Luban-mini_SDK

Version 1.0.0

修订日期： 2026-06-22

修订记录

下表记录了 2025-06-22 (V1.0) 至今的所有修订历史：

表 0-1 修订记录

版本	章节	修订说明
V1.0	-	初版

内容

1. SDK 架构简介

Luban Mini SDK 由下列部分组成：

- 驱动层
  - HAL 驱动，即各硬件模块的 HAL 驱动实现
  - CPU 核心访问接口，如 Cache 操作接口，CPU 寄存器访问接口
  - BSP 驱动，包括板级组件的驱动，如触摸、显示屏等
- 中间件

  中间件层包含已经移植支持的各软件模块：
  - FATFS 文件系统
  - LittleFS 文件系统
  - TLSF 堆管理
  - …

  注：

  通过以上模块可以更方便的构建各种应用。
- 应用层
  - HAL 驱动的使用示例
  - 各中间件组件的使用示例
  - 其他示例

  注：

  应用层提供的内容，主要用于演示如何创建应用，以及如何将各驱动、组件集成到应用。

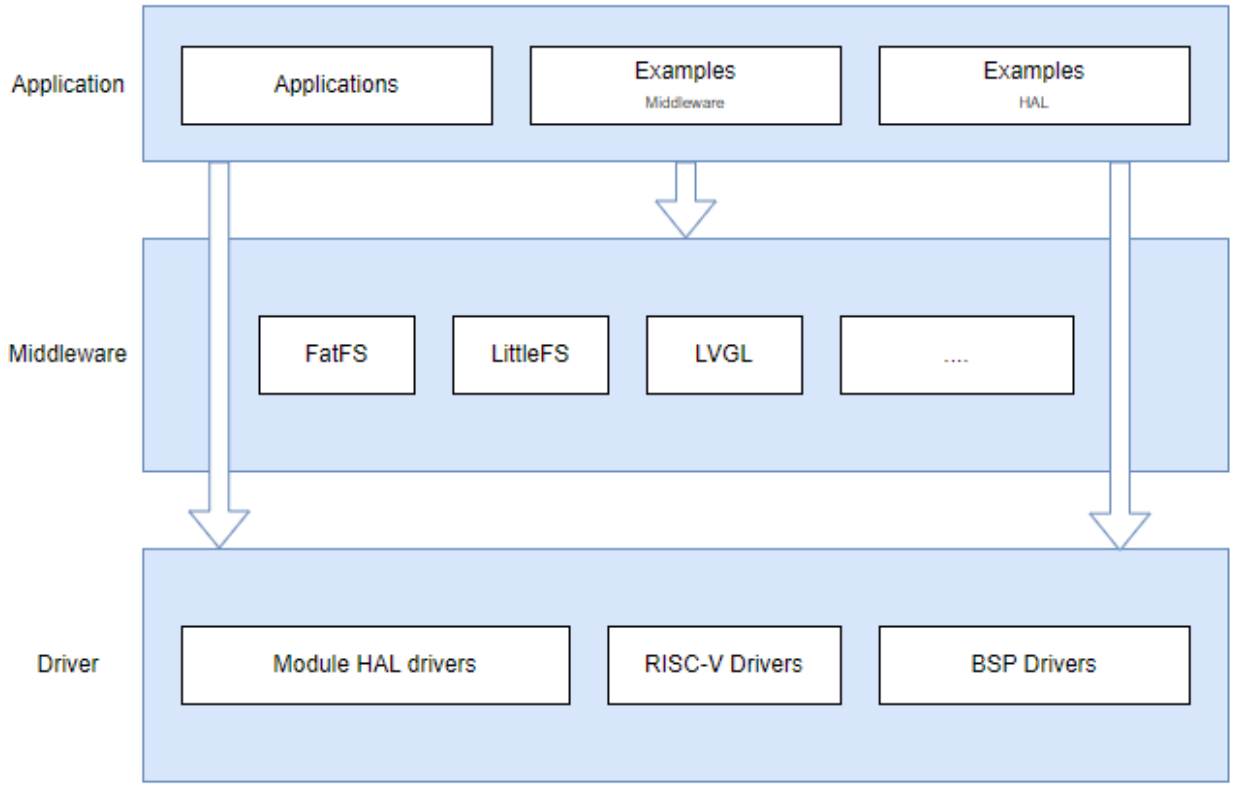


图 1–1 SDK 架构

2. 环境准备

2.1. LubanMini 开发工具

本节将介绍如何安装 Luban Mini Build Tools。

为了简化 Luban Mini 的开发环境设置，以下几种开发环境相关的工具和配置已整理成安装包，一键安装即可使用：

- Windows Eclipse 开发环境
- Windows Msys2 CMake 开发环境
- Linux CMake 开发环境

2.1.1. 下载安装包

从以下任意地址均可获取安装包：

- https://dshare.artinchip.com
- https://aicdoc.artinchip.com/topics/product/download-doc-tool-zip.html

Luban Mini 支持下列开发平台，且不同的开发平台对应不同的安装包，根据操作需求选择：

- Windows：安装文件为 LubanMiniBuildTools-Win32-x86_64-V3.2.0-R2.exe
- Linux：安装文件为 LubanMiniBuildTools-Linux-x86_64-V3.2.0-R1.sh

2.1.2. Windows

对于 Windows 环境：

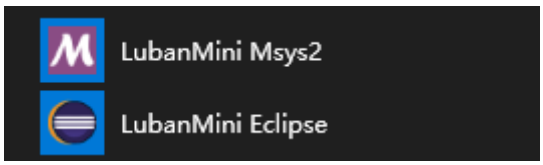
1. 下载完成后，双击安装文件 LubanMiniBuildTools-Win32-x86_64-V3.2.0-R2.exe 并按照页面提示完成安装。

   注：

   该安装不需要管理员权限，仅针对当前用户安装。

   默认安装位置为用户目录的 `App/Data/Local/LubanMiniBuildTools`。

2. 安装完成后，选择以下任意快捷入口方式即可开始使用：



   - Luban Mini Eclipse ： 原有的 Eclipse IDE 程序。

     该 IDE 为 Eclipse 官网原版程序，增加了以下配置：

- 设置 Xuantie GCC 为 RISCV 默认的工具链。
- 设置 Msys2 为默认 build tool，其他保持不变。

- LubanMini Msys2 ：命令行编译环境。

  该环境为 Msys2 官网下载的 msys2-base 原版程序，作为 Luban Mini CMake 编译工具链。

  该环境集成了以下工具，共同构成 Luban Mini CMake 编译工具链：

  - Make
  - CMake
  - python 3.8.10
  - Xuantie GCC

2.1.3. Linux

在 Linux 环境下，执行自解压脚本 LubanMiniBuildTools-Linux-x86_64-V3.2.0-R1.sh 完成安装即可。

该脚本会将 Luban Mini 编译所需的 Xuantie GCC、CMake 安装到用户目录中的 `.LubanMiniBuildTools` 文件夹。

2.2. 准备代码

LubanMini SDK 的代码托管于 Gitee 服务器中，为开源代码。本节描述了如何从对应的仓库地址中下载源代码。

2.2.1. 通过 Git 下载代码

推荐使用 Git 用户端下载代码，可以进行版本管理的同时获取服务器补丁和版本的发布：

2.2.2. 通过网络下载代码

通过网络下载的方式可直接下载压缩包，详细步骤如下：

- 在 Gitee 上打开 LubanMini 仓库，地址为：。
- 在仓库主界面点击克隆/下载按钮。
- 在克隆/下载界面选择下载 ZIP。

2.2.3. 代码结构

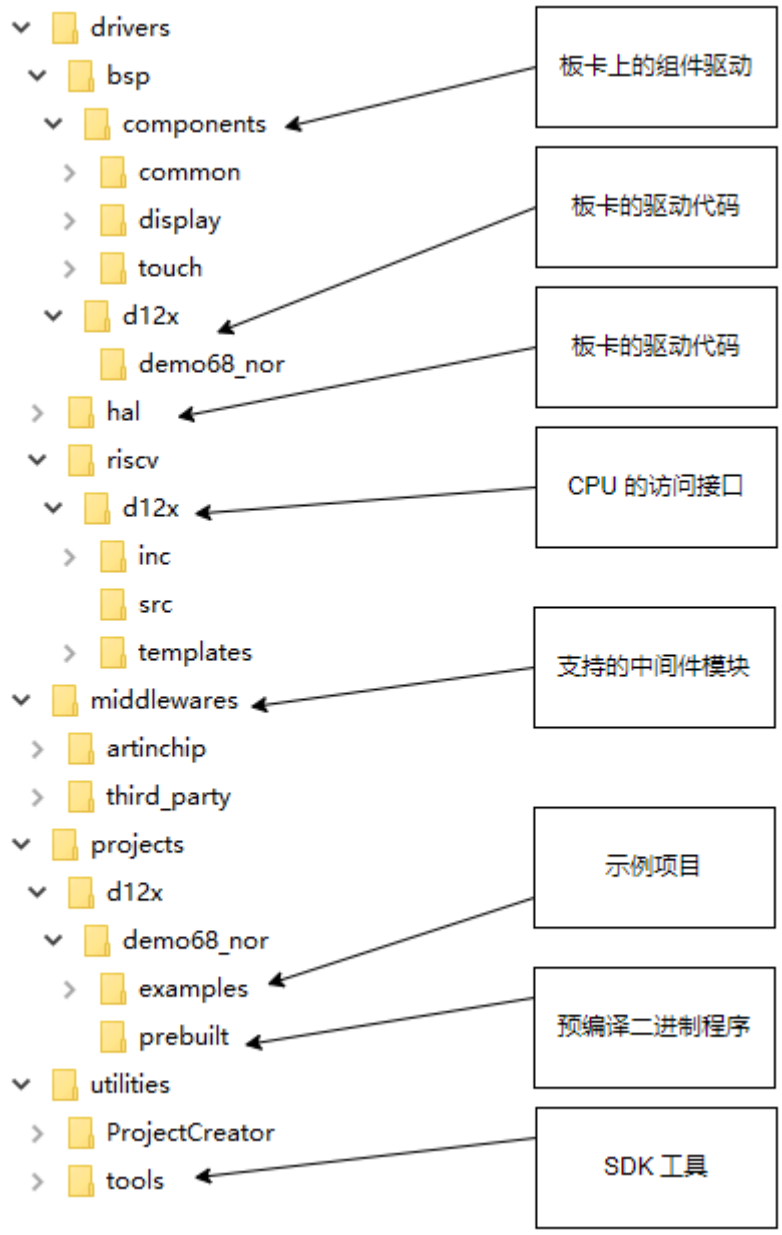


图 2–1 Luban Mini SDK 目录概括

Luban Mini SDK 中有以下重要的文件和目录，其分类和功能如下所述：

```
drivers/
├── bsp/
│   ├── components/         # 板卡上的组件驱动
│   │   ├── common/
│   │   ├── display/
│   │   └── touch/
│   └── d12x/
│       └── demo68_nor/
├── hal/                    # CPU 的访问接口
└── riscv/
```

```
|   ├── d12x/
|   |   ├──inc/
|   |   ├──src/
|   |   └── templates/

middlewares/
├── artinchip/              # 支持的中间件模块
└── third_party/

projects/
├── d12x/
└── demo68_nor/
|   | ├── examples/           # 示例项目
|   | └── prebuilt/           # 预编译二进制程序

utilities/
├── ProjectCreator/
└── tools/
```

2.2.3.1. drivers/ 目录说明

- drivers/hal/ – 提供当前芯片硬件模块的 HAL 驱动代码
- drivers/riscv/d12x/ – 存放芯片的 CPU 内核访问接口以及启动初始化代码
- drivers/riscv/d12x/templates/ – 提供配置模板，如 gcc 链接脚本、HAL 配置文件、中断处理函数列表
- drivers/bsp/d12x/demo68_nor/ – 提供板卡使用的板级驱动
- drivers/bsp/components/ – 提供常用板级设备驱动，如触摸、显示等

2.2.3.2. middlewares/ 目录说明

- middlewares/artinchip/ – ArtInChip 实现或深度定制的中间件模块
- middlewares/third-party/ – 移植自第三方的中间件模块

具体模块的集成和用法，可参考模块目录下的 README_AIC.md。

2.2.3.3. projects/ 目录说明

- projects/d12x/demo68_nor/examples/ – HAL 驱动使用示例和中间件集成示例，源码和配置分离（Eclipse 目录为项目配置，main 目录为源码）
- projects/d12x/demo68_nor/prebuilt/ – 预编译组件的二进制文件

具体示例的介绍，可参考示例下的 README_AIC.md。

2.2.3.4. utilities/ 目录说明

存放 SDK 工具，如烧录镜像的制作工具等。

2.3. 项目配置

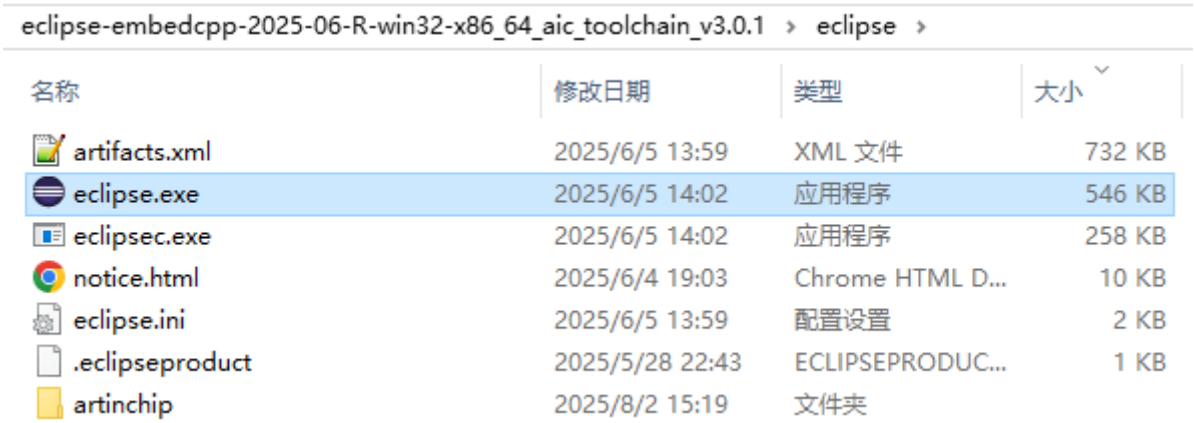
2.3.1. Window–Eclipse

2.3.1.1. 导入项目

2.3.1.1.1. 打开项目

用 Eclipse 打开项目的步骤如下：

1. 点击 Luban Mini Eclipse 打开 Eclipse。



> 注：
>
> 打开 Eclipse 时，可能遇到Microsoft Defender Exclusion Check Error 报错，可参考常见问题中的操作步骤。

2. 点击 FileOpen Projects from File System，从本地文件系统中选择一个项目。

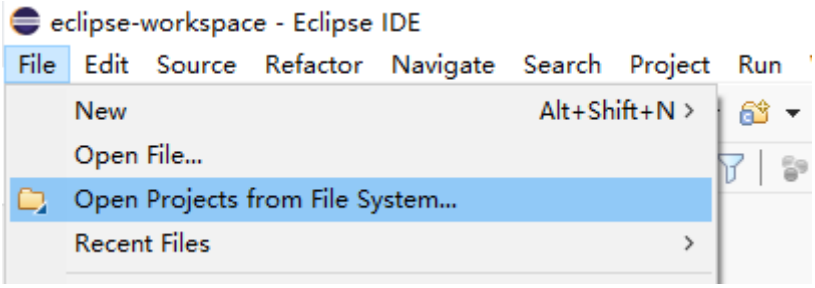
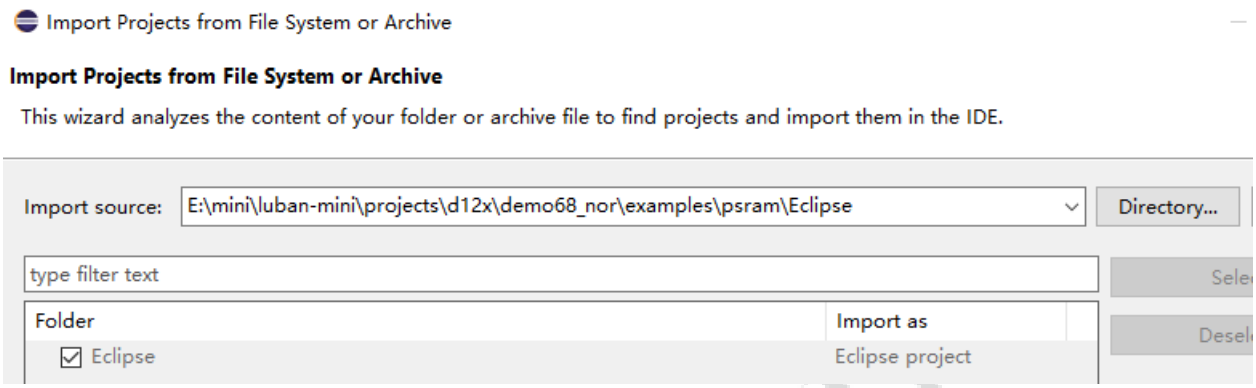


图 2–2 打开项目

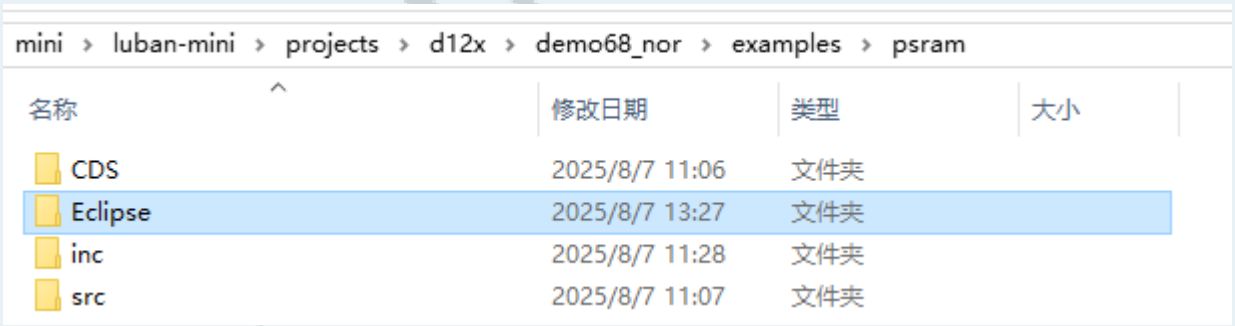
3. 在 Import Projects from File System or Archive 界面的 Import source 处，选择项目所在的 Eclipse 文件夹作为项目路径。



注：

项目文件默认路径 `luban-mini/projects/d12x/demo68_nor/examples` 下包含多个示例，每个示例均为一个独立的项目。

示例中的项目文件存放在 Eclipse 文件夹中，如下所示：



4. 点击 Finish 完成项目导入，Eclipse 工具会自动检测到项目。

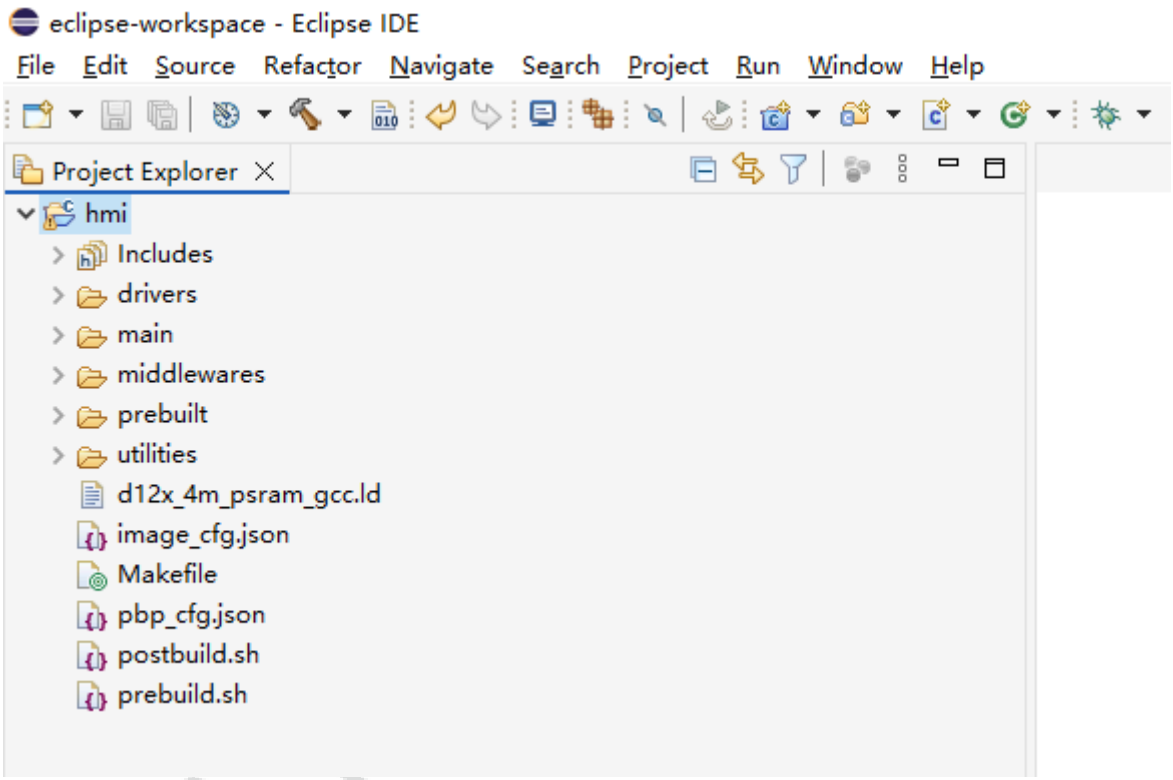


图 2–3 完成导入

2.3.1.2. 项目配置

本节将介绍 Eclipse 中的项目配置操作：

- 项目配置页面
- 常用配置页面
- 添加 include 目录
- 设置链接脚本
- 切换 debug/release
- 设置 prebuild/postbuild
- 配置多线程编译
- 设置 LinkedResources

2.3.1.2.1. 基本项目配置

2.3.1.2.1.1. 进入项目配置页面

通过以下任意方法可以进入项目配置页面：

- 在 Project Explorer 下选中项目后，点击鼠标右键选择 Properties 进入。
- 点击菜单栏中的 Project ＞Properities＞ 进入，如下所示

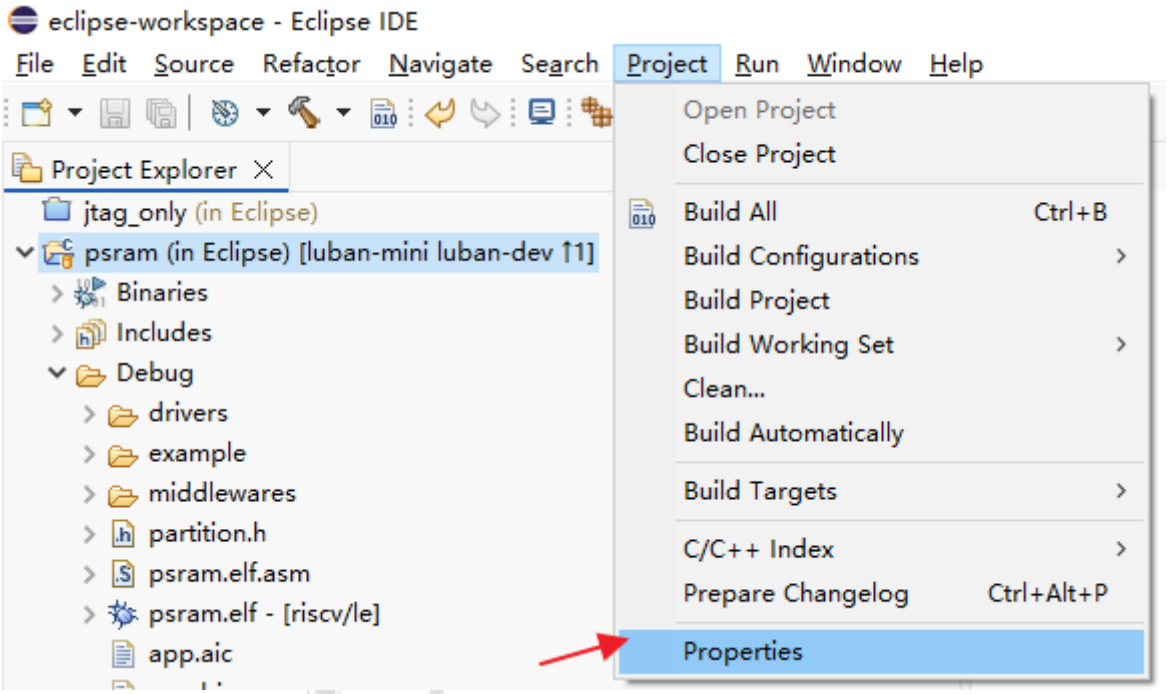


图 2–4 项目设置

2.3.1.2.1.2. 常用配置页面

在 `C/C++ Build/Settings` 页面，可查看常用的项目配置和编译相关配置。

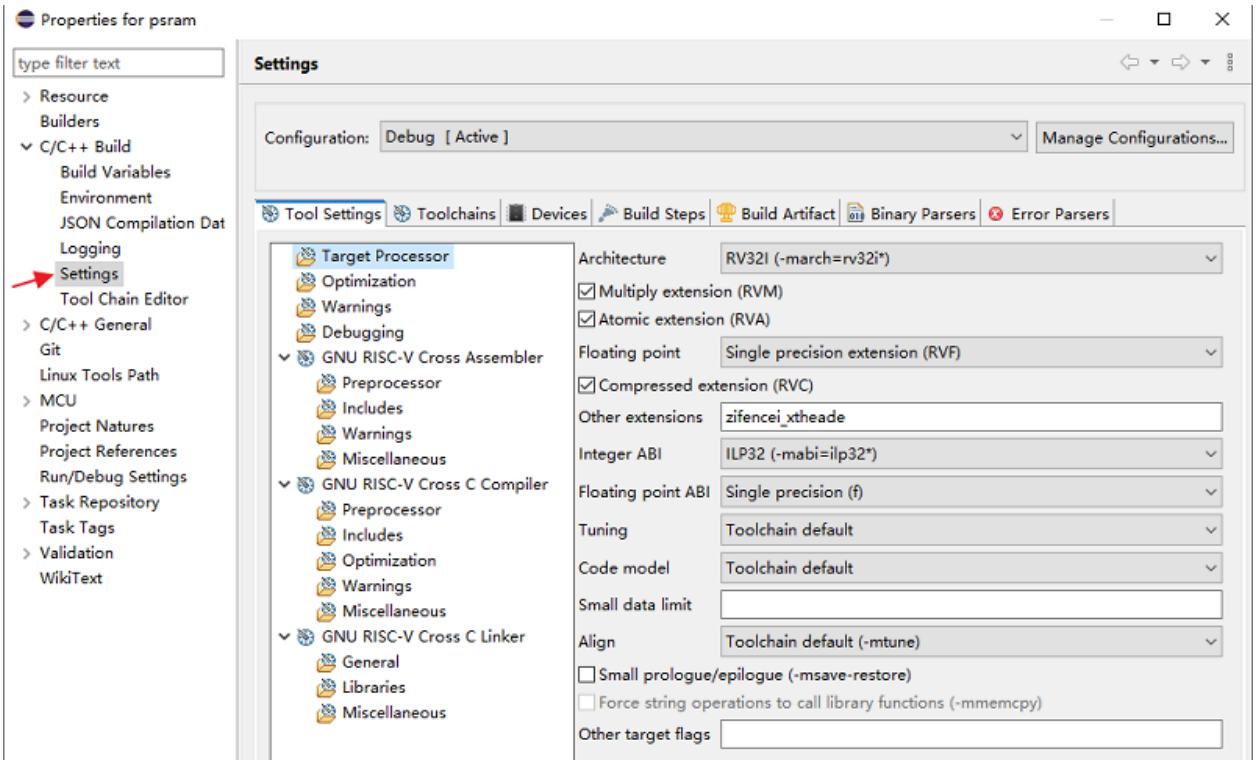


图 2–5 项目配置

2.3.1.2.1.3. 添加 includes 目录

添加汇编代码引用的 'include' 目录：

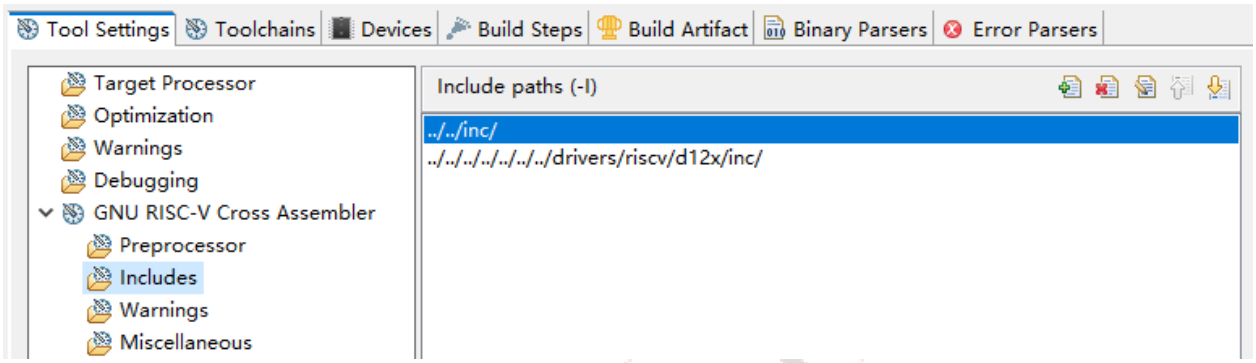


图 2–6 添加 include 目录

添加 C 源码引用的 'include' 目录：

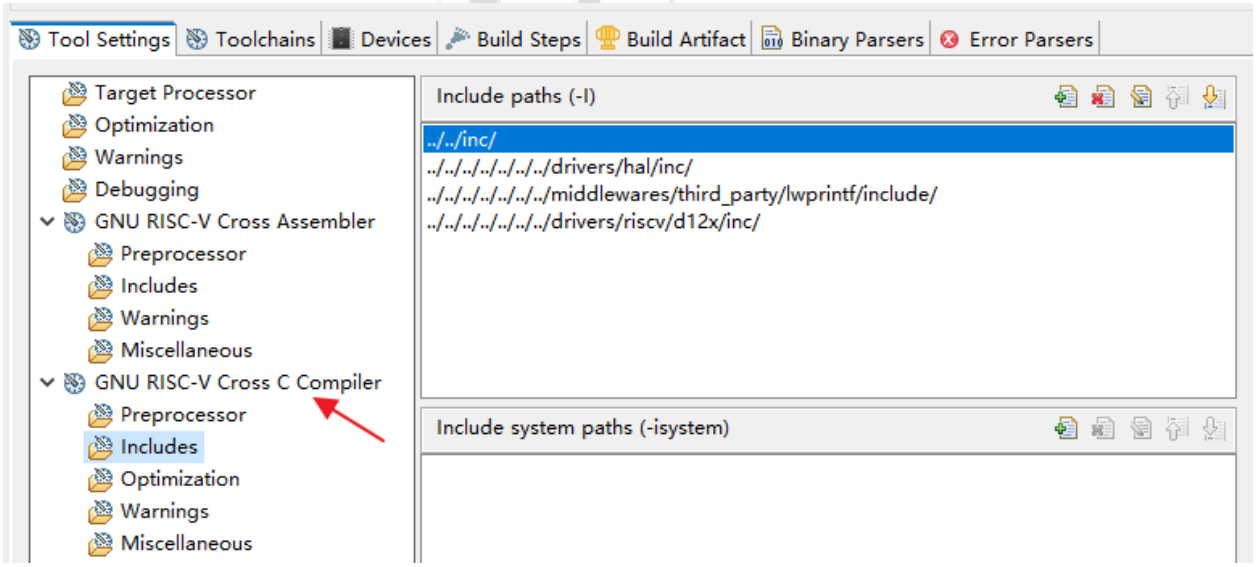


图 2–7 添加 include 目录

2.3.1.2.1.4. 设置链接脚本

在链接脚本配置项中，指定路径：

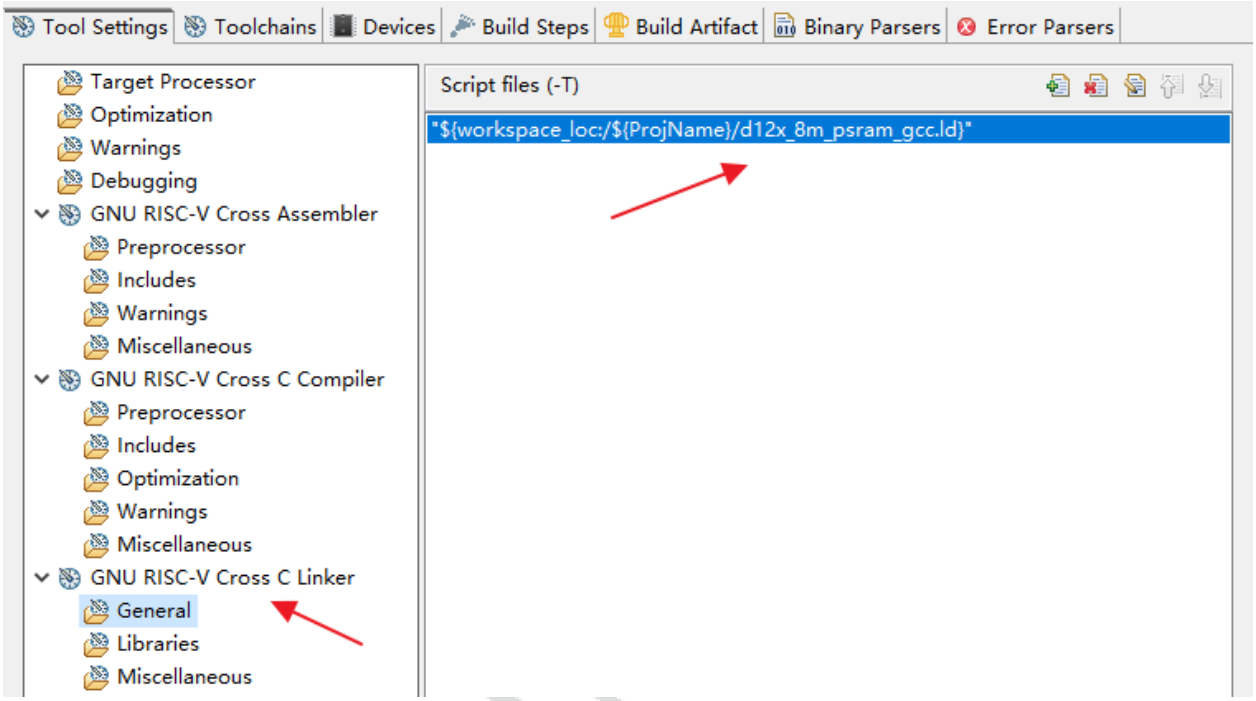


图 2–8 链接脚本

2.3.1.2.1.5. 切换 Debug/Release

切换构建配置：

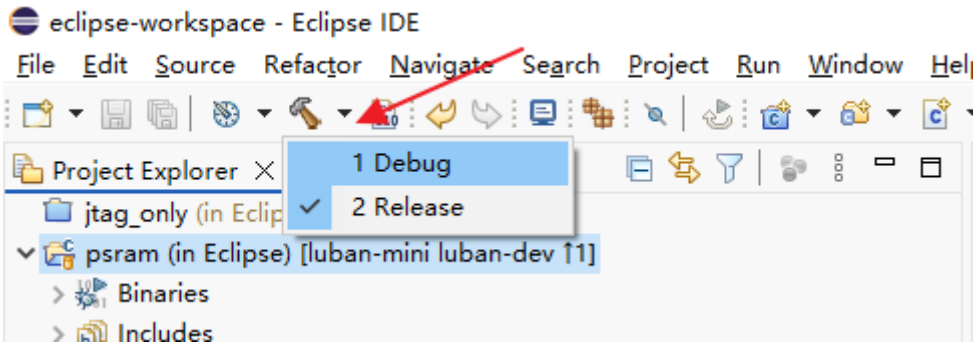


图 2–9 构建配置

2.3.1.2.1.6. prebuild/postbuild

项目支持添加 prebuild.sh 和 postbuild.sh 脚本，用于执行编译前后自动化处理。例如，可在 postbuild.sh 中添加生成 AiBurn image 的处理。

运行环境说明： AIC 提供的 Eclipse 包集成了 Msys2 运行环境，脚本 prebuild.sh/postbuild.sh 将运行在 MinGW 的 Bash 中，可以按照 Bash Shell 语法来编写。

脚本的运行配置在：

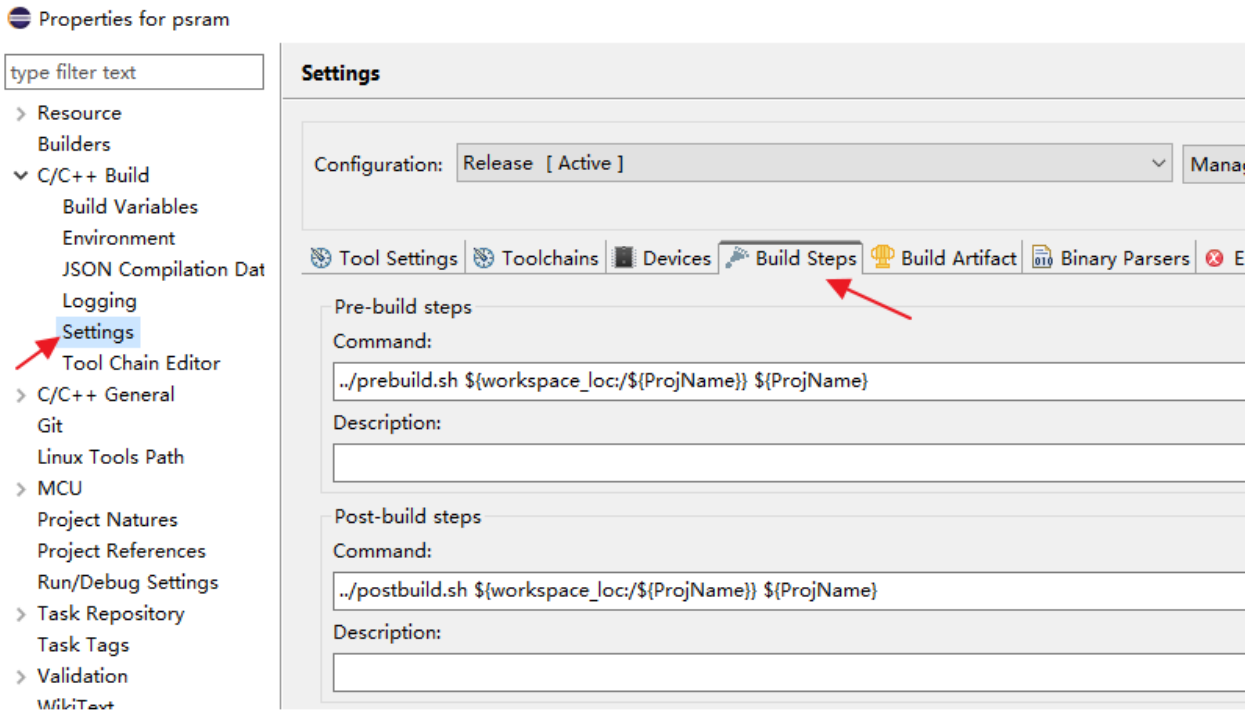


图 2–10 prebuild/postbuild

2.3.1.2.1.7. 多线程编译

单线程编译方便调试编译过程，而多线程编译速度更快。

在开发过程中，可以根据需要勾选或取消下图中的 Enable parallel build 进行切换：

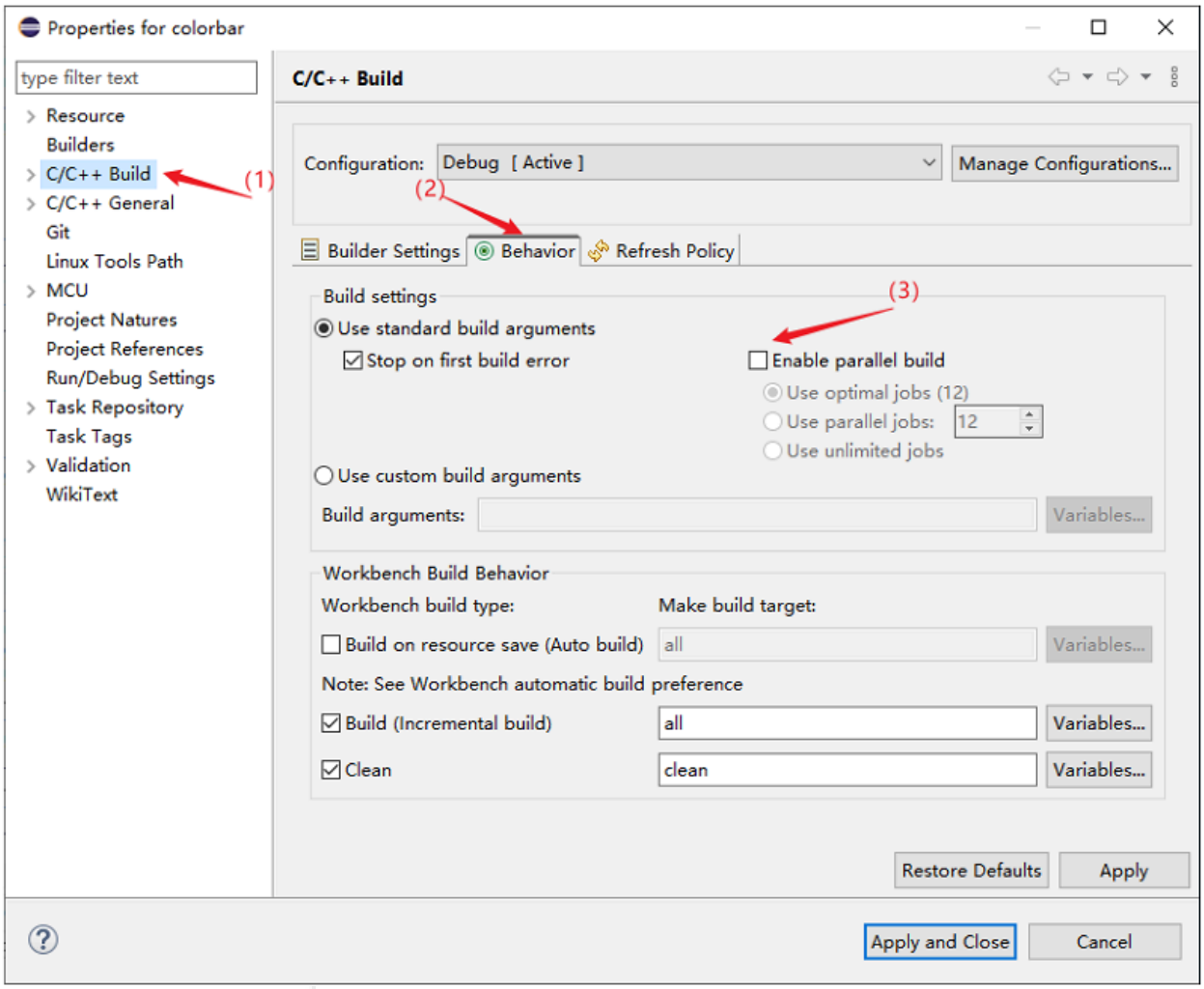


图 2-11 多线程编译

2.3.1.2.2. LinkedResources 设置

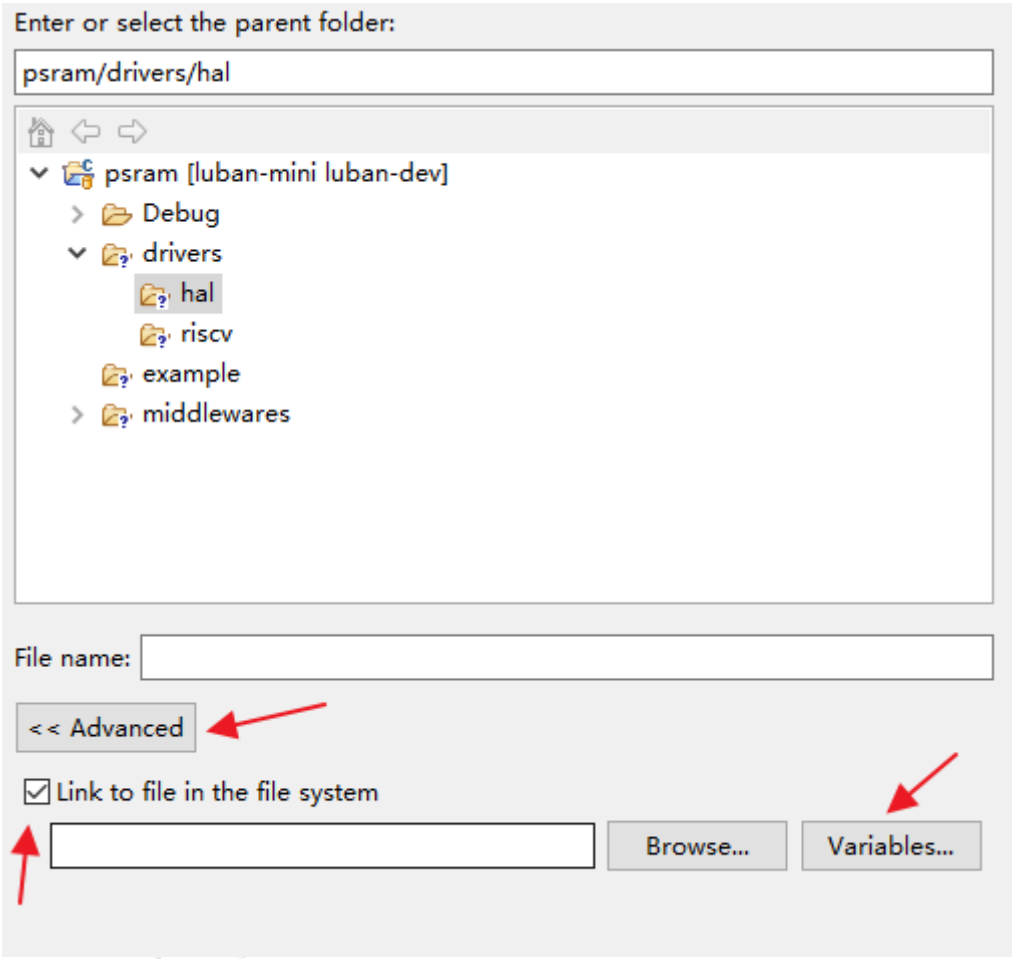
使用 Eclipse 打开一个项目时，默认会将项目文件目录以及子目录的所有文件加入到工程中，编译时也会自动加入编译。

> 注：
>
> Eclipse 的项目文件后缀为 `.project` 和 `.cproject`。

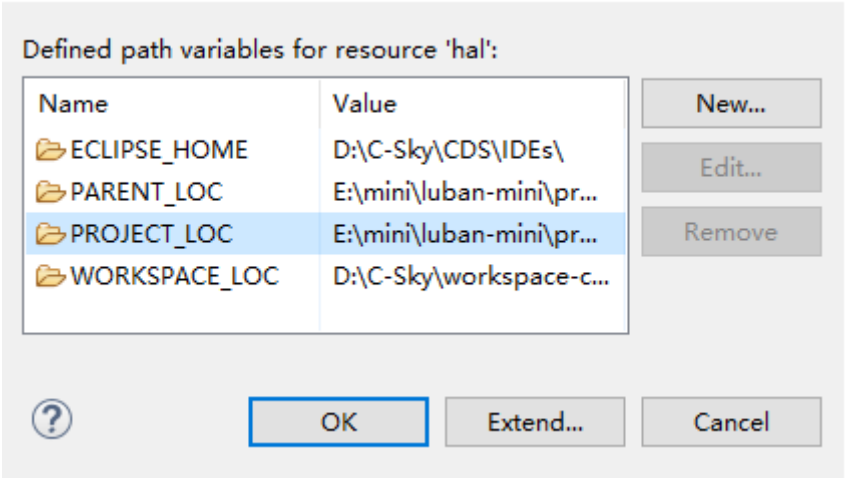
而在实际应用场景中，项目文件目录中可能不会包括所有源码文件，此时需要通过链接文件的方式将源码文件添加至项目文件目录中。

以在 `drivers/hal/src` 中新增一个 `hal_xxx.c` 文件且将其添加到项目的 `drivers/hal` 目录下为例，详细操作步骤如下所示：

1. 在目录 drivers/hal 上点击鼠标右键后，选择 New > File 菜单。
2. 在弹出的页面选择 Advanced 后，勾选 Link to file in the file system。

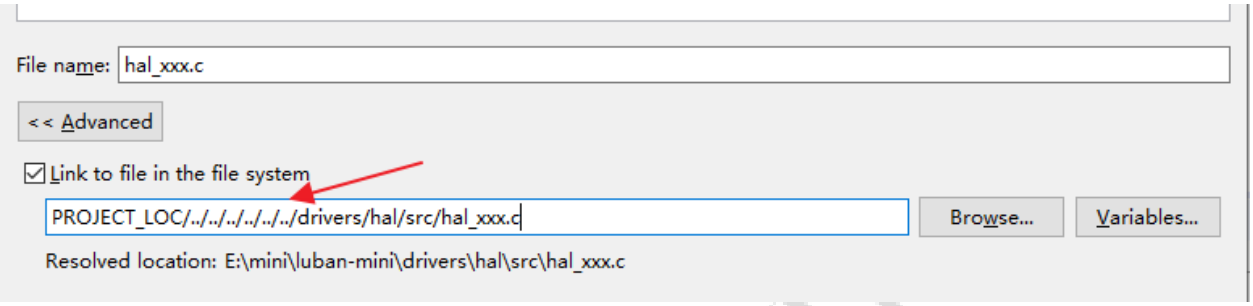


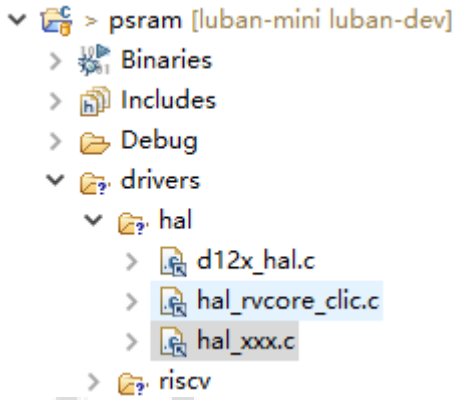
3. 点击 Variables… 后，在弹出窗口中选择 PROJECT_LOC， 即当前项目文件所在的目录路径：



4. 选择以下任意一种方式，添加 PROJECT_LOC 路径：

- 界面操作：填写基于 PROJECT_LOC 的文件相对路径填写完后，点击 Finish 即可。





- 手工编辑：编辑 .project 文件的方式添加：

```
<link>
    <name>drivers/hal/hal_xxx.c</name>
    <type>1</type>
    <locationURI>PARENT-6-PROJECT_LOC/drivers/hal/src/hal_xxx.c</locationURI>
</link>
```

注：

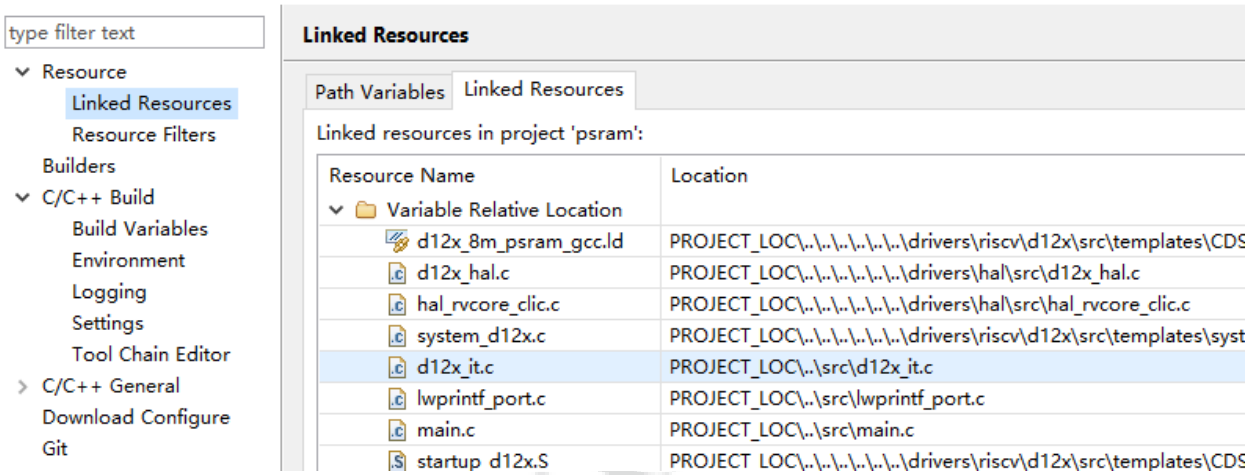
PARENT-6-PROJECT_LOC 中的 PARENT-6：基于 PROJECT_LOC 往上六层相对目录的意思，即 ../../../../../../。

PARENT-6-PROJECT_LOC/drivers/hal/src/hal_xxx.c: 即为`PROJECT_LOC/../../../../../../drivers/hal/src/hal_xxx.c`

5. 按照以下步骤可查看当前项目的 LinkedResources：

a. 打开项目配置页面。

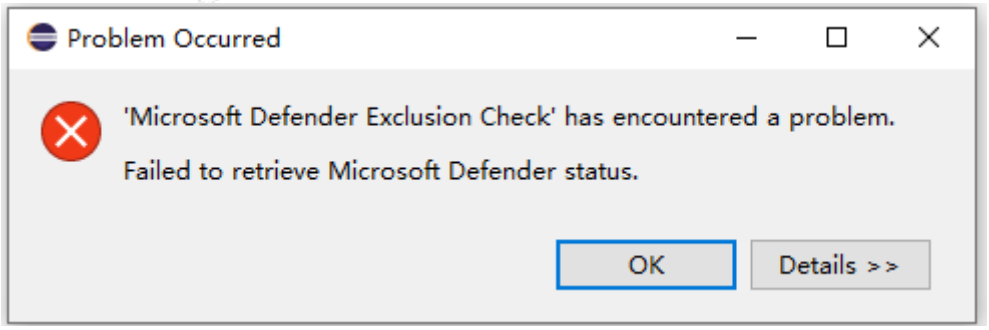
b. 选择 Resource > Lined Resources > Linked Resources



2.3.2. 常见问题(FAQ)

2.3.2.1. Microsoft Defender Exclusion Check Error

在打开 Eclipse 时，可能会遇到 Microsoft Defender Exclusion Check 功能检测不到 Microsoft Defender 状态的错误弹窗，如下图所示：



上述错误可能与使用的 Windows 版本配置有关，按照下列设置方法可阻止弹窗：

1. 在菜单栏中，选择 Window > Preferences：

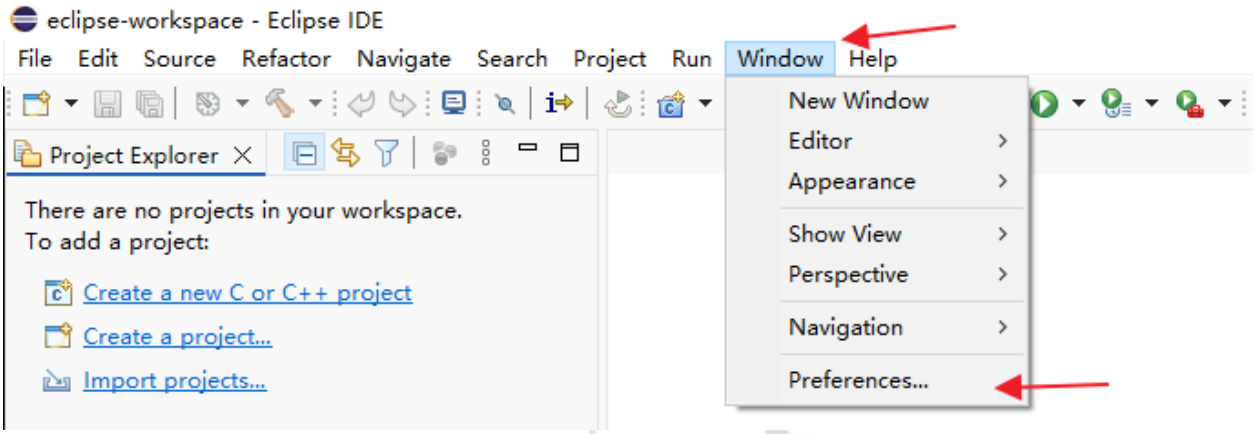


图 2–12 Preferences

2. 在弹出的 Preferences 配置窗口中，打开 General > Startup and Shutdown 配置页，并勾选 Skip exclusion check on startup for all new Eclipse–based installations 选项：

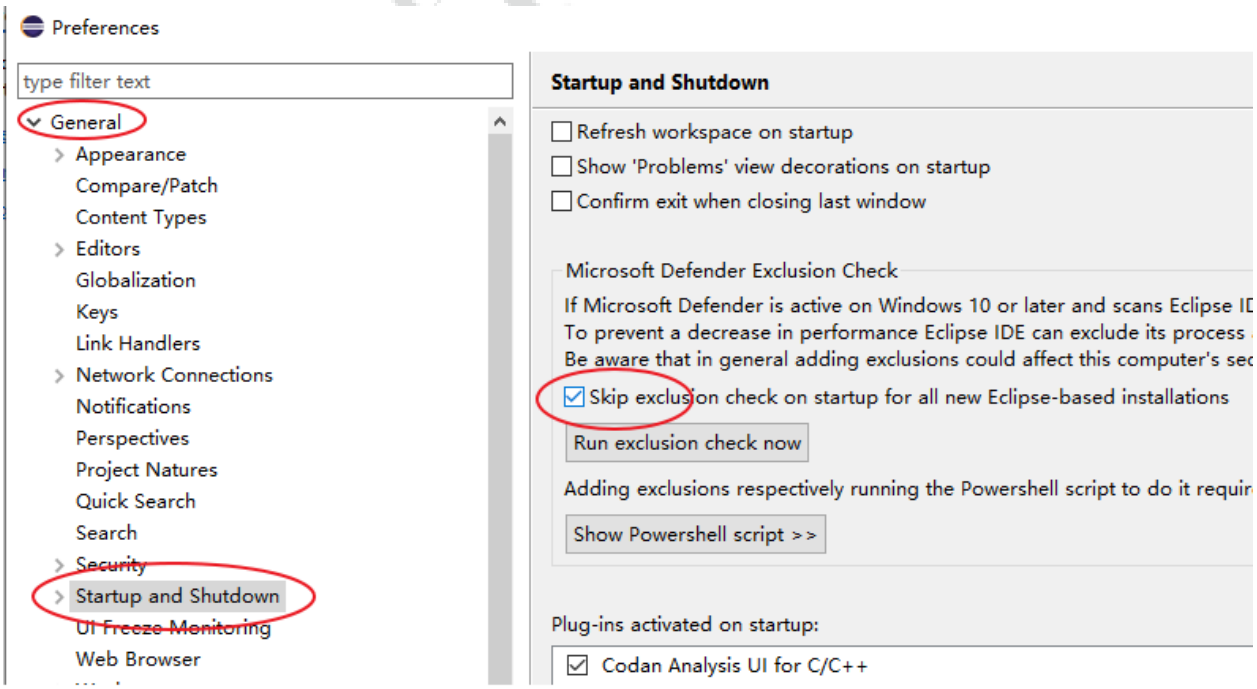


图 2–13 Startup and Shutdown 配置

2.4. JTAG 调试工具

对于 JTAG 调试环境的设置，涉及以下软件和硬件工具：

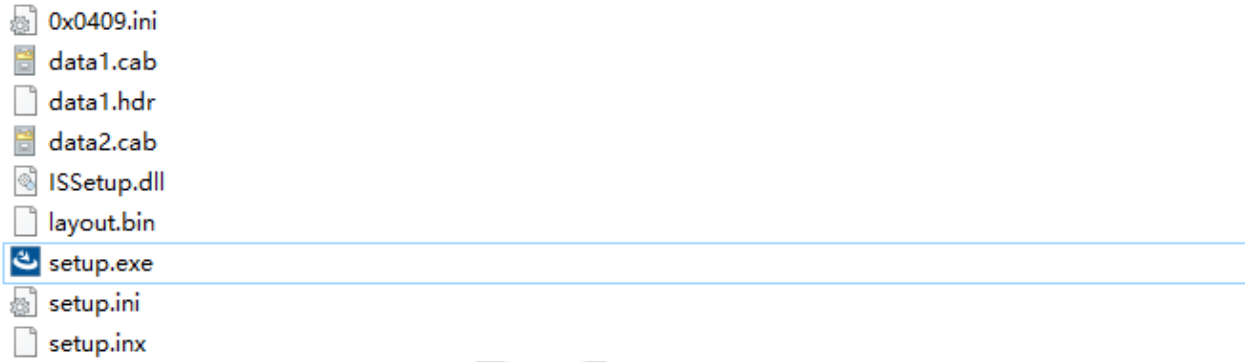
- Xuantie DebugServer 主机工具
- AIC–JTAG 调试器

2.4.1. 安装和配置 DebugServer

本节介绍了 DebugServer 的安装和设置。

2.4.1.1. 安装 DebugServer

解压 XuanTie-DebugServer-windows-V5.18.3-20241119-1930.zip ， 双击 setup.exe 进行安装。



2.4.1.2. 设置 DebugServer

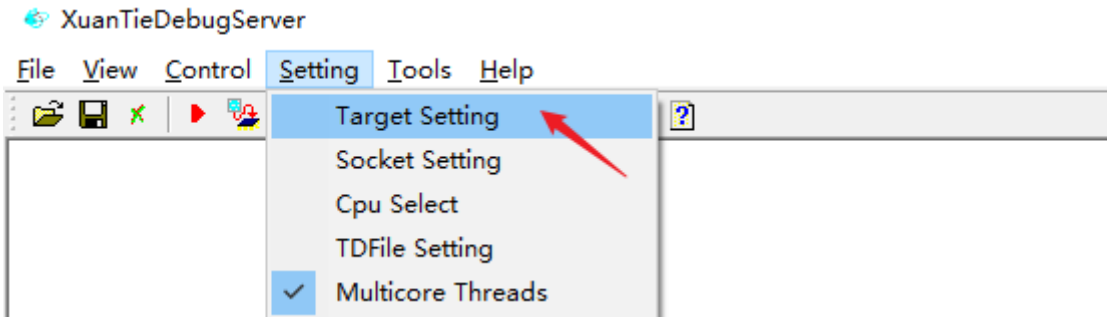
DebugServer 安装完成之后，一般无需额外配置。

在使用时过程中，主要关注以下两项设置即可：

- Target Setting
- Socket Setting

2.4.1.3. 配置 Target Setting

在菜单栏中，选择Setting > Target Setting 进行配置：



注：

在配置 Target Setting 时，需注意 ICE Clk 和 Delay 的设置。

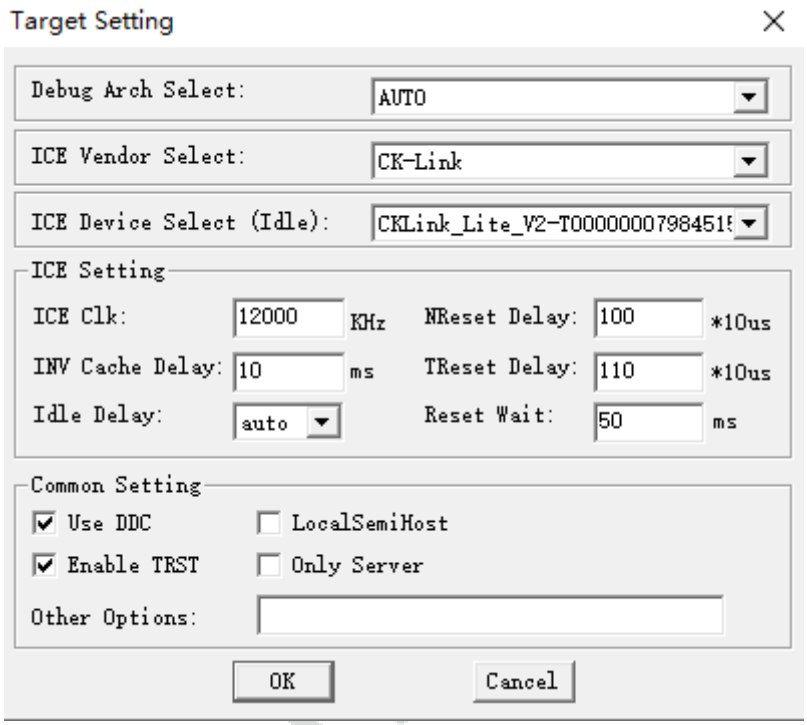
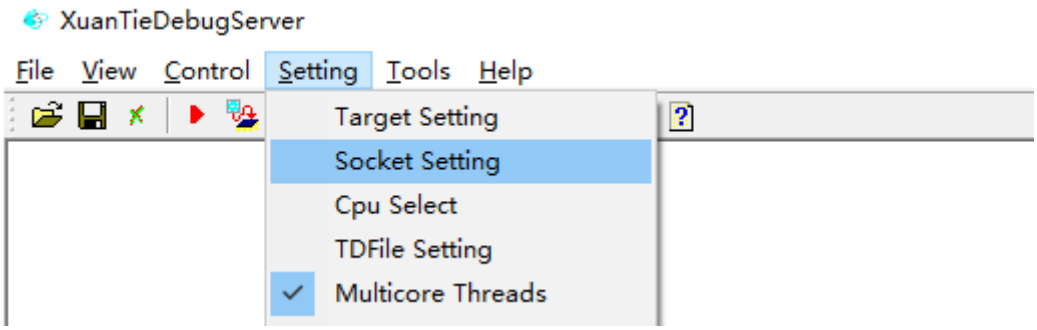


图 2-14 Target Setting 设置参考

2.4.1.4. 配置 Socket Setting

在菜单栏中选择 Setting > Socket Setting：



注：

确保选择一个没有冲突的 Socket 端口。

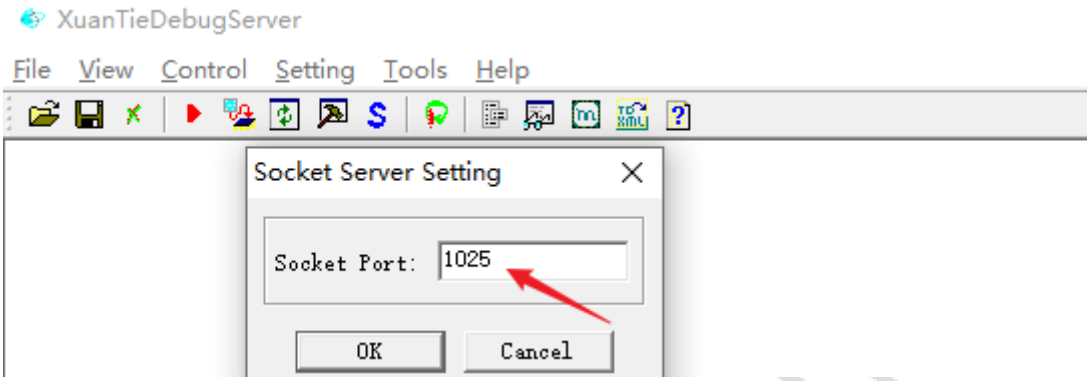


图 2–15 Socket Setting 设置参考

2.4.2. 连接 AIC–JTAG

本节介绍了如何将 AIC–JTAG 调试器与主机和板卡连接。

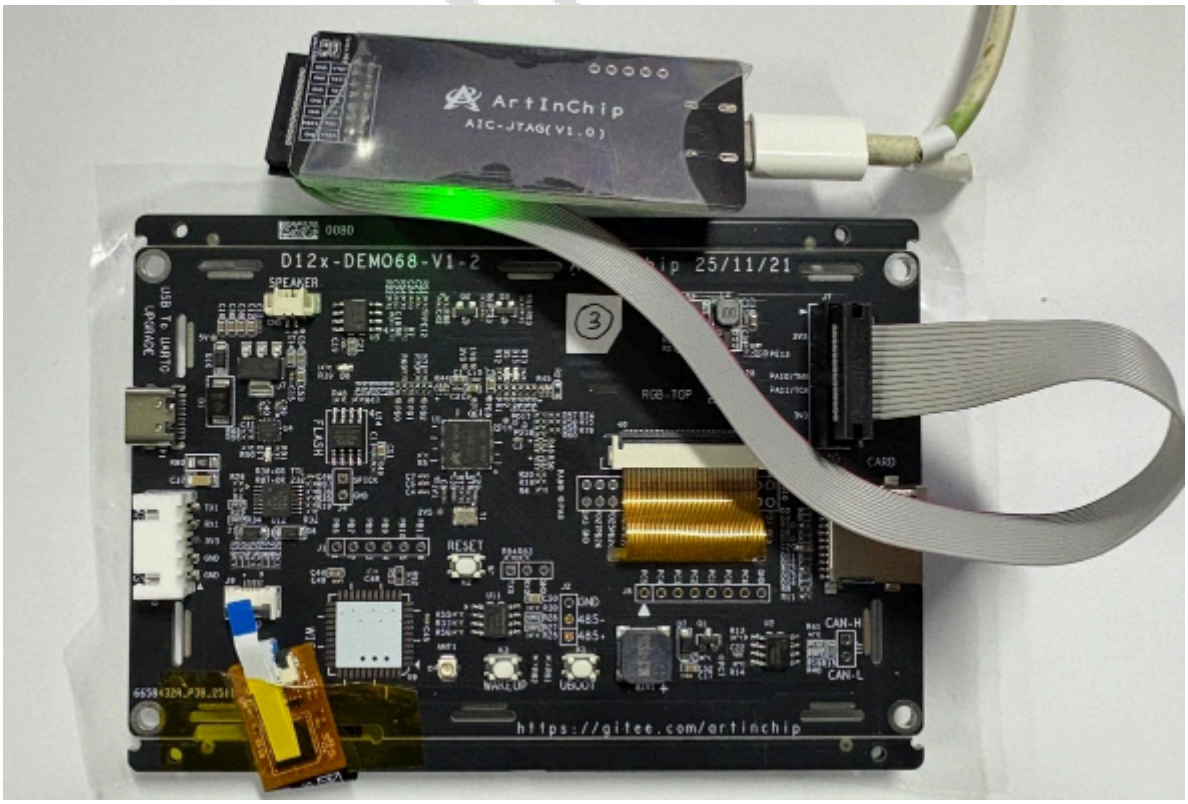


图 2–16 AIC–JTAG 连接板卡示意

等待板卡上电后，点击 DebugServer 上的连接按钮进行连接。

> 注：
>
> 如果出现连接失败，可排查目标板卡上的 JTAG 调试管脚是否被复用。

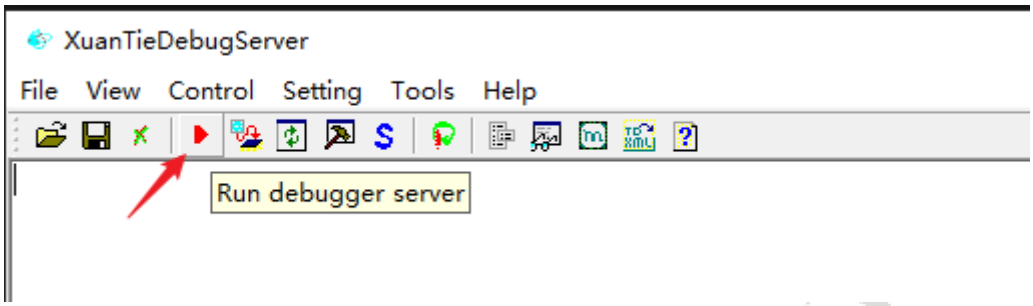


图 2–17 连接目标板卡

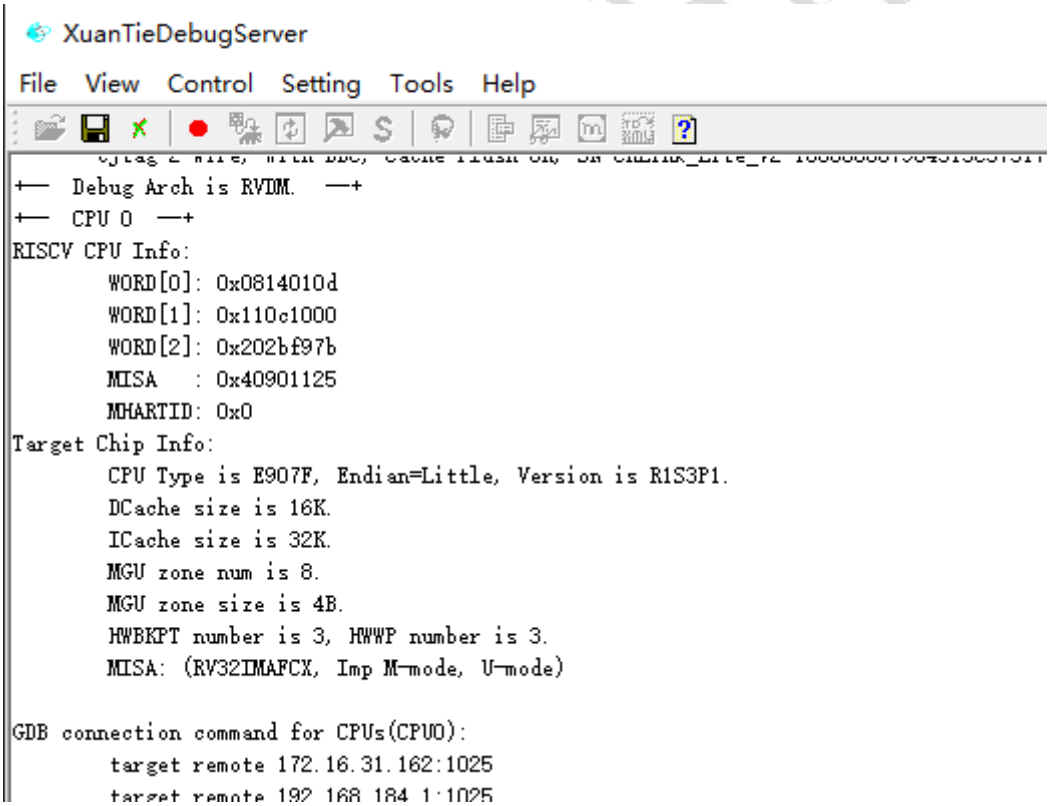


图 2–18 连接成功

3. 编译与调试

在编译项目之前，确保项目配置已完成。具体配置步骤参考项目配置

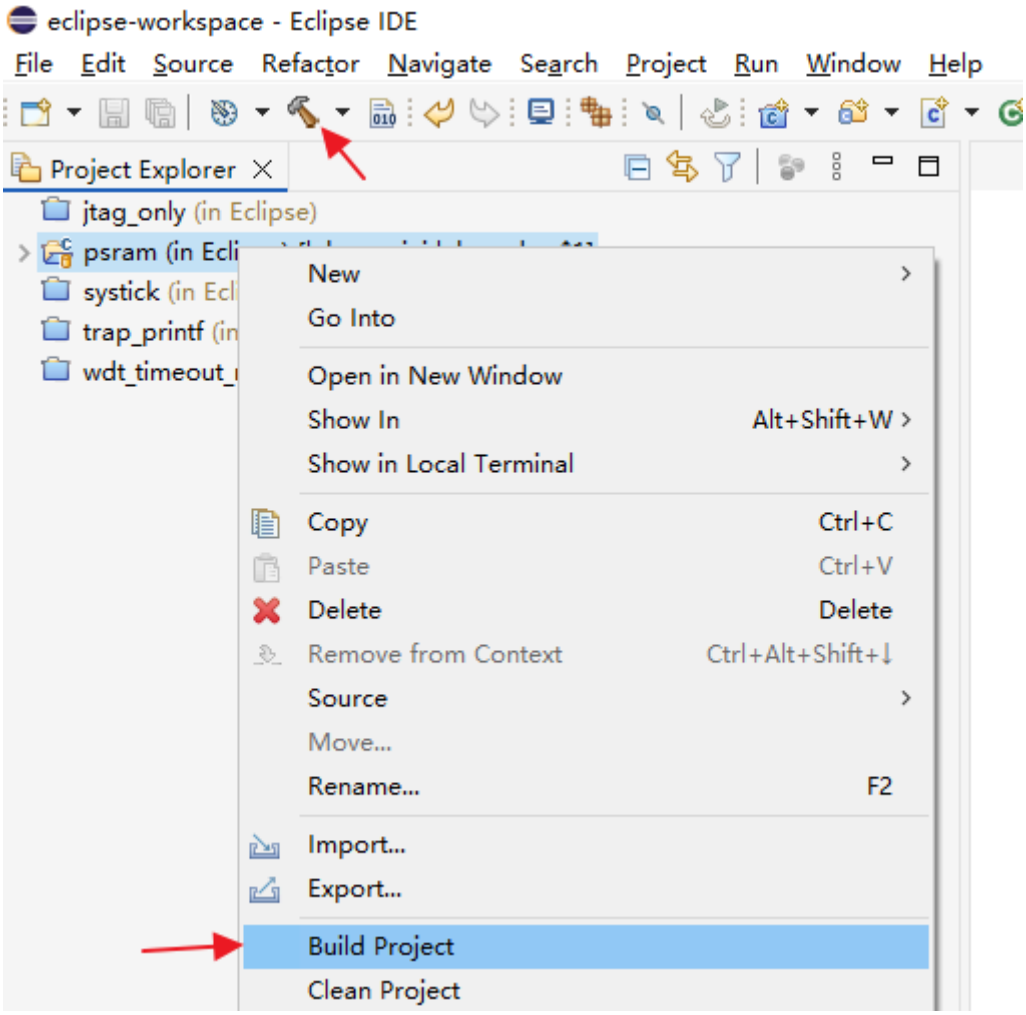
本节介绍了 Luban Mini SDK 开发过程中的编译、调试与烧录操作：

- 项目编译
- 添加 Debug Configuration
- 下载调试
- 编译运行示例

3.1. 项目编译

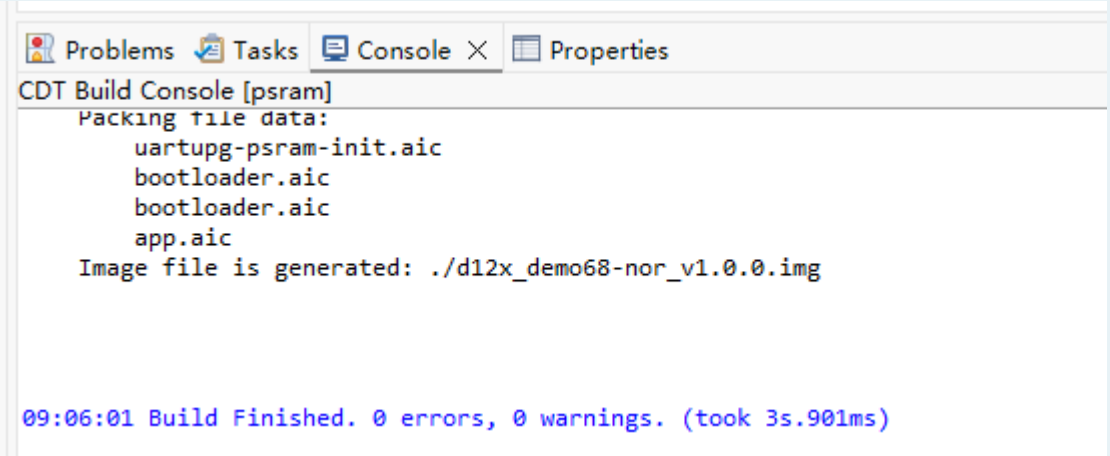
通过以下任意一种方式，编译当前项目：

- 点击工具栏中的 Build 按钮（锤子形状图标）
- 选择项目右键，选择菜单中 Build Project
- 使用快捷键 CTRL+B

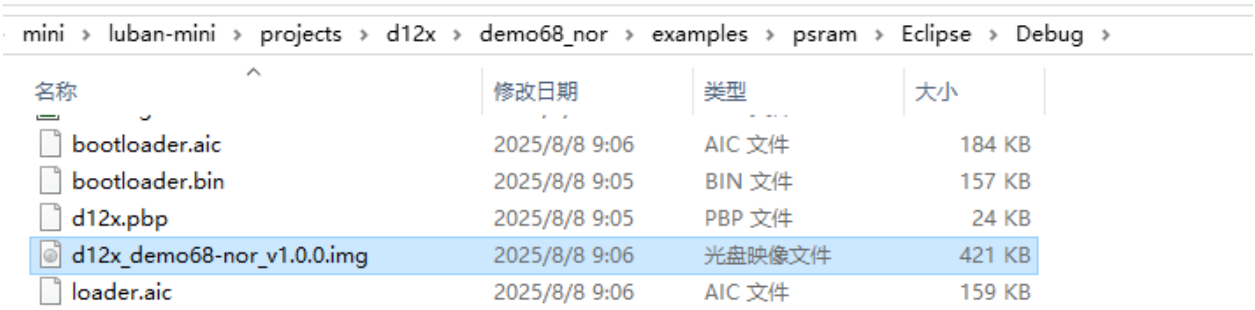


注：

在 Console 窗口可以看到编译过程：



编译输出文件，在 Eclipse 项目目录下，根据当前配置，输出结果放在 `Debug` 或 `Release` 目录中。



3.2. 添加 Debug Configuration

首次调试需要先创建调试配置。点击 Debug 后，会弹出 Debug Configurations 配置窗口。

配置步骤如下：

1. 双击 GDB Hardware Debugging，创建新的调试配置

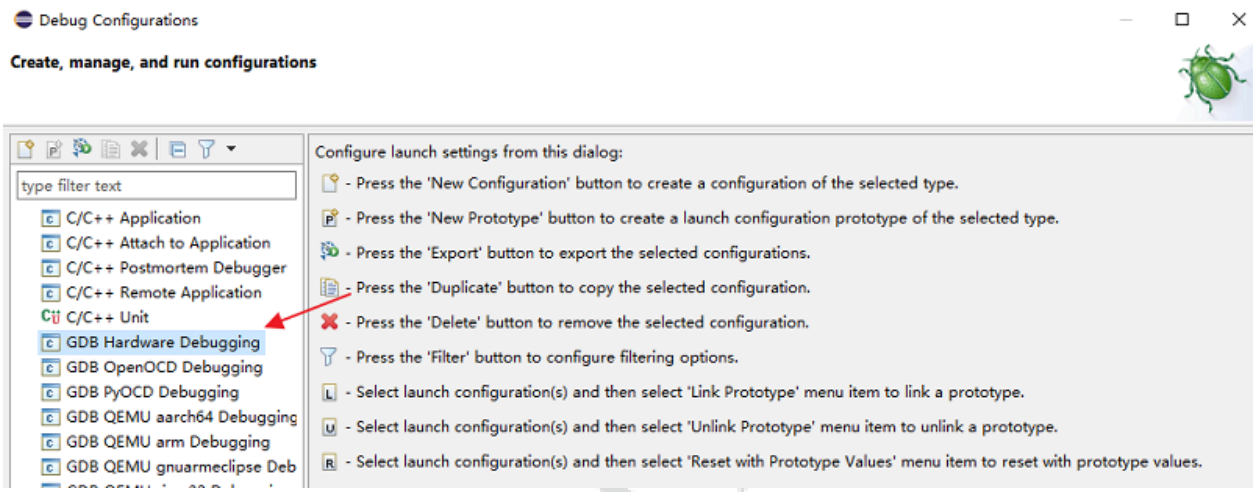


图 3–1 Debug Configurations 配置窗口

2. 点击 Search Project，选择要调试的 ELF 文件

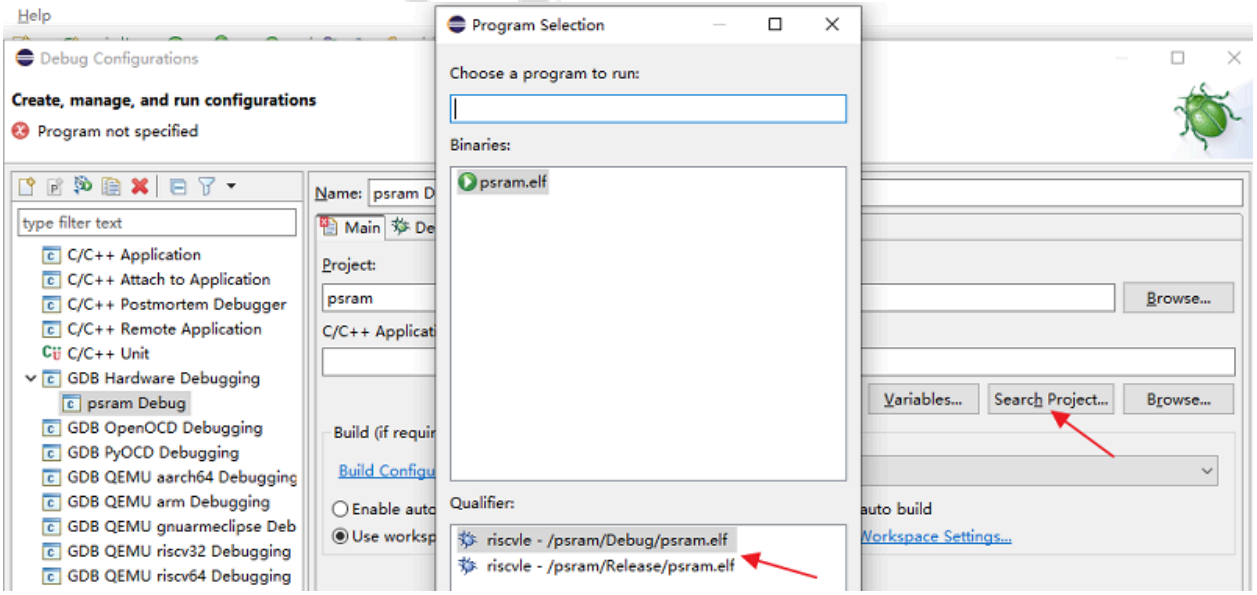


图 3–2 选择 ELF 文件

3. 设置 Debug Server 的端口号。

注：

根据本地运行的 XuanTie Debug Server 端口号，填写到 `Connection` 中，然后点击 Debug 按钮即可开始调试。

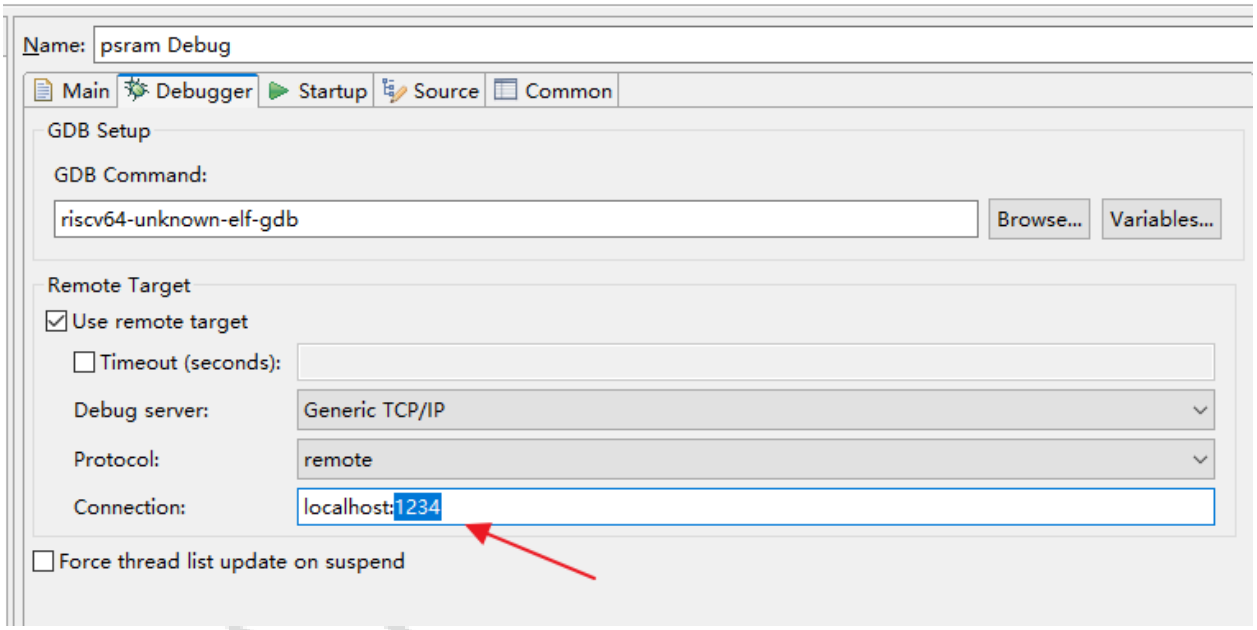


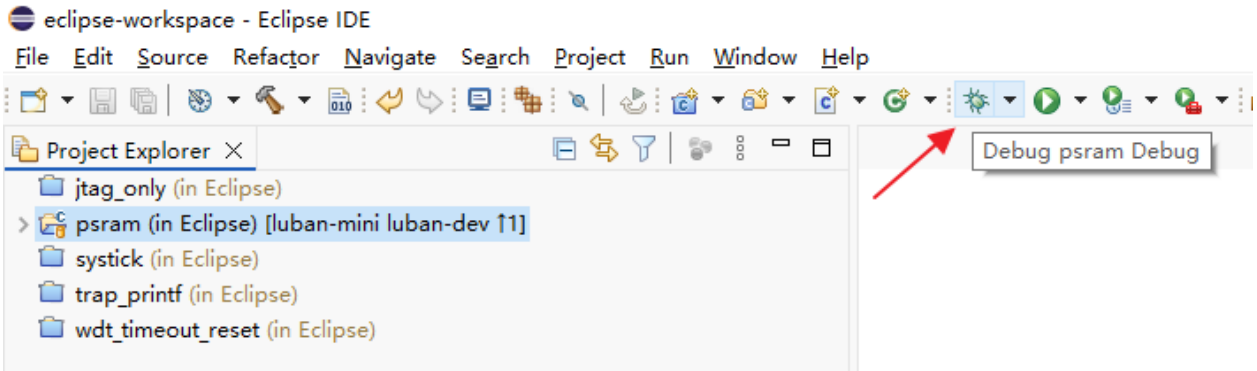
图 3–3 设置端口号

3.3. 下载调试

注：

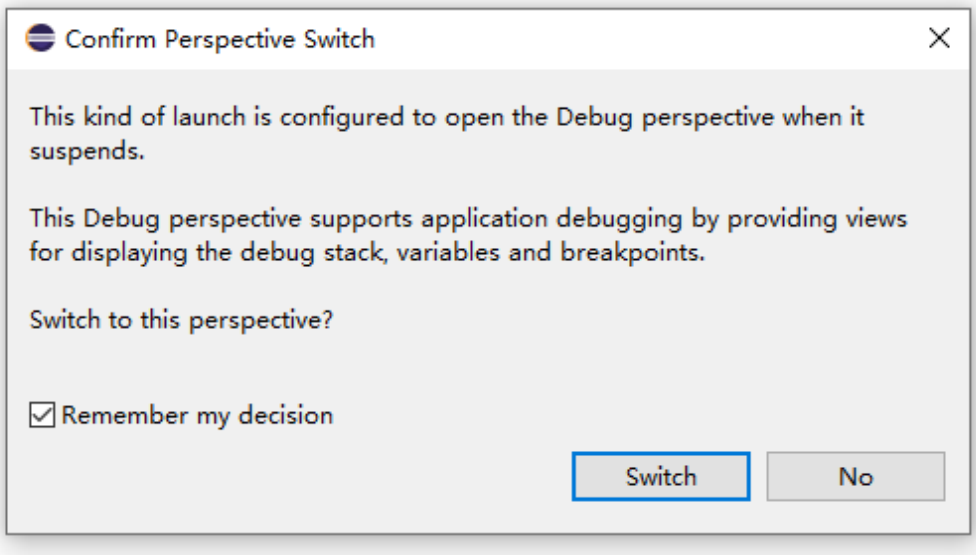
对于 D12x 系列，通常需要使用 PSRAM，因此为了方便 GDB 下载调试，建议先编译 psram 项目并将生成的固件烧录到 Flash 之后再开始调试。

1. 正式调试之前需要确保调试配置已完成，调试配置可参考添加 Debug Configuration。
2. 完成后，点击 Debug 图标即可开始下载 ELF 进行调试。

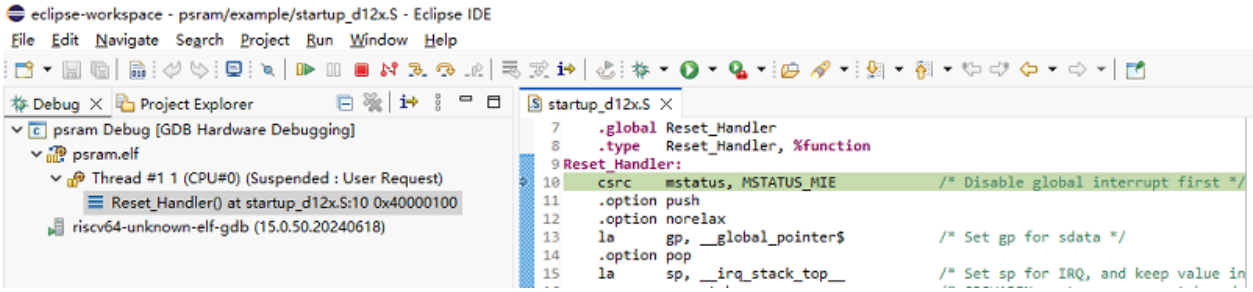


调试过程中常用的快捷键操作，参见调试快捷键

3. 如果弹出视图切换弹框，勾选Remember my decision并且点击 Switch 按钮即可，否则可跳过此步。



4. 调试开始时，会先停留在第一条指令，此时可以直接继续运行，或者新增调试断点后运行。



注：

根据以上操作后，板子每次上电会自动初始化 PSRAM，后续可直接使用 GDB 将应用程序下载到 PSRAM 地址空间进行调试。

3.4. 烧录

编译生成的 img 文件，存放在项目目录下的 `Debug` 或 `Release` 文件夹中。

当前支持使用 AiBurn 或者 SD 卡进行烧录。

详细的烧录操作步骤如下：

- TF 卡烧录：
  1. 准备一张 TF 卡，格式化为 FAT32 文件系统。如果该卡有多个分区，要求文件系统在卡的第一个分区。
  2. 将编译生成的 bootcfg.txt, d12x_demo68-nor_v1.0.0.img 复制到 TF 卡的根目录。
  3. 将 TF 卡插入到板卡的 TF 卡口，重新上电进行烧录。
  4. 烧录完成后，拔卡，系统重启执行烧录的固件。

- AiBurn 烧录：

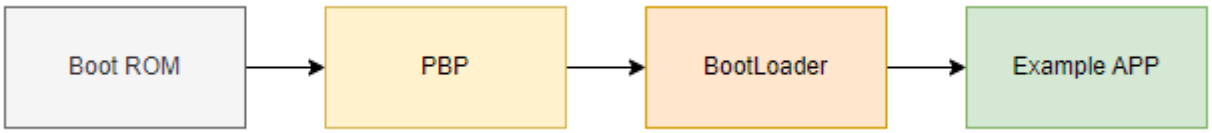
1. 板卡上电后，同时按下 UBOOT 按钮 和 RESET 按钮，让板卡进入 BROM 升级模式。
2. 打开 AiBurn，选择串口烧录模式。
3. 选择正确的 COM 口

> 注：
>
> D12x-DEMO68-V1-2 板卡的烧录串口为 UART0，与供电 USB 口共用。

4. 选择要烧录的 img 文件
5. 点击连接 > 开始，成功后，进行烧录。

3.5. 编译运行示例

D12x 芯片 SRAM 比较小，通常需要使用 PSRAM。SDK 所提供的示例代码，多数是运行在 PSRAM 之中。 由于 PSRAM 需要初始化才能用，因此使用 PSRAM 的示例，其运行流程如下：



- PBP (Pre-Boot-Program)：主要功能是做 PSRAM 初始化，使得 BootLoader 和 Application 可以在 PSRAM 中运行。

注：

PBP 和 BootLoader 在 SDK 中以 prebuilt 二进制的形式提供，在 `image_cfg.json` 中打包到最终烧录的固件。

3.5.1. 烧录运行示例

本示例以`projects/d12x/demo68_nor/examples/display/colorbar/Eclipse`为例，该示例完成对屏幕的初始化，以及芯片的显示模块初始化，显示色块。

烧录运行步骤如下：

1. Eclipse 编译示例：具体操作参照 导入项目，导入示例项目。

   编译后生成的固件在：

   - `projects/d12x/demo68_nor/examples/display/colorbar/Eclipse/Debug`
   - `projects/d12x/demo68_nor/examples/display/colorbar/Eclipse/Release`

2. 烧录：可通过以下两种方式烧录固件。
   - TF 卡烧录：具体操作参考TF 卡烧录
   - AiBurn 烧录：具体操作参考AiBurn 烧录
3. 重新上电运行固件

3.5.2. JTAG 调试示例

使用 JTAG 下载运行示例，需要注意先将 PSRAM 初始化。PSRAM 初始化需要通过烧录固件的方式完成，有以下两种方案可选。

- 方案一： 先使用 TF 卡/AiBurn 烧录的方式，将目标固件烧录到 Flash。

  注：

  后续使用 JTAG 调试前，先将板卡重新上电，待 PSRAM 初始化之后，可以使用 JTAG 下载的方式，将新的应用下载到 PSRAM 进行运行和调试。

- 方案二：编译 `projects/d12x/demo68_nor/examples/basic/psram/Eclipse` 并将该固件烧录到 Flash。

注：

方案一 在上电的时候会先跑一遍示例应用，在某些情况下会影响调试。此时可以编译烧录 `projects/d12x/demo68_nor/examples/basic/psram/Eclipse`， 该固件仅跑完 PBP，不会运行 BootLoader 和应用，运行环境更加单纯。

3.6. 调试快捷键

debug 模式快捷键

作用域	功能	快捷键
全局	单步返回	F7
全局	单步跳过	F6
全局	单步跳入	F5
全局	单步跳入选择	Ctrl+F5
全局	调试上次启动	F11
全局	继续	F8
全局	使用过滤器单步执行	Shift+F5
全局	添加/去除断点	Ctrl+Shift+B
全局	显示	Ctrl+D
全局	运行上次启动	Ctrl+F11
全局	运行至行	Ctrl+R
全局	执行	Ctrl+U

4. 新建应用程序

4.1. 基于示例创建

使用 IDE 零开始配置一个特定芯片、板卡的项目比较繁杂，为了简化新建项目的工作， SDK 提供 ProjectExtractor.exe 工具，使用该工具可以基于示例创建新的项目。在创建新的项目之前，确保开发环境已配置好，具体操作请参考环境准备。

具体操作步骤如下：

1. 打开 ProjectExtractor.exe

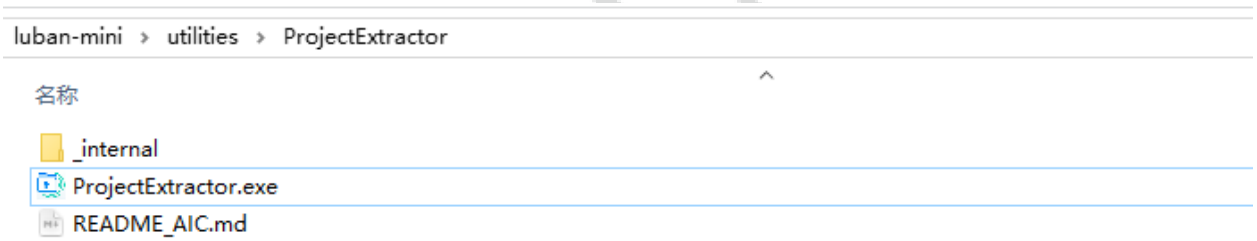


图 4-1 工具目录

2. 使用工具打开 SDK 目录：指定 SDK 的根目录，工具自动扫描 SDK projects 目录中的示例项目，将识别到的示例项目添加到项目列表。
3. 配置输出项目信息：
    a. 提供新项目的名字：工具在提取示例项目时，同步将项目名字修改为新提供的名字。
    b. 提供新项目的保存路径
    c. 在列表中选中要提取的项目
4. 提取项目：点击提取项目按钮，工具自动将指定项目提取到指定路径。

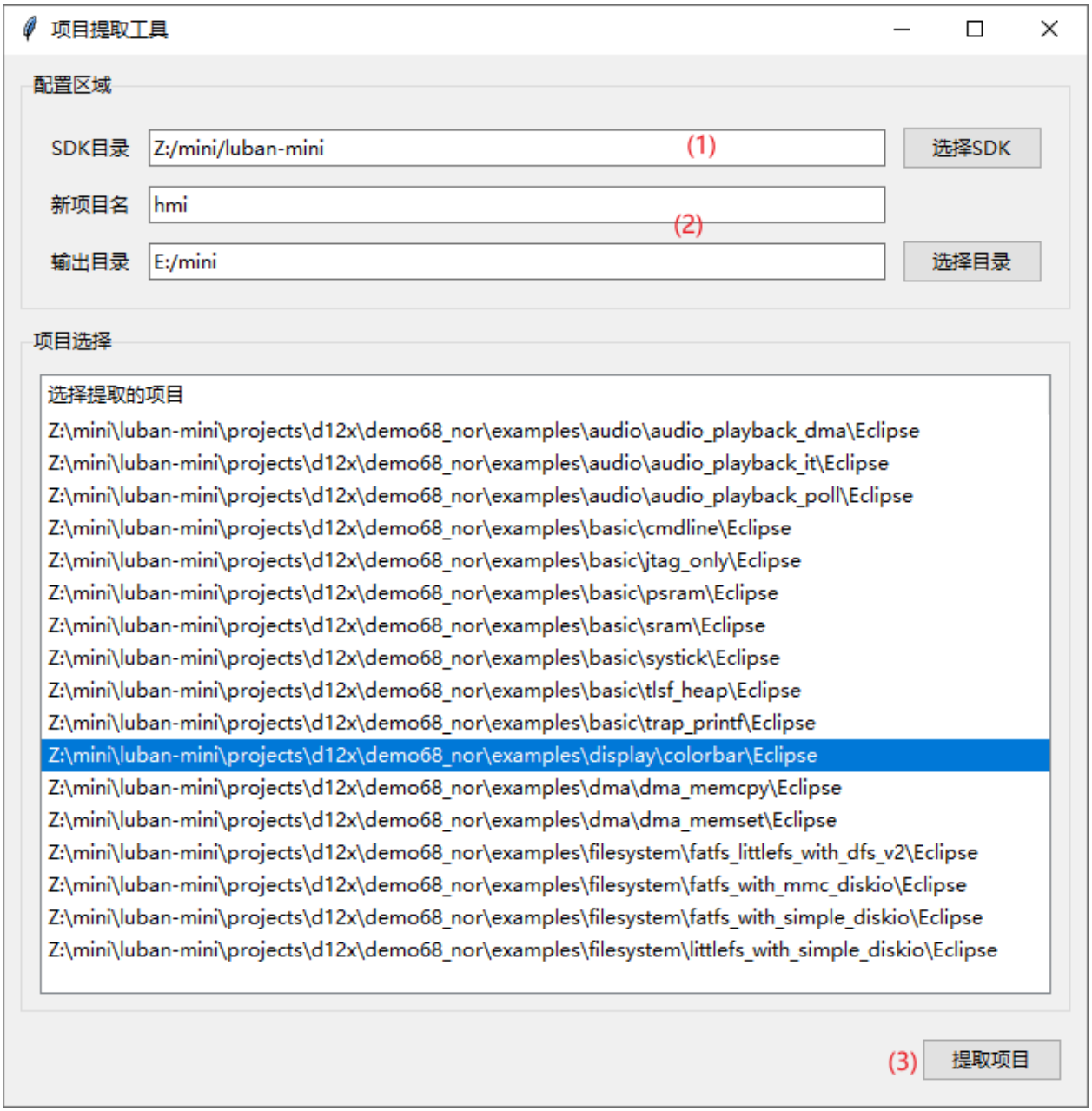


图 4–2 提取操作

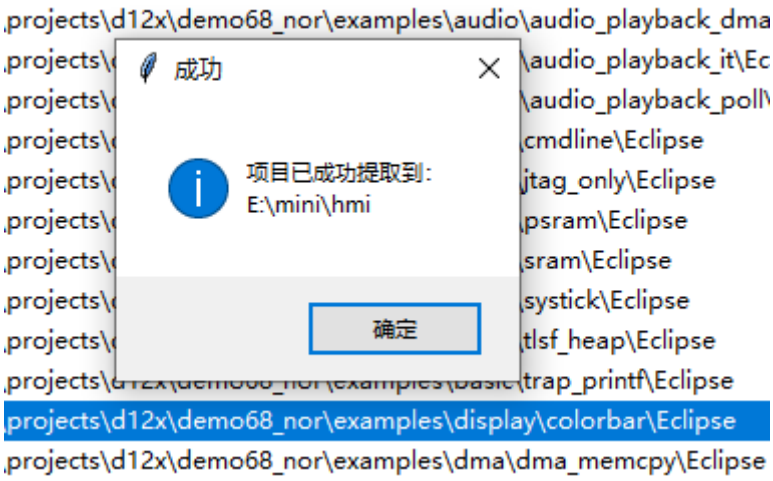


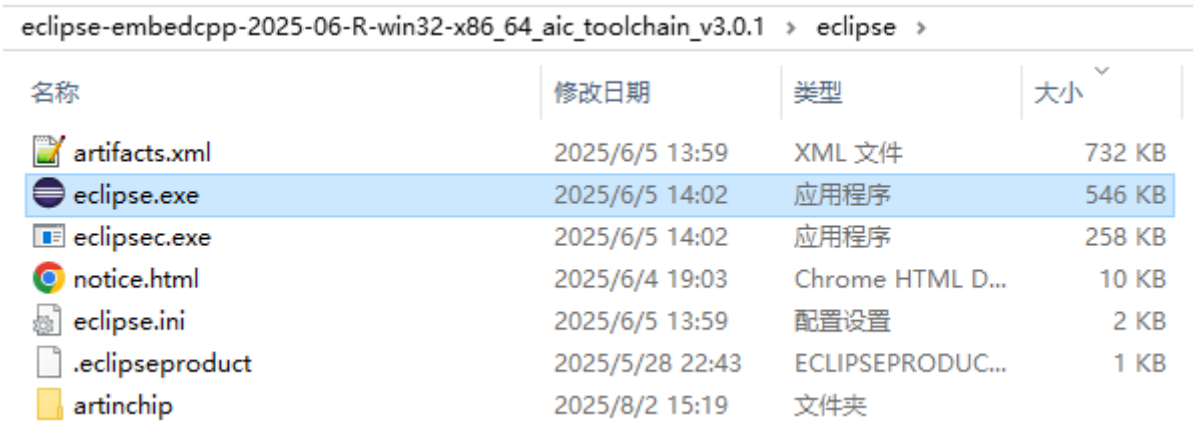
图 4–3 成功提示

4.2. 导入项目

4.2.1. 打开项目

用 Eclipse 打开项目的步骤如下：

1. 点击 Luban Mini Eclipse 打开 Eclipse。



> 注：
>
> 打开 Eclipse 时，可能遇到Microsoft Defender Exclusion Check Error 报错，可参考常见问题中的操作步骤。

2. 点击 FileOpen Projects from File System，从本地文件系统中选择一个项目。

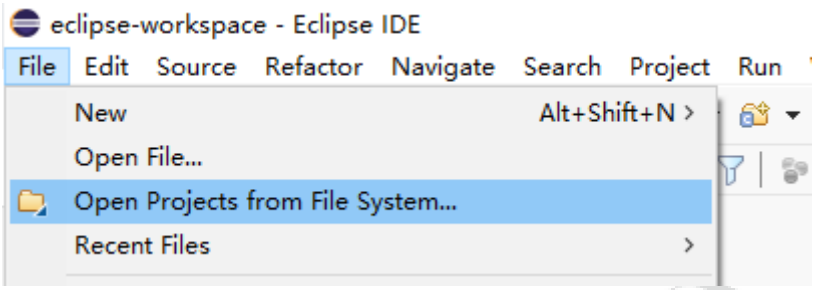
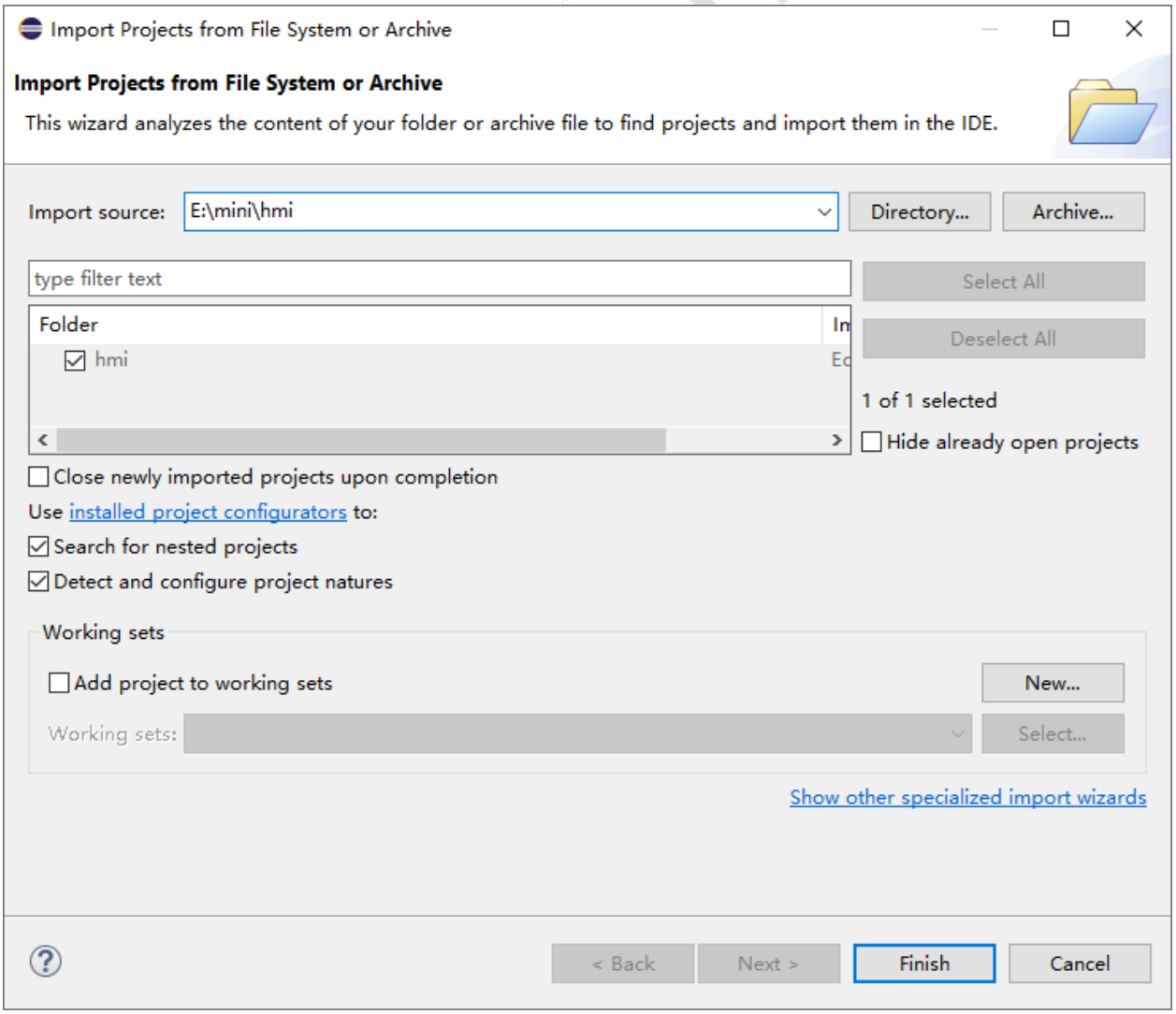


图 4–4 打开项目

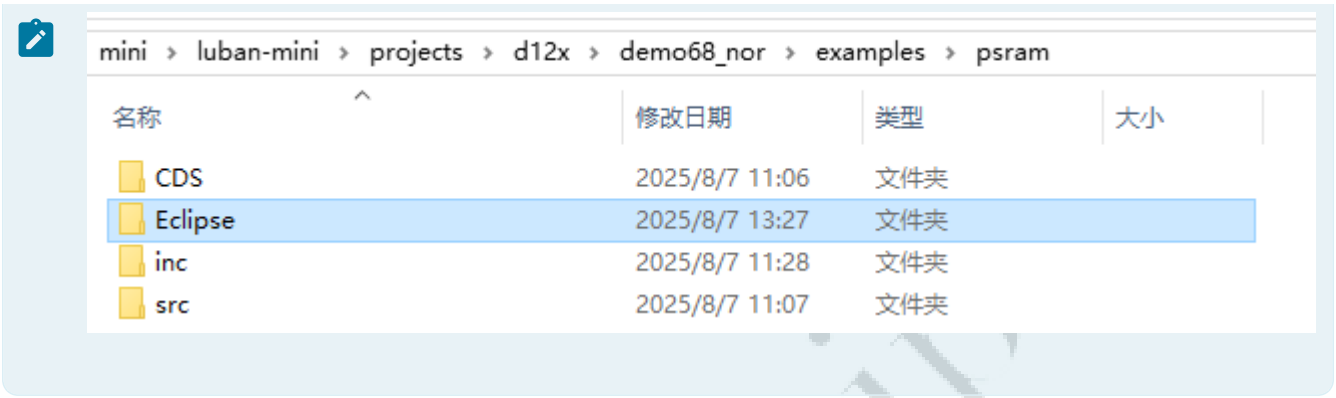
3. 在 Import Projects from File System or Archive 界面的 Import source 处，选择项目所在的 Eclipse 文件夹作为项目路径。



注：

项目文件默认路径 `luban-mini/projects/d12x/demo68_nor/examples` 下包含多个示例，每个示例均为一个独立的项目。

示例中的项目文件存放在 Eclipse 文件夹中，如下所示：



4. 点击 Finish 完成项目导入，Eclipse 工具会自动检测到项目。

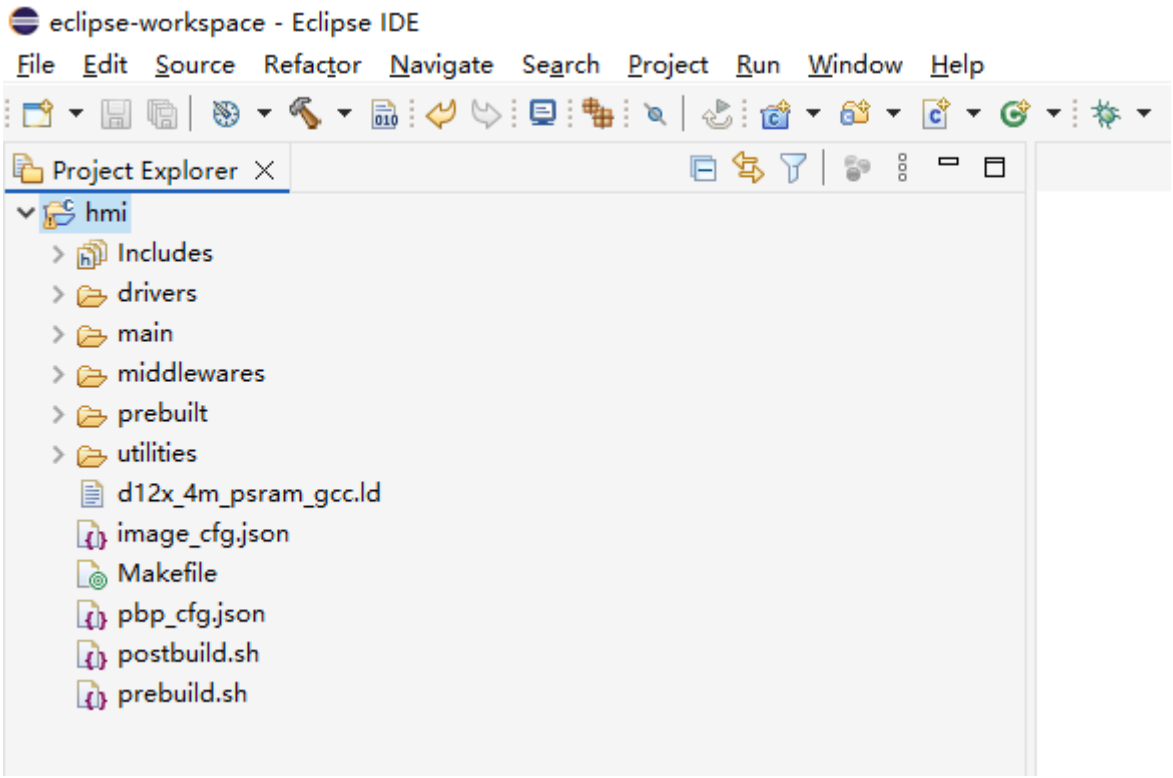


图 4–5 完成导入

4.3. 添加新源码包

新项目需要添加额外的代码到项目中。Eclipse 的项目文件 `.project` 和 `.cproject` 位于根目录，Eclipse 会自动将该目录以及子目录的源文件都加入到项目中进行编译。 因此往该项目添加源码和模块时，只需将源码文件或者源码目录添加到项目目录下。

例如，将 SDK 中的 SFUD 库添加到新项目中，直接将 `spinor_sfud`目录复制过来：

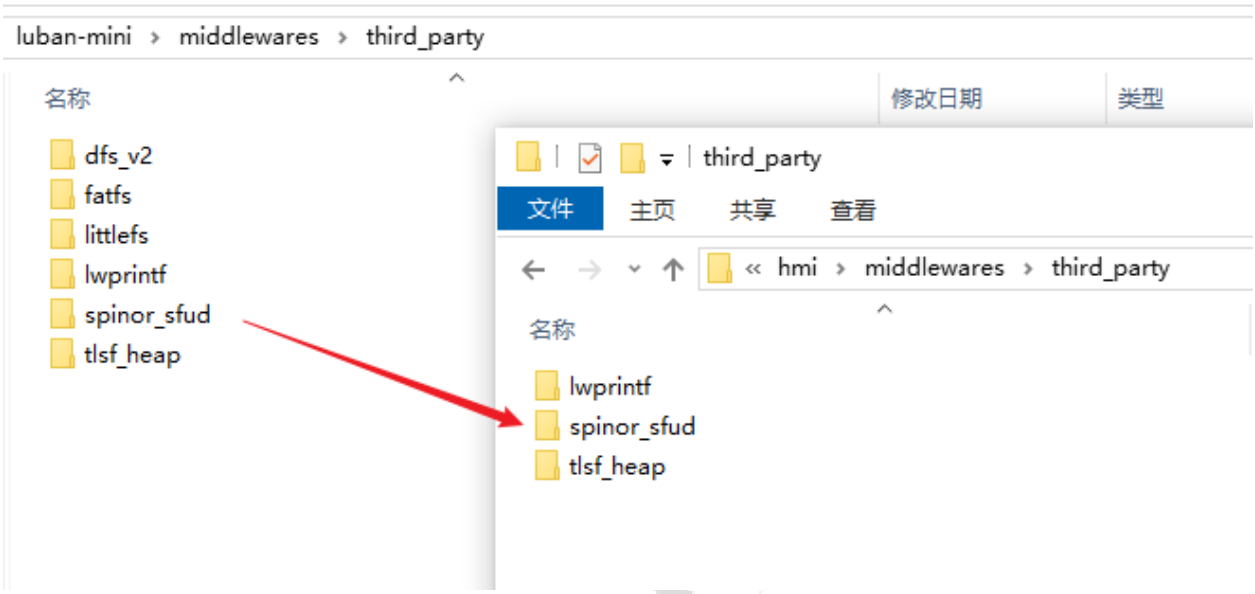


图 4–6 源码复制

将相关的头文件路径添加到项目中，修改 Eclipse 项目配置：

注：

Debug 和 Release 的配置都需要添加

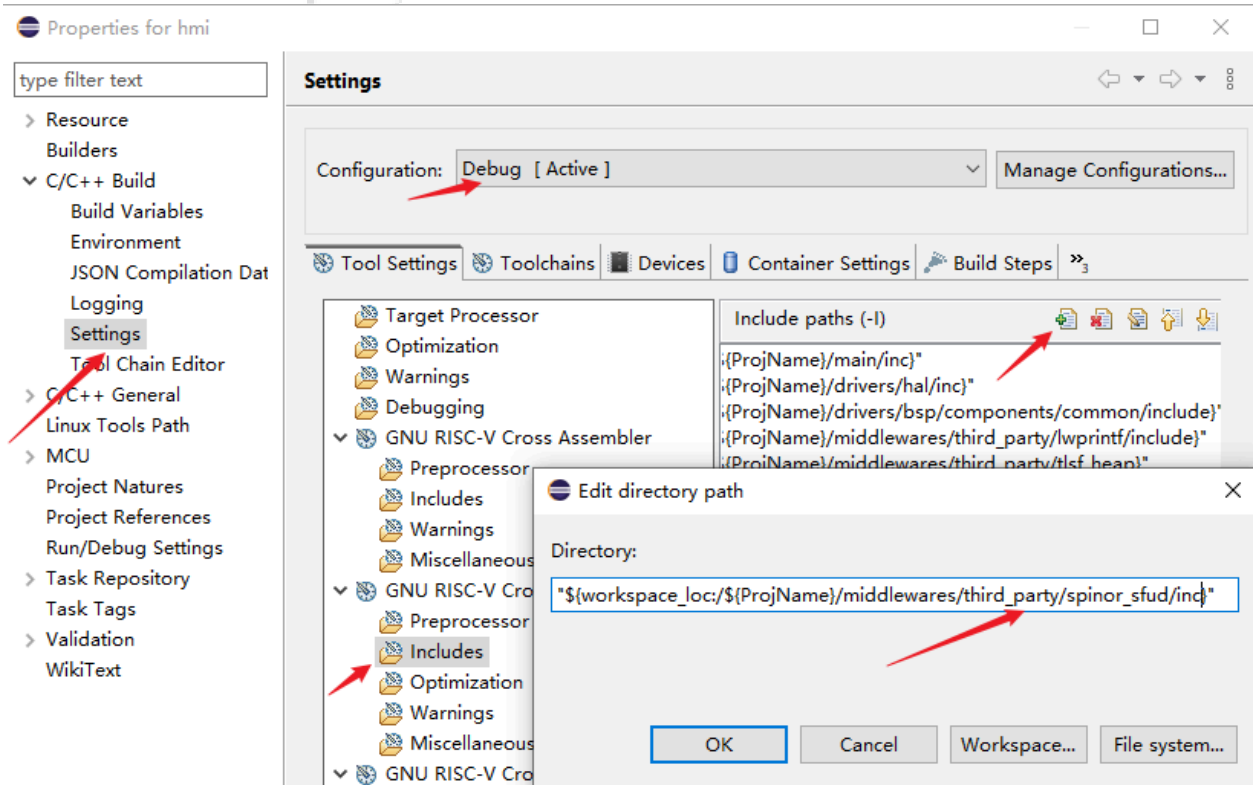


图 4–7 头文件路径

注：

部分源码包需要添加配置文件并且进行配置修改的，需要按要求添加配置文件。 如这里需要将 sfud_cfg_template.h 重命名为 sfud_cfg.h 并修改配置：

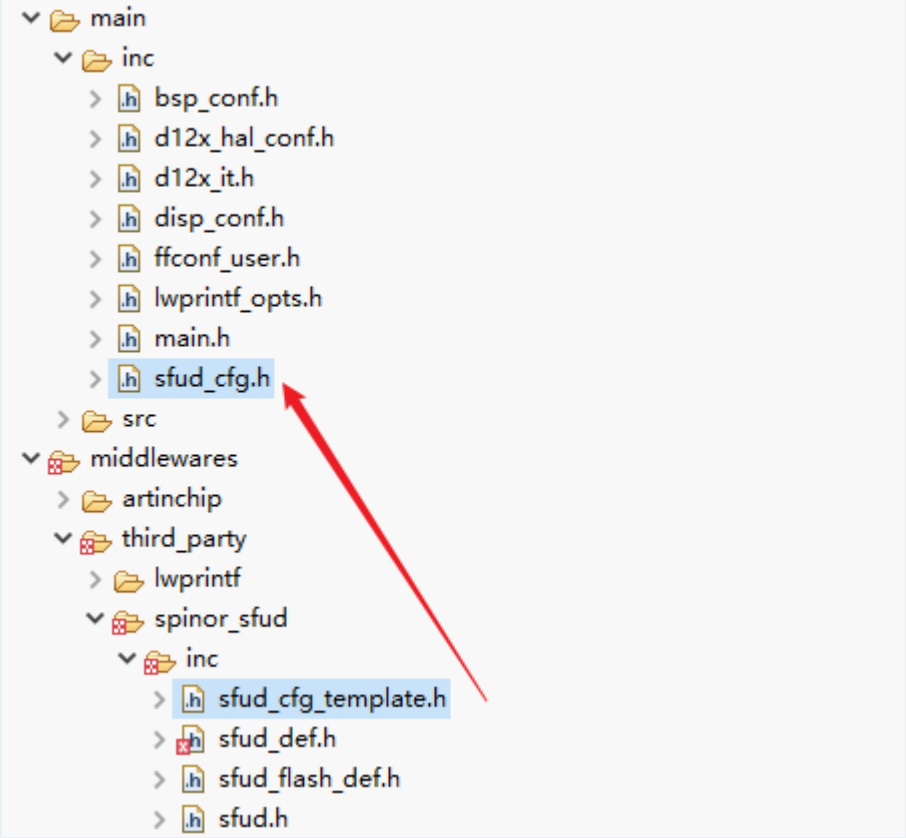


图 4–8 添加配置文件

4.4. 编译与调试

完成源码添加和项目配置后，接下来需要进行编译和调试。 在编译项目之前，确保项目配置已完成。具体配置步骤参考项目配置

本节介绍了 Luban Mini SDK 开发过程中的编译、调试与烧录操作：

- 项目编译
- 添加 Debug Configuration
- 下载调试
- 编译运行示例

4.5. 编译后处理

示例项目提供了后处理脚本：postbuild.sh，用于对编译后的结果进行处理，并且生成烧录文件。

该脚本基于 bash 语法编写，并且由 bash 解析执行。

用户如需添加自定义后处理操作，可直接修改该脚本。