态库不想包含驱动库，那么需要在 Source Location 中删除驱动库目录。

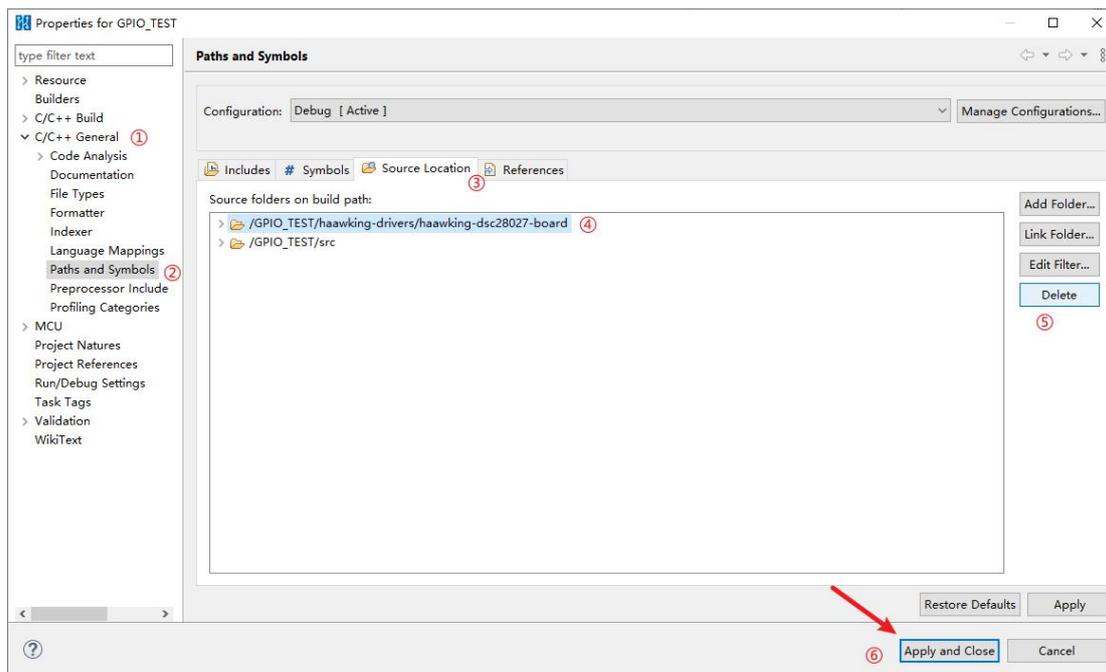
(1) src 目录

(2) Haawking-drivers/haawking-dsc28027-board 目录(驱动库所在目录)



下面演示如何删除 Source Location 中的目录：

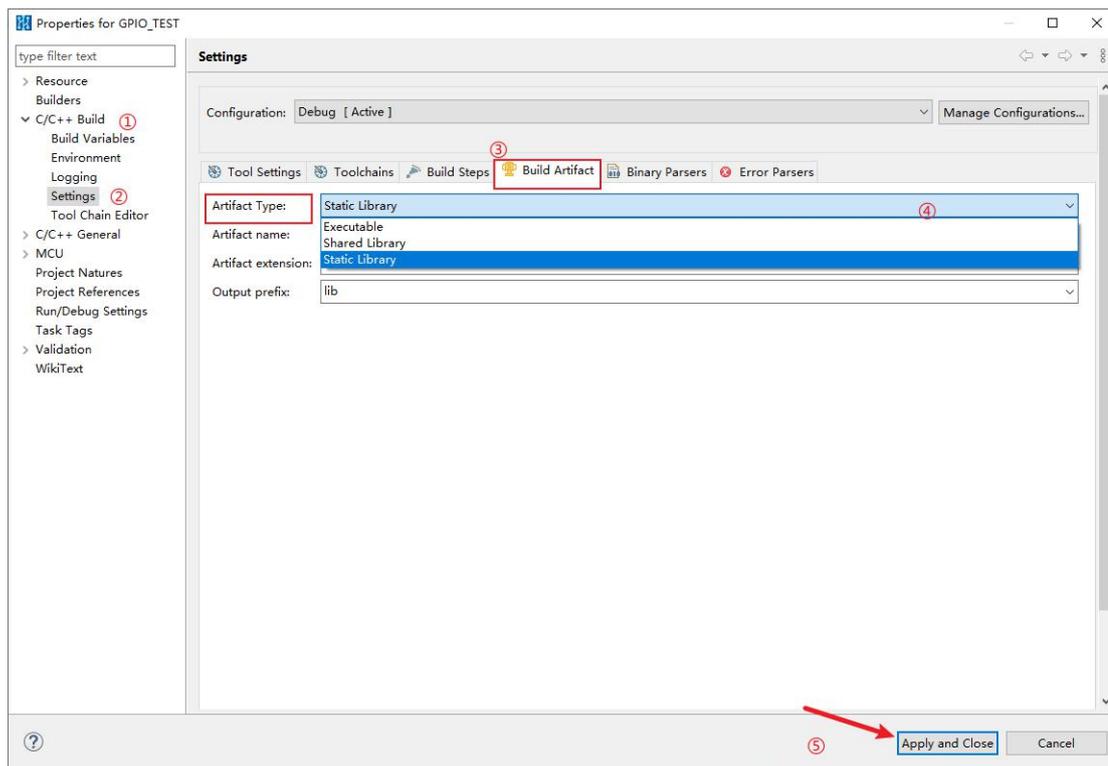
1. 选中项目，右键菜单，点击最下方的“Properties”按钮，打开项目的“Properties”面板。
2. 依次选择 C/C++ General->Paths and Symbols->Source Location
3. 在 Source Location 面板中选中待删除的路径，点击右侧的 Delete 按钮，最后点击“Apply and Close”按钮保存配置即可。
4. 此时，在 Source Location 中删除的目录中的文件就不会被打包进静态库中。



## 2.4.2 生成静态库

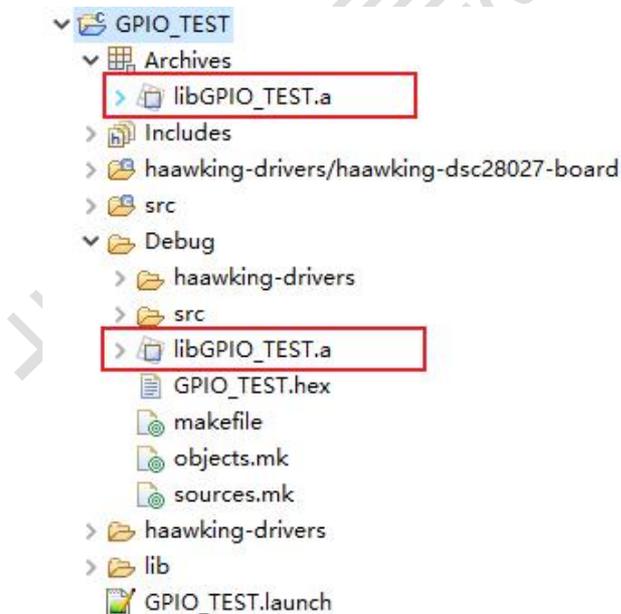
1. 选中项目，右键弹出菜单，点击最下方的“Properties”选项，打开项目的“Properties”面板。
2. 依次选择 C/C++ Build->Settings->Build Artifact, 打开“Build Artifact”面板。
3. 在“Build Artifact”面板中，点击“Artifact Type”的下拉列表，选择“Static Library”选项，最后点击“Apply and Close”保存配置即可。

注: 用户可以按下方图片操作



4. 配置完成后，用户重新编译工程，在工程下的“Archives”选项或 Debug 目录下，即可看到.a 结尾的库文件。

注：“Archives”下的.a 文件与 Debug 目录下的.a 文件是同一个，“Archives”下的.a 文件是 Debug 目录下的.a 文件的映射。



### 2.4.3 使用静态库

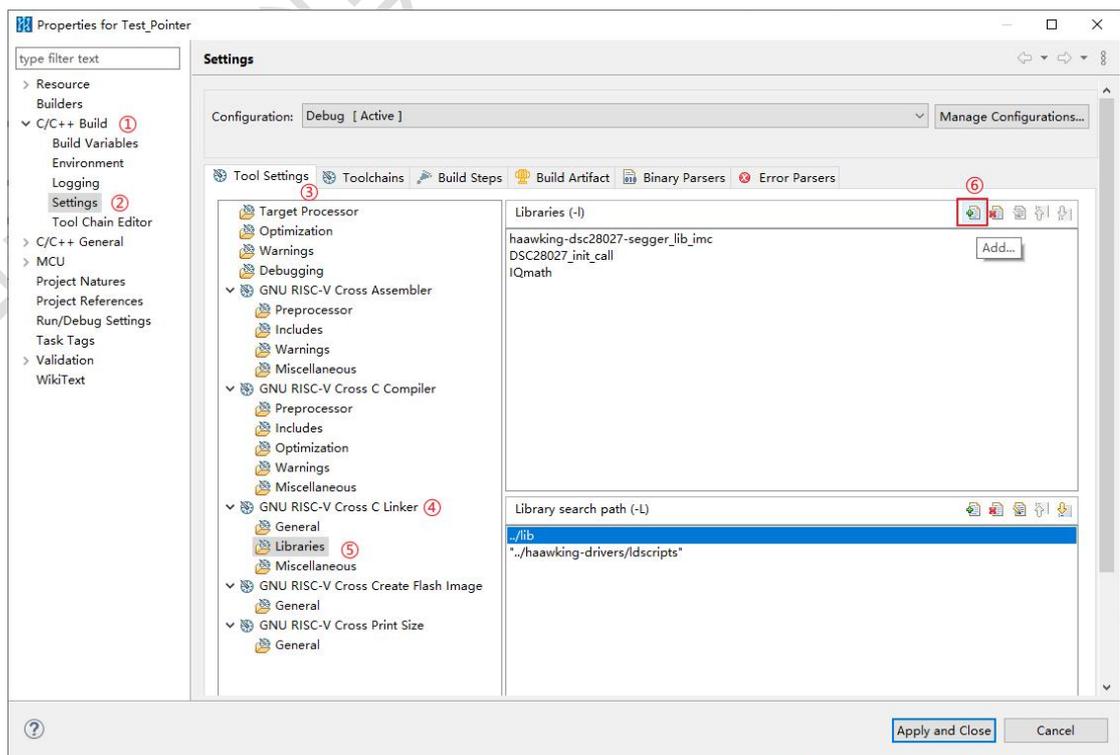
用户生成静态库后,如果想要使用该静态库,需要按照下方的步骤进行配置。

1. 将生成的库文件放至工程下的 lib 文件夹中。



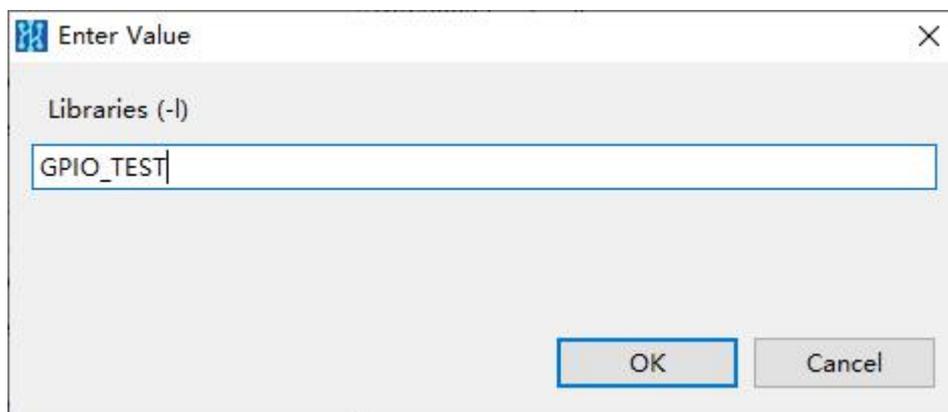
2. 打开项目的“Properties”面板，选择 C/C++ Build->Settings->Tool Settings，打开“Tool Settings”面板。

3. 点击 GNU RISC-V Cross C Linker->Libraries，在右侧第一个面板中点击  按钮。



4. 在弹窗中输入静态库的名称, 点击 OK 即可。

注: 如果要添加名为“libGPIO\_TEST.a”的库, 那么在弹窗中输入的名称只需要“GPIO\_TEST”, 即忽略 lib 前缀和.a 后缀。



5. 最后点击“Apply and Close”按钮保存配置即可。

## 2.5 编译优化介绍

### 2.5.1 -O1 级别优化

- 1、删除无用参数
- 2、合并冗余指令
- 3、内联函数
- 4、有条件地消除无用库调用
- 5、循环不变量: 找出循环不变量并使用“代码移动 (code motion)”将其移出循环体
- 6、删除无用循环
- 7、展开循环
- 8、循环拆分
- 9、循环简化
- 10、删除冗余的指令
- 11、删除无用位
- 12、循环旋转, 修改循环结构, 使其便于编译器优化

## 2.5.2 -O2 级别优化

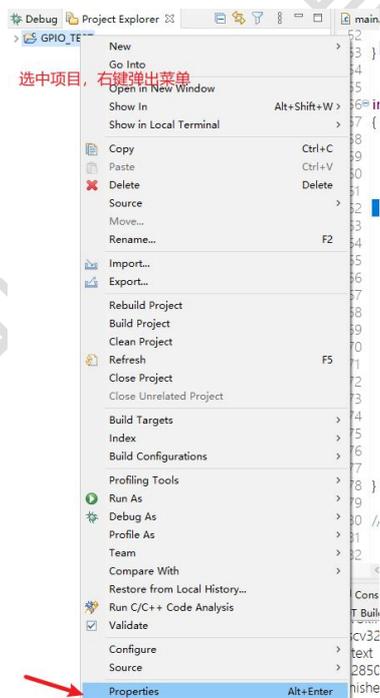
在 O1 的基础上，增加

- 1、删除未使用的异常处理信息
- 2、尾调用优化
- 3、重新组合表达式
- 4、删除无用的全局变量
- 5、合并重复的全局常量
- 6、剥离未使用的函数原型

## 2.6 生成 bin 格式文件

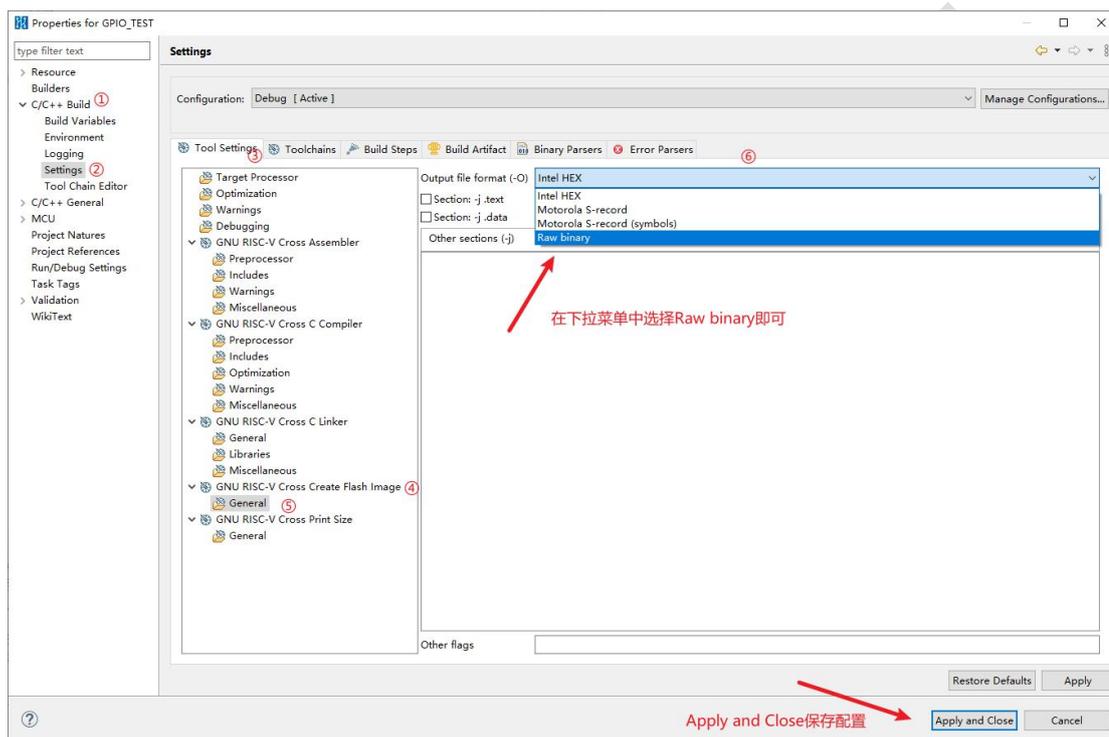
在 Haawking IDE 中，工程编译后，在 Debug 目录下默认会生成 hex 格式的文件，如果用户需要 bin 格式的文件可以按下方步骤操作。

1. 选中项目，右键弹出菜单， 点击底部的“Properties”选项，弹出项目的“Properties 面板”。

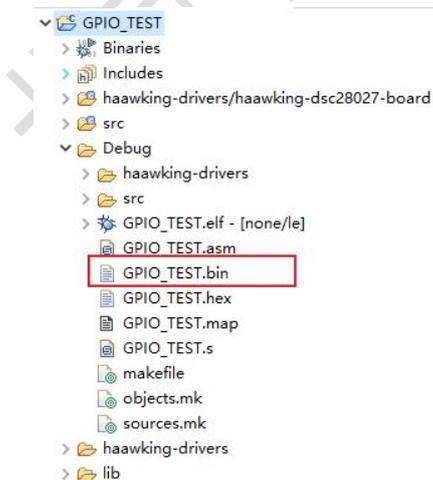


2. 在弹出的菜单中按下方文字提示或图片操作即可。

- (1) 点击左侧菜单中的 C/C++ Build->Settings
- (2) 在菜单右侧选择 Tool Settings->GNU RISC-V Cross Create Flash Image->General
- (3) 在右侧的面板中, 点击 Output file format (-O)选项的下拉菜单, 选择 Raw binary 按钮。
- (4) 点击下方的“Apply and Close”按钮即可。



3. 重新编译工程, 在工程下的 Debug 文件夹中即可看到“.bin”结尾的文件。



## 2.7 下载功能

Haawking IDE 仿真器下载功能暂不支持启动调试后二次下载程序。

### 2.7.1 前置工作

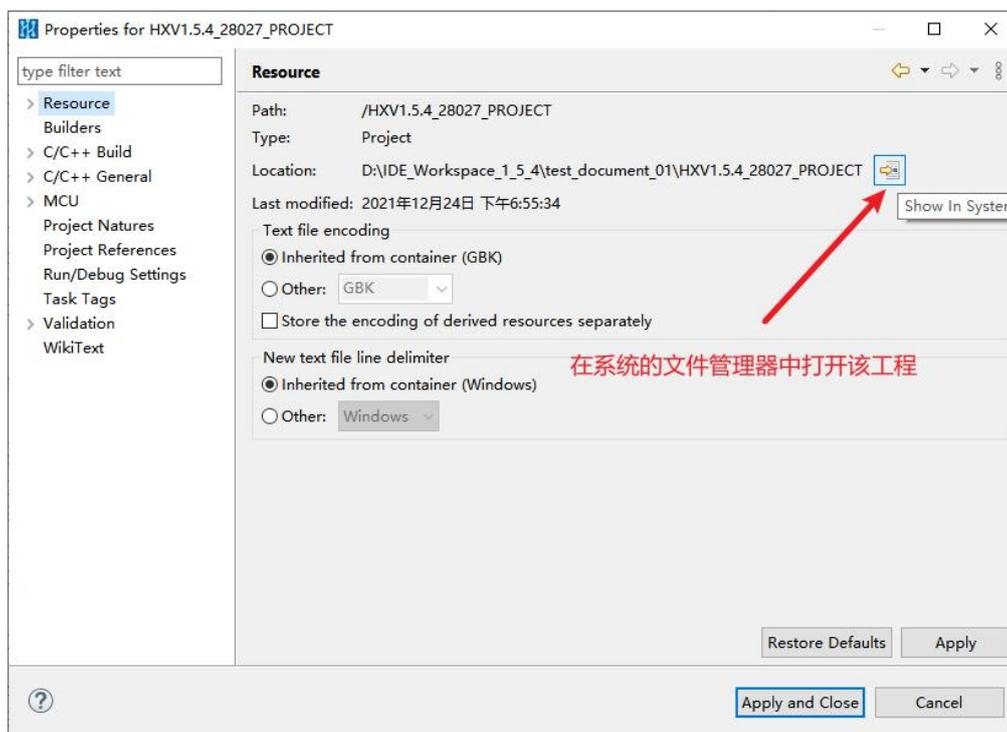
由于该功能并不兼容 Haawking IDE 1.5.3 版本前创建的工程，因此，如果用户希望在旧版工程上使用该功能，首先需要按照下面的步骤更新工程中的配置文件。

1. 首先使用 V1.5.3 版本以后的 IDE 创建一个与旧工程芯片型号一致的工程。

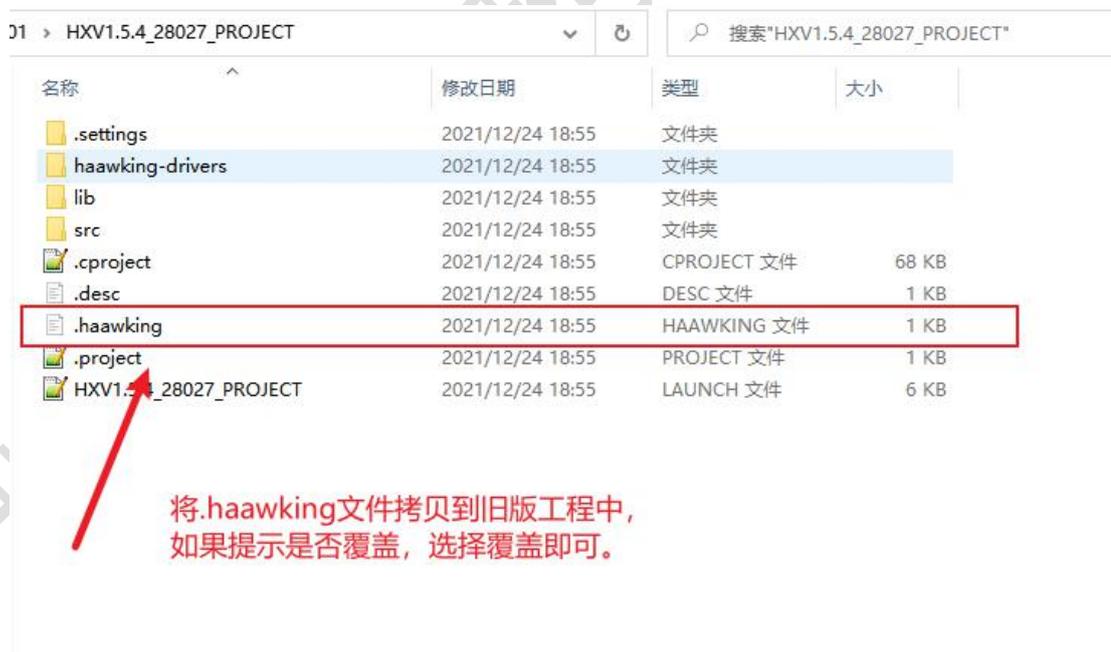


2. 选中该项目，右键弹出菜单，点击“Properties”选项，打开“Properties”面板。

3. 选择左侧的“Resource”选项，点击右侧的  按钮，在文件管理器打开项目。



4. 进入项目目录，拷贝项目中的.haawking 文件，将其复制到旧工程中，如果提示是否覆盖，点击覆盖即可。

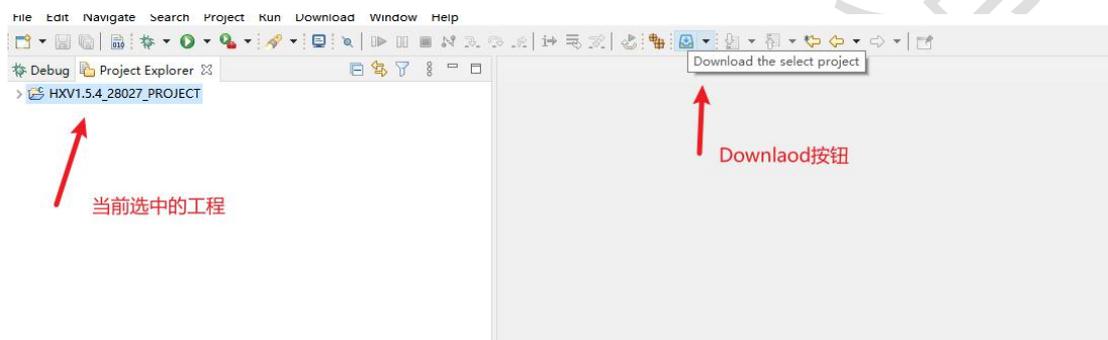


## 2.7.2 使用 IDE 下载程序

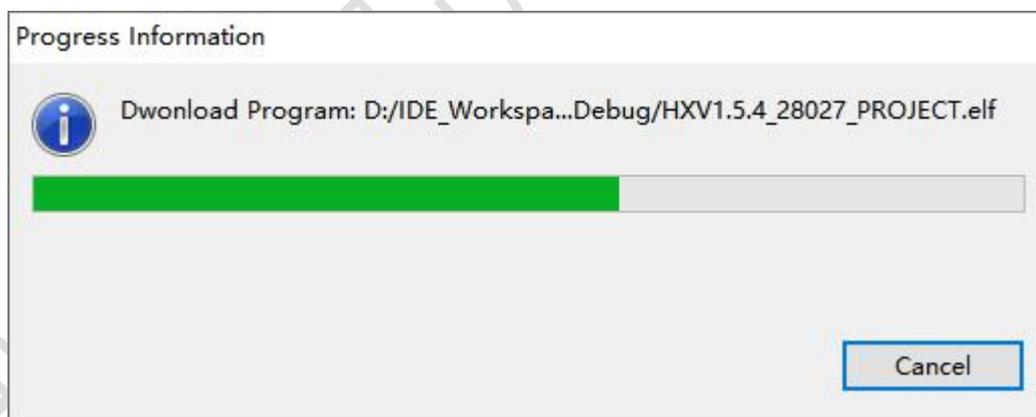
### 2.7.2.1 直接下载工程中的程序

如果用户希望直接下载工程中的程序，那么首先选中待下载的工程，然后直接点击菜单栏上的  按钮，即可开启下载流程。

注：默认下载工程中 Debug 目录下的 elf 文件，如果用户需要下载.hex 格式文件，请看 2.7.2.2 小节。



点击按钮后，IDE 中会出现一个进度条提示下载进度，当下载完成后，进度条会自动关闭，如果下载途中没有任何警告，则代表下载成功。



### 2.7.2.2 下载指定位置的程序

用户如果希望下载任意位置的程序(.elf 文件或.hex)文件，首先需要有一个工程作为下载时的配置文件所在工程，因此用户可以新建一个空的 **FLASH** 工程作为配置文件所在工程。

注：待下载文件所在位置不允许包含中文字符

首先点击菜单栏上下载按钮的 ▾ 选项，然后点击“Select Program to Download”选项。



在出现的弹窗中选择待下载的文件，支持.elf 格式文件和.hex 格式文件的下载，然后选择确定按钮即开启下载流程， 下载过程中如果未出现任何警告，则代表程序下载成功。

如果用户希望查看使用“memory”视图查看芯片内容，可以看文档 3.9 小节，使用“Debug Without Download”功能。

**注：下载文件所在位置不要包含中文字符。**

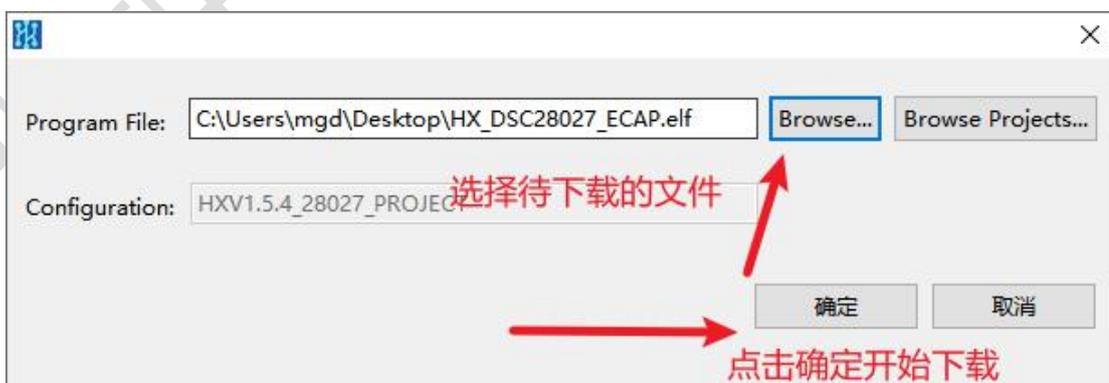
下载面板解释：

Program File: 下载文件所在路径

Browse...: 选择文件资源管理器中的文件

Browse Projects...:选择工作空间中的文件

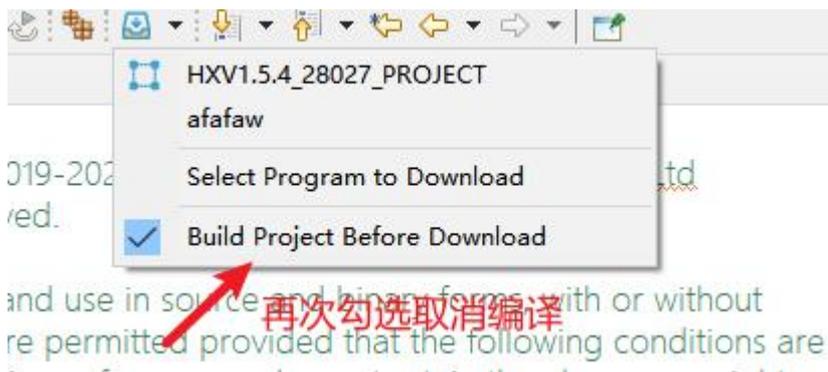
Configuration: 下载时使用的配置文件所在工程，配置文件用于解锁芯片、确定芯片型号等。



### 2.7.2.3 关闭下载前编译工程

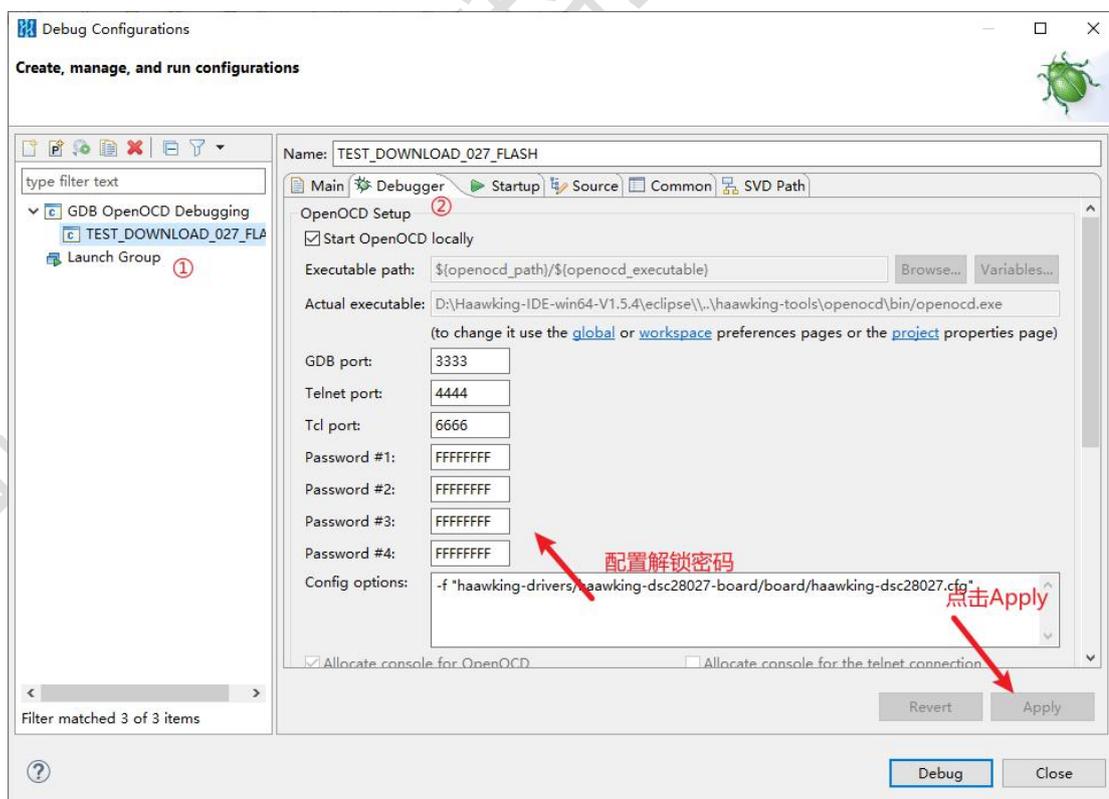
下载功能默认会勾选  Build Project Before Download 选项，即下载前

先编译工程，如果用户不需要编译工程，可以再次点击该按钮，取消下载前编译。



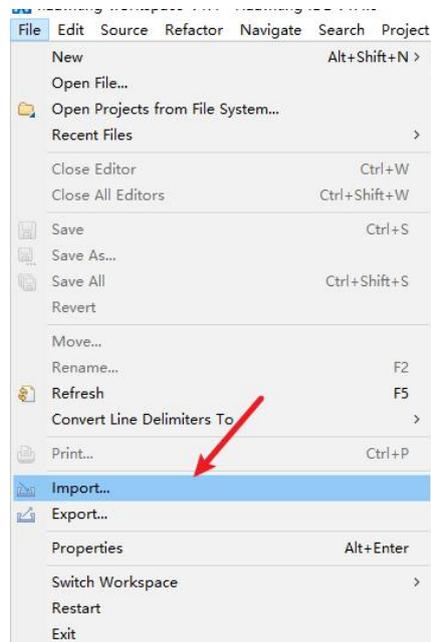
### 2.7.2.4 配置芯片解锁密码

1. 首先右键工程，选择 Debug as->Debug Configurations，打开工程的调试面板
2. 选择工程的调试配置，激活“Debugger”选项卡，在“Debugger”选项卡中配置密码后，点击“Apply”按钮使配置生效，点击“Close”按钮关闭调试配置面板。

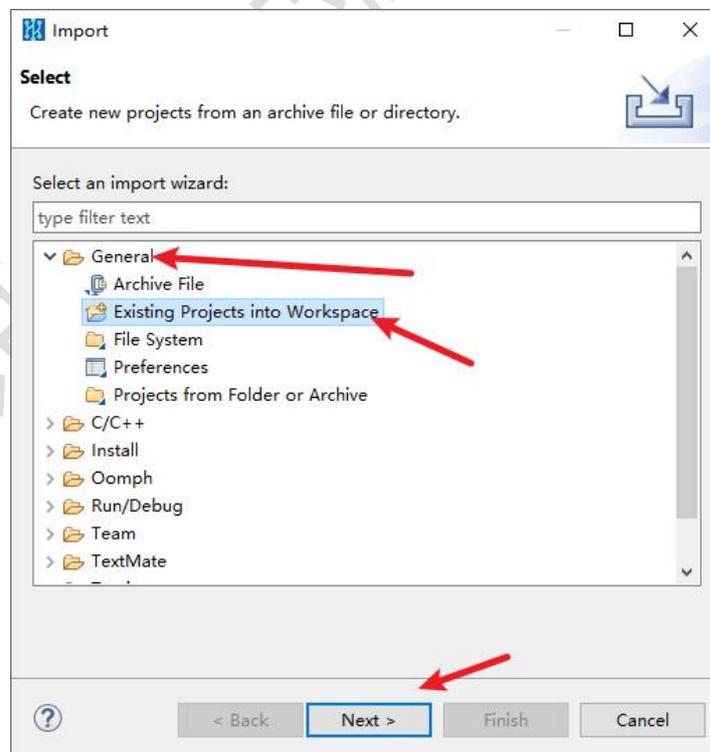


## 2.8 导入工程

在菜单栏点击“File->import”



在“Import”面板中，选择“General->Existing Projects into Workspace”选项，点击“Next”

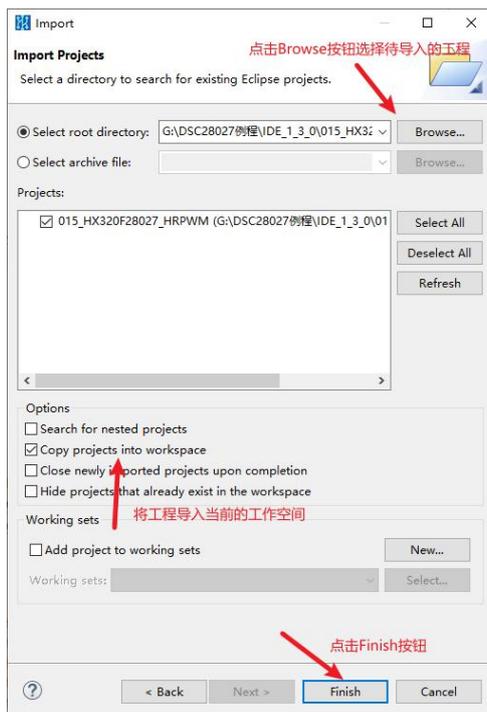


由于待导入的工程有文件夹、压缩包两种格式，因此导入的方式也有所不同，

下面分别就导入文件夹格式工程和压缩包工程进行介绍。

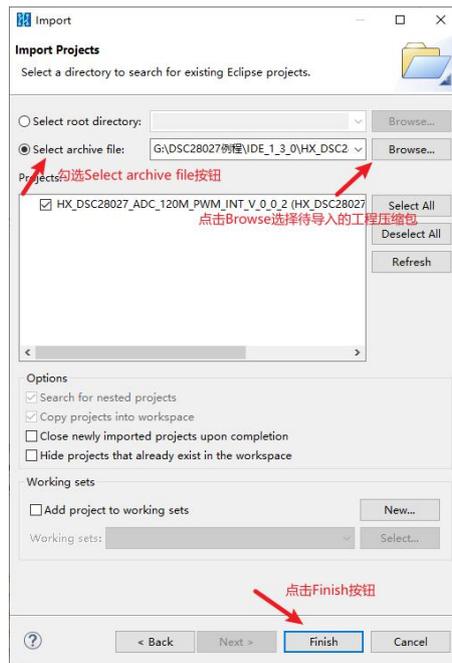
导入文件夹格式的工程时，在 Import 面板中点击“Browse”按钮选择要导入的工程，建议勾选“Copy projects into workspace”按钮，点击“Finish”按钮即可。

注：如果不勾选“Copy projects into workspace”按钮，请确保工程所在的位置不包含中文字符。



导入压缩包格式的工程时，在 Import 面板中勾选“Select archive file”按钮，然后通过“Browse”按钮选择待导入的工程，点击“Finish”按钮即可。

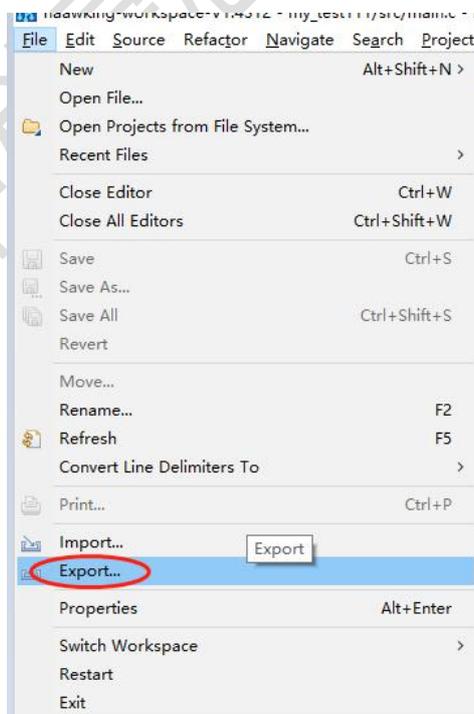
注：压缩包格式的工程仅支持.zip 格式。



## 2.9 导出工程

### 2.9.1 导出压缩包格式工程

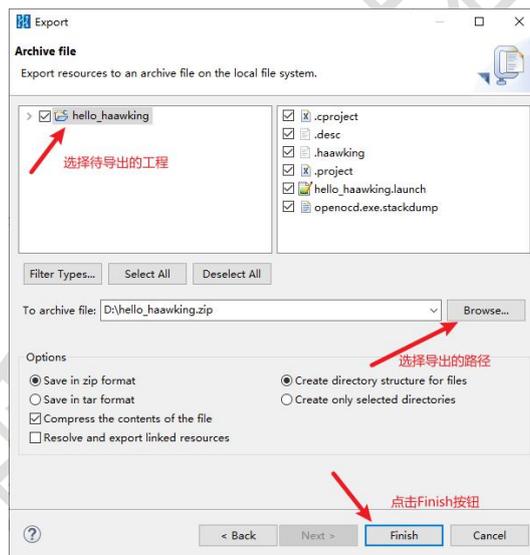
1、在菜单栏点击“File -> Export”



2、在“Export ”面板中，选择“General->Archive File”选项，点击“Next”

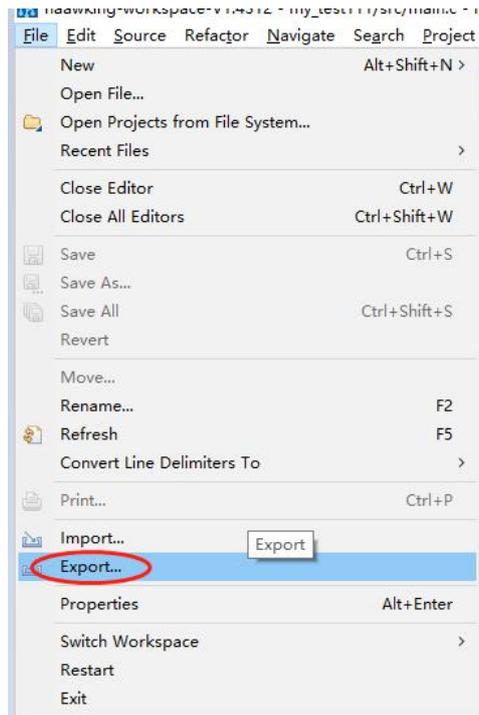


3、在新面板中选中待导出的工程，然后点击“Browse”按钮选择要导出的位置，最后点击“Finish”按钮即可。

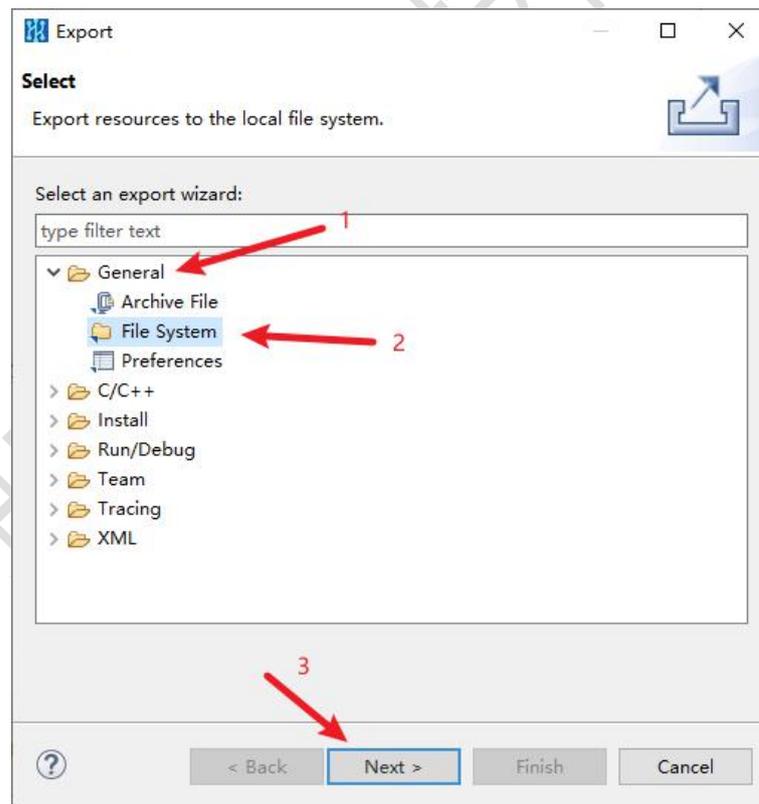


## 2.9.2 导出文件夹格式工程

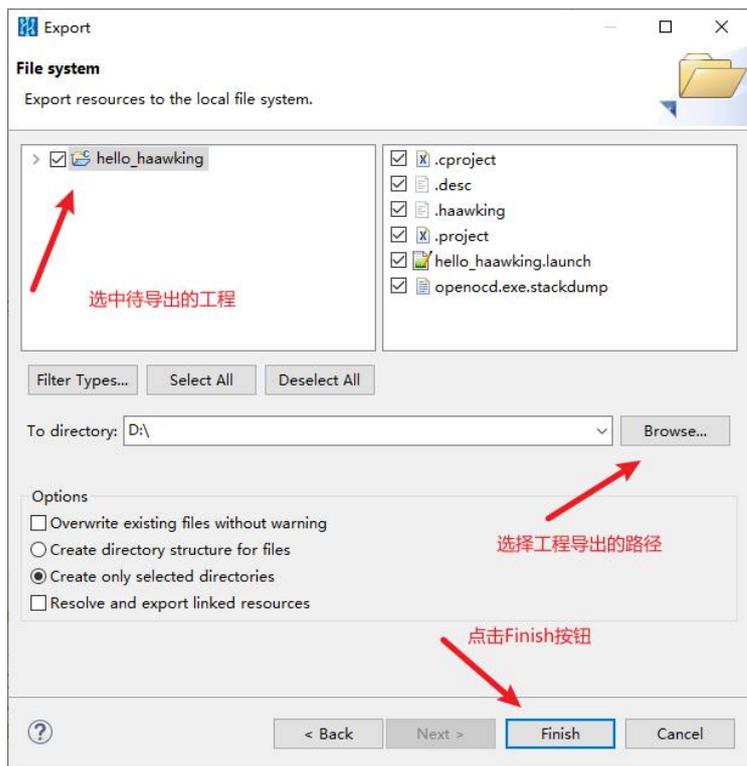
在菜单栏点击“File -> Export”



2、在“Export ”面板中，选择“General->File System”选项，点击“Next”



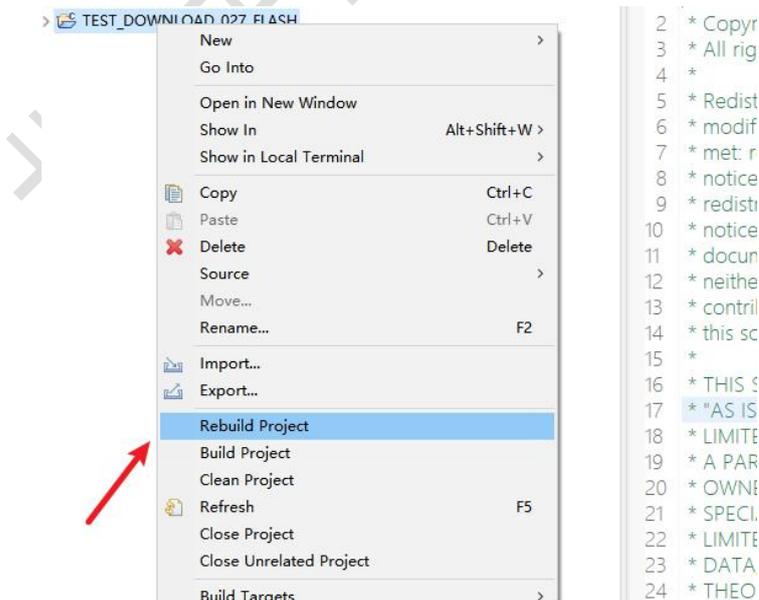
3. 在新面板中选中待导出的工程，然后点击“Browse”按钮选择要导出的位置，最后点击“Finish”按钮即可。



## 2.10使用 Rebuild Project 功能

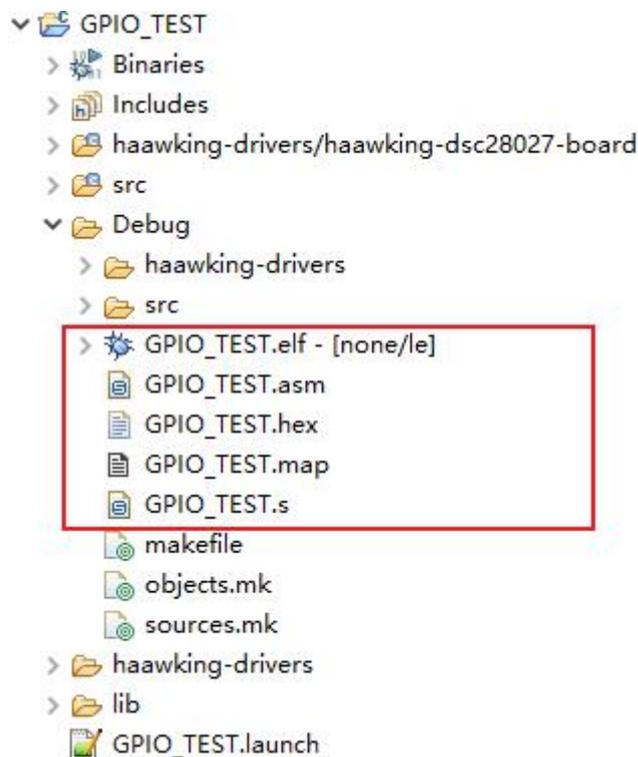
Rebuild Project 功能是 Clean Project 功能与 Build Project 功能的封装，使用 Rebuild Project 时，IDE 会先清除工程 Debug 目录下上次编译生成的相关文件，然后执行全量编译，重新生成可执行文件、反汇编文件等。

1. 选中一个工程，右键弹出菜单，点击 Rebuild Project 选项即可。



## 2.11 Debug 目录下的文件介绍

在 Haawking IDE 中，工程编译完毕后，默认会在 Debug 目录下生成多个类型的文件，下面进行介绍。



.elf 文件：elf 后缀的文件程序的调试和烧入。

.asm 文件：asm 后缀的文件是工程的反汇编文件，其中包含昊芯自定义的汇编指令。

.hex 文件：hex 后缀的文件是指 INTEL HEX 格式文件，用于程序的烧入。

.map 文件：map 后缀的文件是编译器经过编译后，生成的程序中信息的映射文件，该文件中包含程序的内存布局、符号的地址等信息。

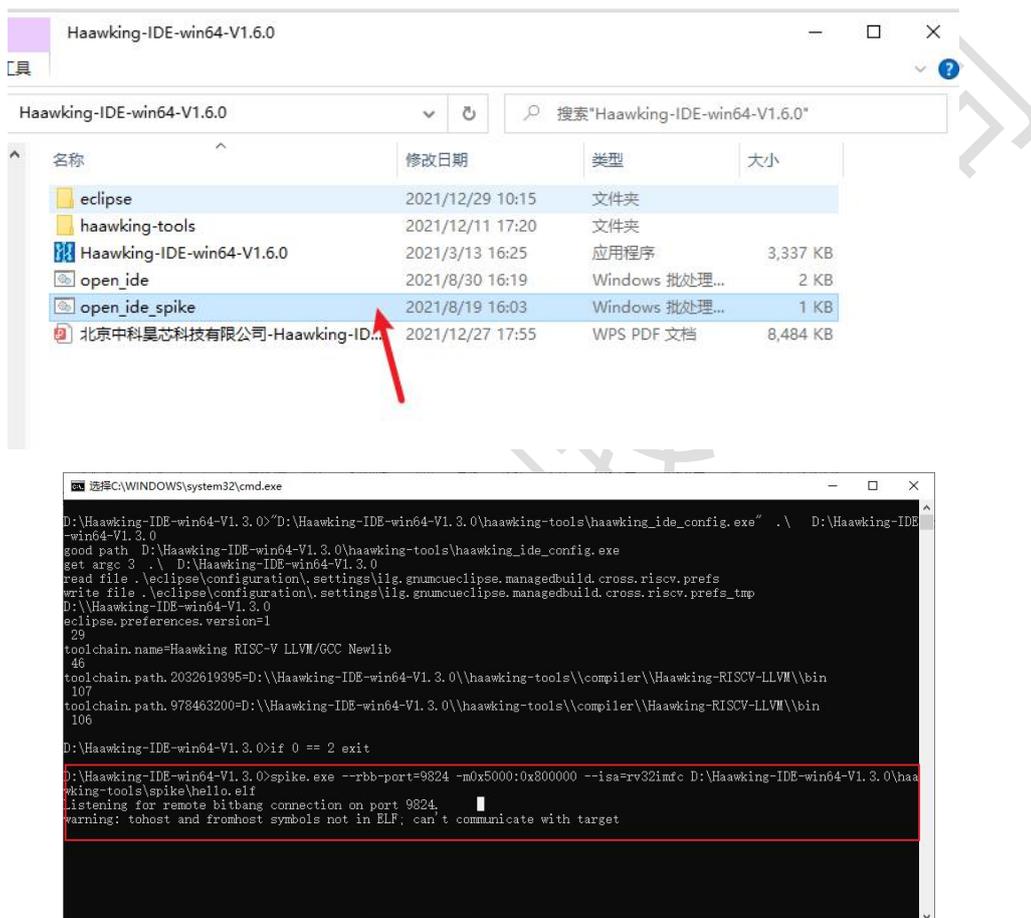
.s 文件：s 后缀的文件同样是工程的反汇编文件，但无法显示昊芯自定义的汇编指令。

## 2.12 用 Spike 模拟器仿真程序

对于没有开发板的客户，可以使用 Haawking Spike Simulator 来进行简单程序的仿真，目前只限于 CPU 程序，不包括外设。

Spike 工程的新建、编译与现有芯片一样，这里就不再进行过多介绍。在调试的时候，当前版本的 IDE（V1.5.1 版本），需要在每次调试之前，手动执行一个 bat 脚本，启动 Spike 模拟器；同时建议选择 RAM 工程（其实两者没啥区别）。

Spike 模拟器的启动脚本在 Haawking IDE 的根目录下，双击即可，启动之后效果如下所示。



然后，回到 Haawking IDE 中，参考第三章的调试过程，就可以进行仿真和调试了。

```

1435
1436 float SIN[360]={};
1437 void FPU_sin_isa();
1438 int main(void)
1439 {
1440     float out;
1441     FPU_sin_isa();
1442
1443     int i;
1444     for(i=0; i<360; i++)
1445     {
1446         //printf("%d %f\n",i,SIN[i],sinf(Rad[i]));
1447     }
1448
1449     return 0;
1450 }
1451
1452 // -----
1453
1454

```

Console Registers Problems Executables Debugger Console Memory

```

test-debug-occd [GDB OpenOCD Debugging]
Info : only one transport option; autoselect 'jtag'
Info : Initializing remote_bitbang driver
Info : Connecting to localhost:9824
Info : remote_bitbang driver initialized
Info : This adapter doesn't support configurable speed
Error: Trying to use configured scan chain anyway...
Warn : Bypassing JTAG setup events due to errors
Started by GNU MCU Eclipse
Error: Trying to use configured scan chain anyway...
Warn : Bypassing JTAG setup events due to errors

```

```

1435
1436 float SIN[360]={};
1437 void FPU_sin_isa();
1438 int main(void)
1439 {
1440     float out;
1441     FPU_sin_isa();
1442
1443     int i;
1444     for(i=0; i<360; i++)
1445     {
1446         //printf("%d %f\n",i,SIN[i],sinf(Rad[i]));
1447     }
1448
1449     return 0;
1450 }
1451
1452 // -----
1453
1454

```

Console Registers Problems Executables Debugger Console Memory

```

test-debug-occd [GDB OpenOCD Debugging]
Info : only one transport option; autoselect 'jtag'
Info : Initializing remote_bitbang driver
Info : Connecting to localhost:9824
Info : remote_bitbang driver initialized
Info : This adapter doesn't support configurable speed
Error: Trying to use configured scan chain anyway...
Warn : Bypassing JTAG setup events due to errors
Started by GNU MCU Eclipse
Error: Trying to use configured scan chain anyway...
Warn : Bypassing JTAG setup events due to errors

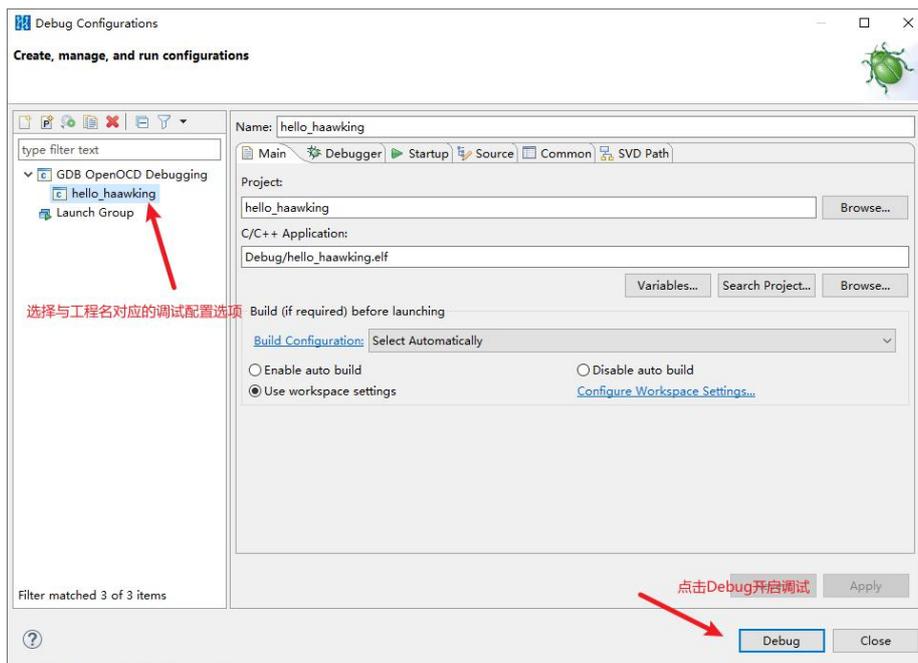
```

```

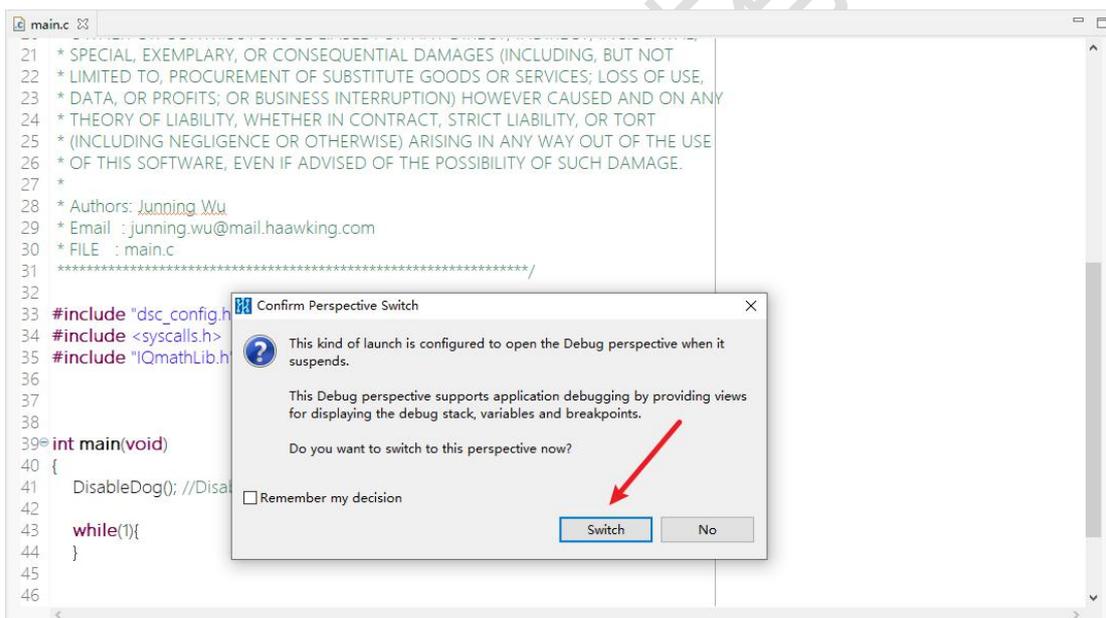
45 #####Compute#####
46 _sin:
47     addi t6, t0, 0           #t6 = t0
48     addi a0, t1, 0          #a0 = t1
49     flw ft10, 0(t2)         #load RadTable - Radian-ft10
50     addi t2, t2, 4          #RadTable address +4
51     fmul.s ft5, ft10, ft1    #ft5= a = Radian*(2.0/(2.0*pi))#float a
52     fcvt.w.s s5, ft5, rtz   #float int s5=(int)a
53     #fcvt.s.w ft6, s5      #int-float ft6=a.0
54     ftrunc.s ft6, ft5      #------isa-----#
55     andi s5, s5, 0x1ff     #k = 0x1fff & (int)a
56     slli s5, s5, 0x2      #s5=2*k
57     #fsub.s ft6, ft5, ft6   #ft6 = b = ft5-ft6 = a -a.0
58     #ffrac.s ft6, ft5     #------isa-----#
59     fmul.s ft6, ft6, ft2   #ft6 =x = bx(2*pi)/512
60     add t6, t6, s5        #t6=t6+k
61     flw ft7, 0(t6)        #FPUsinTable[k]
62     add a0, a0, s5
63     flw ft8, 0(a0)        #FPUcosTable[k]
64     fmul.s ft9, ft3, ft6   #ft9=0.166667*x
65     fmul.s ft9, ft9, ft8   #ft9 = 8.166667*x*FPUcosTable[k]
66     fmsub.s ft9, ft4, ft7, ft9 #ft9 =-0.5x*FPUsinTable[k]-ft9
67     fmadd.s ft9, ft9, ft6, ft8 #ft9 = ft9*x+FPUcosTable[k]
68     fmadd.s ft9, ft9, ft6, ft7 #ft9 = ft9*x+FPUsinTable[k]
69
70 ##### output #####
71     fsw ft9, 0(t3)        #result store to SIN[]
72     addi t3, t3, 4
73     #blcne t4, -84
74     addi t5, t5, 1
75     bne t4, t5, _sin
76
77     flw ft10, 4(sp)
78     flw ft9, 8(sp)
79     flw ft8, 12(sp)
80     flw ft7, 16(sp)

```





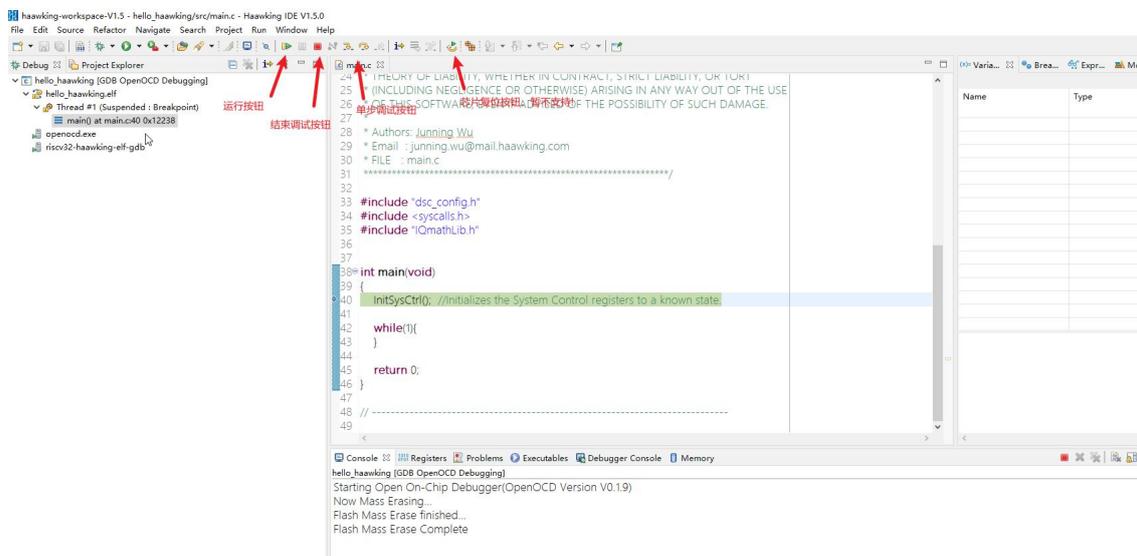
如出现下图所示的界面时，单击“Switch”，选择允许调试透视图的切换。



当“Console”视图中显示出 OpenOCD 相关信息（如下图 1）时并且跳转到仿真界面（如下图 2）



## 3.2 调试模式工具栏介绍



 Resume 按钮的作用为全速运行程序。

 Suspend 按钮的作用为暂停程序运行。

 Terminate 按钮的作用为终止调试。

 Step Info 按钮的作用为单步调试，当遇到函数调用时，会进入函数体内部执行。

 Step Over 按钮的作用也是单步调试，但如果遇到函数调用，Step Over 按钮不会进行函数内部，而是把函数调用作为一步来执行。

 Step Return 按钮的作用是，当使用 Step Info 按钮进入函数调用的内部执行时，执行完该函数，并且返回上一层函数。

 Restart 按钮作用是在调试时复位芯片，使用该功能时有些注意事项，建议用户使用前先阅读 2.5.3 小节的内容。

 Instruction Stepping Mode 按钮: 该按钮用于打开汇编单步调试模式。

## 3.3 断点的使用及断点视图介绍

### 3.3.1 如何添加断点

用户启动调试后，在行号前双击即可添加断点，如下图所示。

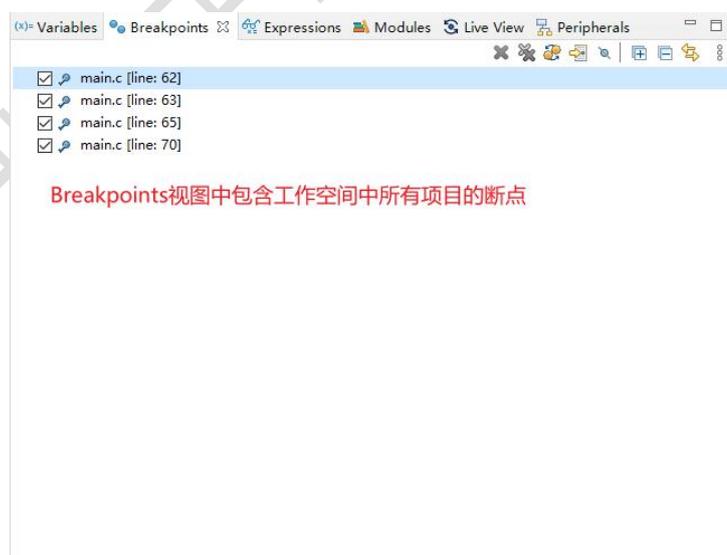


### 3.3.2 如何删除断点

将鼠标移动到断点上，再次双击即可删除该断点。

### 3.3.3 使用 Breakpoints 视图管理断点

在 Haawking IDE 中，“Breakpoints”视图负责管理所有项目的断点。



### 3.3.4 Breakpoints 视图功能按钮介绍

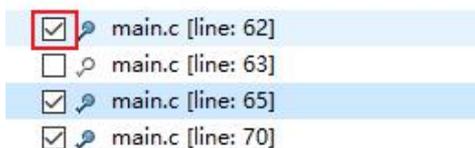
 按钮: 该按钮用户删除选中的断点。

 按钮: 删除 Breakpoints 视图中的所有断点。

 按钮: 跳转到断点所在的文件。

 按钮: 点击该按钮后, 所有的断点都会被跳过, 即程序并不会在断点处停止。

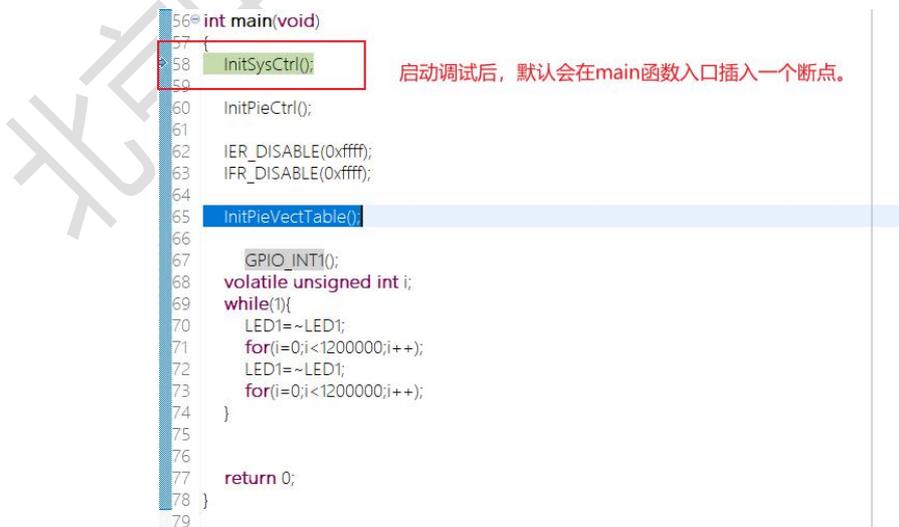
复选框: 断点前的  复选框用于设置断点的生效/失效状态, 当取消选中复选框时, 断点处于失效状态。



### 3.3.5 FLASH 工程使用断点的注意事项

因为在 Flash 工程中, 只允许用户最多加入两个断点 (硬件断点), 但是在进入 Debug 模式后, 默认会在进入 “main()” 函数的第一个函数的位置加入断点, 因此在进入 Debug 模式之前, 工程中最多存在一个生效的断点。

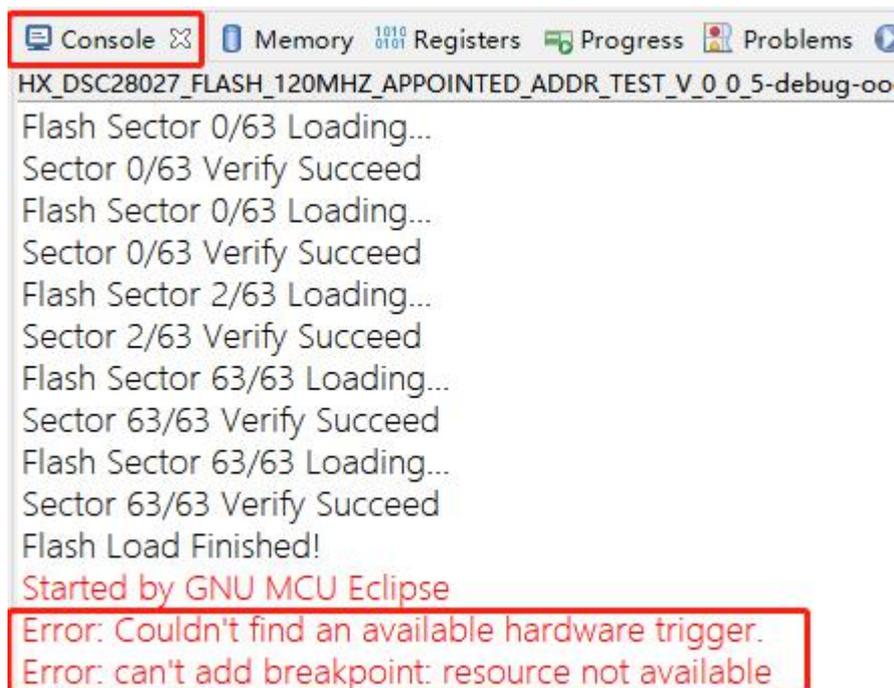
断点的位置见下图:



因此，在 FLASH 项目启动调试前，用户需要在 Breakpoints 视图中管理工程已有的断点，下面提供几种方式。

1. 在 Breakpoints 视图中删除多余的断点或删除全部断点。
2. 在 Breakpoints 视图中点击  按钮，使多余的断点处于失效状态。

**注意：**如果在 Debug 模式下，再加入第三个断点，或进入 Debug 模式之前就断点数量就已经超过了三个，那么控制台就会报如下错误：



```

HX_DSC28027_FLASH_120MHZ_APPOINTED_ADDR_TEST_V_0_0_5-debug-oo
Flash Sector 0/63 Loading...
Sector 0/63 Verify Succeed
Flash Sector 0/63 Loading...
Sector 0/63 Verify Succeed
Flash Sector 2/63 Loading...
Sector 2/63 Verify Succeed
Flash Sector 63/63 Loading...
Sector 63/63 Verify Succeed
Flash Sector 63/63 Loading...
Sector 63/63 Verify Succeed
Flash Load Finished!
Started by GNU MCU Eclipse
Error: Couldn't find an available hardware trigger.
Error: can't add breakpoint: resource not available
  
```

导致芯片无法再继续运行，需要点击关闭按钮  关掉进程，再移除所有断点后重新进入 Debug 模式。

### 3.4 如何使用 Restart 功能复位芯片

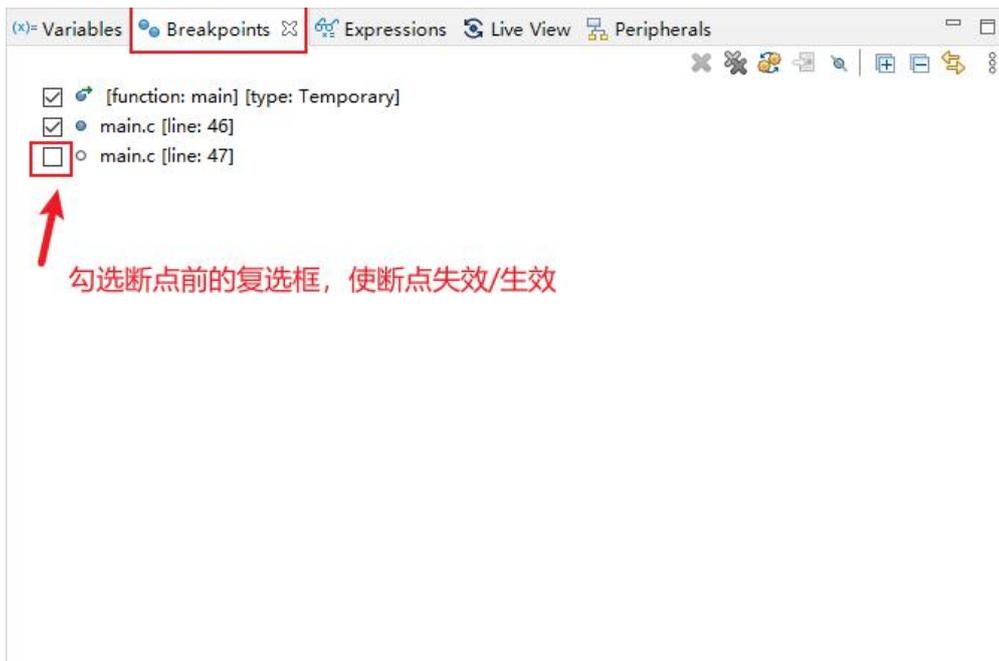
在调试过程中，用户可通过菜单栏上的  Restart 按钮来复位芯片，使 PC 寄存器回到程序的入口位置，但如果用户正在调试的工程是一个 FLASH 工程，那么有一个注意事项，下面进行介绍。

#### 3.4.1 FLASH 工程使用 Restart 功能

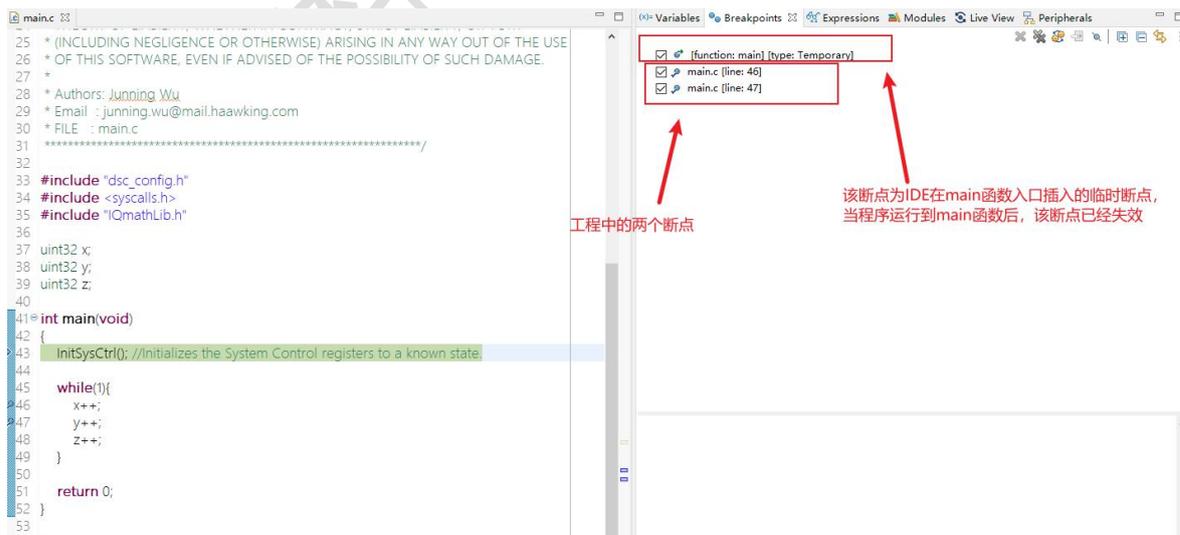
FLASH 工程目前最多同时支持两个断点，但是在使用  前，项目中最多只

允许一个断点生效，这是因为在点击  后，Haawking IDE 会向 main 函数的第一行插入一个断点，如果此时项目中已经存在两个生效断点，那么断点数量就超过了限制，将无法正常使用调试，只能重新启动调试。

因此，用户在 FLASH 工程中使用  前，需要在 Breakpoints 视图中，将工程中断点置为失效状态，即点击断点前的复选框。



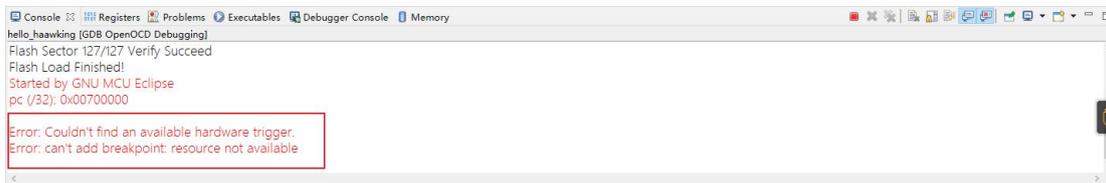
下面演示如果工程中的断点数量大于等于两个时，使用 Restart 功能会出现的问题，下图的示例工程中，在 46、47 行存在两个断点。



如果此时点击  按钮，在 IDE 下方的“Console”面板中，会出现下方的提示，此时调试将无法正常使用，用户需要关闭并重新开启调试。

Error: Couldn't find an available hardware trigger.

Error: can't add breakpoint: resource not available



### 3.5 如何查看变量值

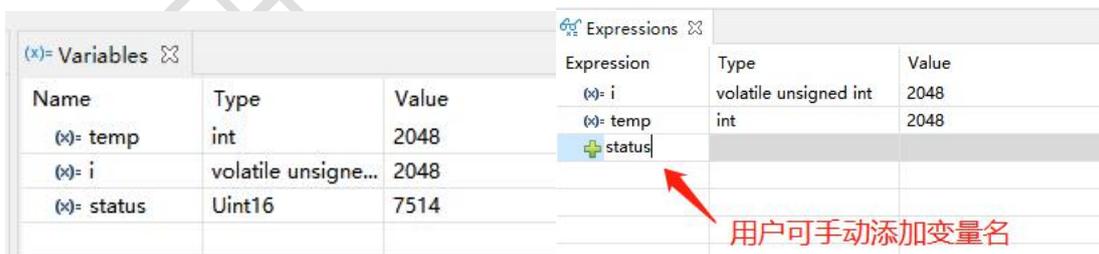
在 Debug 模式下，如果需要查看变量的值可以在右侧窗口栏中点击“Variables”或“Expressions”窗口标签后查看，窗口标签图片见下：



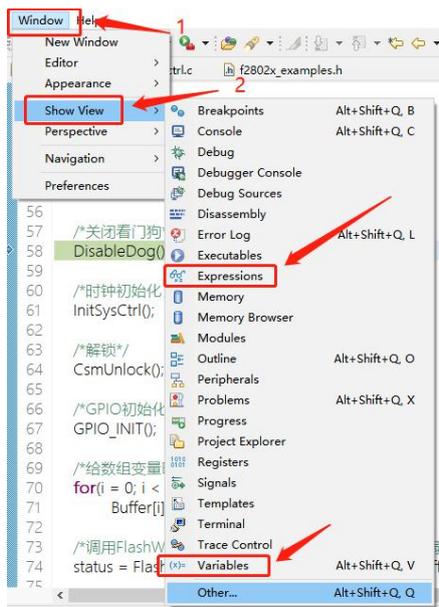
这两个窗口都有显示变量值的功能，区别在于“Variables”中的变量是因程序中声明的变量自动填充的，而“Expressions”中需要用户手动添加需要查看的变量名，变量名的类型和变量的值会自动显示。

**注意：**当芯片在全速运行时，这两个窗口中的变量值不会实时更新，只有当芯片处于暂停的状态或单步调试的模式时，可查看变量值。如果需要查看变量的实时状态值，用户可参阅“2.5.7 使用实时刷新功能”章节。

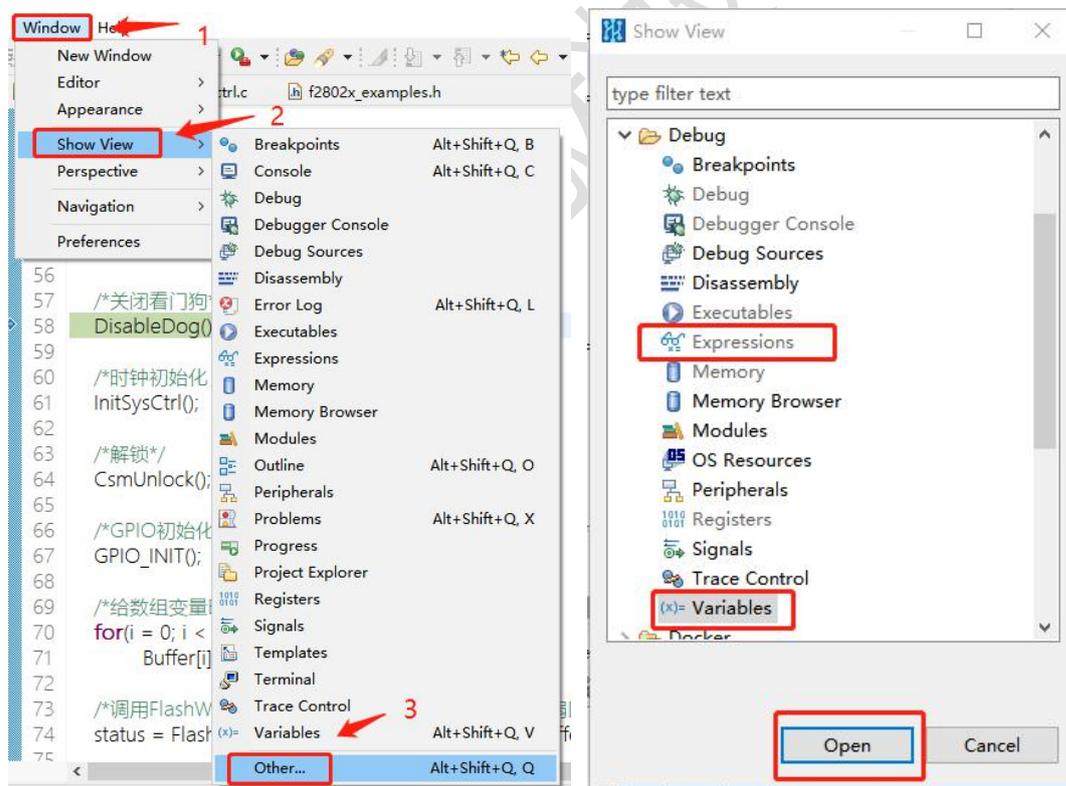
“Variables”和“Expressions”两个窗口的内容显示如下图：



如果进入 Debug 模式中，没有显示这两个窗口时，用户可在工具栏的“Window”->“Show View”中查找。位置如下图



或在工具栏的“Window” -> “Show View” -> “Other...” 中的“Debug” 下单击选中“Variables” 或 “Expressions”，再点击“Open” 就可以显示在 IDE 的右侧窗口栏。操作图片如下：



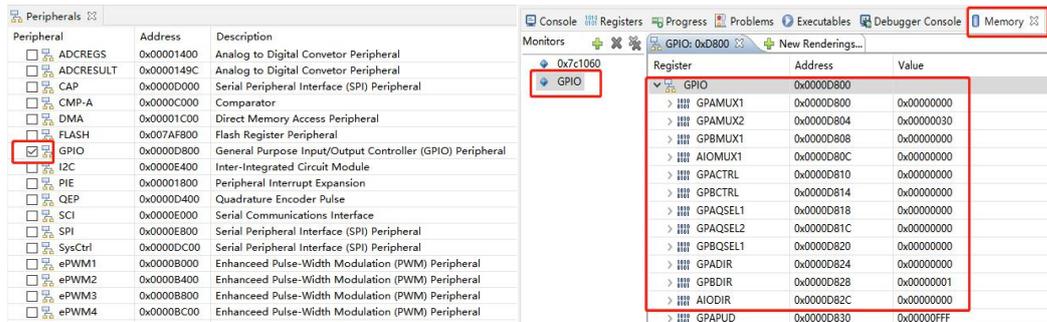
### 3.6 如何查看外设寄存器的值

在 Debug 模式下，如果需要查看芯片外设寄存器的值可以在右侧窗口栏中点

击“Peripherals”窗口标签后查看，窗口标签图片见下：



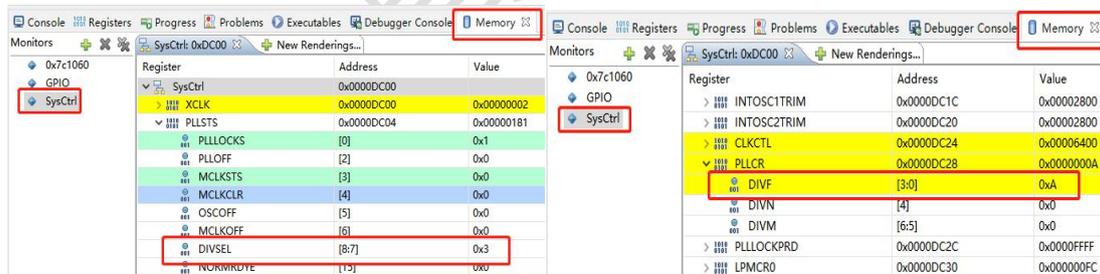
打开后会显示该芯片的所有外设模块名称。点击对应模块名称前面的方块，会在 IDE 下面“Memory”的窗口队列中显示该名称。操作方式如下图：



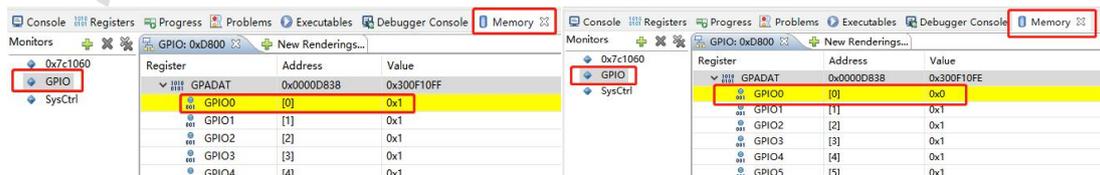
GPIO 模块中的全部寄存器的寄存器名称、地址和数值会显示在该窗口中。

接下来具体说明一下“SysCtrl”和“GPIO”模块中寄存器状态的查看过程。

当程序编译完成后，进入 Debug 模式后，可单步调试 或打断点在“InitSysCtrl();”函数之后，当运行完“InitSysCtrl();”函数后，可查看倍频系数和分频系数是否已经写入“PLLCR”和“PLLSTS”寄存器中，数据显示见下图：



查看数据后，继续运行 GPIO 配置程序“GpioDataRegs.GPATOGGLE.bit.GPIO0 = 1;”和“GpioDataRegs.GPATOGGLE.bit.GPIO0 = 0;”后，数据显示见下图：



**注意：**当芯片在全速运行时，这两个窗口中的变量值不会实时更新，只有当芯片处于暂停的状态或单步调试的模式时，可查看变量值。如果需要查看变量的实时状态值，用户可参阅“2.5.8 使用实时刷新功能”章节。

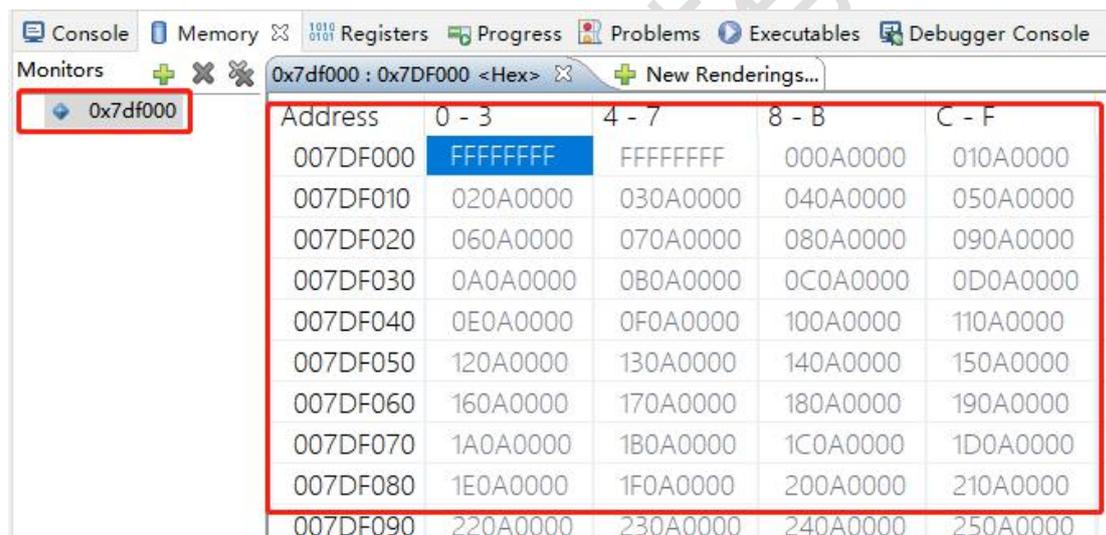
## 3.7 使用 Memort 视图查看存储器中的值

### 3.7.1 如何查看指定地址处的数据

在 Debug 模式下，可输入存储器或寄存器的地址查看该地址的数值。具体操作见下图：



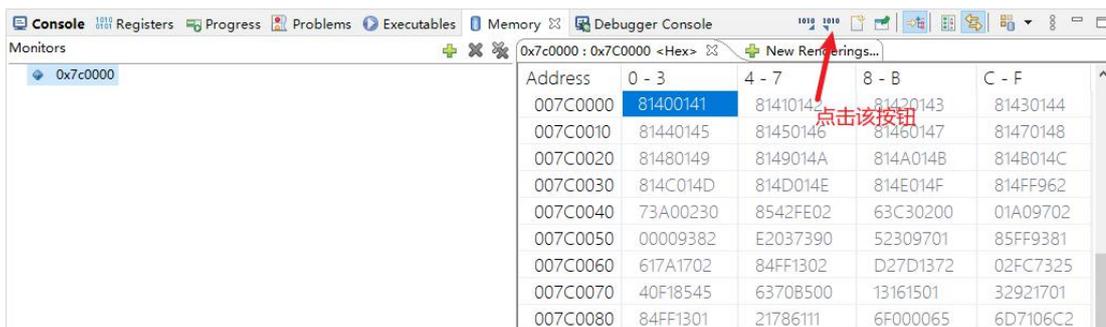
完成这两步操作就可以查看该地址中的数据了。



**注意：**当芯片在全速运行时，Memory 视图中的值不会实时更新，只有当芯片处于暂停的状态或单步调试的模式时，可查看变量值。

### 3.7.2 如何导出 Memory 视图中的数据

1. 点击 Memory 视图中的  按钮，弹出“Export Memory”弹窗。



2. 在弹窗中分别配置下方的选项， 点击 OK 按钮即可。

- (1) Format: 导出的文件格式
- (2) Start address: 起始地址
- (3) End address: 结束地址
- (4) Length: 自定义导出数据的长度
- (5) File Name: 导出文件的名称

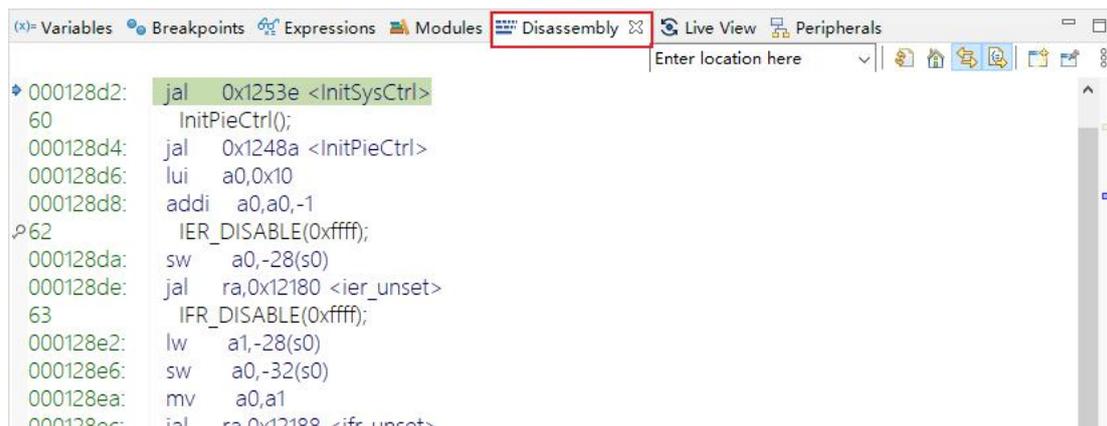


### 3.8 如何进入汇编调试

1、在工具栏点击  “Instruction Stepping Mode” 按钮。



2、查看右侧 “Disassembly” 显示栏



3、单步汇编调试继续使用工具栏中的 “Step Into” 按钮



### 3.9 使用 Debug Without Download 功能

功能介绍: 该功能用于不擦除芯片的内容时启动调试, 调试启动后, 用户可以使用 Memory 视图查看芯片中的内容, 或者调试芯片中的程序。

#### 3.9.1 前置工作

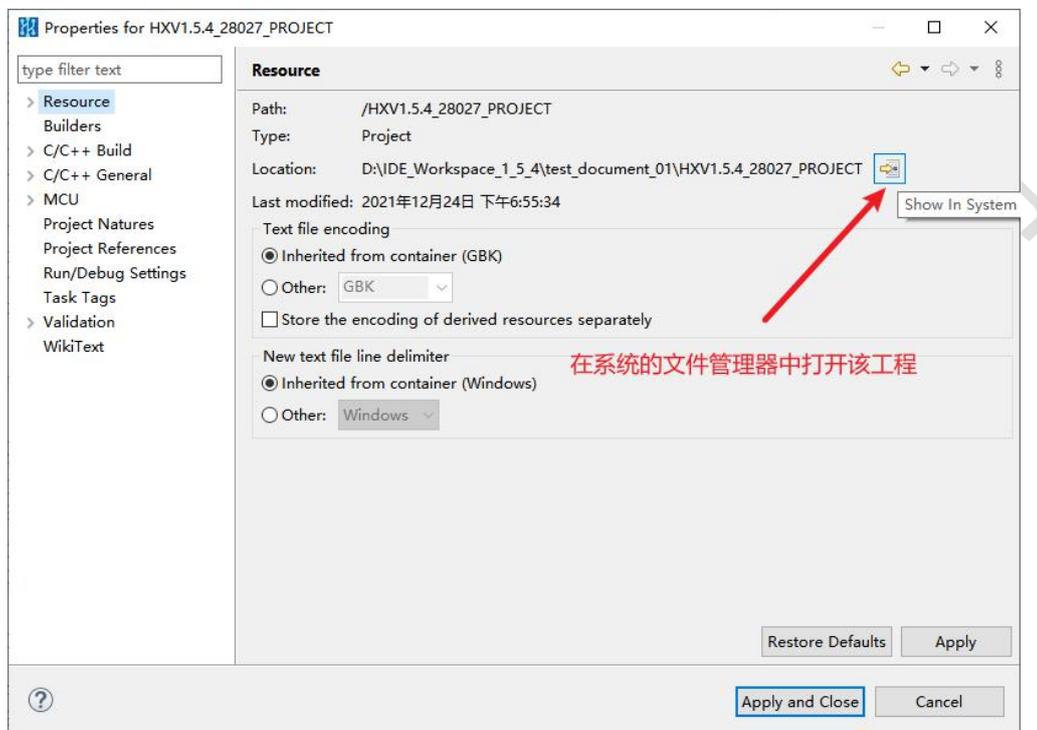
由于该功能并不兼容 Haawking IDE 1.5.3 版本前创建的工程, 因此, 如果用户希望在旧版工程上使用该功能, 首先需要按照下面的步骤更新工程中的配置文件。

1. 首先使用 V1.5.3 版本及以后的 IDE 创建一个与旧工程芯片型号一致的工程。

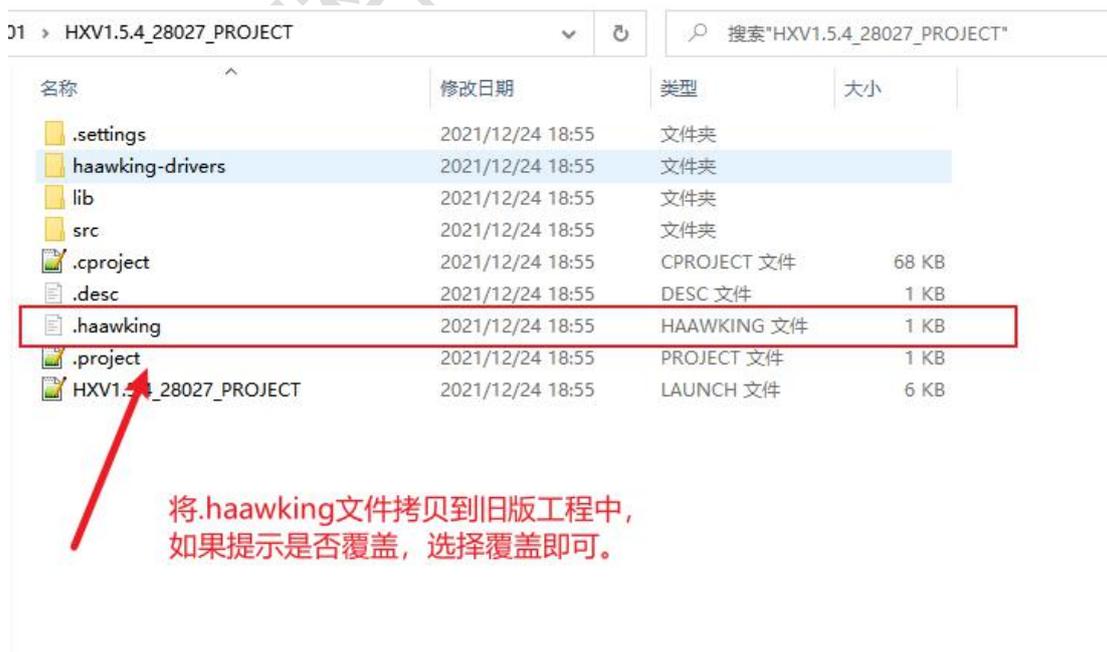


2. 选中该项目，右键弹出菜单，点击“Properties”选项，打开“Properties”面板。

3. 选择左侧的“Resource”选项，点击右侧的  按钮，在文件管理器打开项目。

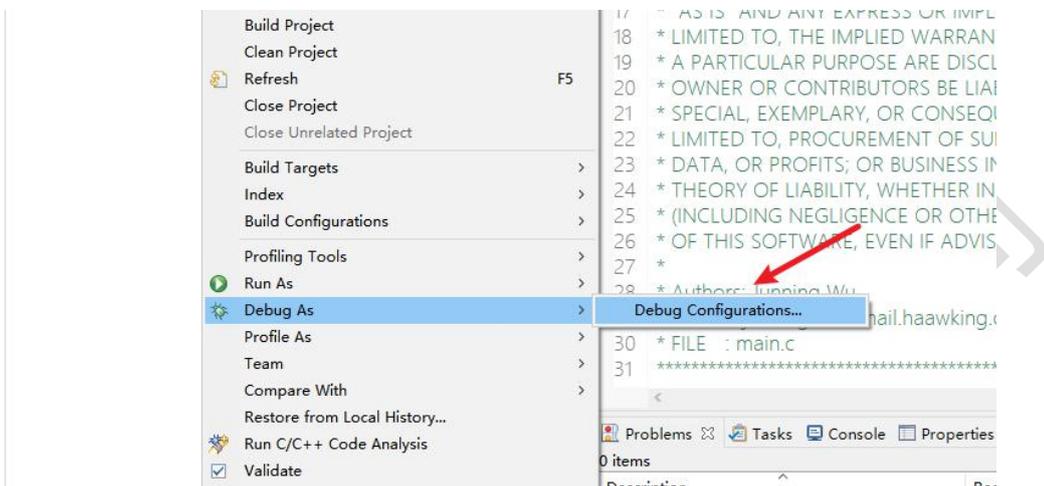


4. 进入项目目录，拷贝项目中的.haawking 文件，将其复制到旧工程中，如果提示是否覆盖，点击覆盖即可。



### 3.9.2 启动 Debug Without Download

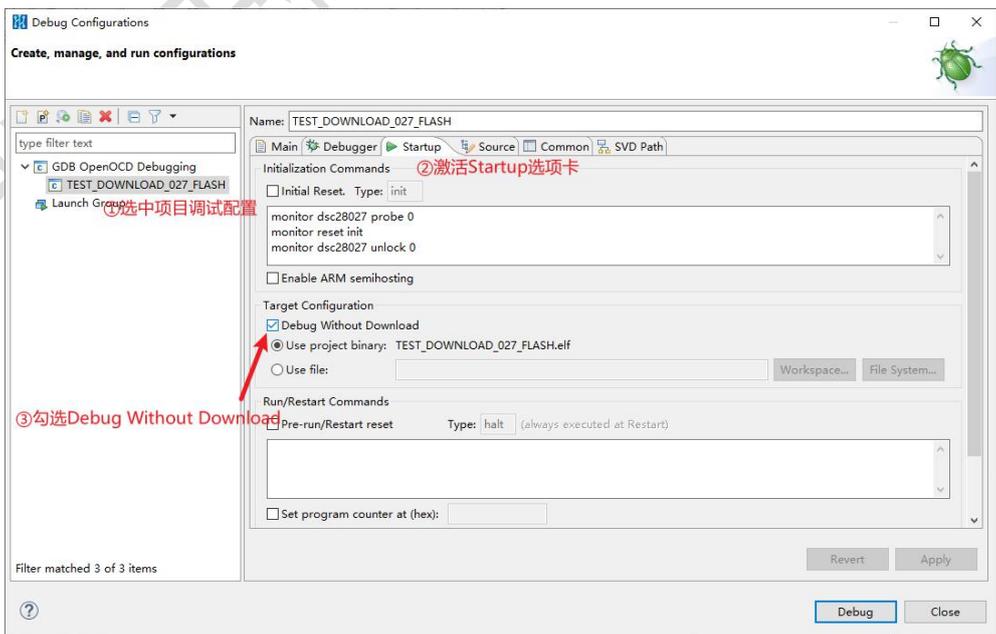
1. 右键工程，选择“Debug As->Debug Configurations”，打开工程的 Debug 配置面板。



2. 选中工程的调试配置(与工程同名)，激活右侧的“Startup”选项卡，勾选“Debug Without Download”复选框即可。

由于 Debug Without Download 功能需要启动调试，因此需要一个调试信息所在的文件，默认情况下会使用当前工程下的 elf 文件作为调试信息所在文件，用户也可以指定任意的文件作为调试文件。

- (1) Use project binary 选项: 使用工程下的 elf 文件作为调试文件
- (2) Use file 选项: 用户自定义一个 elf 文件作为调试文件



3. 点击 Debug 进入调试。

### 3.9.3 Debug Without Download 功能注意事项

如果用户希望使用“Debug Without Download”功能调试芯片中的程序，那么用于启动“Debug Without Download”的工程中的程序需要与芯片中的程序保持一致，否则可能无法达到正常调试的效果。

## 3.10 使用实时刷新功能

由于 Haawking IDE 的 Expressions 视图不支持在芯片运行时查看变量的值，因此，用户想在芯片运行时查看变量的值需要使用“Live View”视图。

### 3.10.1 实时刷新视图使用注意事项

1. 实时刷新视图仅支持添加**全局变量**，并且仅支持添加全局变量名称。

例:

如果程序中定义了一个名为“test\_struct\_val”的变量，那么对于实时刷新视图来说，“test\_struct\_val”是一个合法的变量名，但是“test\_struct\_val.test\_int\_member”则是一个非法的变量名。

```
1. struct {
2.     int test_int_member;
3. }test_struct_val;
```

2. 请勿向实时刷新视图添加占用内存过大的变量，否则可能会导致 IDE 出现卡顿。

3. 实时刷新视图默认处于暂停刷新状态，如果用户需要使用实时刷新，则需要手动点击  按钮来开启实时刷新。

4. 实时刷新模块目前仅支持变量的查看，不支持变量的写入。

5. 实时刷新模块目前不支持变量的波形图显示。

6. 实时刷新视图目前仍处于开发阶段，部分功能存在问题，具体的问题请

看 3.9.8 小节中的内容。

### 3.10.2 实时刷新视图按钮介绍

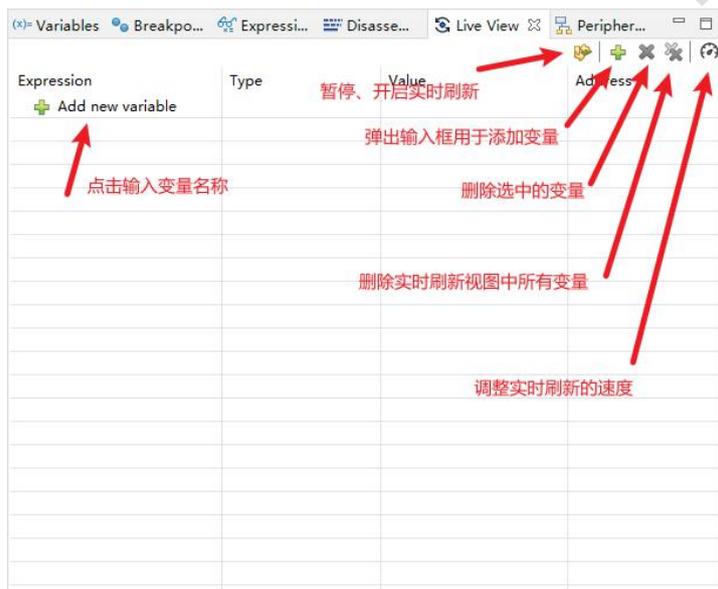
 Start Refresh/Stop Refresh 按钮用于开启、暂停实时刷新

 Add Expression To LiveView 按钮用于向实时刷新视图添加变量，它会弹出一个输入框，让用户输入待添加的变量名称

 Remove Selection Variable 按钮用于删除选中变量

 Remove All Variables 按钮用于删除实时刷新视图中所有的变量

 Change Refresh Interval 按钮用于调整实时刷新的速度，默认的刷新间隔为 200ms



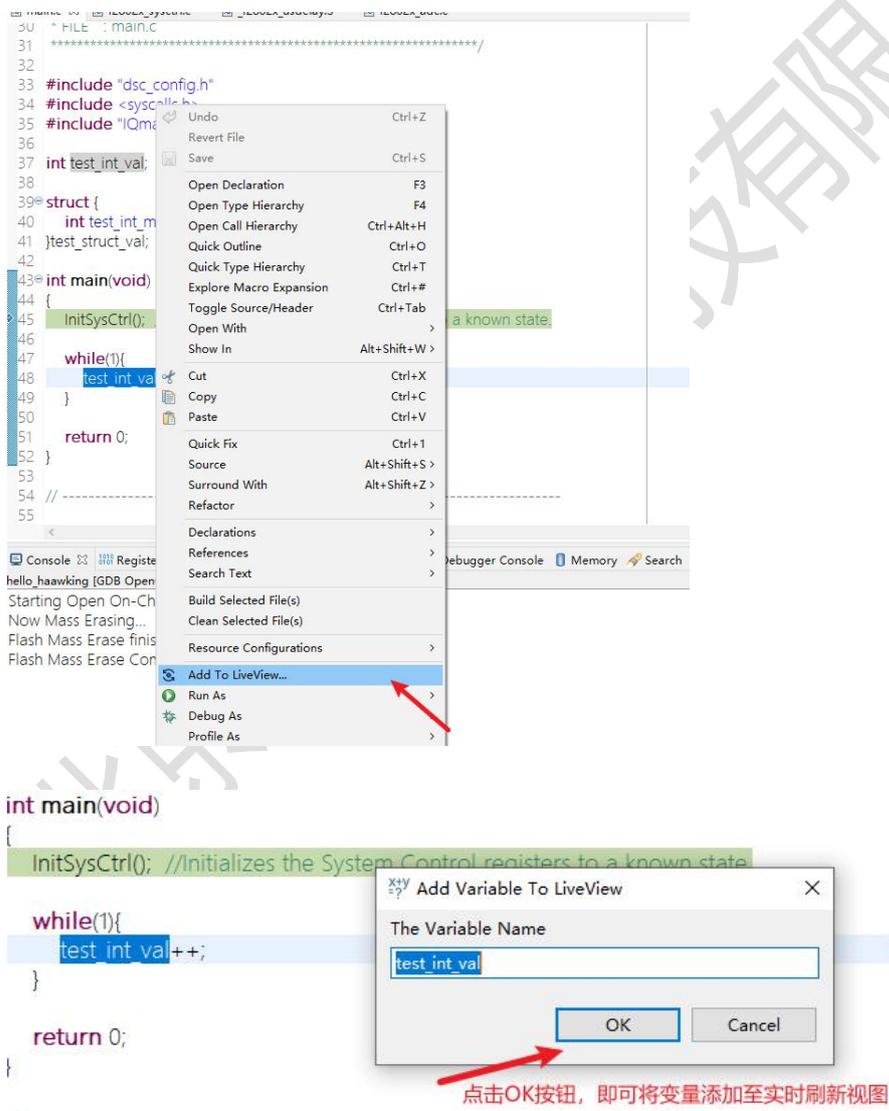
### 3.10.3 向实时刷新视图添加变量

向实时刷新视图添加调试变量的方式有两种。

第一种方式：未开启 Debug 时，通过点击实时刷新视图中的  按钮，或者直接编辑“Add new variable”元素来输入变量名称。

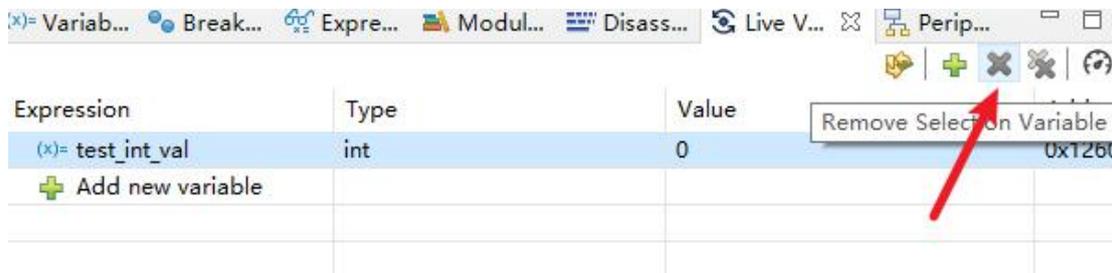


第二种方式为，在开启 Debug 后，在编辑视图中选中要添加的变量，然后右击，选择“Add To LiveView”按钮，同样会弹出输入框，框中已经包含刚才选中的变量名称，点击 OK 即可将其加入实时刷新视图进行调试。



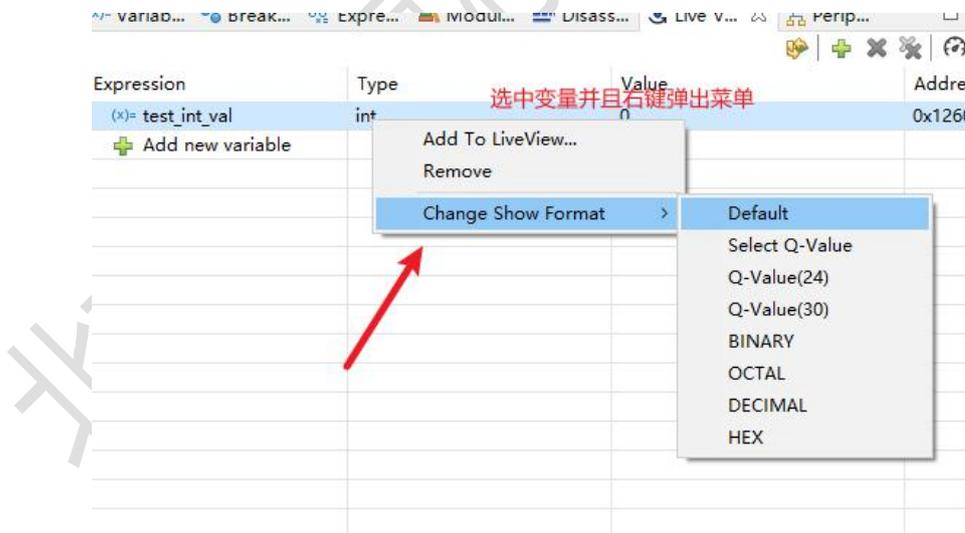
### 3.10.4 删除实时刷新视图中的变量

如果想要删除一个变量，首先在实时刷新视图中选中该变量，然后点击视图中的  按钮即可删除选中的变量。



### 3.10.5 切换变量输出格式

实时刷新视图支持多种输出格式， 默认的输出格式为按照其类型进行解析的数据。如果想要切换变量的输出格式，首先需要选中一个元素，然后右击，在弹出的菜单中选择“Change Show Format”即可切换显示格式。



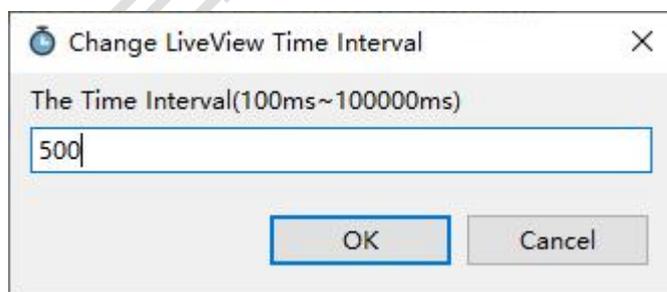
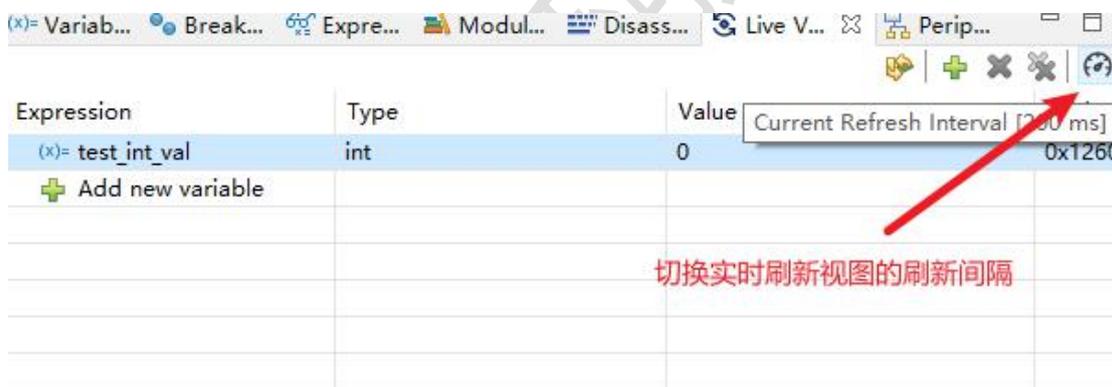
“Change Show Format” 菜单中选项的含义:

Default	该按钮是将变量按照其类型进行解析
Select Q-Value	该按钮将会弹出一个输入框, 在其中输入具体的 IQ 数值

Q-Value (24)	该按钮将会将数据按照 IQ24 格式显示
Q-Value (30)	该按钮将会将数据按照 IQ30 格式显示
BINARY	该按钮将会显示变量数据对应的 2 进制形式
OCTAL	该按钮将会显示变量数据对应的 8 进制形式
DECIMAL	该按钮将变量转换为有符号整数进行显示
HEX	该按钮将会显示变量数据对应的 16 进制形式

### 3.10.6 切换刷新间隔

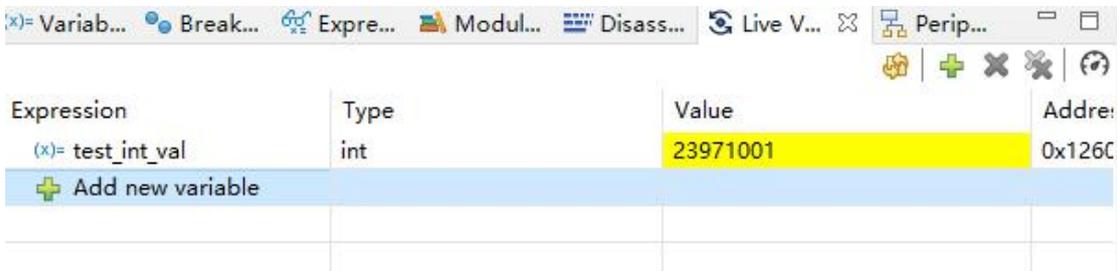
点击视图右上角的  按钮，将会弹出输入框用来输入新的刷新闻隔，单位为毫秒，默认刷新闻隔为 200ms，实时刷新视图允许的最小间隔为 100ms，用户通常使用默认的刷新闻隔即可。



### 3.10.7 数据存在变化时的显示效果

如果一个数据距离上次刷新时值有变化，那么该变量的背景色将变为黄色。

注：实时刷新视图第一次启动时，默认处于暂停刷新状态，需要手动点击  按钮来开启刷新。



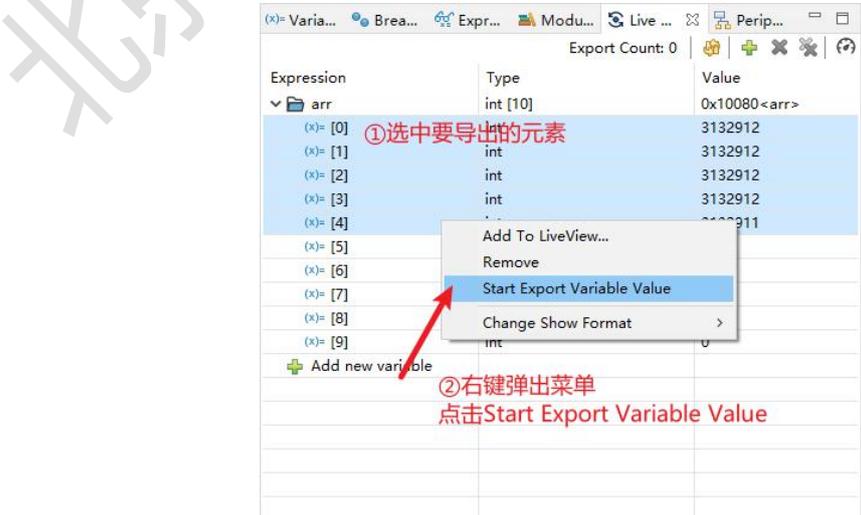
### 3.10.8 导出实时刷新视图中的数据

该功能用于导出实时刷新视图中的数据，导出数据的格式为 csv 文件，如下图所示。

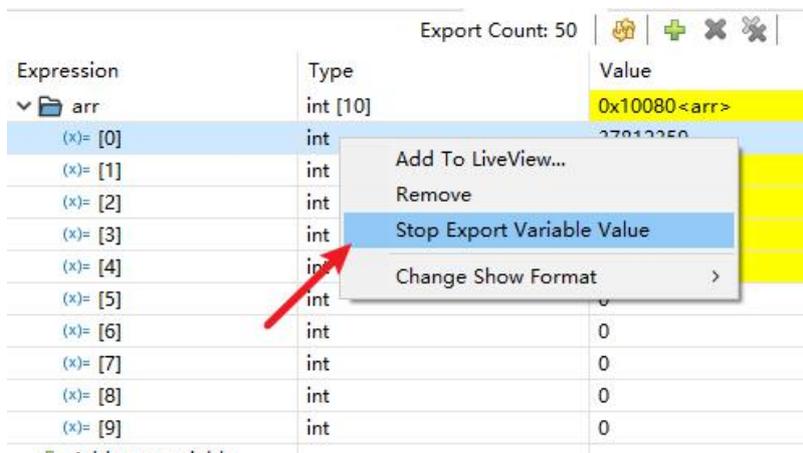
注：如果要导出实时的数据，请确保实时刷新视图不处于暂停状态

	A	B	C	D	E
1	arr-[0]	arr-[1]	arr-[2]	arr-[3]	arr-[4]
2	3132912	3132912	3132912	3132912	3132911
3	3132912	3132912	3132912	3132912	3132911
4	3132912	3132912	3132912	3132912	3132911
5	3132912	3132912	3132912	3132912	3132911
6	3132912	3132912	3132912	3132912	3132911
7	3132912	3132912	3132912	3132912	3132911
8	3132912	3132912	3132912	3132912	3132911
9	3132912	3132912	3132912	3132912	3132911
10	3132912	3132912	3132912	3132912	3132911
11	3132912	3132912	3132912	3132912	3132911
12	3132912	3132912	3132912	3132912	3132911
13	3132912	3132912	3132912	3132912	3132911
14	3132912	3132912	3132912	3132912	3132911
15	3132912	3132912	3132912	3132912	3132911
16	3132912	3132912	3132912	3132912	3132911
17	3132912	3132912	3132912	3132912	3132911
18	3132912	3132912	3132912	3132912	3132911
19	3132912	3132912	3132912	3132912	3132911

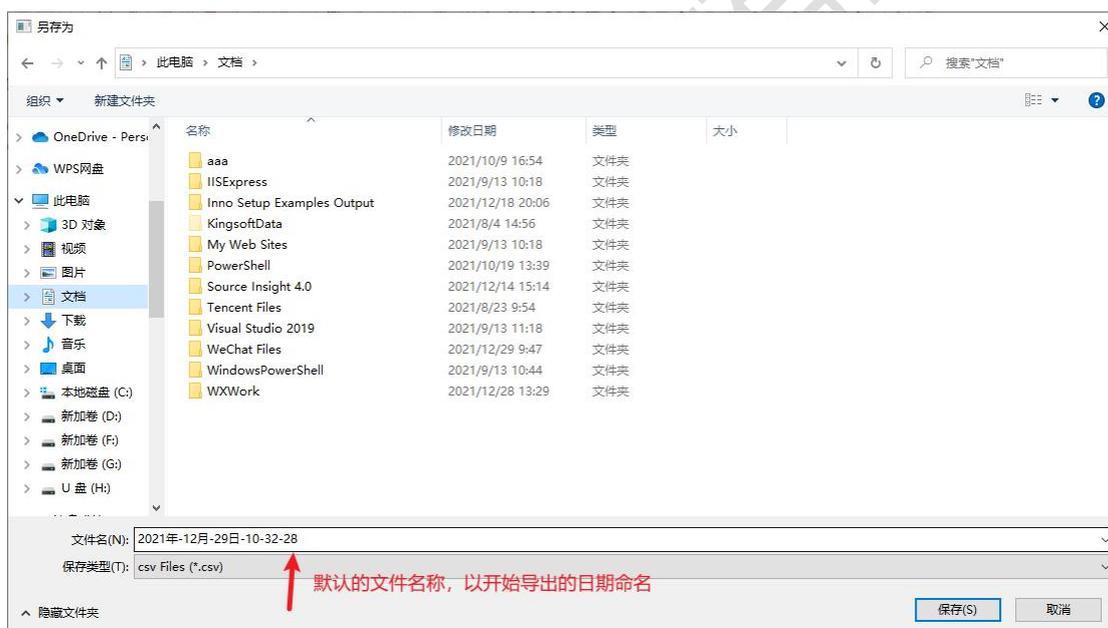
1. 首先在实时刷新视图中选中待导出的元素，可以按住 Ctrl 键选中多个元素，然后右键菜单，点击“Start Export Variable Value”按钮。



2. 当用户需要暂停导出时，在实时刷新视图右键，点击“Stop Export Variable Value”按钮。

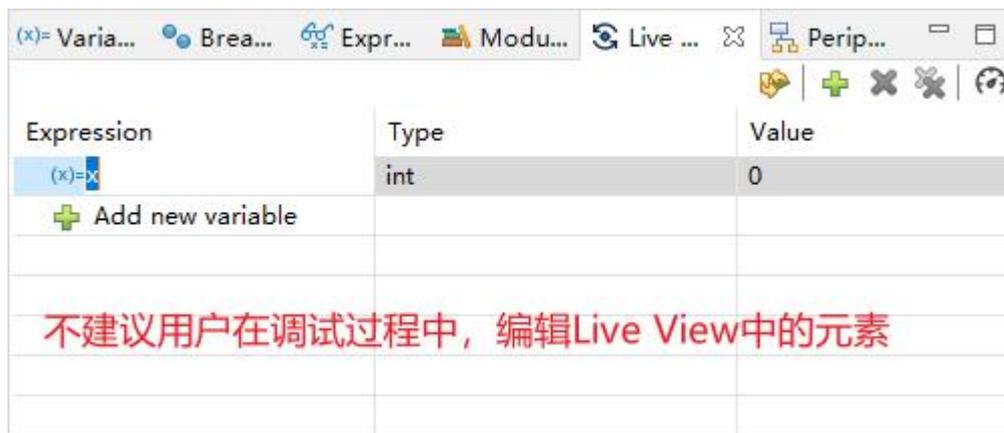


3. 点击“Stop Export Variable Value”按钮后，会弹出保存文件弹窗，保存导出的文件即可。



### 3.10.9 实时刷新视图存在的问题

编辑变量功能: 如果用户在调试过程中，编辑了“Live View”中的元素，相当于删除了旧的元素，会出现删除功能存在的问题。同时还可能造成用户卡顿。



### 3.11 如何加密解密

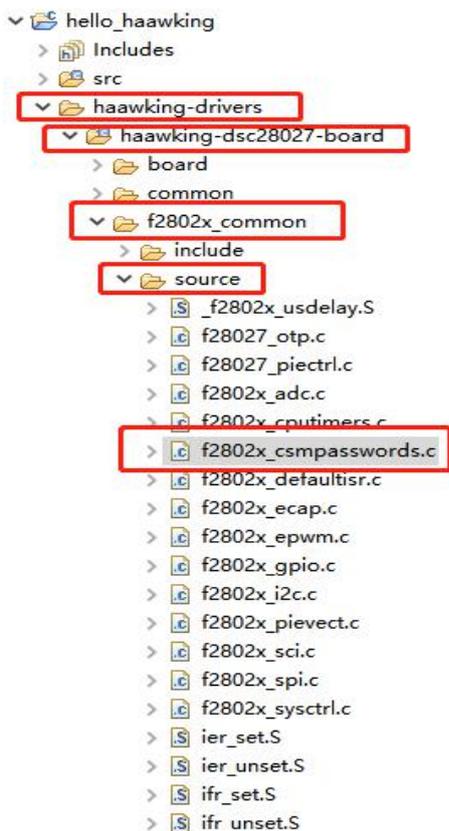
#### 3.11.1 加密操作

我们的 IDE 没有直接设置加密密码的位置，而是通过在一个密码文件中修改对应的密钥进行加密操作。

如 28027 就是在 f2802x\_common/source/f2802x\_csmpasswords.c 文件中。首先在工程目录中，

- 点击 “haawking-drivers”
- > “haawking-dsc28027-board”
- > “f2802x\_common”
- > “source” 中，找到 “f2802x\_csmpasswords.c” 文件，双击该文件。

该文件的位置如下：



打开后就可以看到四组密钥，其中内容如下图，对该密钥进行设置，**切记不要写全零（全“0”）**，否则会锁芯片，该芯片就不能再使用了。

<pre> 1 2 #include "F2802x_Device.h" // Headerfile Include File 3e/***** 4 password file 5 *****/ 6 7 volatile struct CSM_PWL CODE_SECTION("CsmPwl") CsmPwl = 8 { 9     0xFFFFFFFF, 10    0xFFFFFFFF, 11    0xFFFFFFFF, 12    0xFFFFFFFF, 13 }; 14 </pre>	<pre> 1 2 #include "F2802x_Device.h" // Headerfile Include File 3e/***** 4 password file 5 *****/ 6 7 volatile struct CSM_PWL CODE_SECTION("CsmPwl") CsmPwl = 8 { 9     0x1234FFFF, 10    0xFFFFFFFF, 11    0xFFFFFFFF, 12    0xFFFFDECB, 13 }; 14 </pre>
---	---

我们对密钥进行设置后，通过 Debug 的 方式对芯片进行加密。

### 3.11.2 解密操作

完成加密操作后，当需要再次 Debug 的时候，需要进行解密操作。

点击“Debug”的下拉选项

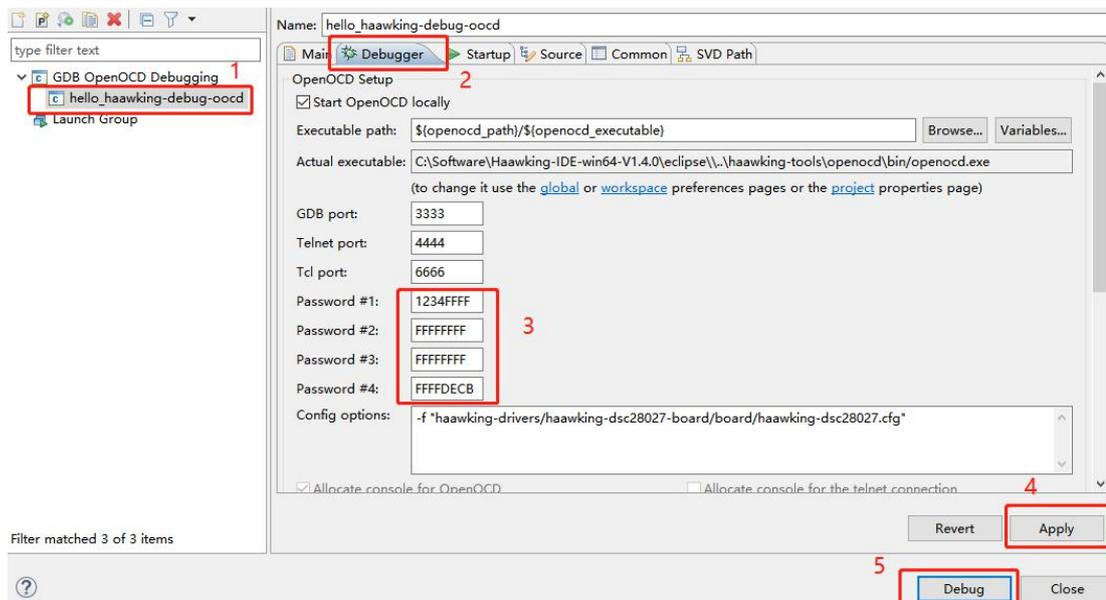
-> “Debug Configurations”

-> “Debugger” 窗口下 Debugger

-> “Password #1” Password #1:

- > “Password #2” Password #2: FFFFFFFF
- > “Password #3” Password #3: FFFFFFFF
- > “Password #4” Password #4: FFFFFFFF

输入框中，输入修改后的密钥，修改后见下图。

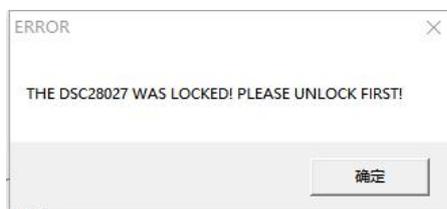


-> 再点击 “Apply”

-> 点击 “Debug”，

完成解密步骤后，就可以对芯片再次进行调试了。

**注意**，如果不进行解密操作，就无法对芯片进行调试。当点击 “Debug” 后，会有弹窗提示，提示用户芯片是处于加密状态，需要解密后才可以进行调试，弹出提示如下图。



在解密状态下，仍然可以继续配置新的密钥，但一定要记住密钥。如果需要把芯片的加密状态改为解密状态，需要再将密钥配置为全 “F”，通过 Debug 相关的配置就可以实现了。

```
1
2 #include "F2802x_Device.h" // Headerfile Include File
3 /*****
4 password file
5 *****/
6
7 volatile struct CSM_PWL CODE_SECTION(".CsmPwl") CsmPwl =
8 {
9     0x1234FFFF,
10    0xFFFFFFFF,
11    0xFFFFFFFF,
12    0xFFFFDECB,
13 };
14
```

----->

```
1
2 #include "F2802x_Device.h" // Headerfile Include File
3 /*****
4 password file
5 *****/
6
7 volatile struct CSM_PWL CODE_SECTION(".CsmPwl") CsmPwl =
8 {
9     0xFFFFFFFF,
10    0xFFFFFFFF,
11    0xFFFFFFFF,
12    0xFFFFFFFF,
13 };
14
```

北京中科昊芯科技有限公司

## 4 审核

版本编号	变更状态	简要说明	变更人	变更日期	批准人	批准日期

北京中科昊芯科技有限公司