XHSC 小华半导体
XIAOHUA SEMICONDUCTOR

32 位微控制器

基于 VSCode 搭建 HC32 系列 MCU 的 GCC 编译及调试环境

应用笔记

Rev1.01　2025 年 06 月

适用对象

产品系列	产品型号	产品系列	产品型号	产品系列	产品型号
HC32F4A0	ALL	**HC32F4A2**	ALL	**HC32F467**	ALL
HC32F460	ALL	**HC32F451**	ALL	**HC32F452**	ALL
HC32F448	ALL	**HC32F472**	ALL	**HC32F334**	ALL
HC32F115	ALL	**HC32F155**	ALL	**HC32F120**	ALL
HC32F4A8	ALL	**HC32M120**	ALL	**HC32M441**	ALL
HC32A4A8	ALL	**-**	-	**-**	-

声　明

★ 小华半导体有限公司（以下简称：“XHSC”）保留随时更改、更正、增强、修改小华半导体产品和/或本文档的权利，恕不另行通知。用户可在下单前获取最新相关信息。XHSC 产品依据购销基本合同中载明的销售条款和条件进行销售。

★ 客户应针对您的应用选择合适的 XHSC 产品，并设计、验证和测试您的应用，以确保您的应用满足相应标准以及任何安全、安保或其它要求。客户应对此独自承担全部责任。

★ XHSC 在此确认未以明示或暗示方式授予任何知识产权许可。

★ XHSC 产品的转售，若其条款与此处规定不同，XHSC 对此类产品的任何保修承诺无效。

★ 任何带有“®”或“™”标识的图形或字样是 XHSC 的商标。所有其他在 XHSC 产品上显示的产品或服务名称均为其各自所有者的财产。

★ 本通知中的信息取代并替换先前版本中的信息。

目　录

1 概述

VSCode（Visual Studio Code）是一款由微软开发且跨平台的免费源代码编辑器，具有良好的编码体验，同时它还具有丰富的插件生态系统，用户可以根据自己的需求安装对应的插件来扩展编辑器的功能。GCC（GNU Compiler Collection）是 GNU 开发的自由开源编译器套件，支持多种编程语言（包括 C、C++等）。

本文将介绍基于 VSCode，利用其插件扩展功能，在 Windows（Win 10）、Linux（Ubuntu 20.04）和 MacOS（10.14.6 Intel）上搭建 HC32 系列 MCU 的 GCC 编译及调试环境，并使用 XHSC 官网 HC32F4A0_Template 模板工程进行实际的编译及调试演示。

2 环境搭建步骤

2.1 下载 ARM GNU 工具链

通过访问 ARM 官网[1]，下载 ARM GNU 工具链。本文以“10.3-2021.10”版本为例，如图 2-1 所示，点击下载对应系统的“gcc-arm-none-eabi-10.3-2021.10”压缩包。然后将下载的压缩包解压至本地目录，如图 2-2 示例，该路径后续将会用于插件配置。

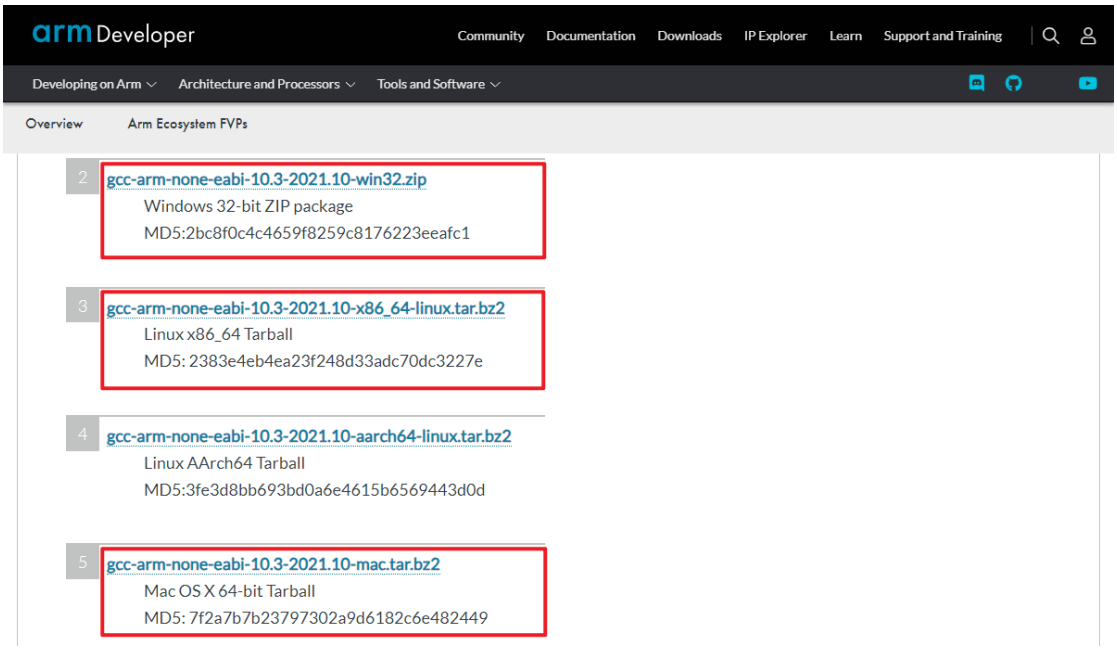

图 2-1 官网工具链下载

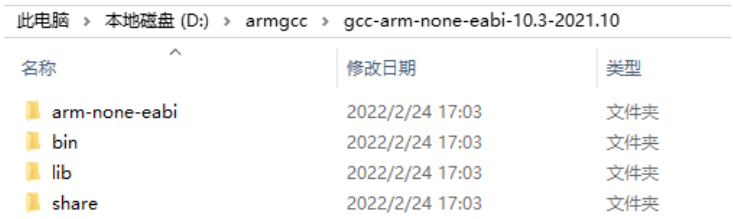

图 2-2 工具链解压存放目录

2.2 安装及配置 EIDE 插件

2.2.1 安装

在 VSCode 扩展市场，搜索“EIDE”，找到“Embedded IDE”插件[2]，并进行安装。插件安装后，会在 VSCode 的侧边栏出现一个芯片形状的“EIDE”按钮。

由于插件还需下载一些必要的依赖文件，时间取决于网络环境，点击“EIDE”按钮查看，当出现图 2-3 的左侧 EIDE 操作视图时，表明插件安装及依赖文件下载已完成。

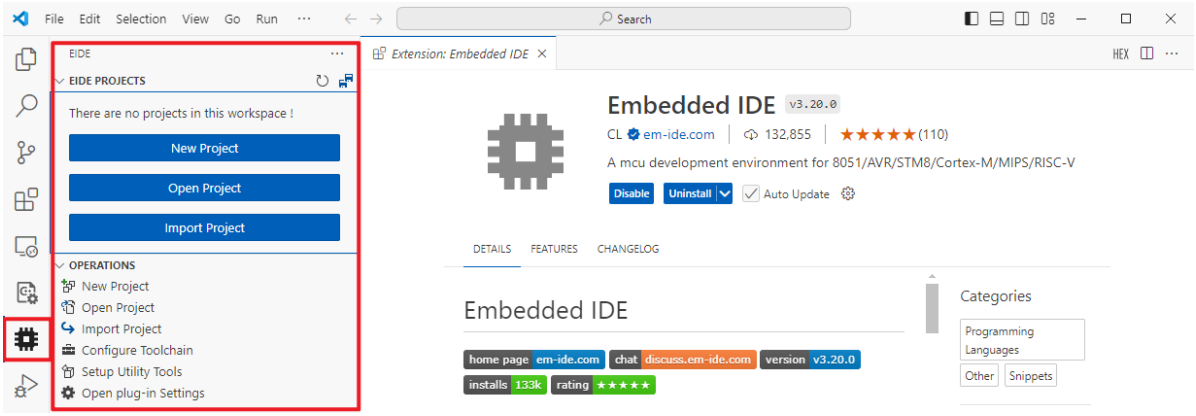

图 2-3 EIDE 插件安装完成视图

对于 Linux/ MacOS 系统，由于插件不支持自动为系统安装.NET6 runtime，因此可能会出现“Not found .NET6 runtime on your PC, please install it!”的提示，需要手动进行安装。

- Ubuntu 20.04，安装方法为：

1) 添加 Microsoft 包存储库

```
wget https://packages.microsoft.com/config/ubuntu/20.04/packages-microsoft-prod.deb -O packages-microsoft-prod.deb
```

```
sudo dpkg -i packages-microsoft-prod.deb
```

```
rm packages-microsoft-prod.deb
```

2) 安装 runtime

```
sudo apt-get update && sudo apt-get install -y dotnet-runtime-6.0
```

其它 Ubuntu 系统版本安装.NET6 runtime 方法，可参考微软提供的教程[3]。

- MacOS，安装方法为：

1) 从微软.NET6 下载网站[4]，下载安装程序，根据指引进行安装

图 2-4 下载.NET6 runtime

2) 创建一个软链接

```
sudo ln -s /usr/local/share/dotnet/dotnet /usr/local/bin/dotnet
```

2.2.2 配置

进入 EIDE 的插件配置界面，找到“EIDE.ARM.GCC:Install Directory”配置项，填入前面下载并解压的 ARM GNU 工具链的路径，Windows 下路径示例如图 2-5 所示（该路径在 Windows 环境下的 json 文件中会自动被转换为以\\进行分隔），Linux、MacOS 下路径示例分别如图 2-6、图 2-7 所示。

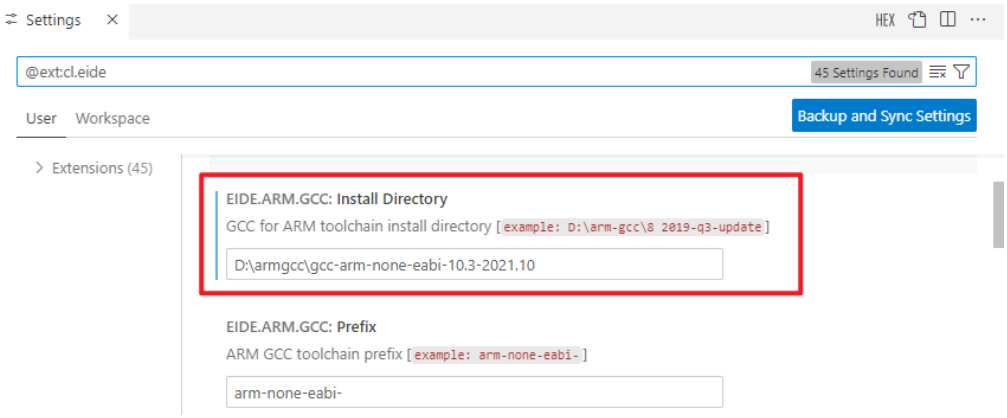

图 2-5 EIDE 插件配置（Windows）

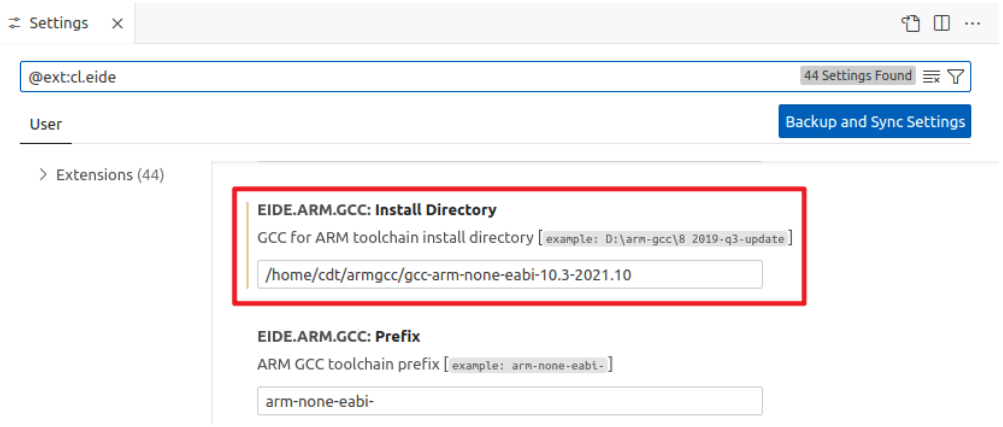

图 2-6 EIDE 插件配置（Linux）

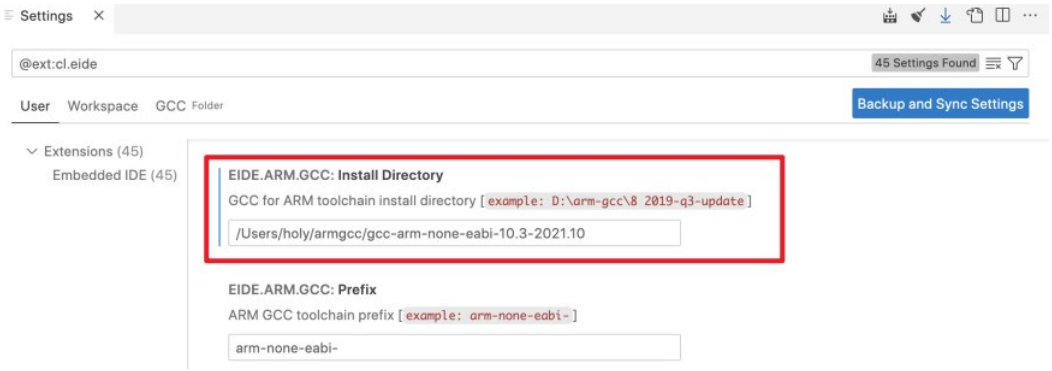

图 2-7 EIDE 插件配置（MacOS）

2.3 安装及配置 Cortex-Debug 插件

2.3.1 安装

在 VSCode 扩展市场，搜索“Cortex-Debug”插件[5]，并进行安装。

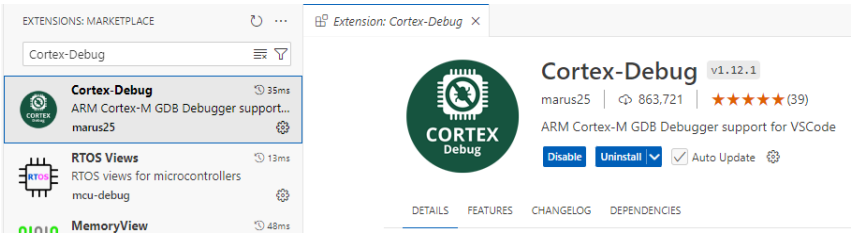

图 2-8 Cortex-Debug 插件安装

2.3.2 配置

进入 Cortex-Debug 的配置界面，找到“Cortex-debug:Arm Toolchain Path”配置项，并点击“Edit in settings.json”。在 json 文件中，填入 ARM GNU 工具链的 bin 目录。如图 2-10、图 2-11 和图 2-12 所示，分别是各系统下的配置示例。

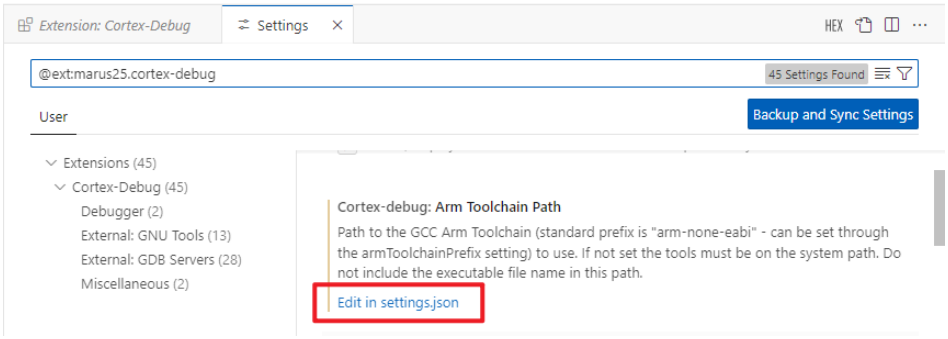

图 2-9 Cortex-Debug 插件配置 UI 界面

图 2-10 Cortex-Debug 插件配置 json 文件（Windows）

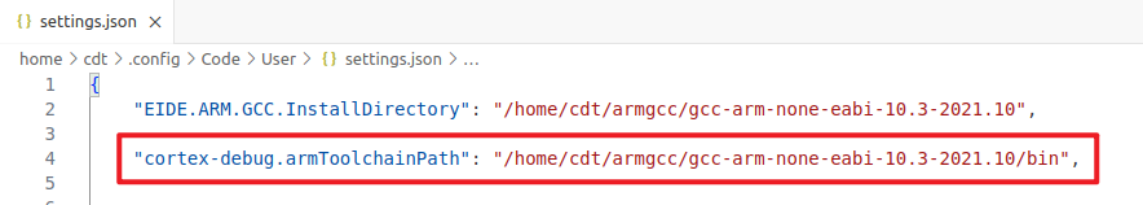

图 2-11 Cortex-Debug 插件配置 json 文件（Linux）

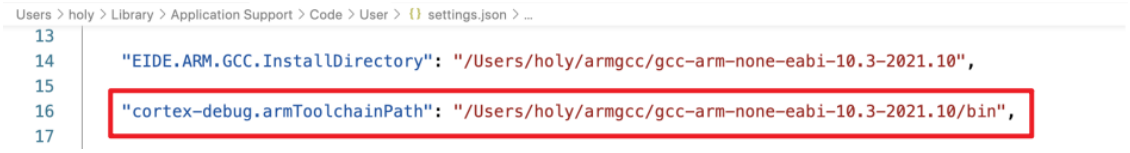

图 2-12 Cortex-Debug 插件配置 json 文件（MacOS）

注意：*json* 文件中，*Windows* 路径需要以\\进行分隔，同时注意与前面 *EIDE* 插件配置工具链路径的区别，*Cortex-Debug* 插件要求指定到 *bin* 目录级别。

3 编译 GCC 工程

3.1 导入 GCC 工程

从 VSCode 侧边栏选择 EIDE 视图，然后点击“Import Project”，在弹出的 Project Type 选项中，选择“Eclipse embedded gcc Projects”类型。

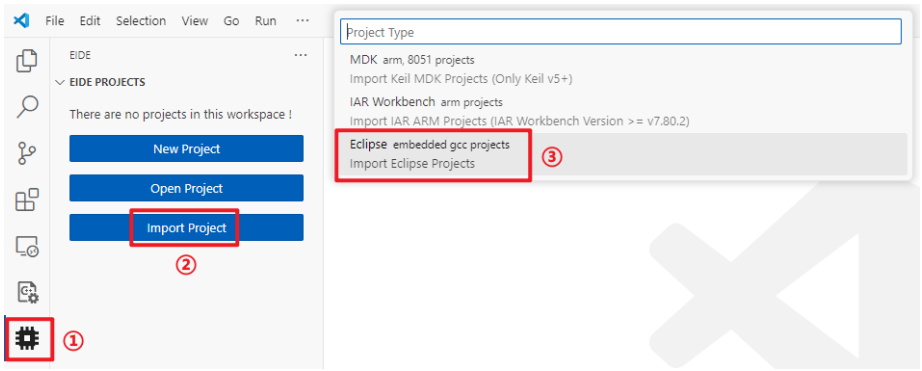

图 3-1 导入工程

然后在弹出的文件选择对话框中，按指引选择 XHSC 提供的 gcc 工程“.cproject”文件（对于 Linux/MacOS 系统，需先显示隐藏文件），然后点击“Import”。

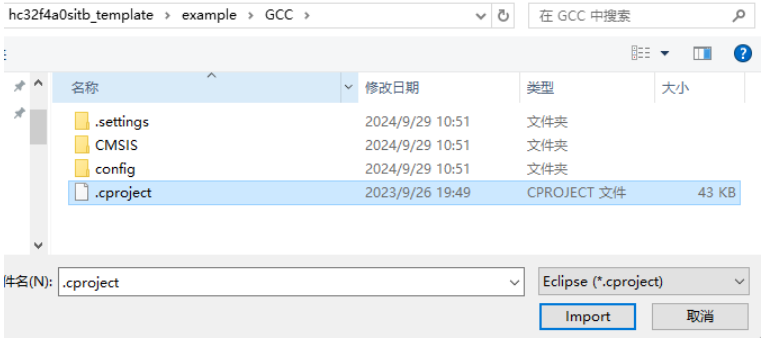

图 3-2 选择 GCC 工程文件（Windows）

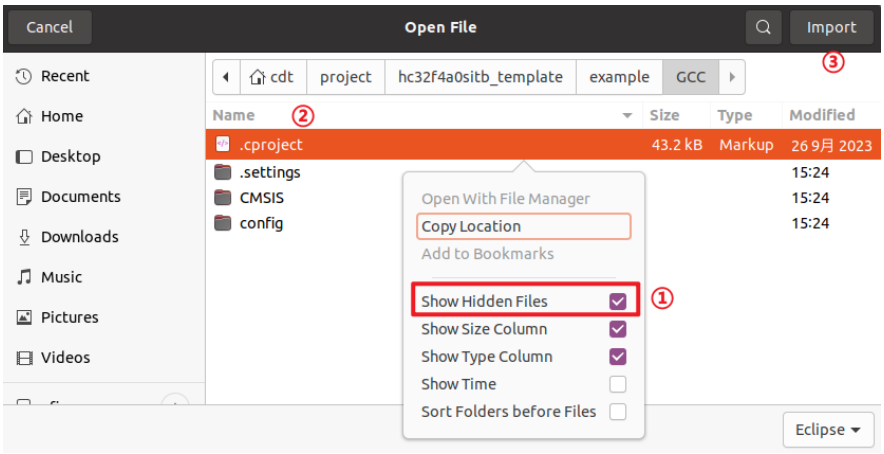

图 3-3 选择 GCC 工程文件（Linux）

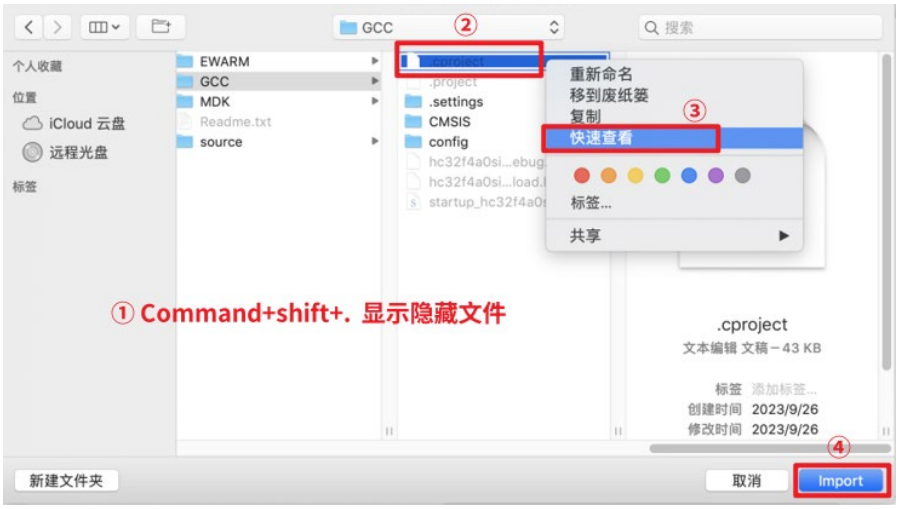

图 3-4 选择 GCC 工程文件（MacOS）

由于部分参数不兼容，无法被完全正确导入，EIDE 会创建一个警告文本（修复这些不兼容的配置后，可以删除该警告文件）。同时，注意右下角切换到 workspace 的提示，需点击“Continue”以进入工作空间。

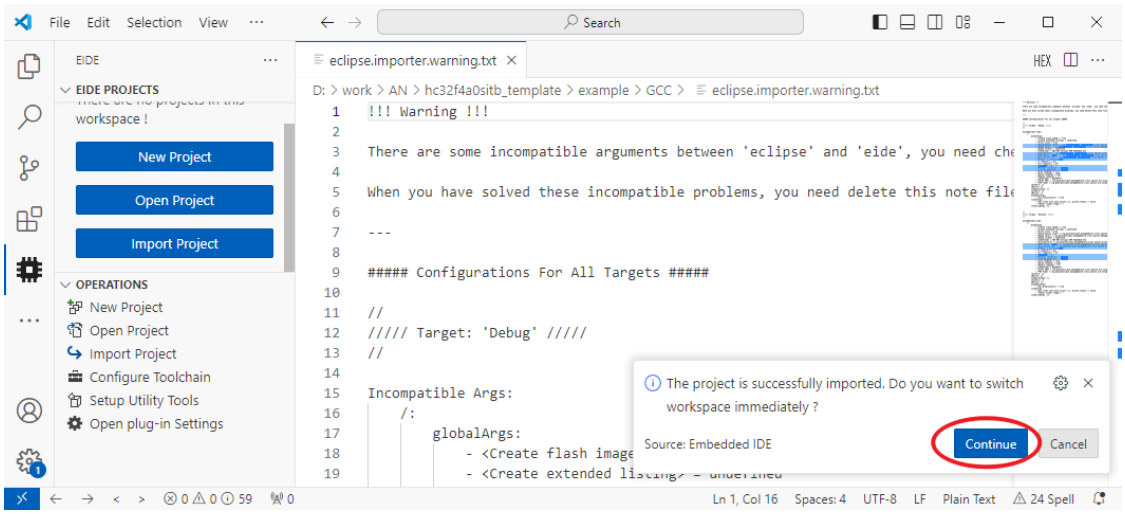

图 3-5 导入提示信息

注意：使用 *EIDE*“*Open Project*”打开已有工程，同样需要点击“*Continue*”切换到对应的工作空间。

3.2 配置 GCC 工程

由于前面提到部分参数无法被完全正确导入的原因，需要先选择对应的工程模式“Debug 或 Release”，然后再通过“Builder Option”进行编译选项配置来进行修复。

3.2.1 修复 Link 文件路径

导入后 Link 文件路径为：“../config/linker/hc32f4a0xi.ld”，需修改为“./config/linker/HC32F4A0xI.ld”，并保存。Windows/ MacOS 对文件名可不区分大小写，但 Linux 下文件名大小写需与实际文件严格对应。

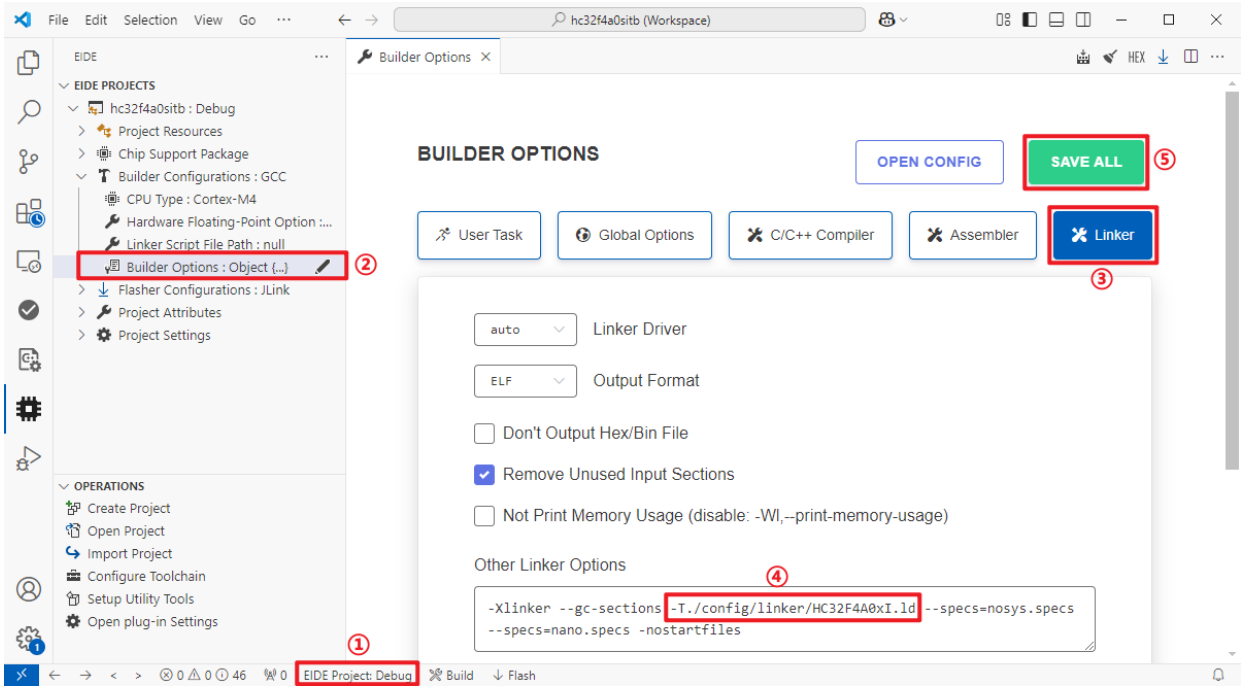

图 3-6 修复 Link 文件路径

3.2.2 配置编译选项

按需配置“Builder Options”中的各个选项，如配置优化等级、警告等级、C/C++标准等，并保存。推荐配置如下图所示。

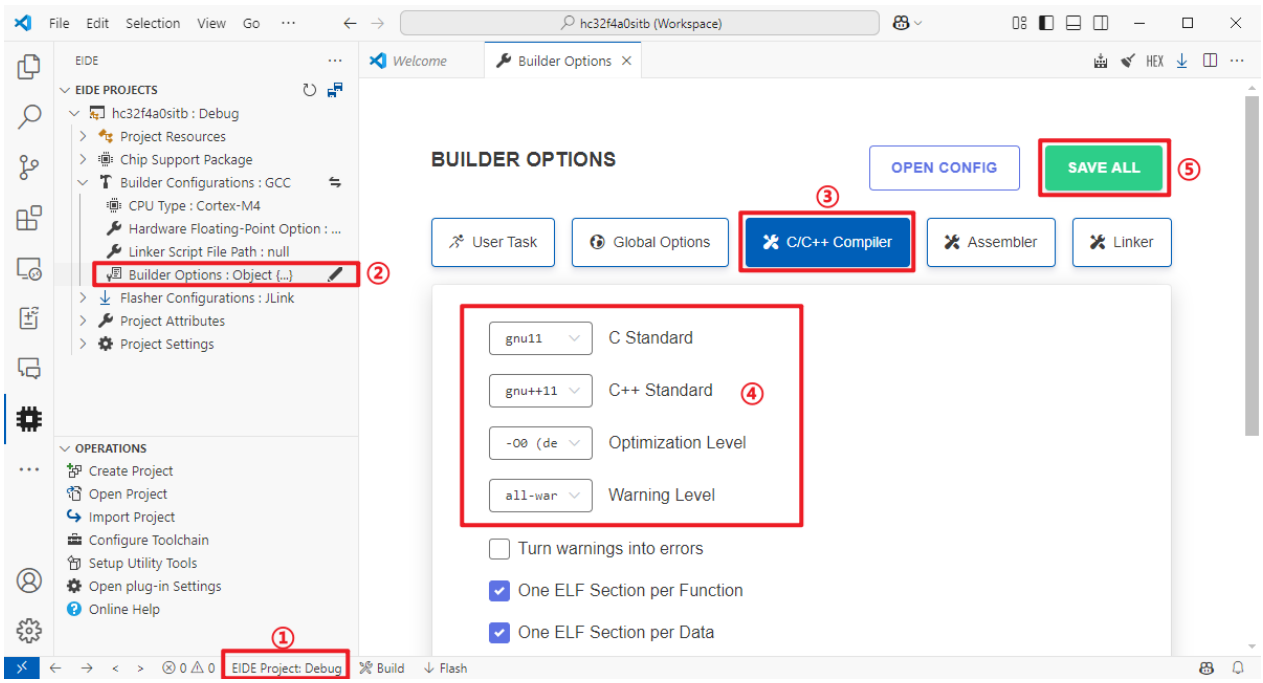

图 3-7 配置编译器选项

3.3 编译程序

点击界面的任一“Build”按钮，即可对程序进行编译，编译信息在 TERMINAL 中进行输出展示。

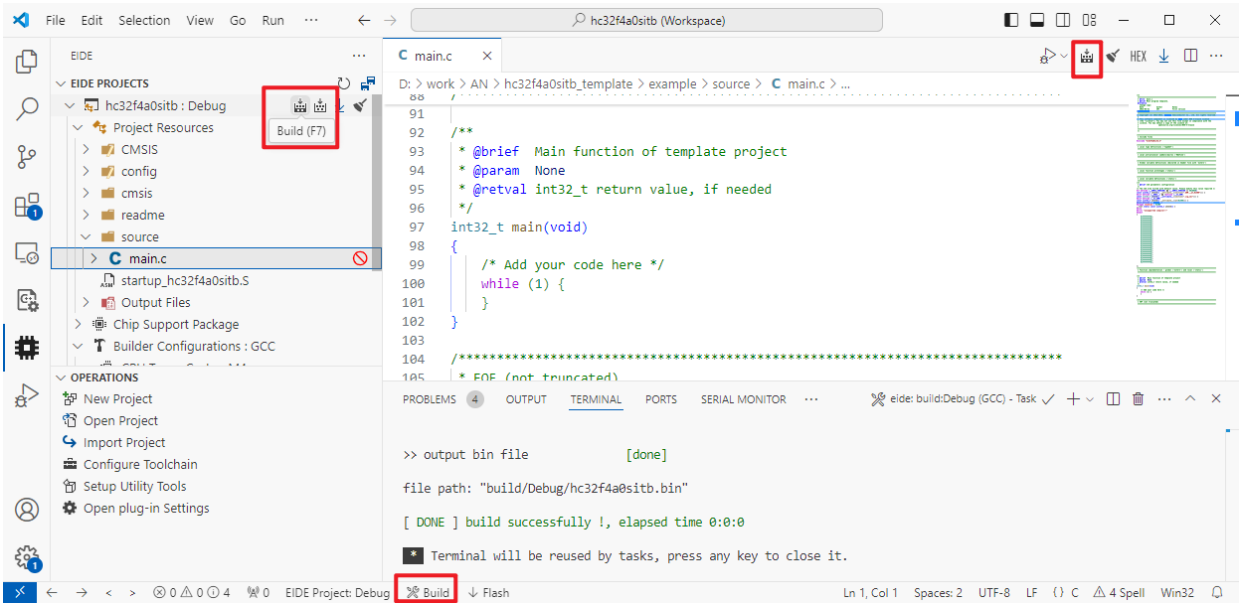

图 3-8 编译程序

4 下载及调试

4.1 驱动准备

下载及调试用的硬件工具，依赖对应的软件进行驱动。XH-Link 及 Jlink 可通过 pyOCD 进行驱动，Jlink 也可选择使用官方程序进行驱动。因此进行下载及调试前，应确保对应的驱动软件被正确安装并可以与调试工具建立连接。

4.1.1 pyOCD 安装

推荐通过下载小华 pyOCD 仓库的 hc32_pr 分支源码[6]，在终端输入下述指令进行安装。

```
pip install .
```

注意：

- *pyOCD* 的安装路径可能不在系统 *PATH* 中，需添加，以确保可以直接从终端中使用 *pyocd* 命令。
- *Linux* 由于权限的原因，*pyOCD* 可能无法访问到调试器，需要将目标 *USB* 设备添加到规则中[7]：以 *XH-Link* 设备为例，在 *pyOCD* 仓库的 “*udev/50-cmsis-dap.rules*” 文件中，将 *XH-Link* 的 *VID/PID* 信息按图 *4-1* 所示添加到文件中，然后将该文件使用指令拷贝到系统目录 “*/etc/udev/rules.d*”，重新拔插 *USB* 设备，*pyOCD* 即可访问到 *XH-Link*。

```
sudo cp 50-cmsis-dap.rules /etc/udev/rules.d
```

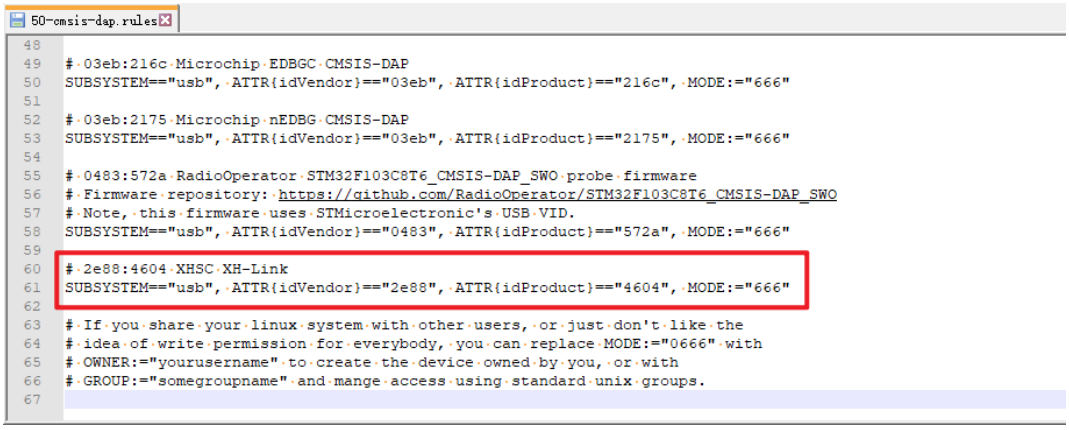

图 4-1 添加 XH-Link 规则

4.1.2 Jlink 驱动安装

如果需要使用 Jlink 官方程序直接驱动 Jlink 下载器，则访问 SEGGER 官网下载对应系统的安装包进行安装即可[8]。如果官方 Jlink 驱动还未支持目标的 HC32 芯片，可联系 XHSC 获得支持。

对于 Linux/ MacOS：

- Ubuntu 建议下载 deb 驱动包，使用 apt 进行安装。

```
sudo apt install ./JLink_Linux_V812_x86_64.deb
```

- EIDE 插件调用的命令文件是“JLink”，但 Jlink 驱动在 Linux/MaxOS 下命令文件可能只有“JLinkExe”，此种情况需要建立一个软链接，使“JLink”与“JLinkExe”命令等价。

```
sudo ln -s JLinkExe JLink
```

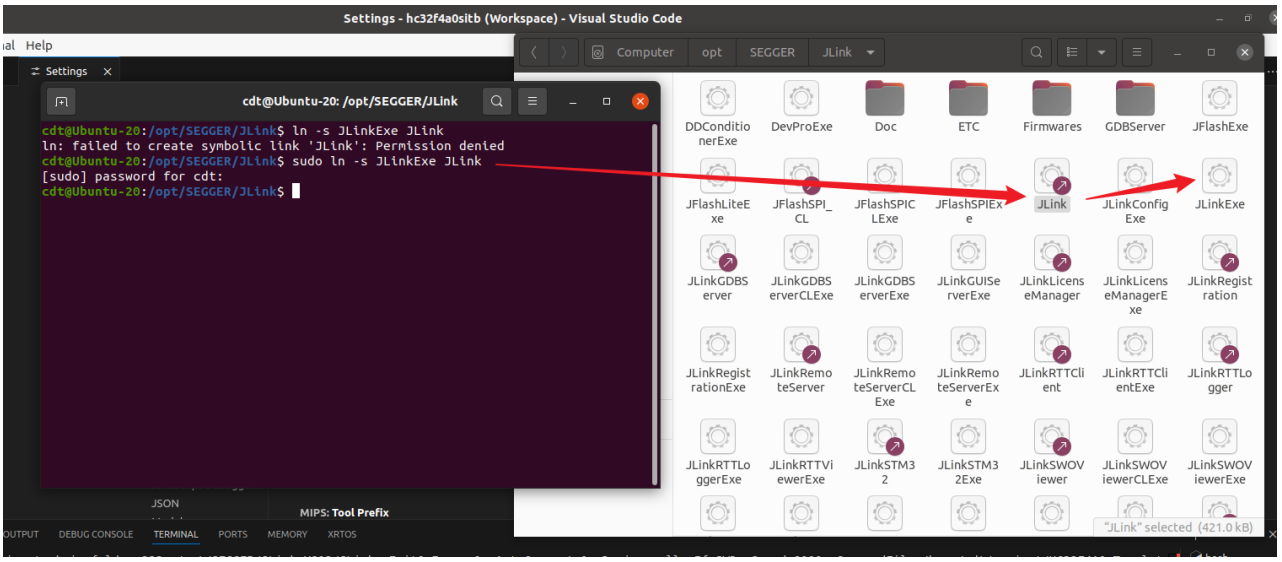

图 4-2 添加 Jlink 软链接

4.1.3 驱动路径配置

通常情况下，只要 pyOCD 或 Jlink 驱动安装路径在系统 PATH 中，是无需在插件设置中配置其路径的。如果确实安装路径不方便添加到系统 PATH，或者存在多个版本驱动的情况，必须对驱动路径进行指定时，需分别在 EIDE 和 Cortex-Debug 插件中配置。

- 对于下载程序功能：在 EIDE 插件中配置驱动路径，且目前本文使用的插件版本仅支持 Jlink 路径的指定。

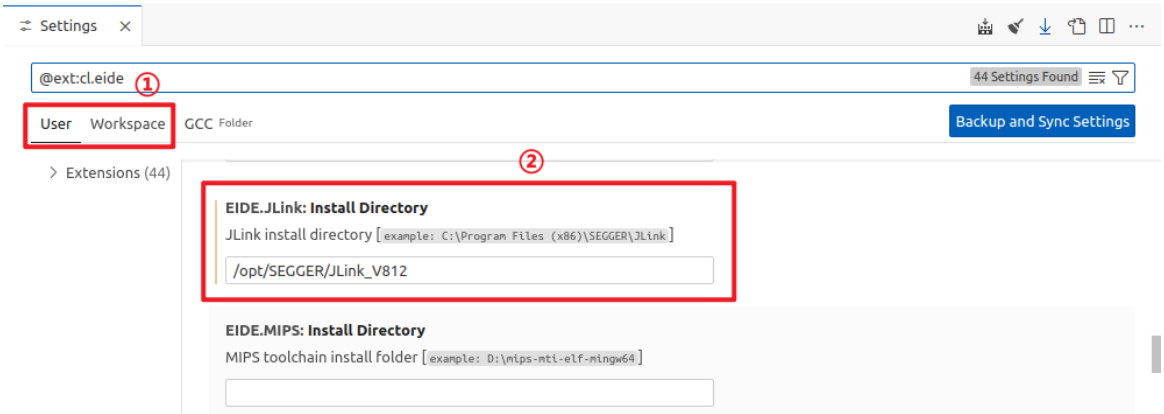

图 4-3 EIDE 插件配置 Jlink 安装路径

- 对于调试程序功能：在 Cortex-Debug 插件中配置驱动可执行文件的具体路径，该配置是在 json 文件中进行，各系统下的配置示例，分别如图 4-6、图 4-7 和图 4-8 所示。

图 4-4 Cortex-Debug 插件配置 Jlink 路径 UI

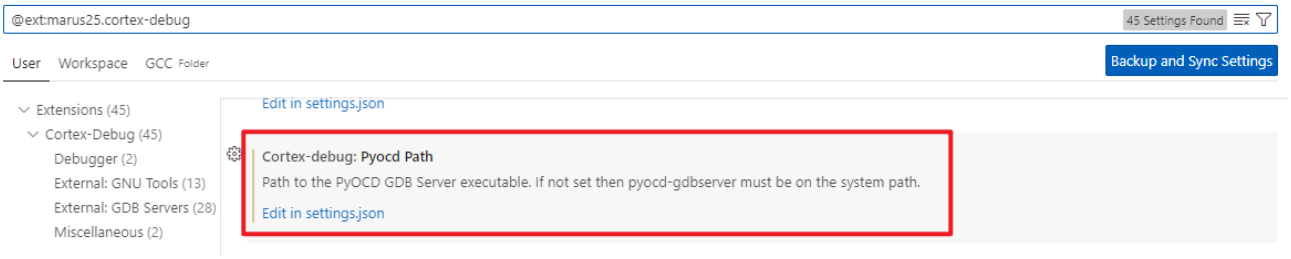

图 4-5 Cortex-Debug 插件配置 pyOCD 路径 UI

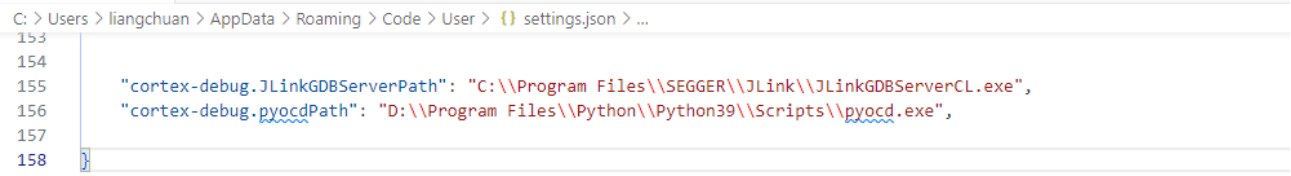

图 4-6 Cortex-Debug 插件配置驱动路径示例（Windows）

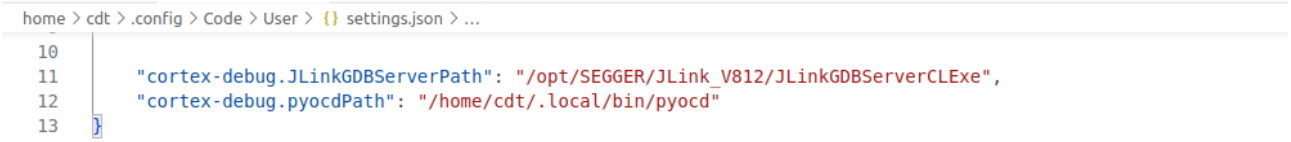

图 4-7 Cortex-Debug 插件配置驱动路径示例（Linux）

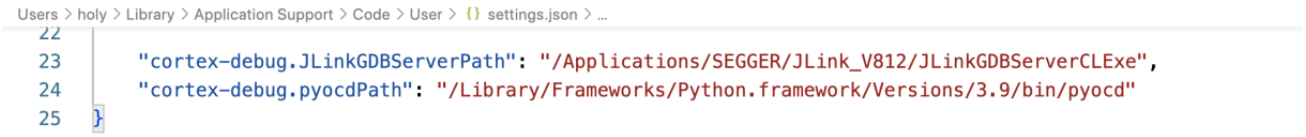

图 4-8 Cortex-Debug 插件配置驱动路径示例（MacOS）

备注：通常插件的配置在 *User* 标签下进行全局配置即可，因系统环境的差异，若发现在 *User* 标签配置下无法生效，可尝试在 *Workspace* 标签下再次进行配置。

4.2 下载程序

4.2.1 配置下载器

点击 EIDE 的“Flasher Configuration”配置按钮，选择下载工具，并配置目标芯片型号及下载速度等参数。以 pyOCD 为例进行下载，配置如图 4-9 所示。

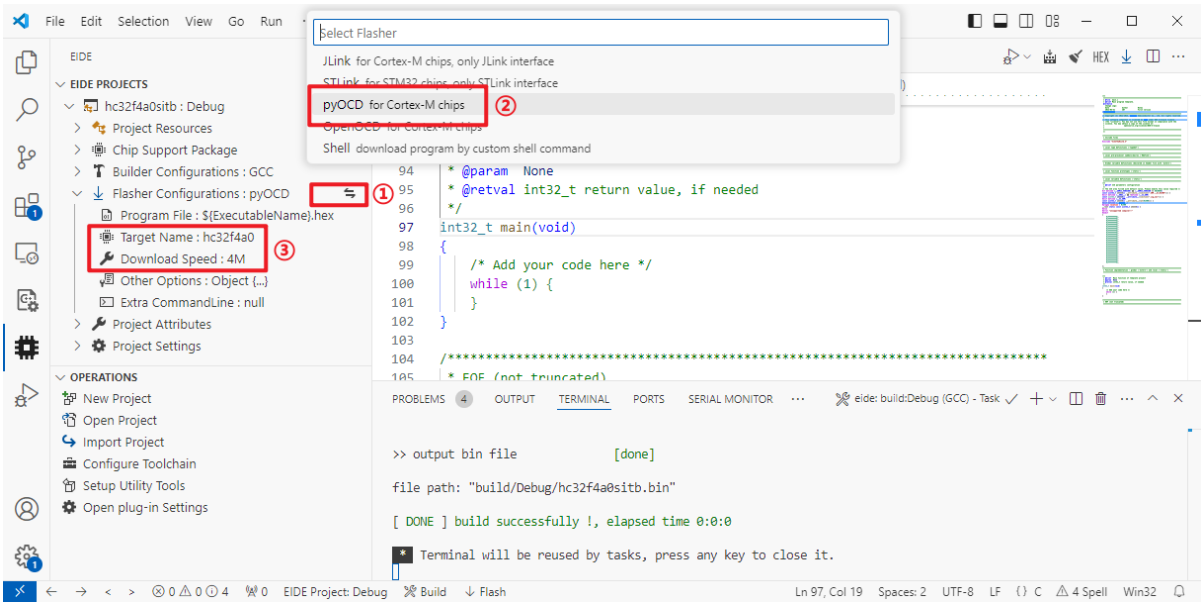

图 4-9 配置下载器

4.2.2 下载程序

点击界面任一“Flash”按钮，即可进行程序下载。

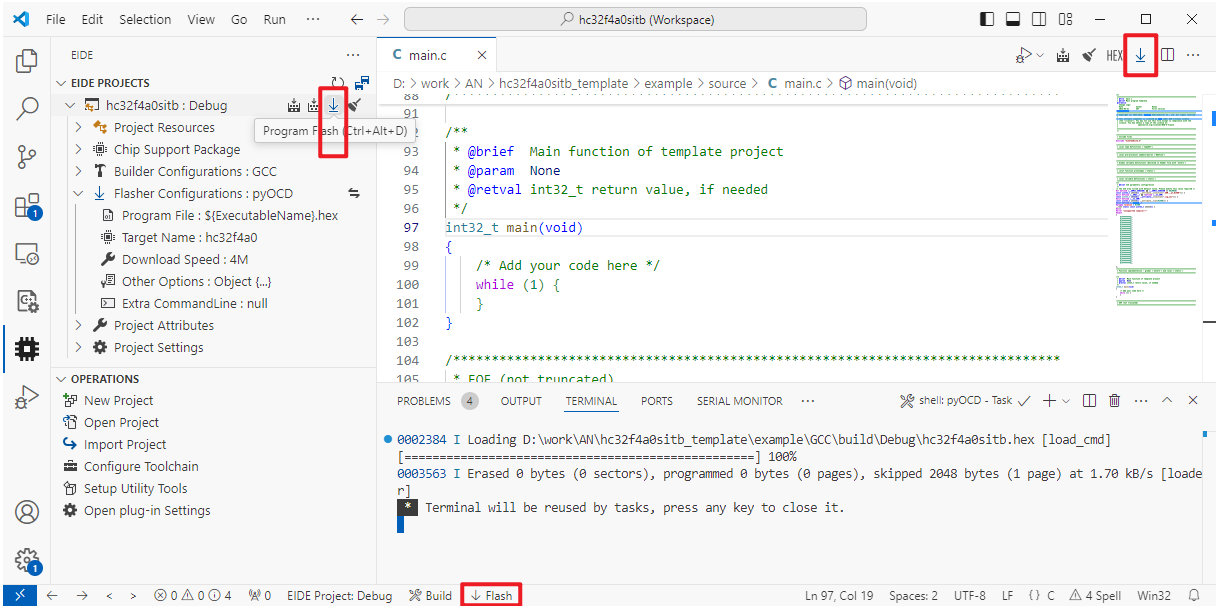

图 4-10 下载程序

4.3 调试程序

4.3.1 配置调试器

要启动调试，需要一个“launch.json”配置文件。在工程上右键，选择“Generate Debugger Configuration”，选择对应的调试器，在弹出的配置模板上，点击“Create”按钮即可生成。

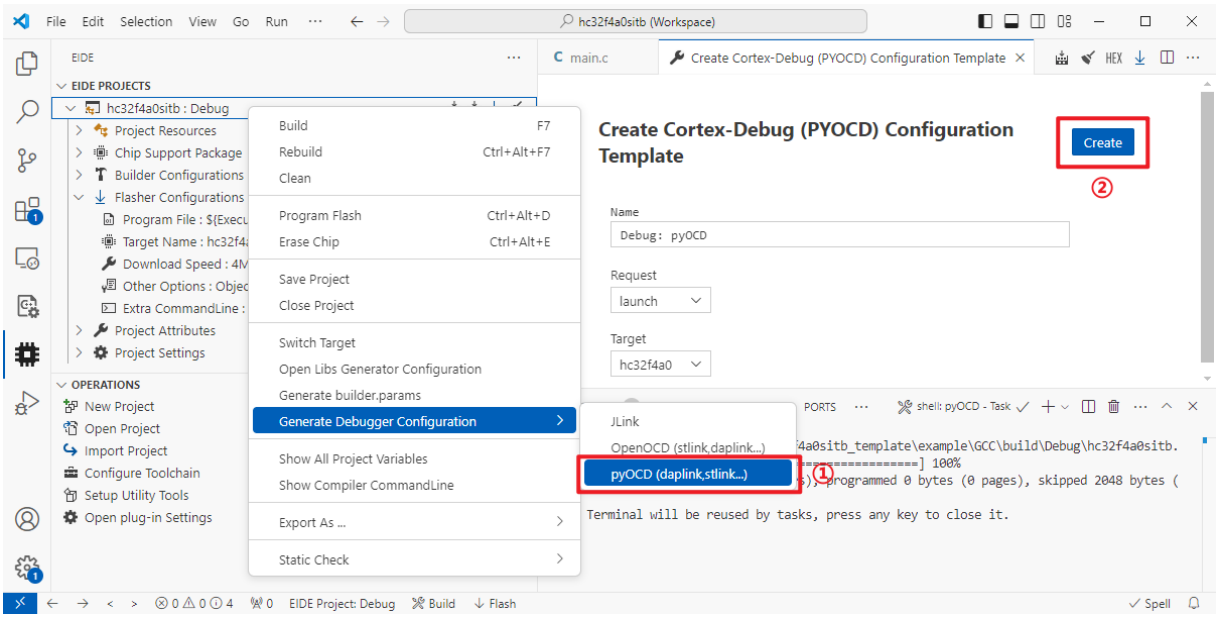

图 4-11 配置调试器

生成的调试配置“launch.json”文件位于工程“.vscode/”目录下，可以按需往 json 文件中添加配置属性。如添加 svd 文件，以便调试时可以查看芯片外设寄存器。使用 DDL 样例时，svd 文件路径按实际填写，或者拷贝至工程“config/”目录下。更多可配属性，请参考 Cortex-Debug 插件官方说明[9]。

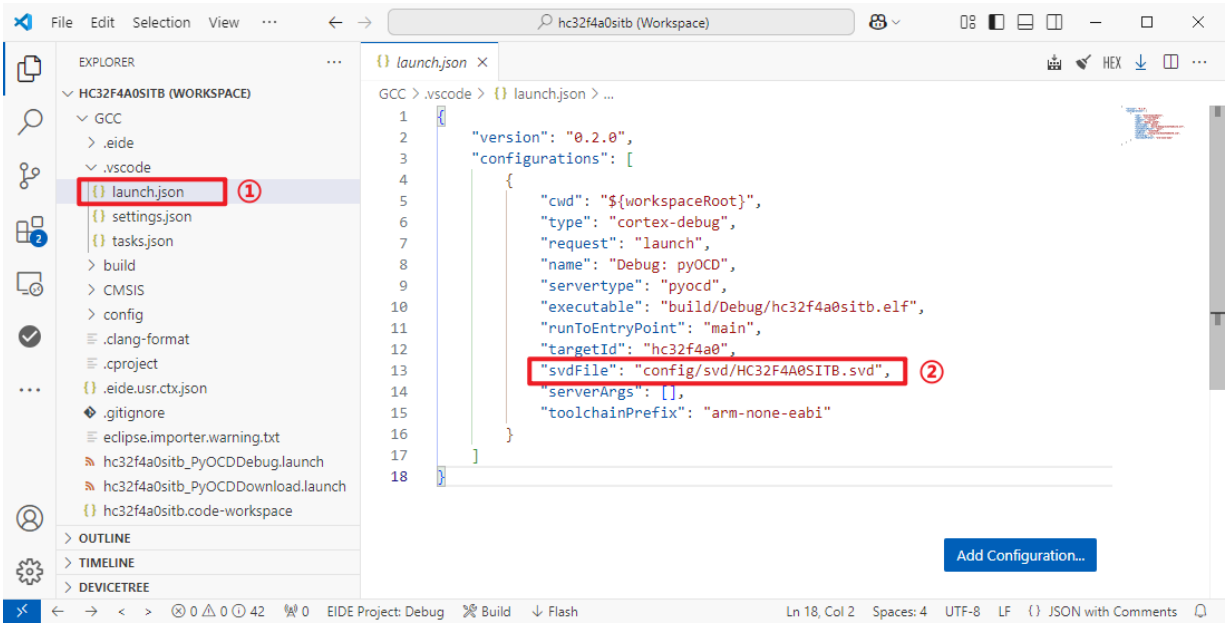

图 4-12 调试 Json 文件添加 svd 文件配置

4.3.2 调试程序

按下快捷键 F5，或者点击侧边栏“Debug”窗口中的“Start Debugging”三角形按钮，便可启动调试。

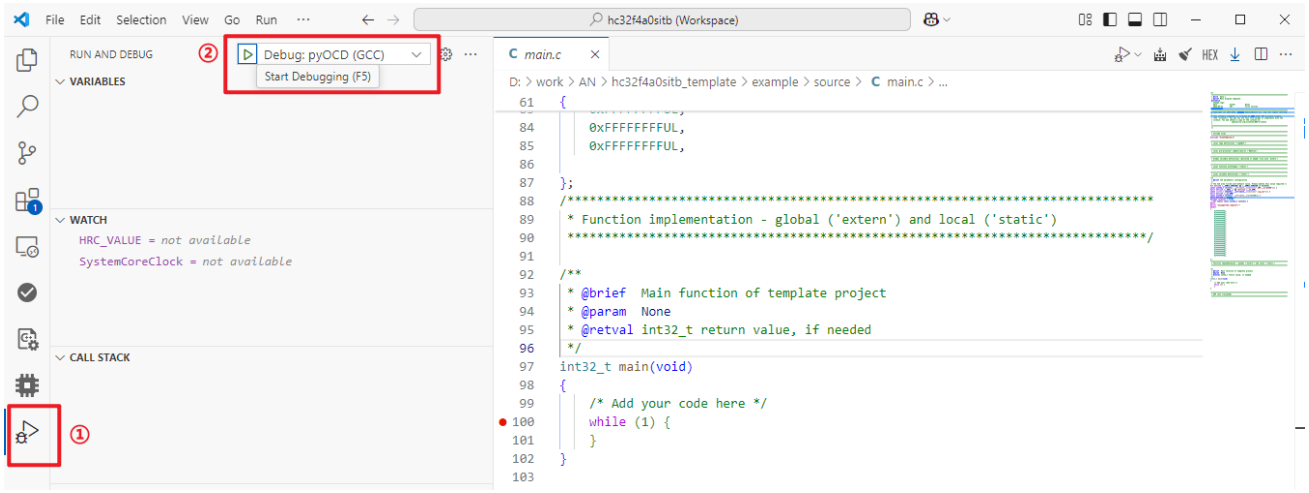

图 4-13 启动调试

在调试界面，可以查看程序变量、外设寄存器值、Memory 等信息。

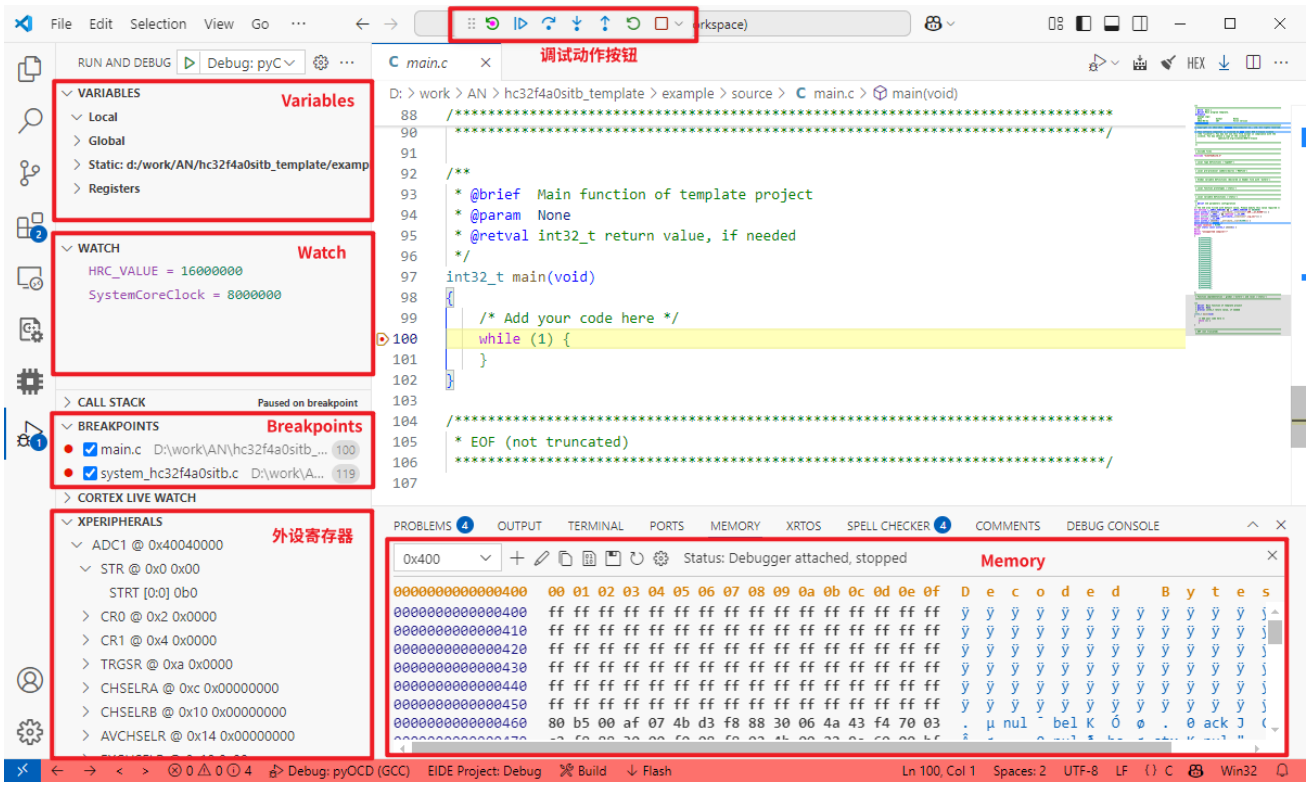

图 4-14 调试主要界面

备注：若无图 *4-14* 中的 *Memory* 窗口，请安装“*MemoryView*”插件[10]。

5 总结

本文基于 VSCode，介绍了在 Windows、Linux 和 MacOS 上，HC32 系列 MCU 的 GCC 编译及调试环境的搭建方法，并使用模板工程进行了实际的编译及调试演示，可以给客户提供参考。

6 参考

[1] Arm, "GNU Arm Embedded Toolchain," Arm Developer. [Online]. Available: https://developer.arm.com/downloads/-/gnu-rm. [Accessed: Dec. 30, 2024].

[2] GitHub0null, "EIDE: An embedded development environment for mcs51/stm8/avr/cortex-m/riscv on VsCode," GitHub. [Online]. Available: https://github.com/github0null/eide. [Accessed: Dec. 30, 2024].

[3] Microsoft, "Install .NET on Ubuntu," Microsoft Learn. [Online]. Available: https://learn.microsoft.com/zh-cn/dotnet/core/install/linux-ubuntu-install?tabs=dotnet6&pivots=os-linux-ubuntu-2004. [Accessed: Dec. 30, 2024].

[4] Microsoft, ".NET 6.0 Download," .NET Official Website. [Online]. Available: https://dotnet.microsoft.com/zh-cn/download/dotnet/6.0. [Accessed: Dec. 30, 2024].

[5] Marus, "cortex-debug: Visual Studio Code extension for enhancing debug capabilities for Cortex-M Microcontrollers," GitHub. [Online]. Available: https://github.com/Marus/cortex-debug. [Accessed: Dec. 30, 2024].

[6] HDSCMCU, "pyOCD: Open Source Python Library for Programming and Debugging Arm Cortex Microcontrollers (HC32 Branch)," GitHub. [Online]. Available: https://github.com/hdscmcu/pyOCD/tree/hc32_pr. [Accessed: Dec. 30, 2024].

[7] HDSCMCU, "pyOCD: udev Rules for Linux (HC32 Branch)," GitHub. [Online]. Available: https://github.com/hdscmcu/pyOCD/blob/hc32_pr/udev/README.md. [Accessed: Dec. 30, 2024].

[8] SEGGER, "J-Link Software and Documentation Pack," SEGGER. [Online]. Available: https://www.segger.com/downloads/jlink/. [Accessed: Dec. 30, 2024].

[9] Marus, "cortex-debug: Debug Attributes Documentation," GitHub. [Online]. Available: https://github.com/Marus/cortex-debug/blob/master/debug_attributes.md. [Accessed: Dec. 30, 2024].

[10] MCU-Debug, "MemView: Memory View to examine memory from programs being debugged," GitHub. [Online]. Available: https://github.com/mcu-debug/memview. [Accessed: Dec. 30, 2024].

版本修订记录

版本号	修订日期	修订内容
Rev1.00	2025/03/25	初版发布。
Rev1.01	2025/06/26	新增 HC32A4A8 系列型号。