QSTM32
SDK 快速开发指导

版本：2.1

日期：2026-01-21

状态：受控文件

上海移远通信技术股份有限公司（以下简称“移远通信”）始终以为客户提供最及时、最全面的服务为宗旨。如需任何帮助，请随时联系我司上海总部，联系方式如下：

上海移远通信技术股份有限公司
上海市闵行区田林路 1016 号科技绿洲 3 期（B 区）5 号楼　　邮编：200233
电话：+86 21 5108 6236　　邮箱：info@quectel.com

或联系我司当地办事处，详情请登录：http://www.quectel.com/cn/support/sales.htm。

如需技术支持或反馈我司技术文档中的问题，请随时登录网址：
http://www.quectel.com/cn/support/technical.htm 或发送邮件至：support@quectel.com。

前言

移远通信提供该文档内容以支持客户的产品设计。客户须按照文档中提供的规范、参数来设计产品。同时，您理解并同意，移远通信提供的参考设计仅作为示例。您同意在设计您目标产品时使用您独立的分析、评估和判断。在使用本文档所指导的任何硬软件或服务之前，请仔细阅读本声明。您在此承认并同意，尽管移远通信采取了商业范围内的合理努力来提供尽可能好的体验，但本文档和其所涉及服务是在“可用”基础上提供给您的。移远通信可在未事先通知的情况下，自行决定随时增加、修改或重述本文档。

使用和披露限制

许可协议

除非移远通信特别授权，否则我司所提供硬软件、材料和文档的接收方须对接收的内容保密，不得将其用于除本项目的实施与开展以外的任何其他目的。

版权声明

移远通信产品和本协议项下的第三方产品可能包含受移远通信或第三方材料、硬软件和文档版权保护的相关资料。除非事先得到书面同意，否则您不得获取、使用、向第三方披露我司所提供的文档和信息，或对此类受版权保护的资料进行复制、转载、抄袭、出版、展示、翻译、分发、合并、修改，或创造其衍生作品。移远通信或第三方对受版权保护的资料拥有专有权，不授予或转让任何专利、版权、商标或服务商标权的许可。为避免歧义，除了正常的非独家、免版税的产品使用许可，任何形式的购买都不可被视为授予许可。对于任何违反保密义务、未经授权使用或以其他非法形式恶意使用所述文档和信息的违法侵权行为，移远通信有权追究法律责任。

商标

除另行规定，本文档中的任何内容均不授予在广告、宣传或其他方面使用移远通信或第三方的任何商标、商号及名称，或其缩略语，或其仿冒品的权利。

第三方权利

您理解本文档可能涉及一个或多个属于第三方的硬软件和文档（“第三方材料”）。您对此类第三方材料的使用应受本文档的所有限制和义务约束。

移远通信针对第三方材料不做任何明示或暗示的保证或陈述，包括但不限于任何暗示或法定的适销性或特定用途的适用性、平静受益权、系统集成、信息准确性以及与许可技术或被许可人使用许可技术相关的不侵犯任何第三方知识产权的保证。本协议中的任何内容都不构成移远通信对任何移远通信产品或任何其他硬软件、设备、工具、信息或产品的开发、增强、修改、分销、营销、销售、提供销售或以其他方式维持生产的陈述或保证。此外，移远通信免除因交易过程、使用或贸易而产生的任何和所有保证。

隐私声明

为实现移远通信产品功能，特定设备数据将会上传至移远通信或第三方服务器（包括运营商、芯片供应商或您指定的服务器）。移远通信严格遵守相关法律法规，仅为实现产品功能之目的或在适用法律允许的情况下保留、使用、披露或以其他方式处理相关数据。当您与第三方进行数据交互前，请自行了解其隐私保护和数据安全政策。

免责声明

1) 移远通信不承担任何因未能遵守有关操作或设计规范而造成损害的责任。
2) 移远通信不承担因本文档中的任何因不准确、遗漏、或使用本文档中的信息而产生的任何责任。
3) 移远通信尽力确保开发中功能的完整性、准确性、及时性，但不排除上述功能错误或遗漏的可能。除非另有协议规定，否则移远通信对开发中功能的使用不做任何暗示或法定的保证。在适用法律允许的最大范围内，移远通信不对任何因使用开发中功能而遭受的损害承担责任，无论此类损害是否可以预见。
4) 移远通信对第三方网站及第三方资源的信息、内容、广告、商业报价、产品、服务和材料的可访问性、安全性、准确性、可用性、合法性和完整性不承担任何法律责任。

文档历史

修订记录

版本	日期	作者	变更描述
-	2025-05-01	Jerry Chen	文档创建
1.0	2025-05-15	Jerry Chen	初始版本
2.0	2025-08-08	Jerry Chen	更新 5.1 章节图 17
2.1	2026-01-21	Jerry Chen	项目名更改为 UniKnect

目录

表格索引

图片索引

1 引言

Quectel UniKnect Project 是移远通信专为开发者设计的一套软件框架，在该框架下可直接调用 API 来实现各种功能，开发者只需专注于自己的业务逻辑，而无需处理 MCU 和模组之间复杂的 AT 交互数据，从而使移远模组的开发过程更加简单、友好。

本文档主要介绍 Quectel UniKnect Project SDK 的使用方法，主要包括 SDK 的目录结构、编译框架、编译环境搭建、硬件环境准备、项目构建、开发过程中的编译、清理、下载、调试等。

2 SDK 目录结构

Quectel UniKnect Project 的 SDK 目录结构如*图 1* 所示。

```
├── .vscode            # VSCode 调试环境配置（可选，脚本自动生成）
├── apps               # 应用程序目录，包含各功能 example 和 app 入口
├── build              # 构建输出目录，包含编译生成的中间文件和可执行文件
├── quectel            # Quectel 相关代码适配目录
├── system             # 系统平台目录，包含各型号 MCU 驱动适配 HAL 相关代码
├── tools              # 工具脚本目录，包含交叉编译工具链、脚本、配置等
├── .clang-format      # Clang 代码风格规范文件
├── .editorconfig      # 跨编辑器格式统一配置文件
├── .gitignore         # Git 版本控制忽略规则
├── build.bat          # SDK 脚本，执行构建、编译、下载、调试等命令
├── CMakeLists.txt     # CMake 项目构建主配置（脚本自动生成）
├── CMakePresets.json  # CMake 项目构建预设参数（脚本自动生成）
└── README.md          # 说明文档
```

图 1：SDK 目录结构

3 SDK 编译框架

Quectel UniKnect Project SDK 的编译框架支持 STM32 微处理器全系型号，不同型号 MCU 共用一套 SDK，在构建项目时的初始化配置中予以区分，且所有操作自动化完成，无需用户手动配置，充分体现了本项目的友好性。

3.1. 主要特性

- **构建、编译、清理、下载、调试**
- SDK 内置交叉编译工具链，编译环境无需手动搭建，**开箱即用**
- 不同型号 MCU 的所有相关参数、文件，**自动适配**
- ***CMakeLists.txt***、***CMakePresets.json*** 等构建文件**自动生成**
- 配套自动化脚本，**一键完成**上述所有操作，**无需手动配置**

3.2. 操作方式

1. **命令行：** 不依赖任何 IDE 或代码编辑器
2. **GUI：** VSCode + 插件（相关参数自动生成，无需关注）

3.3. 自动化配置

- HAL 驱动相关代码的自动配置
- Flash 链接脚本.ld 文件的自动配置
- OpenOCD 调用.cfg/.svd/interface/target 文件的自动配置
- 编译器、链接器、宏、编译文件、包含目录等的自动配置
- CMakeLists.txt / CMakePresets.json / xxx.json 等文件的自动生成
- 芯片型号、软件版本缺省时的自动配置
- *build.bat* 脚本对用户输入异常参数的防呆处理。

3.4. 编译框架逻辑图

Quectel UniKnect Project 的 SDK 编译框架逻辑图如*图 2* 所示。

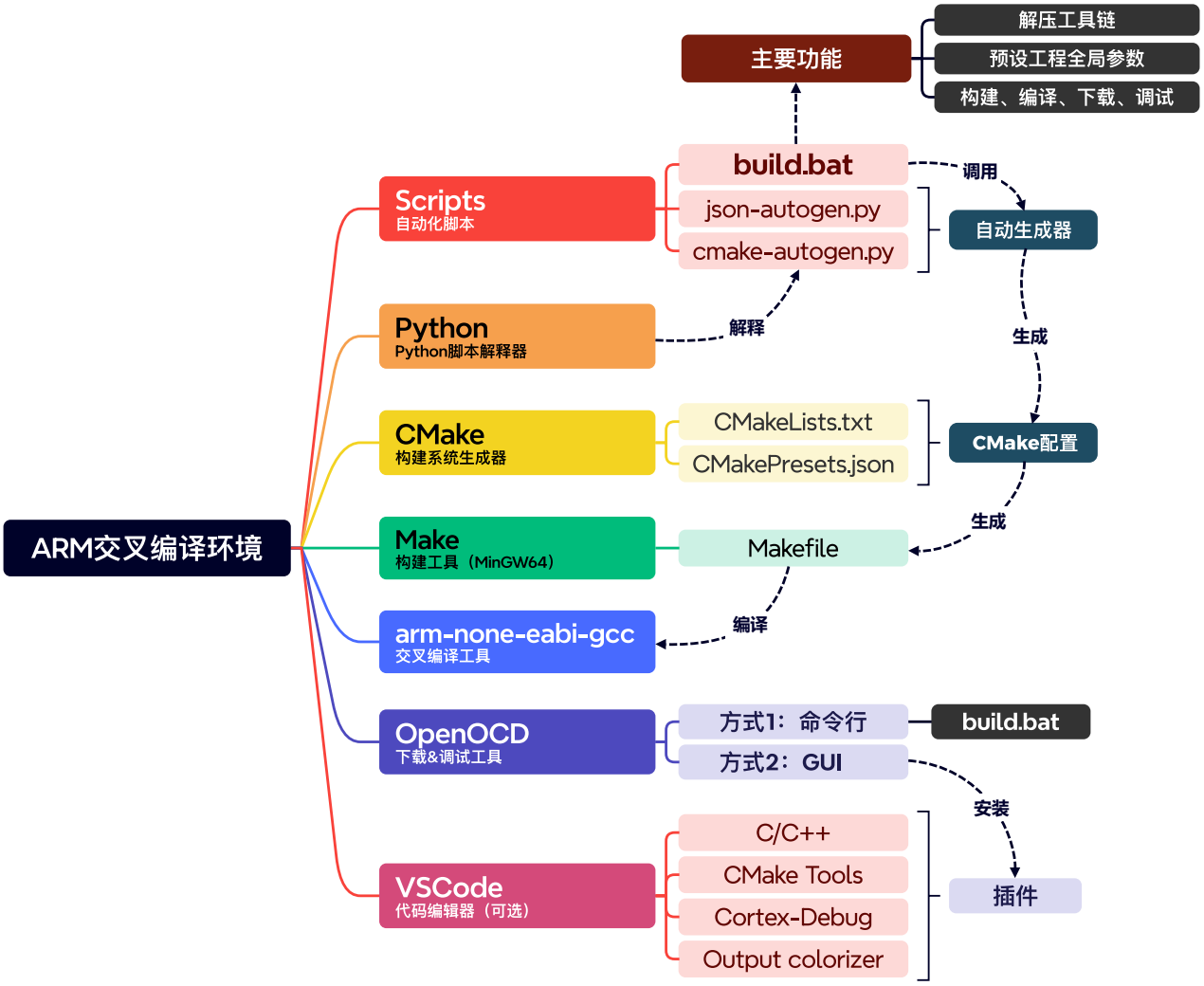

图 2：SDK 编译框架逻辑图

4 SDK 编译环境

4.1. 概述

Quectel UniKnect Project SDK 编译环境运行的主机系统建议使用 64 位版 Windows10。

SDK 中已集成了完整的编译工具链，用户可选择使用 SDK 内置工具链，也可自行搭建编译环境，二者在搭建难度、占用空间方面的对比如***表 1*** 所示。

表 1：SDK 编译环境搭建方式对比

对比	使用 SDK 内置编译工具链（推荐）	用户自行搭建
搭建难度	无需搭建，开箱即用	按本文档 4.3 章节搭建，可能略有波折
	★☆☆☆☆	★★★★☆
占用空间	完整工具链 2G，7z 压缩后 332M 解压时间约 4 分钟	不占空间
	★★★★☆	☆☆☆☆☆

4.2. 使用 SDK 内置编译工具链

为简化项目开发流程，减少客户在搭建编译环境问题上的精力消耗，实现开箱即用，SDK 内置了完整的编译工具链。首次使用 SDK，执行 ***build.bat*** 脚本时，工具链会自动解压，解压路径：tools\toolchain

工具链解压后的目录结构如***图 3*** 所示。

```
├── arm-gcc      # 交叉编译工具
├── cmake        # 构建系统生成器
├── mingw64      # Make 等构建工具
├── openocd      # 下载和调试工具
└── python       # Python 脚本解释器
```

图 3：SDK 内置编译工具链目录结构

4.3. 自行搭建编译环境

如果用户不想使用 SDK 内置工具链，选择自行搭建开发环境，本节将介绍如何从零开始搭建。

备注

选择使用 SDK 内置编译工具链的用户，本节可跳过不看。

4.3.1. 安装 ARM-GCC

1. 下载：*arm-gnu-toolchain-13.3.rel1-mingw-w64-i686-arm-none-eabi.zip*
2. 解压至 ***D:\Toolchain\arm-gcc***，如*图 4* 所示。

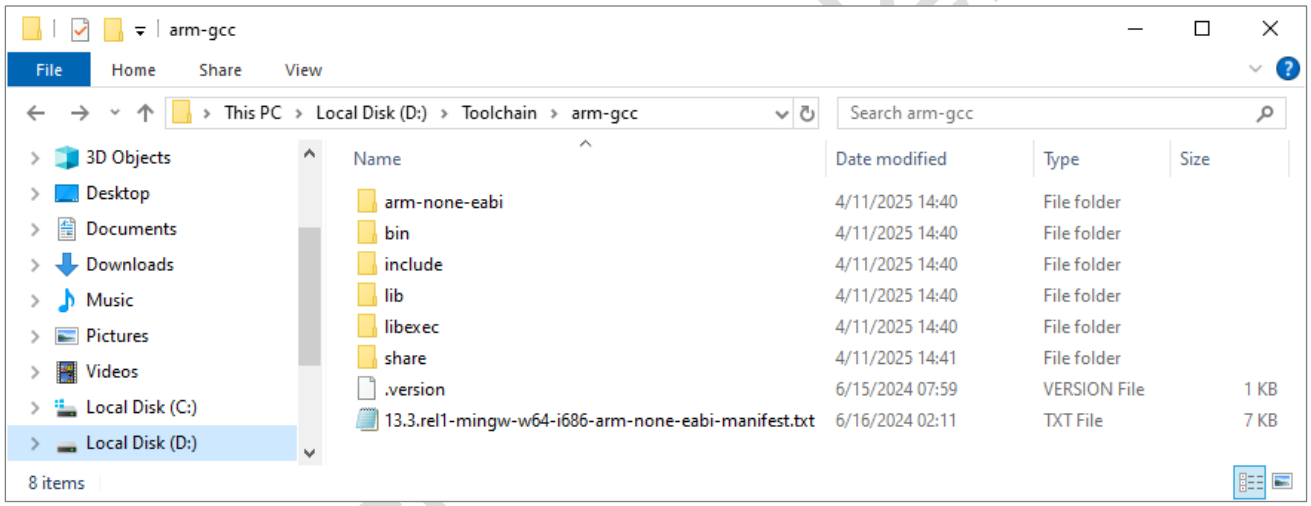

图 4：安装 ARM-GCC

3. 添加 ***D:\Toolchain\arm-gcc\bin*** 到环境变量 **Path**。
   右键【This PC】➔【Properties】➔【Advanced system settings】
   然后按照*图 5* 操作：

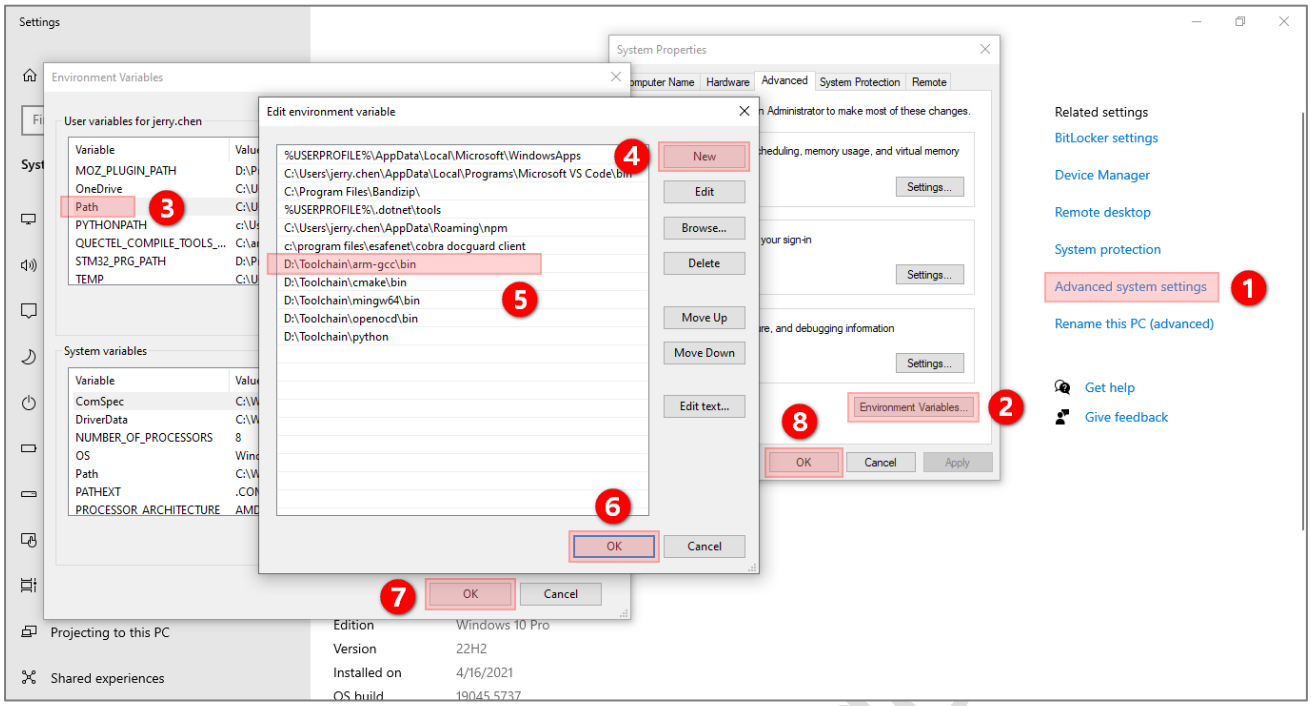

图 5：设置环境变量

4. 验证安装 ***arm-none-eabi-gcc -v***

   如***图 6*** 所示，说明 ARM-GCC 已安装成功，且环境变量已配置成功。

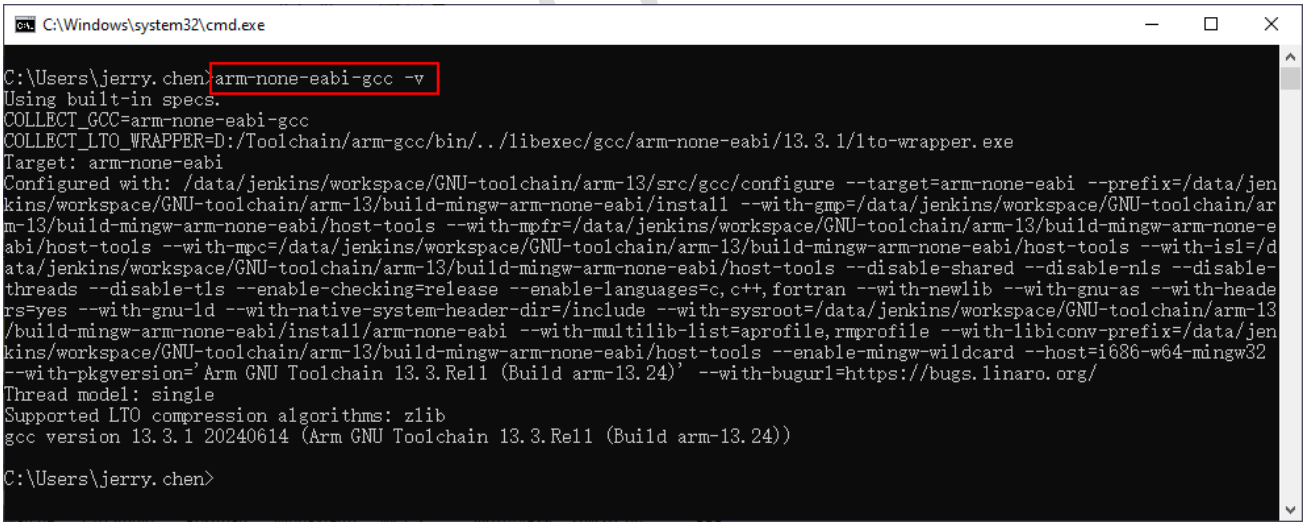

图 6：验证 ARM-GCC

4.3.2. 安装 CMake

1. 下载：*cmake-3.31.3-windows-x86_64.msi*
2. 安装至 ***D:\Toolchain\cmake***
   注意保持 ☑ Add CMake to the PATH environment variable 勾选，否则后续需要手动添加环境变量，如***图 7*** 所示。

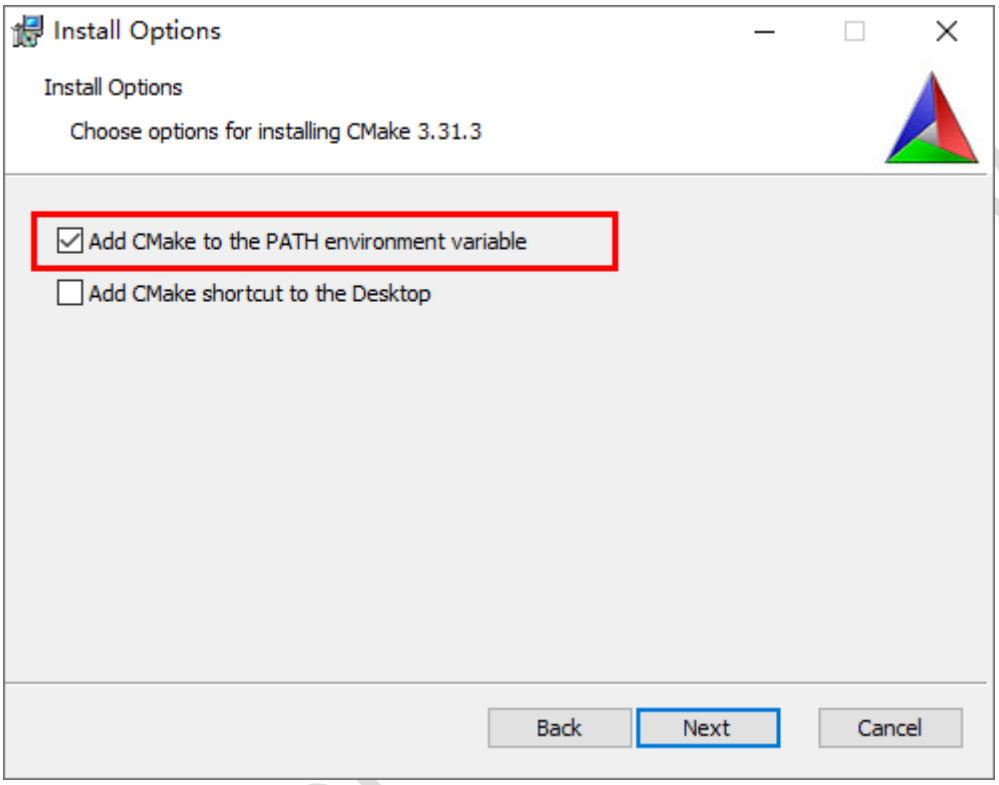

图 7：安装 CMake

5. 验证安装 ***cmake --version***
   如***图 8*** 所示，说明 CMake 已安装成功，且环境变量已配置成功。

图 8：验证 CMake

4.3.3. 安装 MinGW

1. 下载：x86_64-14.2.0-release-posix-seh-ucrt-rt_v12-rev2.7z
2. 解压至 ***D:\Toolchain\mingw64***，如***图 9*** 所示。

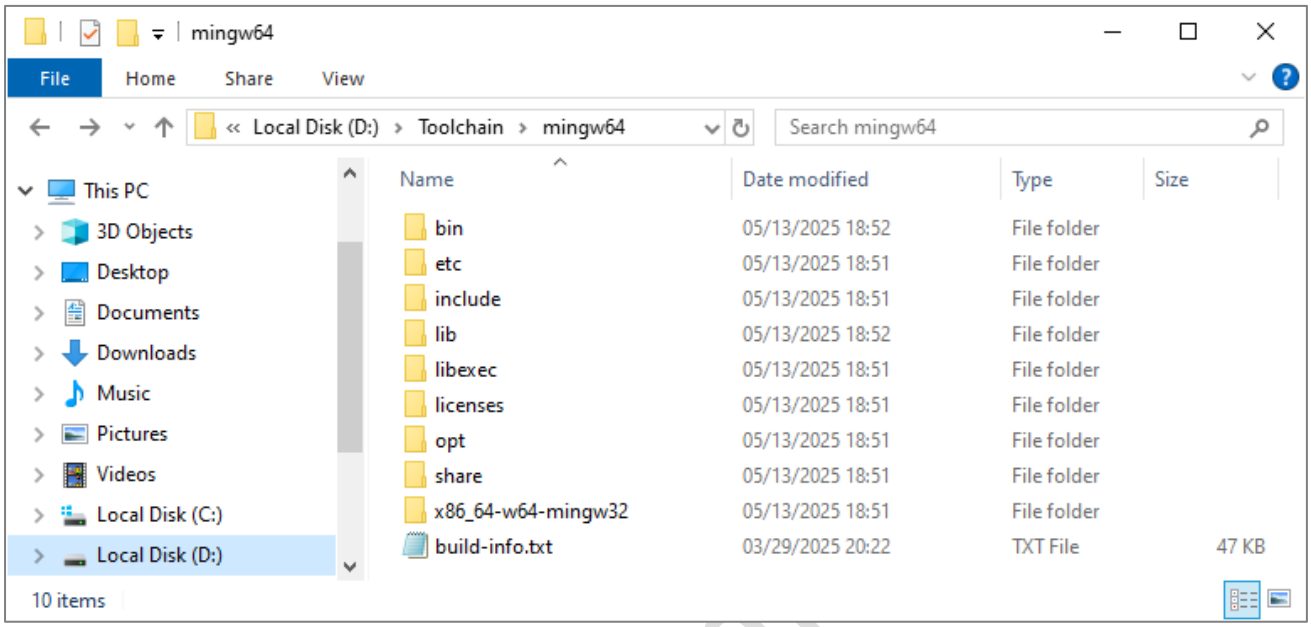

图 9：安装 MinGW

3. 将 ***D:\Toolchain\mingw64\bin*** 目录下的 ***mingw32-make.exe*** 复制一份副本，并重命名为 ***make.exe***，如***图 10*** 所示。

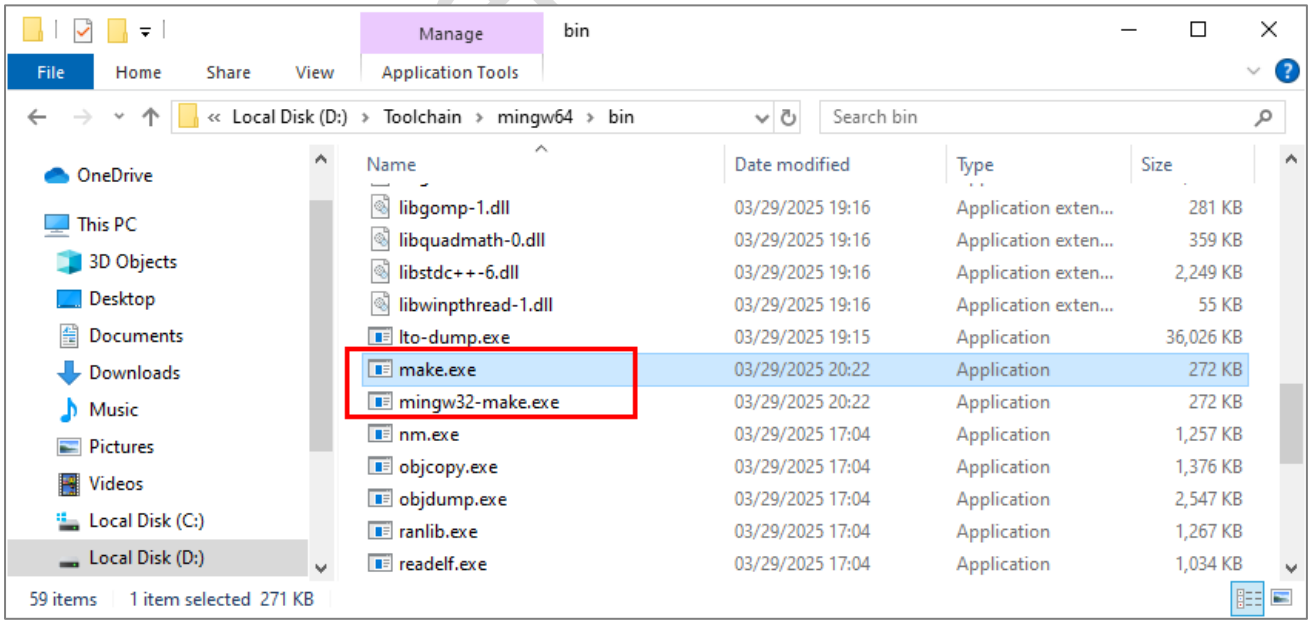

图 10：复制并重命名 make.exe

4. 添加 ***D:\Toolchain\mingw64\bin*** 到环境变量 **Path**，方法同 4.3.1。
5. 验证安装 ***make -v***
   如***图 11*** 所示，说明 MinGW 已安装成功，且环境变量已配置成功。

图 11：验证 MinGW

4.3.4. 安装 OpenOCD

1. 下载：*openocd-0.12.0-20240916.7z*
2. 解压至 ***D:\Toolchain\openocd***，如***图 12*** 所示。

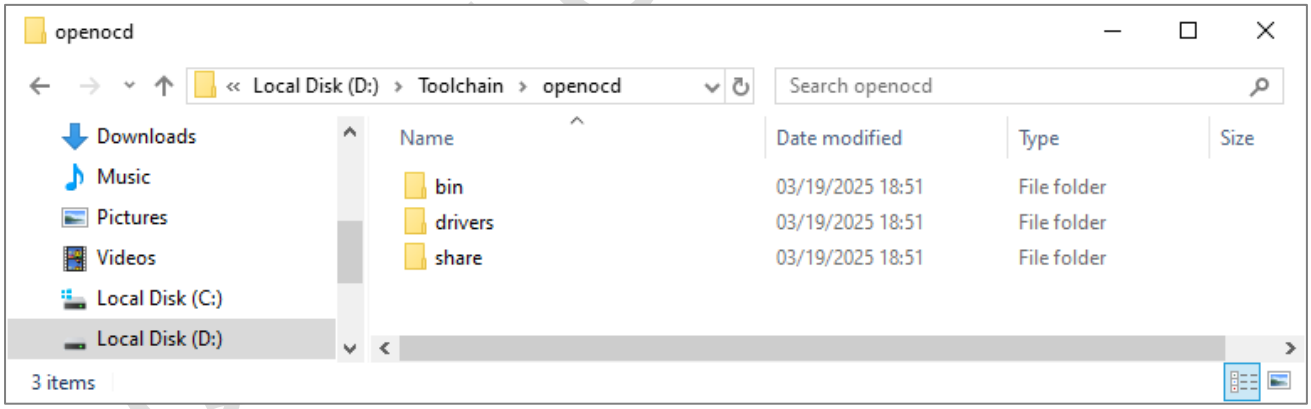

图 12：安装 OpenOCD

3. 添加 ***D:\Toolchain\openocd\bin*** 到环境变量 **Path**，方法同 4.3.1。
4. 验证安装 ***openocd -v***
   如***图 13*** 所示，说明 OpenOCD 已安装成功，且环境变量已配置成功。

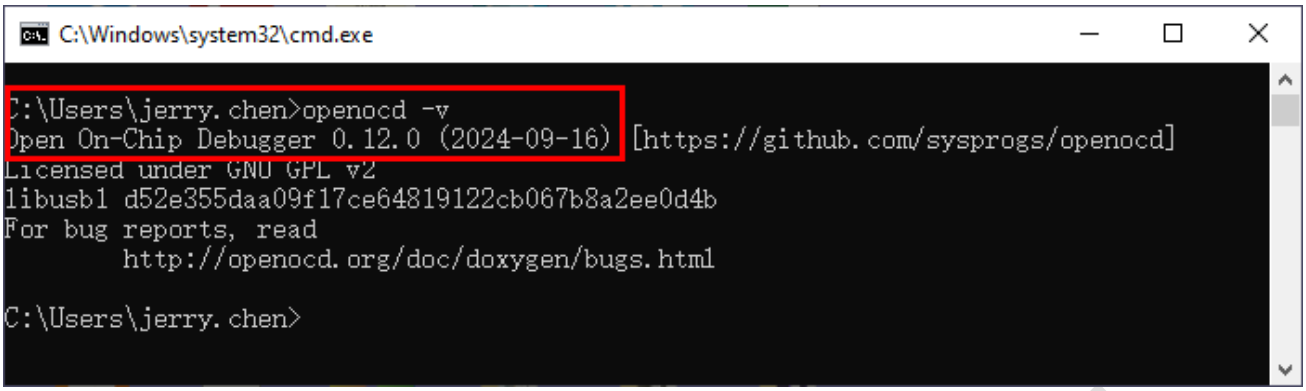

图 13：验证 OpenOCD

4.3.5. 安装 Python

1. 下载：*python-3.9.6-amd64.exe*
2. 安装 Python。
   注意要先勾选 ☑ Add Python 3.9 to PATH，然后再点击 Install Now，否则后续需要手动添加环境变量，如***图 14*** 所示。

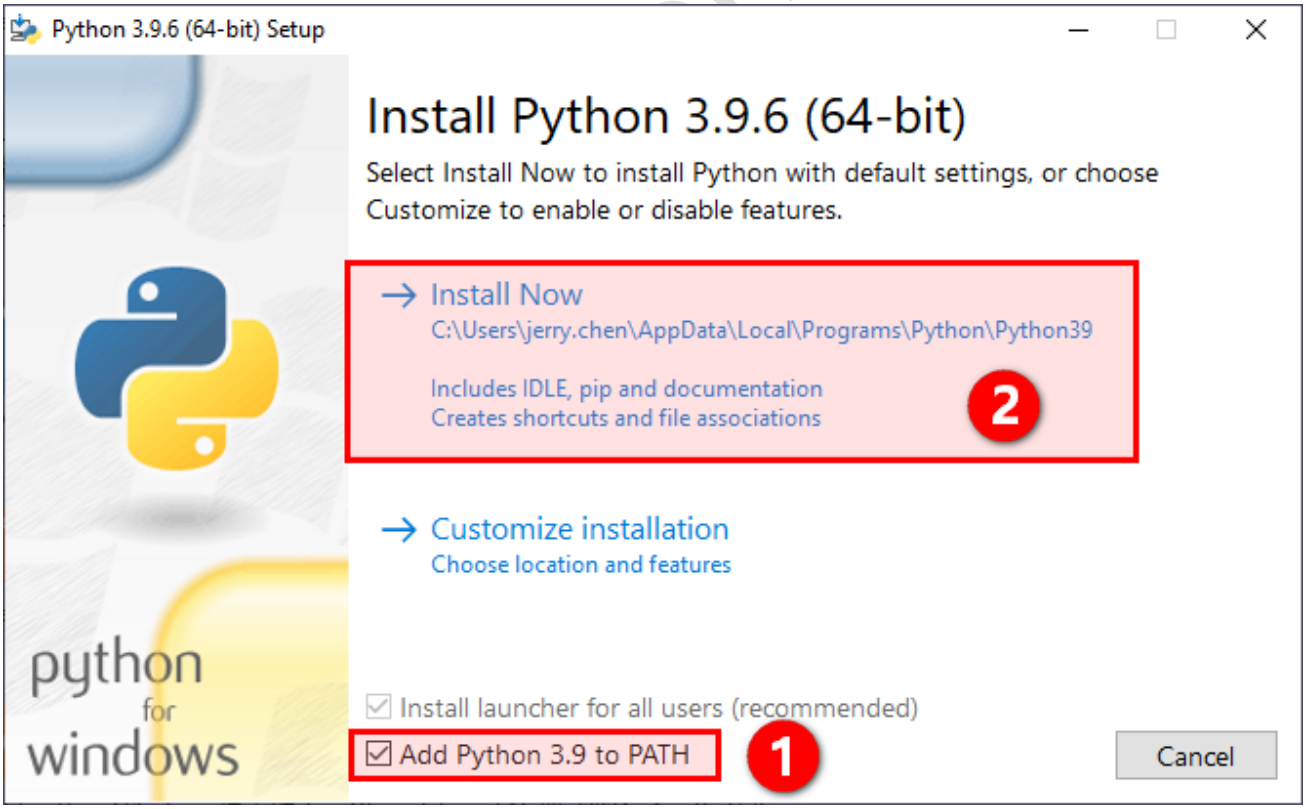

图 14：安装 Python

3. 验证安装 ***python***

   如***图 15*** 所示，说明 Python 已安装成功，且环境变量已配置成功。

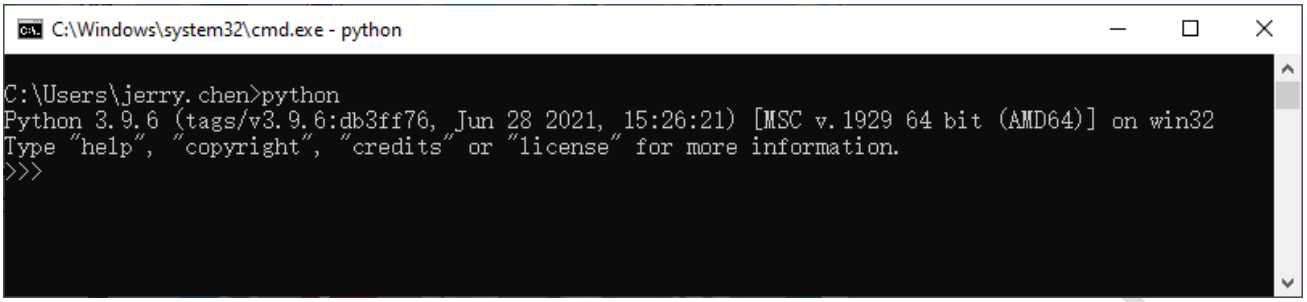

图 15：验证 Python

4.3.6. 删除 SDK 内置编译工具链

删除 ***tools*** 目录下的 ***toolchain.7z*** 工具包，如果先前已解压，则 ***tools\toolchain*** 文件夹也需一并删除，如***图 16*** 所示。

删除后，构建和编译脚本无法找到内置的工具链，会自动适配系统 **Path** 环境变量下的工具链。

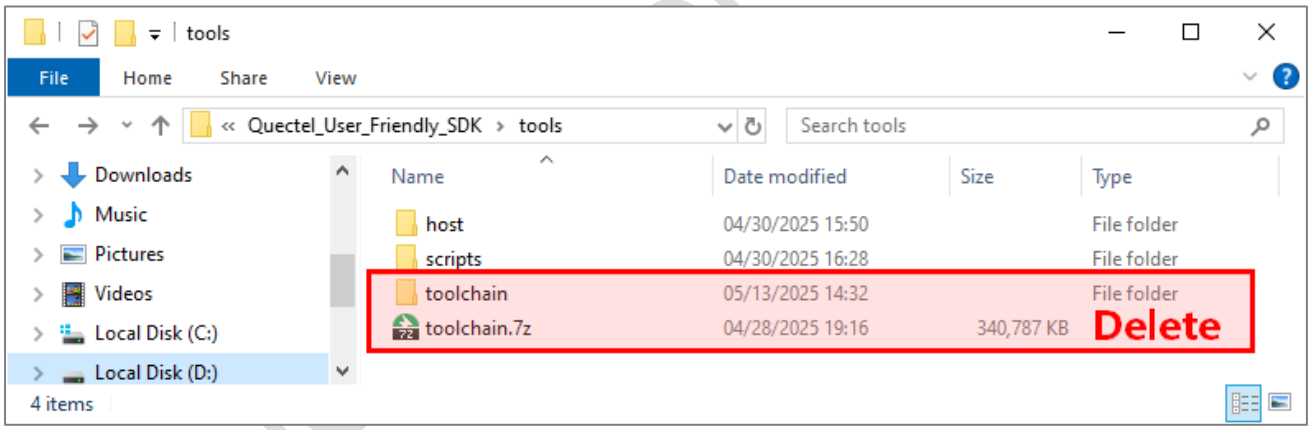

图 16：删除 SDK 内置编译工具链

5 硬件环境准备

5.1. 组件装配和线缆连接

在进行软件开发前，用户首先需要将硬件环境准备就绪。
请按照*图 17* 进行必要的组件装配和线缆连接。

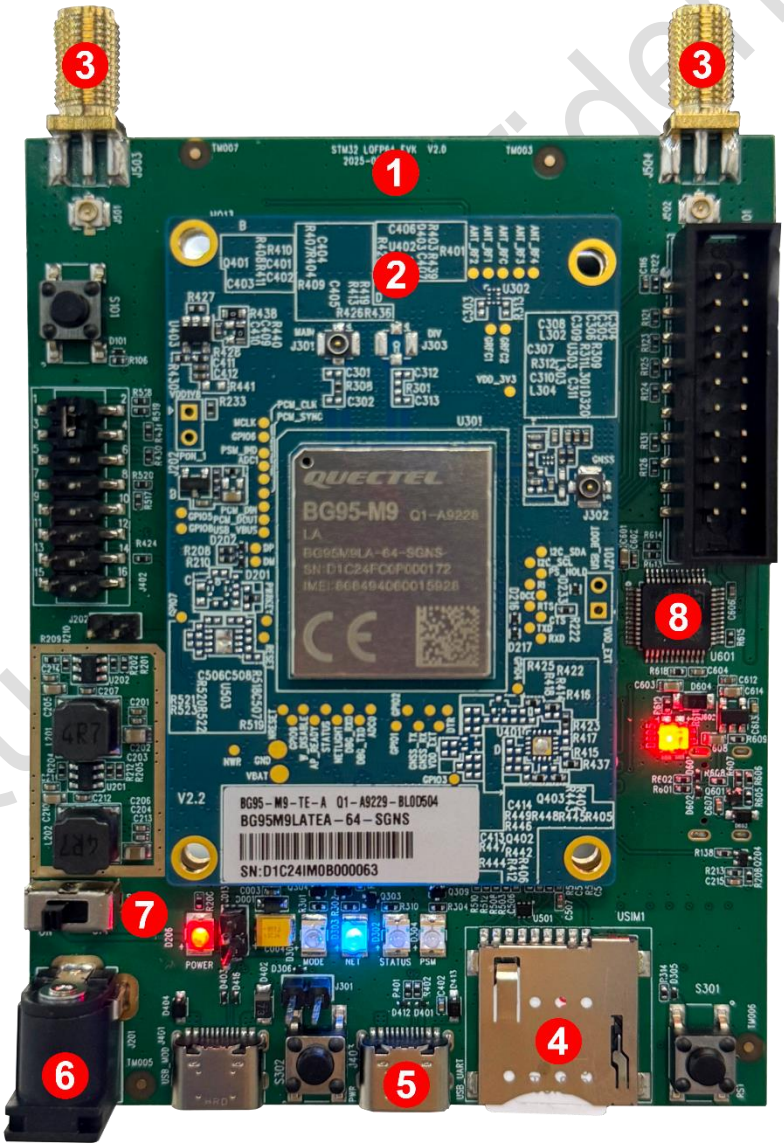

图 17：硬件环境准备

❶ EVK 底板
❷ TE-A 核心板
❸ 天线
❹ Micro SIM 卡
❺ Type-C USB 线（Debug）
❻ 5V-DC 电源适配器
❼ Power Switch（左 ON，右 OFF）
❽ ST-Link 调试器

硬件连接就绪后，将 ❼ Power Switch 拨到左侧，使硬件上电运行。

5.2. 安装驱动程序

5.2.1. 模组驱动

如需通过 USB 对模组收发 AT 命令或抓取 log，则应先根据 ❷ TE-A 核心板上的模组型号，安装对应的驱动。如果不需要上述操作，可暂不安装模组驱动。

例如模组型号是 BG95，则对应的驱动是：*Quectel_LTE&5G_Windows_USB_Driver_V2.2.4.zip*

想要获取模组对应的驱动程序，可发送邮件至：support@quectel.com

5.2.2. 调试器驱动

下载并安装 ❽ ST-Link 调试器的驱动：

- ST-Link : *https://www.st.com.cn/zh/development-tools/stsw-link009.html*

将 ST-Link 插入 PC，且成功安装驱动后，设备管理器中会显示 STM32 STLink，如*图 18* 所示。

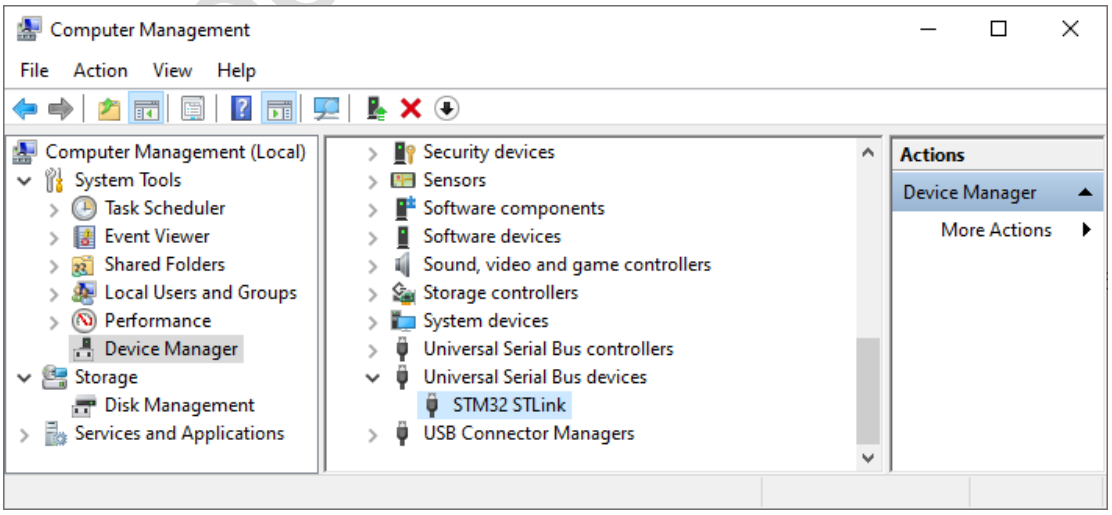

图 18：设备管理器中的调试器

5.2.3. 串口驱动

1. 下载驱动程序：*CP210x Universal Windows Driver*
2. 解压安装包 ➔ 右键单击【**silabser.inf**】 ➔ 点击【**Install**】，如*图 19* 所示。

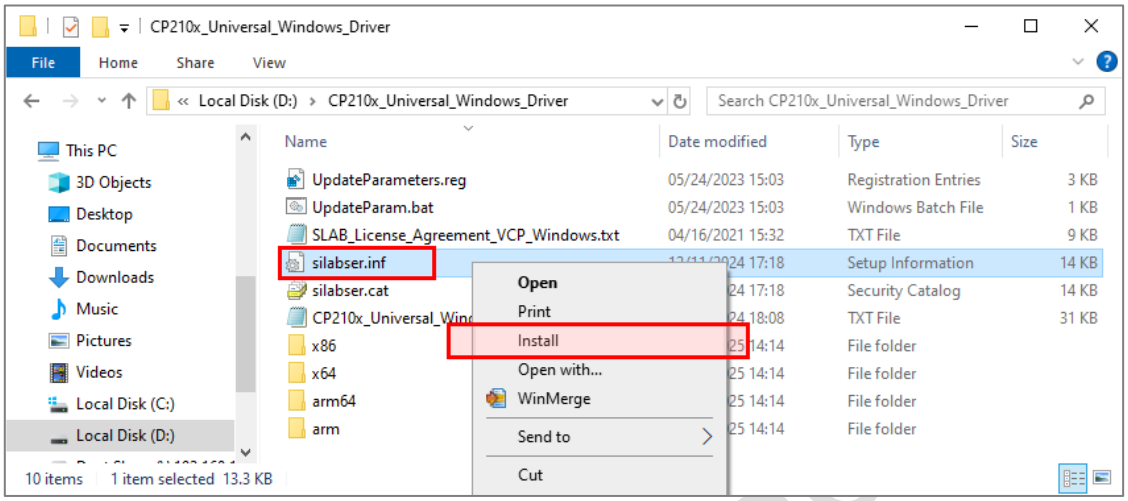

图 19：安装串口驱动程序

串口驱动程序安装成功后，打开设备管理器，会显示 Ports 列表。

其中的【**Silicon Labs Quad CP2108 USB to UART Bridge: Interface 1 (COM7)**】是用户后续要使用的 MCU Debug 端口，如*图 20* 所示。

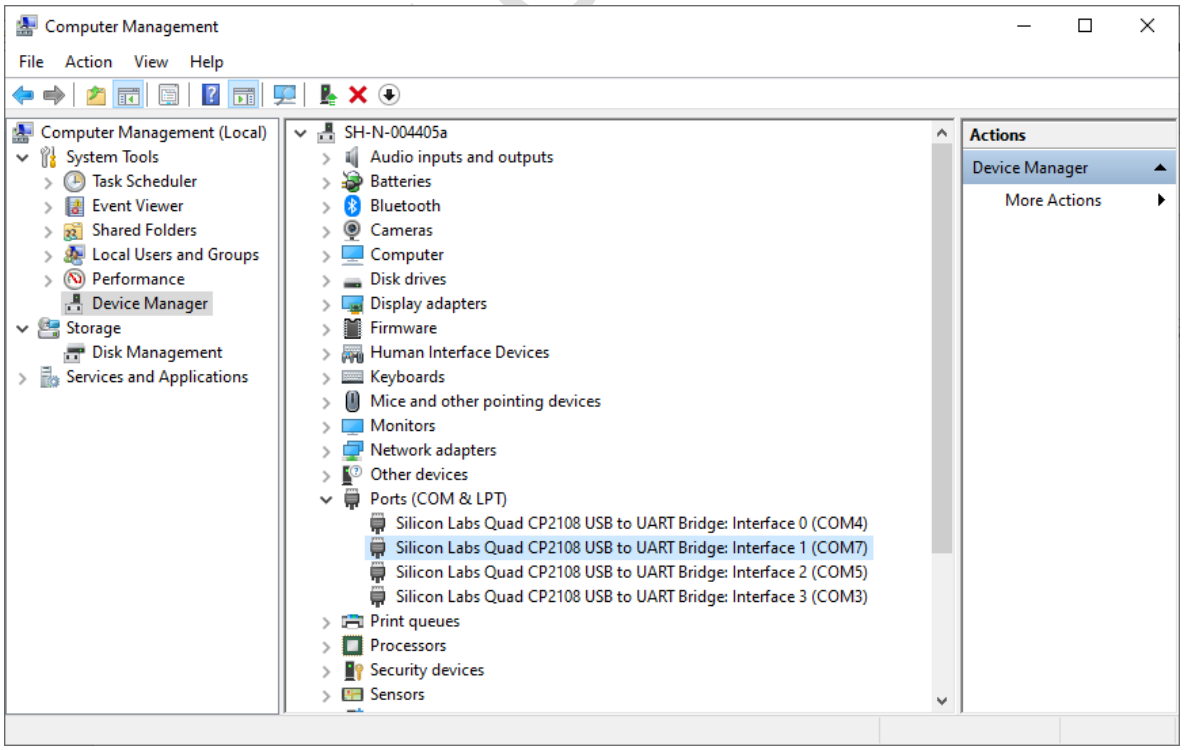

图 20：PC 设备管理器

随后打开串口终端交互工具，如 MobaXterm，端口选择 **COM7**，如*图 21* 所示。

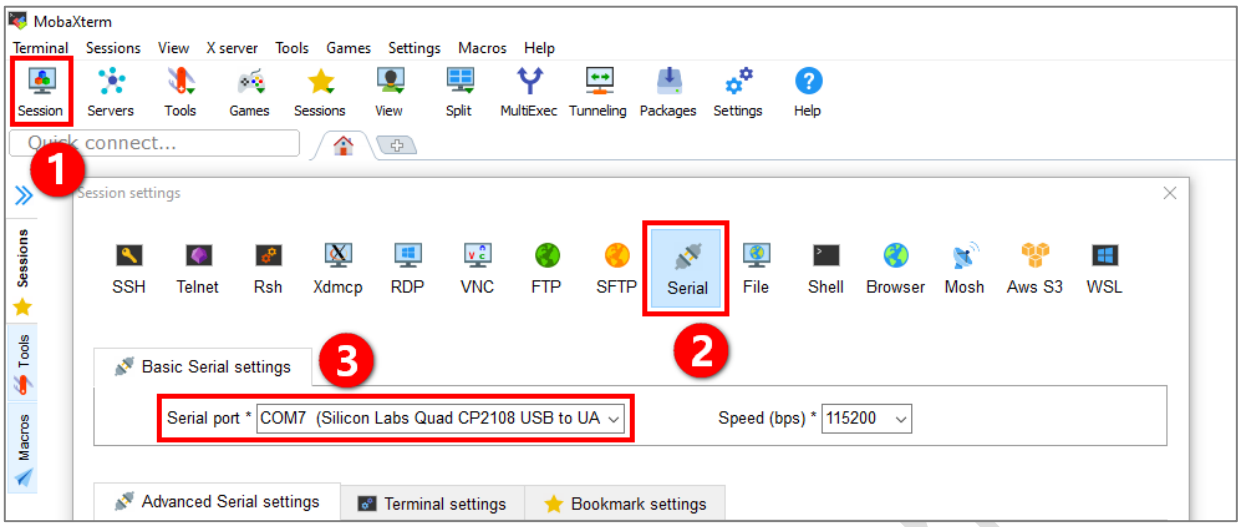

图 21：串口端终交互工具

备注

1. 端口号因电脑而异，不一定是 **COM7**，以实际为准，但要选择 **Interface 1**
2. 更详细的硬件使用说明，请参考*文档 [1]*

6 项目构建

Quectel UniKnect Project SDK 支持多种型号 MCU，使用本 SDK 开发前，需要执行构建命令，以配置指定的 MCU 型号、版本号、编译链接依赖、生成 CMakeLists.txt / xxx.json 等必要的构建文件，具体构建原理可参考***第 3 章***。

6.1. 构建命令

```
build.bat config [ChipType] [Version]
```

备注

build.bat config 命令后面可带 [芯片型号] [版本号] 两个参数，例如：
build.bat config STM32F413RGT6 your_firmware_version

[芯片型号] [版本号] 两个参数在缺省的情况下，使用上次配置的芯片型号和版本号。
若首次使用无先前配置记录，则芯片默认使用 **STM32F413RGT6**，版本号默认使用格式 **Quectel_UFP_Chip_Date**，例如 **Quectel_UFP_STM32F413RGT6_20250430**

命令行日志如***图 22*** 所示。

构建成功后，会在 SDK 根目录下自动生成***.vscode*** 和 ***build*** 两个目录，以及 ***CMakeLists.txt*** 和 ***CMakePresets.json*** 两个文件，如***图 23*** 所示。

图 22：构建命令执行日志

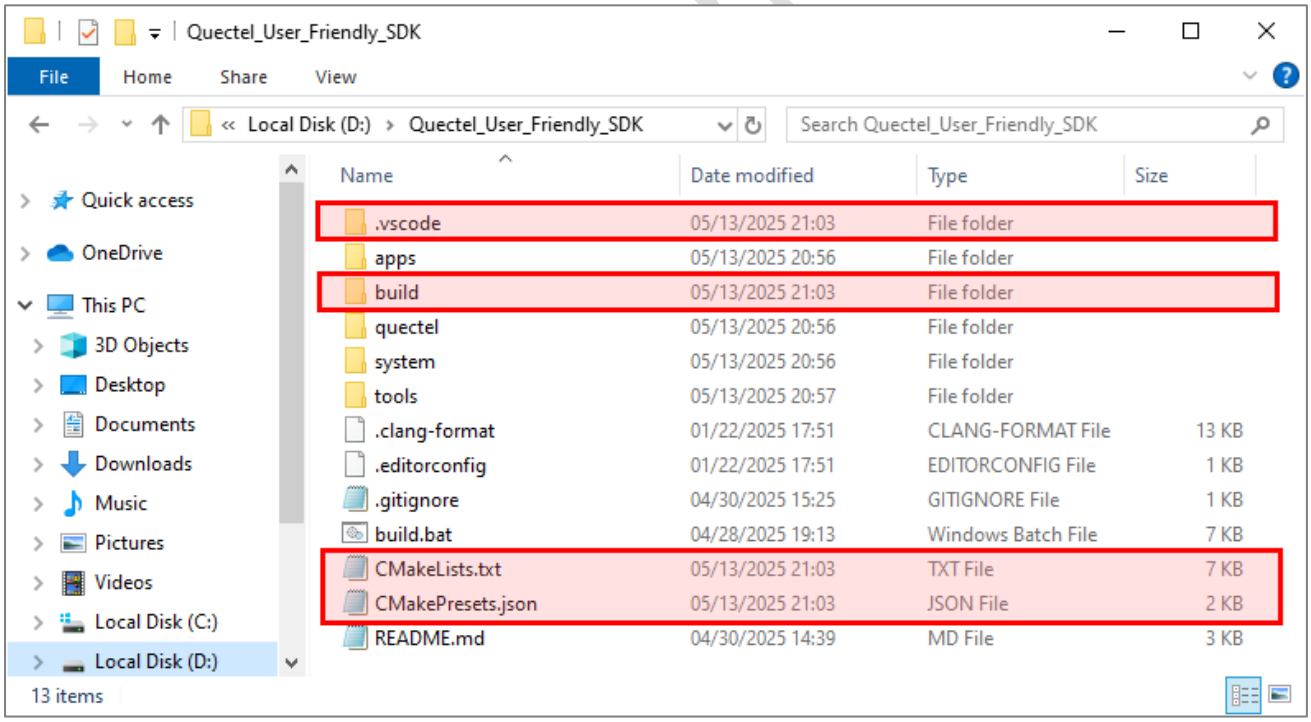

图 23：构建命令自动生成的文件

7 开发调试

Quectel UniKnect Project SDK 提供 2 种开发调试的操作方式，用户可根据习惯任选其一，或二者结合使用：

1. **命令行：** 不依赖任何 IDE 或代码编辑器
2. **GUI：** VSCode + 插件（相关参数自动生成，无需关注）

备注

选择使用命令行操作方式的用户，只参考 7.1 章节即可。
选择使用 GUI 操作方式的用户，只参考 7.2 章节即可。
二者结合使用的用户，7.1 和 7.2 需同时参考。

7.1. 命令行操作

7.1.1. 编译

编译命令：

```
build.bat all
```

执行 ***build.bat all*** 命令的日志如***图 24*** 所示，说明已编译成功。
编译成功后，会在 ***build*** 目录下生成 ***elf / hex / bin / map*** 等目标文件，如***图 25*** 所示。

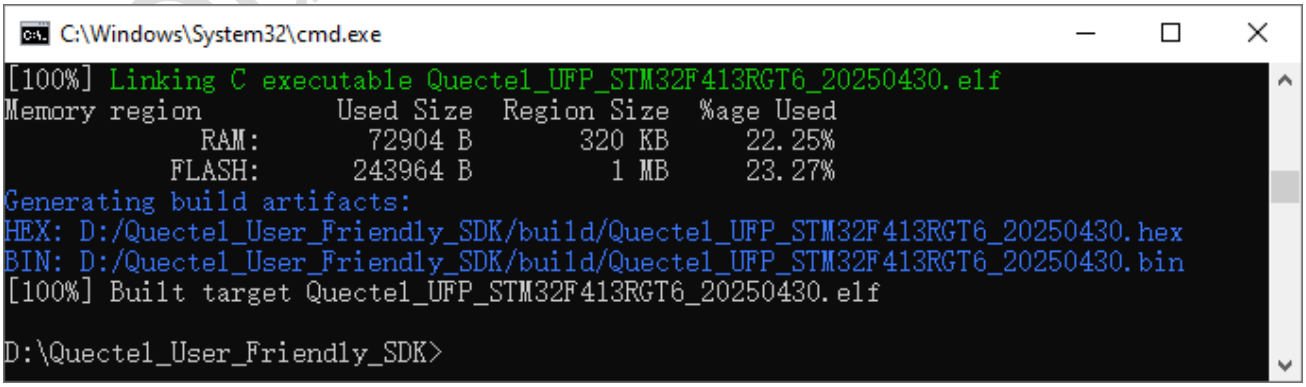

图 24：编译日志

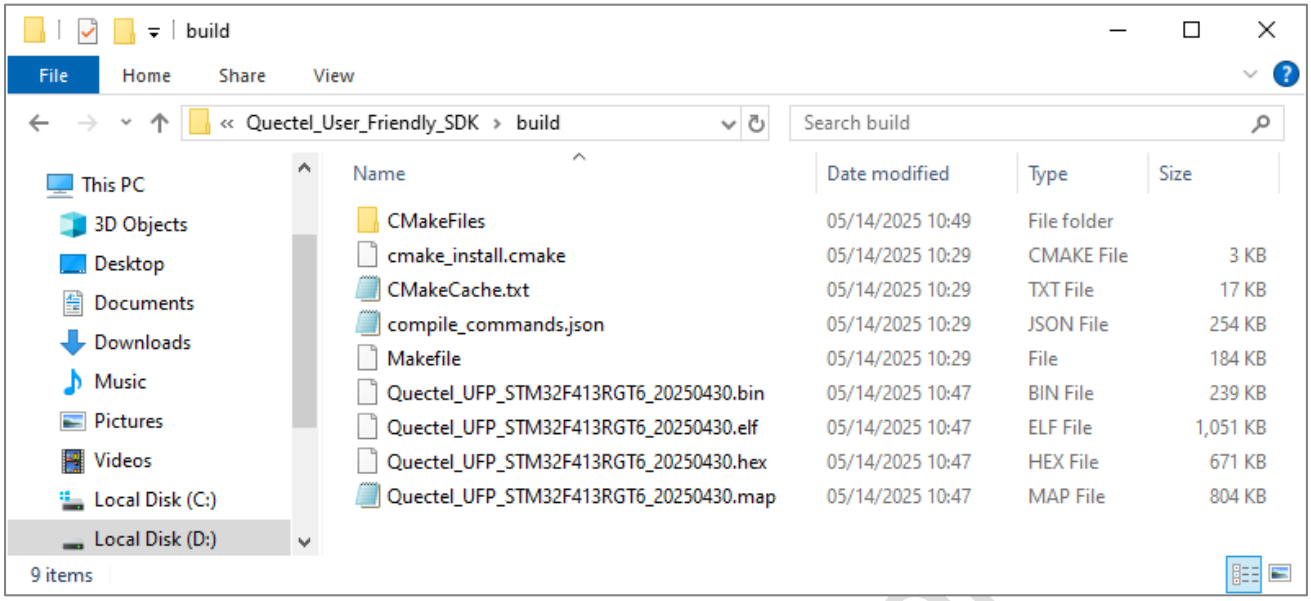

图 25：编译产生的目标文件

7.1.2. 清理

清理命令：

```
build.bat clean
```

执行 ***build.bat clean*** 命令的日志如*图 26* 所示，说明已清理成功。
如构建命令未成功执行，或需更彻底的清理，请直接删除 SDK 根目录下的 ***build*** 目录。

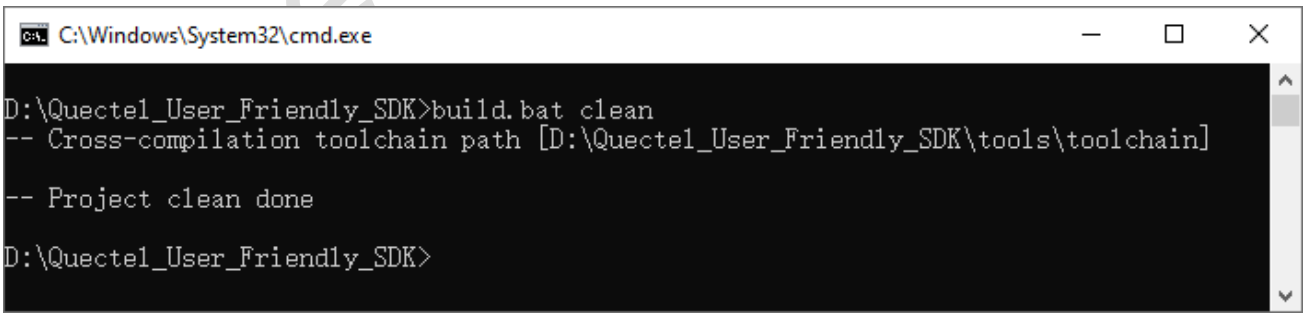

图 26：清理日志

7.1.3. 下载

下载命令：

```
build.bat download
```

执行 ***build.bat download*** 命令的日志如***图 27*** 所示，说明已下载成功。
MCU 开机日志如***图 28*** 所示，说明已启动成功。

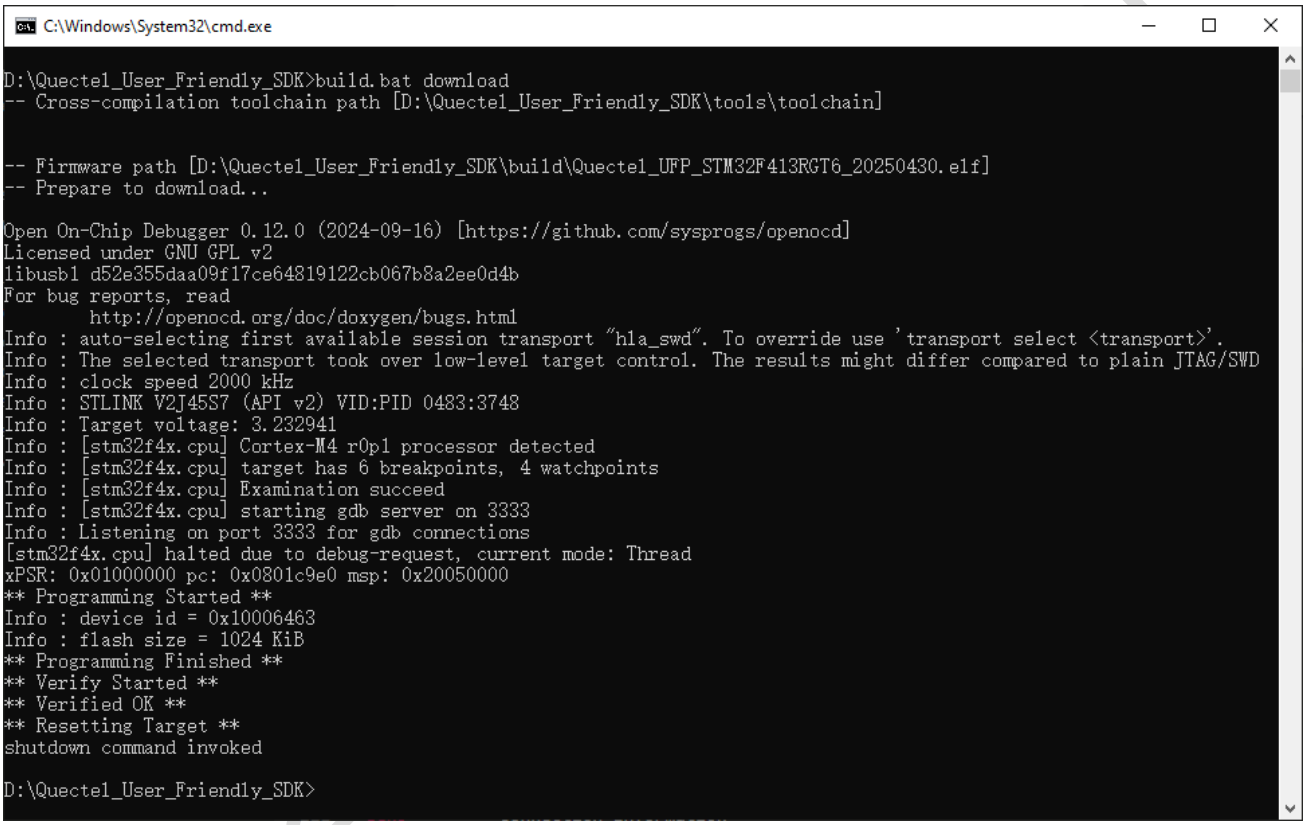

图 27：下载日志

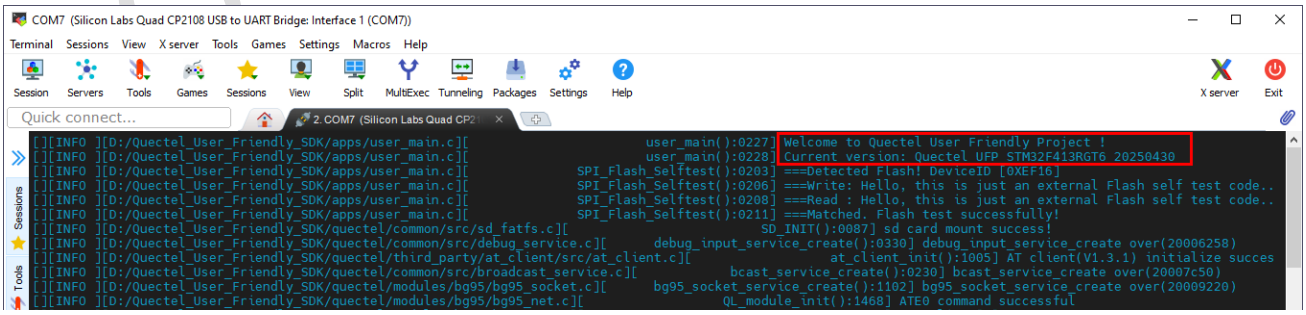

图 28：开机日志

7.1.4. 调试

调试命令：

```
build.bat debug
```

执行 ***build.bat debug*** 命令后，系统会启动一个名为 OpenOCD 的 GDB 服务进程，出现如***图 29*** 所示的日志界面后，说明已成功进入调试模式。

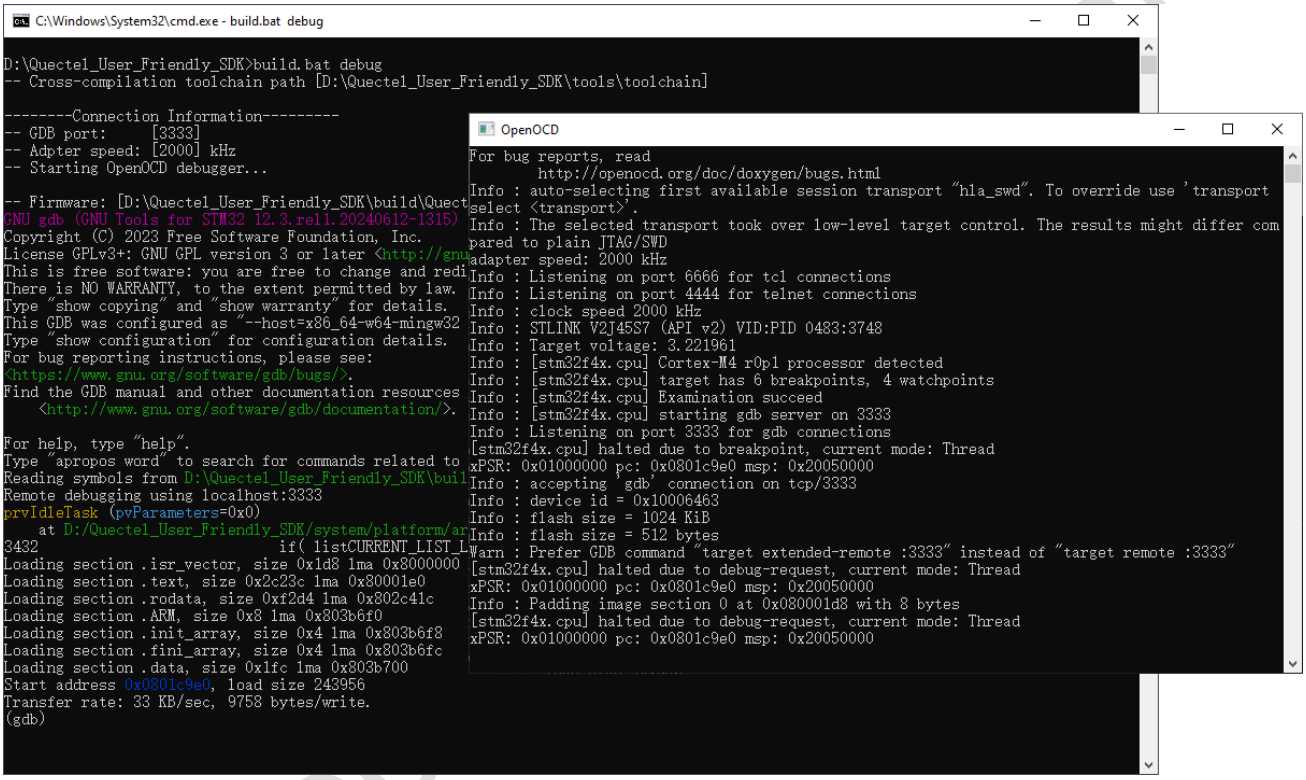

图 29：GBD 调试界面

GDB 调试常用命令可参考***表 2***。

表 2：GDB 调试常用命令

功能	命令	描述	示例
设置断点	b <位置> 或 break	在函数或地址处设置断点	b main b *0x08001234
单步跳过	n 或 next	执行下一行代码（跳过函数）	n
单步进入	s 或 step	执行下一行代码（进入函数）	s
继续运行	c 或 continue	继续执行到下一个断点或结束	c
查看内存	x/<格式> <地址>	查看内存 格式：x 十六进制	x/4x 0x20000000 （查看 4 个 32 位值）
查看变量值	p <变量> 或 print	打印变量或表达式	p cnt p (uint32_t*)0x20000000
查看寄存器	p/x $r0 ~ p/x $r15	查看 ARM 寄存器	p/x $sp（查看栈指针）
查看所有寄存器	info reg	查看所有寄存器	info reg
查看调用栈	bt 或 backtrace	查看函数调用栈	bt
切换栈帧	f <编号> 或 frame	切换到指定层级的栈帧	frame 1
删除断点	delete <编号>	删除指定断点	delete 2
列出源码	l 或 list	查看当前或指定位置的源码	list 20,30 （显示 20-30 行）
复位 MCU	monitor reset	复位 MCU	monitor reset
退出调试	q 或 quit	退出 GDB	q

7.2. GUI 操作

7.2.1. 安装 VSCode 及插件

1. 下载：*https://code.visualstudio.com*
2. 安装 VSCode，按提示操作即可。
3. 安装 VSCode 插件。
   打开 VSCode，点击界面左侧 **Extensions** 按钮，或按【**Ctrl + Shift + X**】唤出插件管理器，在搜索栏中搜索并安装以下 4 个插件，如***图 30*** 所示。

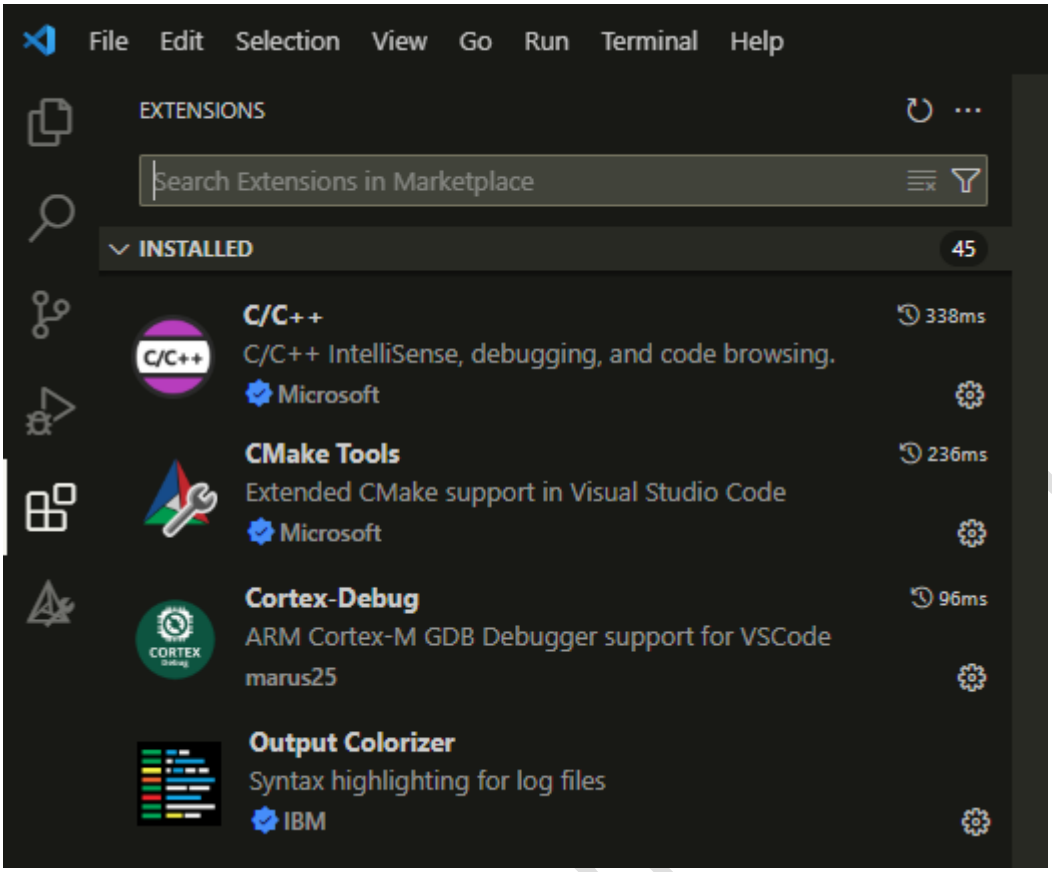

图 30：安装 VSCode 插件

7.2.2. 配置

首先打开 Quectel_UniKnect_SDK 工程。

使用快捷键【**Ctrl + K　Ctrl + O**】，或点击 VSCode 菜单栏【File】➔【Open Folder…】，选择 SDK 工程文件夹，如*图 31* 所示。

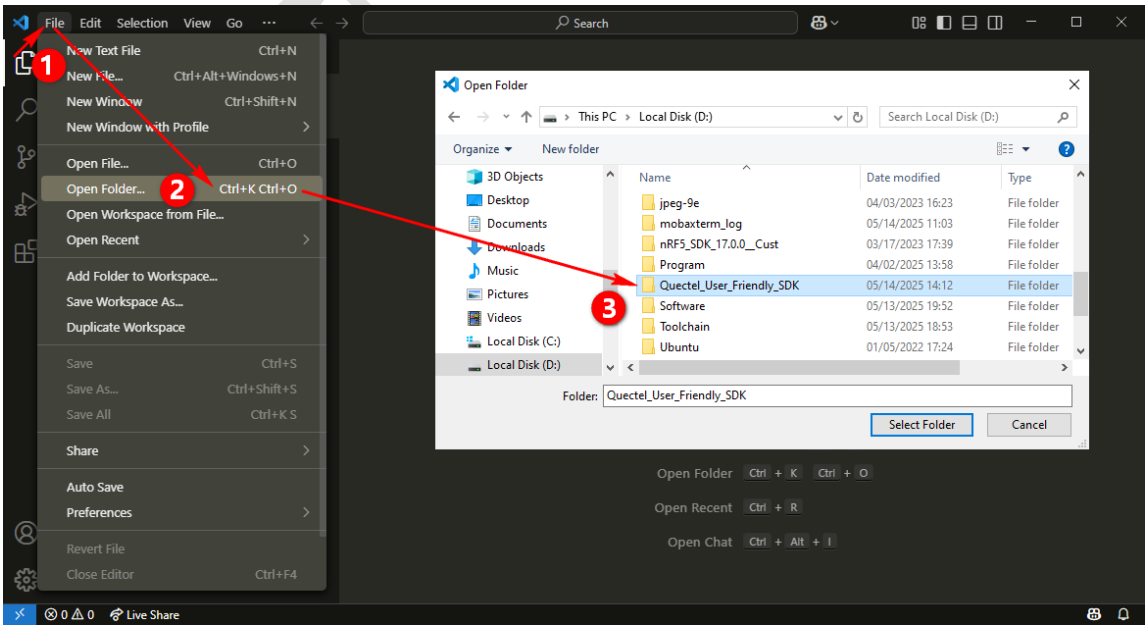

图 31：VSCode 中打开工程文件夹

工程文件夹打开后，VSCode 会自动加载 CMake Tools 插件，并自动进行相关配置。如果没有自动配置，或删除了 **build**，可点击 CMake Tools 插件中的【**Delete Cache and Reconfigure**】按钮，进行重新配置，如*图 32* 所示。

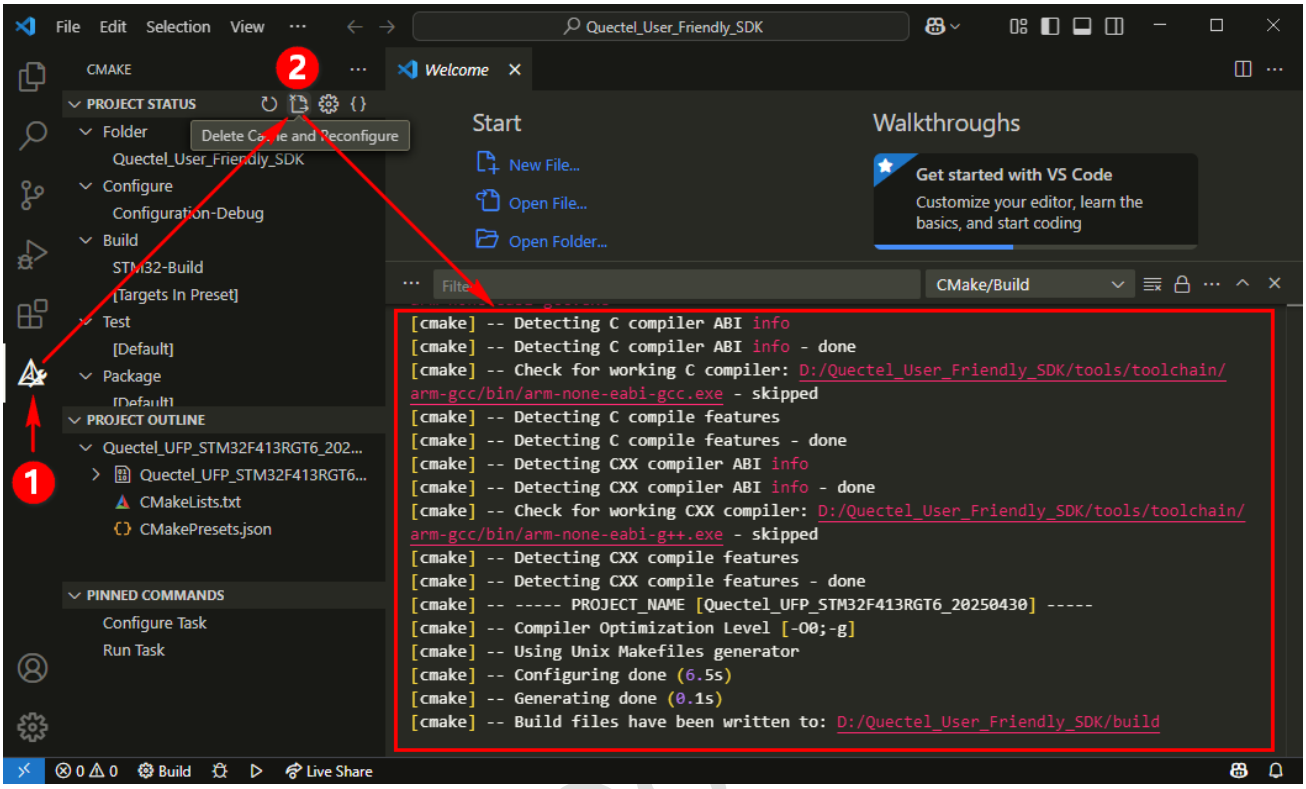

图 32：VSCode 中执行 CMake 配置

7.2.3. 编译

使用快捷键【**F7**】，或点击 CMake Tools 插件中的【**Build all projects**】按钮，如*图 33* 所示。

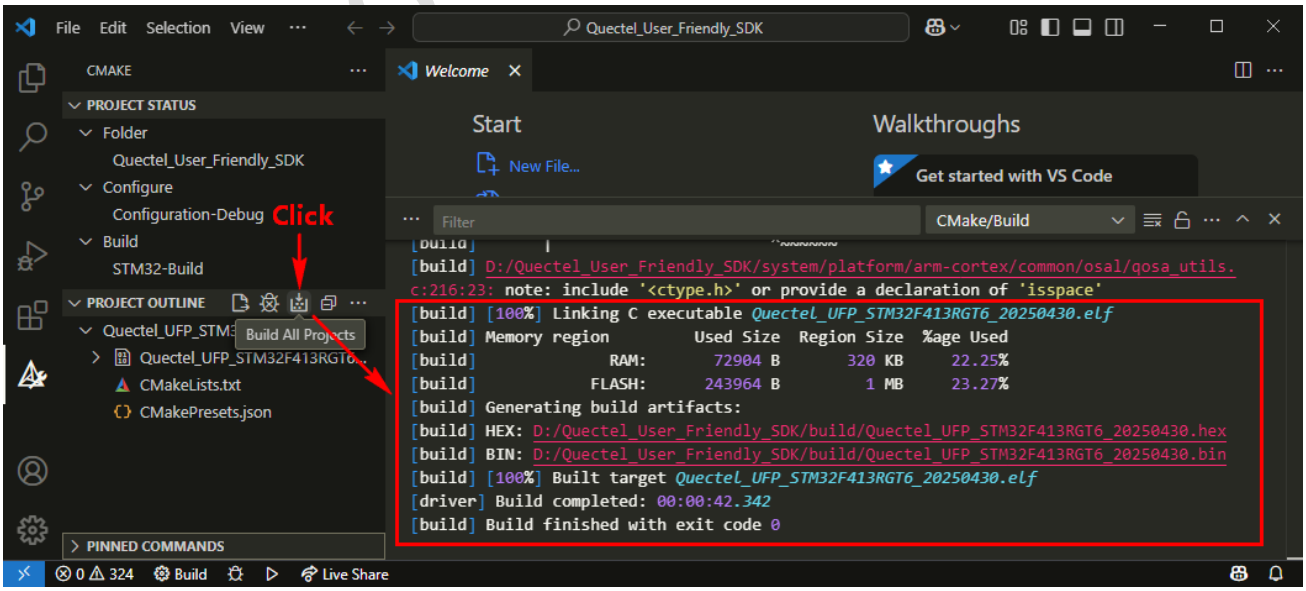

图 33：VSCode 中执行编译

7.2.4. 清理

点击 CMake Tools 插件中的【**Clean all projects**】按钮，如*图 34* 所示。
如构建命令未成功执行，或需更彻底的清理，请直接删除 SDK 根目录下的 ***build*** 目录。

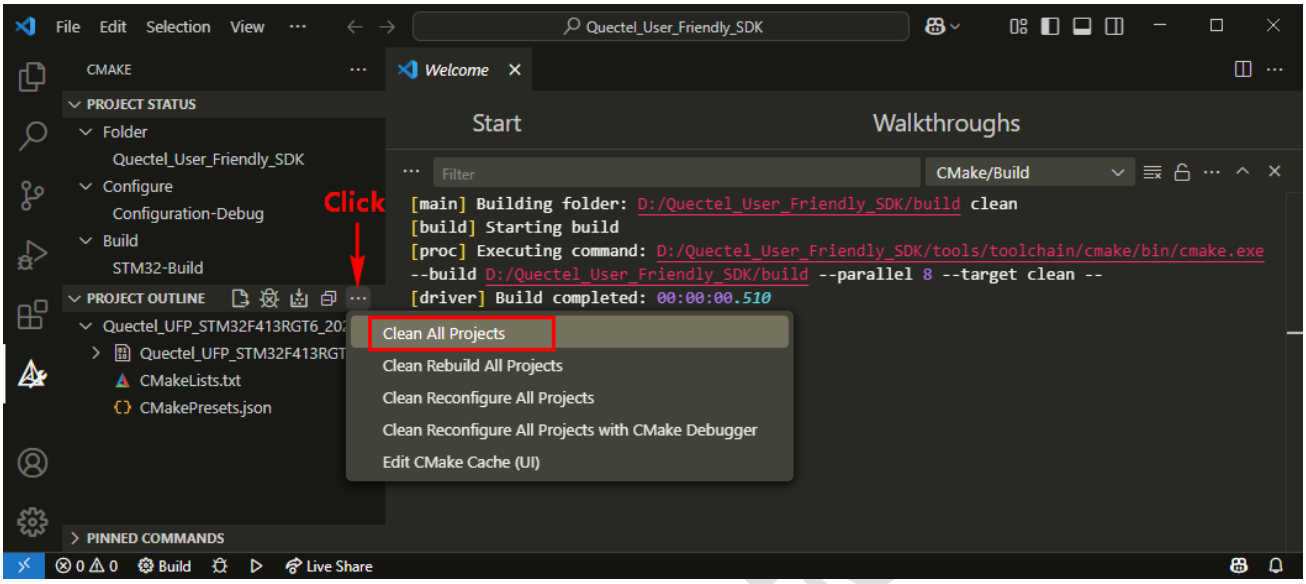

图 34：VSCode 中执行清理

7.2.5. 下载

使用快捷键【**Ctrl + Shift + B**】，或点击 VSCode 菜单栏【Terninal】➔【Run Task…】，如*图 35* 所示。在弹出的任务面板中，点击【**Download**】任务，开始下载，如*图 36* 所示。

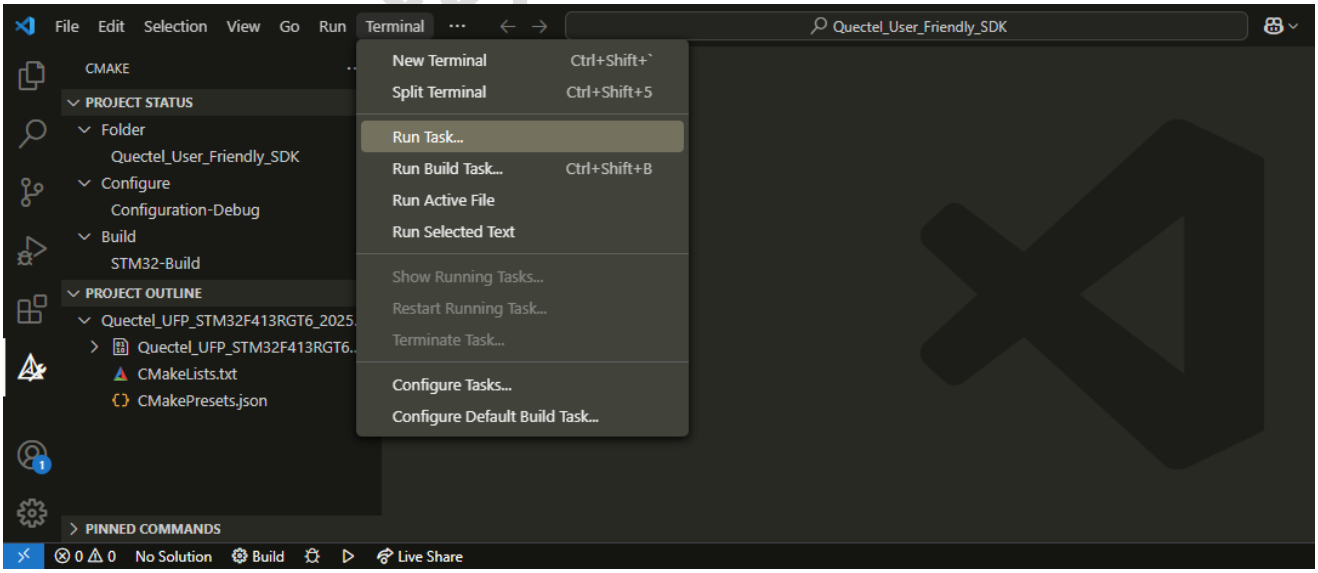

图 35：VSCode 中打开任务面板

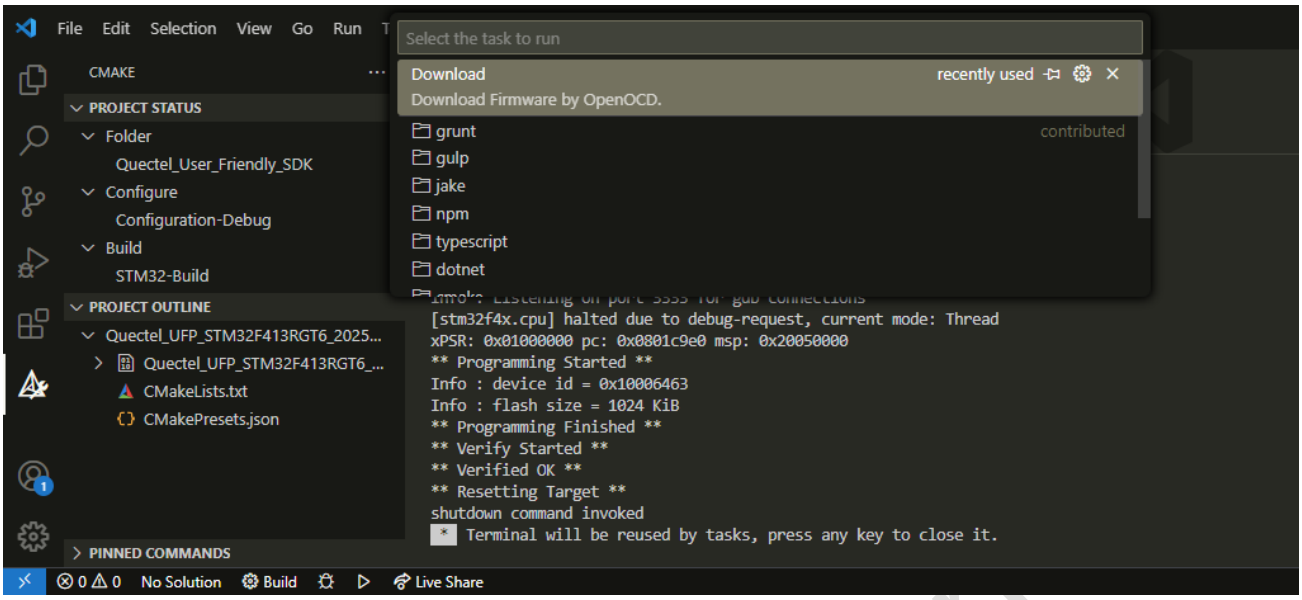

图 36：VSCode 中执行下载

7.2.6. 调试

点击 CMake Tools 插件中的**【Configure All Projects with CMake Debugger】**按钮，配置完成后，下方的调试安钮会变成 **Debug with OpenOCD**，如*图 37* 所示。

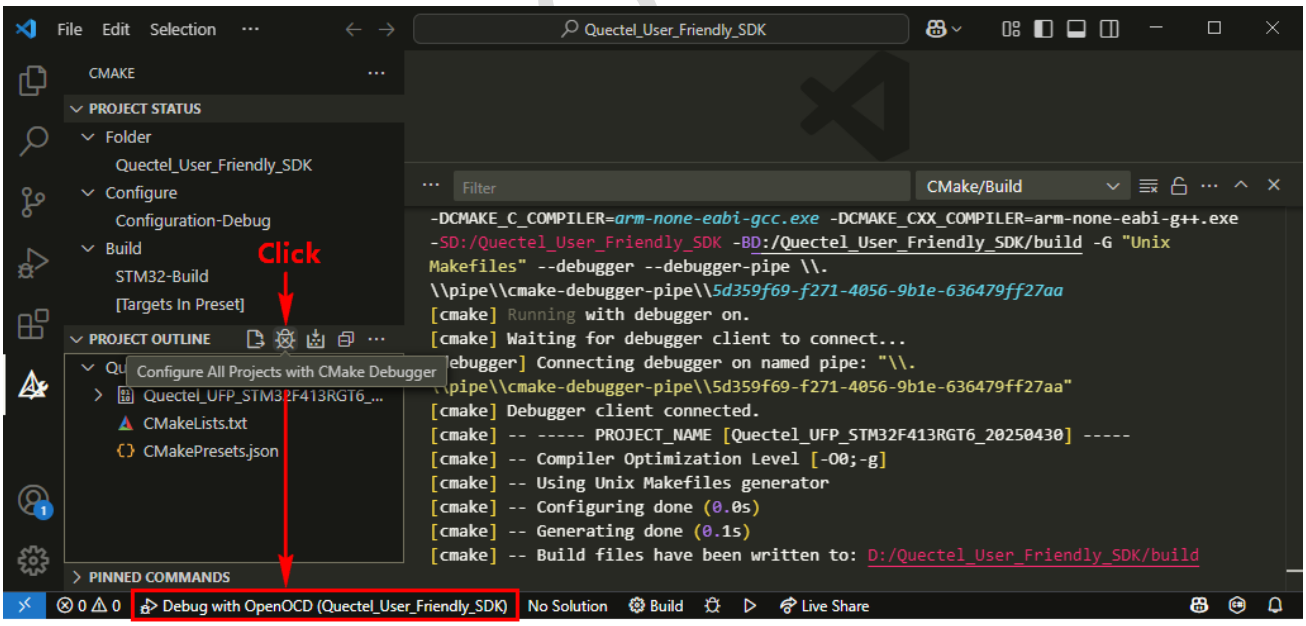

图 37：VSCode 中配置调试器

接着再点击下方的调试按钮【**Debug with OpenOCD**】，上方会唤起配置面板，点击【**Debug with OpenOCD**】启动调试进程。

启动成功后，会弹出 **Debug panel**，并且程序默认断点在 main()函数入口处，如*图 38* 所示。

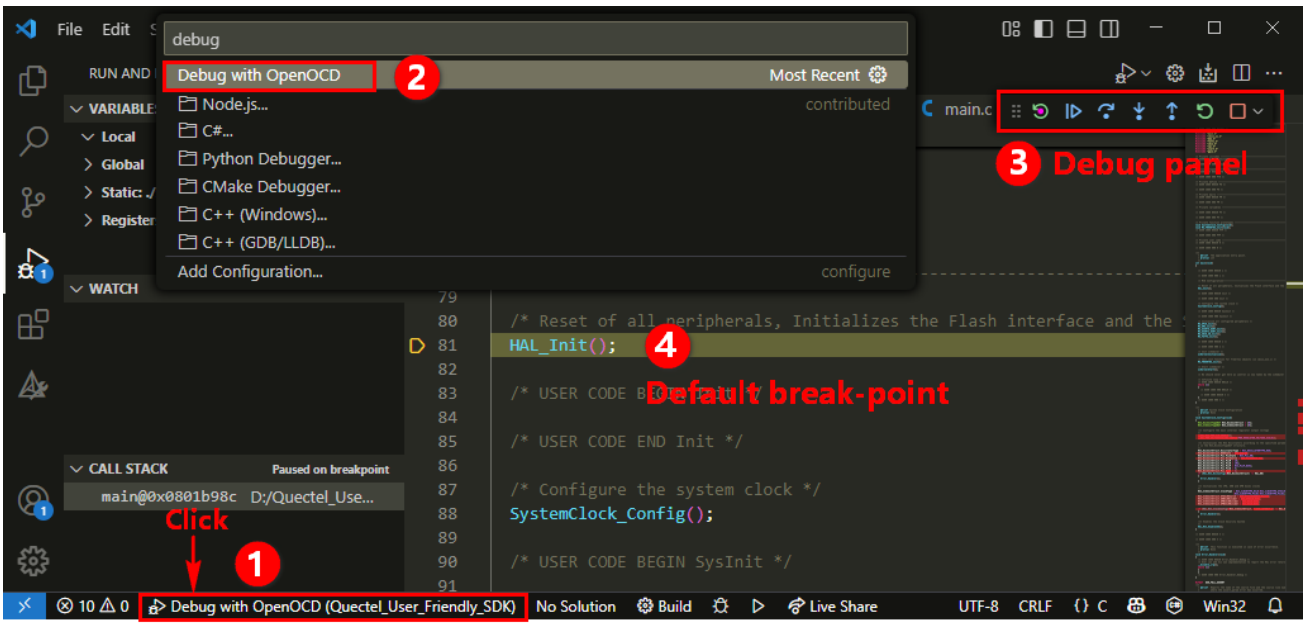

图 38：VSCode 中执行调试

点击执行过【**Configure All Projects with CMake Debugger**】后，除上述方法外，还可使用快捷键【**F5**】启动调试进程。

后续即可进行设置断点、单步调试、查看变量、查看调用栈、复位等调试操作，如*图 39* 所示。

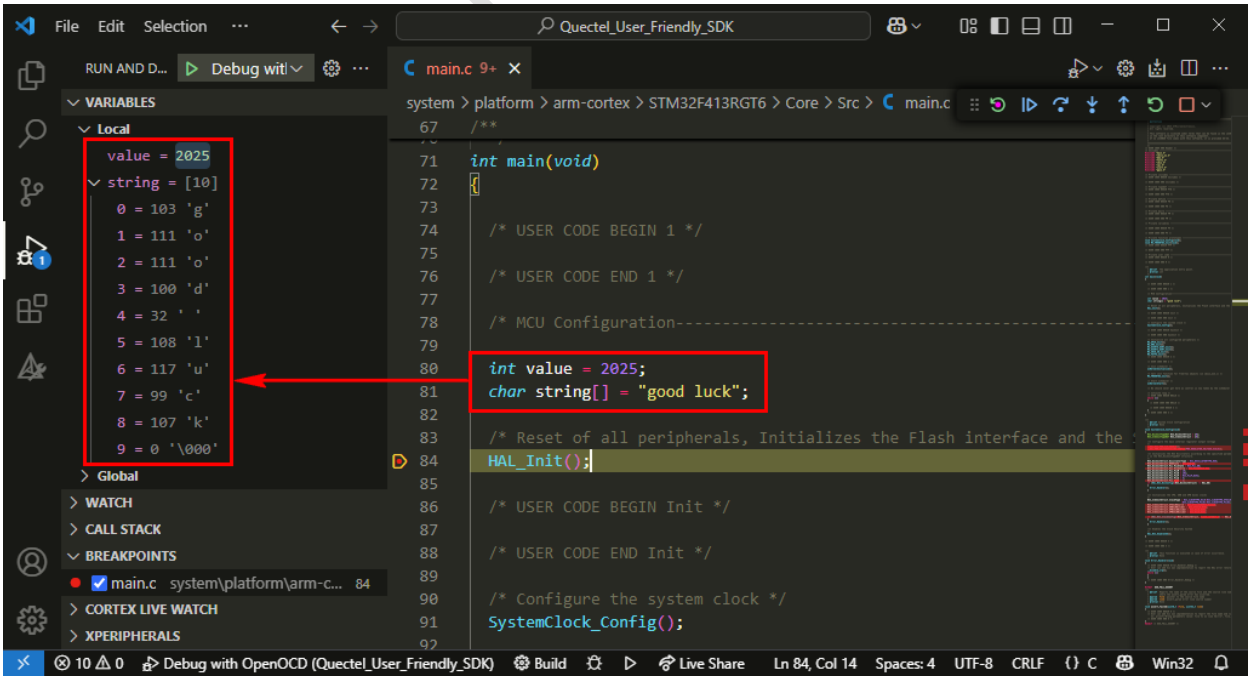

图 39：VSCode 中调试查看变量

8 附录 参考文档及术语缩写

表 3：参考文档

文档名称
[1] STM32 LQFP64 V2.0 载板使用说明 V1.0-0605
[2] Quectel_QSTM32_Test_Guide_V2.0
[3] Quectel_LTE_Standard(U)系列_AT 命令手册_V1.1

表 4：术语缩写

缩写	英文全称	中文全称
API	Application Programming Interface	应用程序接口
AT	Attention Command	AT 命令
EVK	Evaluation Kit	评估套件
GCC	GNU Compiler Collection	GNU 编译器套件
GUI	Graphical User Interface	图形用户界面
HAL	Hardware Abstraction Layer	硬件抽象层
IDE	Integrated Development Environment	集成开发环境
IoT	Internet of Things	物联网
JSON	JavaScript Object Notation	JavaScript 对象表示法
LED	Light Emitting Diode	发光二极管
MCU	Micro Controller Unit	微控制器单元
RTOS	Real-Time Operating System	实时操作系统

SDK	Software Development Kit	软件开发工具包
UART	Universal Asynchronous Receiver/Transmitter	通用异步收发器
USB	Universal Serial Bus	通用串行总线
(U)SIM	(Universal) Subscriber Identity Module	（通用）用户身份识别模块