Nuclei Development Tool Guide

Release 2025.02

Nuclei

Jun 25, 2025

CONTENTS

CHAPTER

ONE

INTRODUCTION

This user guide mainly talked about how to use Nuclei Development Tools, including Nuclei Studio IDE, Nuclei RISC-V Toolchain, Nuclei OpenOCD, Nuclei QEMU and Nuclei Model.

Nuclei Studio IDE is built on Eclipse Embedded CDT plugins, mainly optimized for Nuclei RISC-V Processor to improve user experience in IDE.

Nuclei RISC-V Toolchain is built on RISC-V GNU and LLVM toolchain(gcc/llvm/binutils/gdb/newlib) and also include Nuclei C Runtime Library, it provide good support for Nuclei RISC-V Processor.

Nuclei OpenOCD is built on RISC-V OpenOCD, adding nuspi flash support, cjtag support, customized csr support, nuclei openocd flashloader support.

Nuclei QEMU is built on QEMU project, adding Nuclei N/NX/UX RISC-V processor support, which works with Nuclei SDK and Nuclei Linux SDK.

Nuclei Model uses spike as the RISC-V ISA simulator and adds support for Nuclei ' s N/NX/UX RISC-V processors, it supports near cycle-level simulation and SystemC TLM 2.0 Nuclei EvalSoC modeling.

Note

To get a pdf version of this documentation, please click Nuclei Development Tool User Guide

You can also find our **Nuclei Studio Supply Documents** in https://doc.nucleisys.com/nuclei_studio_supply/, which is used for application notes using Nuclei Studio and Nuclei Tools.

If you have issues in this user guide, please send us an pull request in https://github.com/Nuclei-Software/nuclei-tool-guide repo to help us improve it.

If you have issues in our Nuclei Tools, please send us an issue in this repo or related tool repo to help us improve it.

- Nuclei Studio: https://github.com/Nuclei-Software/nuclei-studio
- Nuclei RISC-V Toolchain: https://github.com/riscv-mcu/riscv-gnu-toolchain
- Nuclei OpenOCD: https://github.com/riscv-mcu/riscv-openocd
- Nuclei Qemu: https://github.com/riscv-mcu/qemu
- Nuclei DLink: https://github.com/nuclei-Software/nuclei-dlink

CHAPTER

TWO

NUCLEI STUDIO IDE

2.1 Nuclei Studio IDE 简介

Note

- Nuclei Studio 出视频教程啦，相关内容在 芯来科技视频号中持续更新中，您可以在微信中搜索 芯来科技视频号并关注，以便获取到我们最新的更新内容。
- 在使用 Nuclei Studio, Nuclei Tools 过程中，如查有问题，可以查阅 https://doc.nucleisys.com/nuclei_studio_supply/内容，也可以向我们提交相关 Issue。

一款高效易用的集成开发环境（Integrated Development Environment，IDE）对于任何 MCU 都显得非常重要，软件开发人员需要借助 IDE 进行实际的项目开发与调试。ARM 的商业 IDE 软件 Keil，在中国大陆很多嵌入式软件工程师均对其非常熟悉。但是商业 IDE 软件（譬如 Keil）存在着授权以及收费的问题，各大 MCU 厂商也会推出自己的免费 IDE 供用户使用，譬如瑞萨的 e2studio 和 NXP 的 LPCXpresso 等，这些 IDE 均是基于开源的 Eclipse 框架，Eclipse 几乎成了开源免费 MCU IDE 的主流选择。

Nuclei Studio IDE 正是芯来公司，基于 Eclipse IDE 开发的一款针对芯来 RISC-V 处理器 IP 核产品的集成开发环境工具。

Eclipse 平台采用开放式源代码模式运作，并提供公共许可证（提供免费源代码）以及全球发布权利。Eclipse 本身只是一个框架平台，除了 Eclipse 平台的运行时内核之外，其所有功能均位于不同的插件中。开发人员既可通过 Eclipse 项目的不同插件来扩展平台功能，也可利用其他开发人员提供的插件。一个插件可以插入另一个插件，从而实现最大程度的集成。

由于 Eclipse IDE 已经在社区被大量使用，一些常见的使用方法在 Eclipse IDE 里面，如果没有和硬件或者 CPU 绑定，一般情况下是可以借鉴其他人写的关于 Eclipse IDE 的使用教程，这里推荐几个常用的教程网站：

- https://help.eclipse.org/latest/index.jsp
- https://eclipse-embed-cdt.github.io/
- https://mcuoneclipse.com/

Eclipse IDE 平台具备以下几方面的优势。

- 社区规模大

  Eclipse 自 2001 年推出以来，已形成大规模社区，这为设计人员提供了许多资源，包括图书、教程和网站等，以帮助他们利用 Eclipse 平台与工具提高工作效率。Eclipse 平台和相关项目、插件等都能直接从 eclipse.org 网站下载获得。

- 持续改进

  Eclipse 的开放式源代码平台帮助开发人员持续充分发挥大规模资源的优势。Eclipse 在以下多个项目上不断改进。

  – 平台项目——侧重于 Eclipse 本身。
  – CDT 项目——侧重于 C/C++ 开发工具。

  - PDE 项目——侧重于插件开发环境。

- 源码开源

  设计人员始终能获得源代码，总能修正工具的错误，它能帮助设计人员节省时间，自主控制开发工作。

- 兼容性

  Eclipse 平台采用 Java 语言编写，可在 Windows 与 Linux 等多种开发工作站上使用。开放式源代码工具支持多种语言、多种平台以及多种厂商环境。

- 可扩展性

  Eclipse 采用开放式、可扩展架构，它能够与 ClearCase、SlickEdit、Rational Rose 以及其他统一建模语言（UML）套件等第三方扩展协同工作。此外，它还能与各种图形用户接口（GUI）编辑器协同工作，并支持各种插件。

Nuclei Studio IDE 充分利用上述 Eclipse IDE 优势，结合社区成熟的 Eclipse embedded CDT, Linux Tools 等插件，并研发自有插件满足 RISC-V 嵌入式开发的日常需求:

- 工程创建，管理，编译和调试
- 支持多种调试方案，例如 OpenOCD，JLink，Nuclei DLink，Nuclei Qemu 等
- 支持扩展调试方案，方便扩展支持更多调试器
- 支持多种编译器，包括 gcc, clang, zcc
- 提供快捷的工程常用设置工具 **Nuclei Settings**
- 支持基于 Nuclei ETrace 软硬件方案的 Onchip Trace
- 支持基于 gprof、gcov 的大幅增强 profiling 和 code coverage 方案
- 支持 Nuclei PacKage(NPK) 软件包方案，可以便捷的扩展支持软件开发包和 Nuclei Studio 解耦，实现 软件包导入 **-> Project Wizard** 的便捷方案, 这种方案已经得到广泛的应用，例如芯来科技自有的 **Nuclei SDK**.
- 更快捷的工程和工作空间的打开方式
- 更好用的 Launchbar 功能

2.2 Nuclei Studio 更新说明

2.2.1 2025.02 版更新说明

2025.02 版本是基于 eclipse Cpp 2024-06 开发，CDT 版本到 Eclipse CDT 2024-06，升级了芯来科技的工具版本至 2025.02，优化了部分原有功能，新增了调试及代码性能分析等功能，以及解决了 2024.06 版中存在的缺陷。

升级 Eclipse Cpp 版本

在 Nuclei Studio 2025.02 基于 Eclipse Cpp 2024-06 版本开发此版本。基础的 CDT 版本，升级到了 11.6.1。

升级 RISC-V Toolchain、OpenOCD、QEMU 版本

在 Nuclei Studio 2025.02 版本中集成了 Nuclei RISC-V Toolchain 2025.02 版，具体信息可以查看：https://github.com/riscv-mcu/riscv-gnu-toolchain/releases/tag/nuclei-2025.02 。

在 Nuclei Studio 2025.02 版本中集成了 OpenOCD 2025.02 版，具体信息可以查看：https://github.com/riscv-mcu/riscv-openocd/releases/tag/nuclei-2025.02 。

在 Nuclei Studio 2025.02 版本中集成了 Nuclei Qemu 2025.02 版，具体信息可以查看：https://github.com/riscv-mcu/qemu/releases/tag/nuclei-2025.02 。

新增对更多新核的支持

增加了对 N200E、N202、N202E、NX1000、NX1000F、NX1000FD、UX1000、UX1000F CPU 的核配套支持。

新增 Flash Programming 功能

为了满足用户将编译好的二进制文件直接下载到硬件开发板的需求，Nuclei Studio 新增了 Flash Programming 功能。该功能允许用户快速、便捷地将编译好的二进制文件直接下载到硬件开发板中，极大提升了开发和调试的效率。

具体参见*Flash Programming 功能* (page 137) 。

新增了 Nuclei NICE Wizard

Nuclei NICE Wizard 是一个集成在 Nuclei Studio 上的工具，旨在简化和加速 NICE (自定义指令扩展) 和 VNICE (向量化自定义指令扩展) 指令的创建过程。

具体参见*Nuclei NICE Wizard* (page 241) 。

新增 Nuclei Model 功能的使用

Nuclei Model 是芯来科技为 Nuclei Near Cycle Model 开发了专门的运行工具，为了提供更简洁高效的用户体验，在 RVProf 的基础上进行了功能简化，推出了新的 Model 工具。

具体参见*Nuclei Model* (page 248) 。

升级 Nuclei Near Cycle Model 版本

Nuclei Near Cycle Model，是由芯来科技自主研发的仿真测试和性能分析工具，可以帮助研发人员在项目初期进行一些必要的仿真测试和程序性能分析。在此版本中全面支持 Nuclei CPU 200，300，600，900，1000 系列的 CPU，同时支持 Windows 和 Linux 系统下使用。

具体参见*Nuclei Near Cycle Model* (page 248) 。

新增 Live Watch 功能

Live Watch 是芯来科技研发的实时监控工具，专为开发者设计，旨在帮助开发者更高效地调试和优化代码。

具体参见*Live Watch 功能的使用* (page 254)。

ZCC 升级

在 Nuclei Studio 2025.02 版本中集成了 ZCC 3.2.5 版，并加入芯来科技支持的软件库。具体信息可以查看：https://www.terapines.com/products/zcc

2.2.2 2024.06 版更新说明

本版本是一次比较重大的版本升级，2024.06 版本升级了 CDT 版本到 Eclipse CDT 2024-06，升级了芯来科技的工具版本至 2024.06，优化了部分原有功能，新增了调试及代码性能分析等功能，以及解决了 2024.02 版中存在的缺陷。

升级 Eclipse CDT 版本

在 Nuclei Studio 2024.06 版本中基础的 CDT 版本，升级到了 11.6.0，并基于 Eclipse CDT 2024-06 版本开发此版本。

升级 RISC-V Toolchain、OpenOCD、QEMU 版本

在 Nuclei Studio 2024.06 版本中集成了 Nuclei RISC-V Toolchain 2024.06 版，具体信息可以查看：https://github.com/riscv-mcu/riscv-gnu-toolchain/releases/tag/nuclei-2024.06 。

在 Nuclei Studio 2024.06 版本中集成了 OpenOCD 2024.06 版，具体信息可以查看：https://github.com/riscv-mcu/riscv-openocd/releases/tag/nuclei-2024.06 。

在 Nuclei Studio 2024.06 版本中集成了 Nuclei Qemu 2024.06 版，具体信息可以查看：https://github.com/riscv-mcu/qemu/releases/tag/nuclei-2024.06 。

新增对 U600 和 UX1000 的支持

配合 U600 和 UX1000 核的发布，同步增加了对 U600 和 UX1000 核配套支持。

优化 NPK 软件包管理

优化 Nuclei Package Management 中对 NPK 包依赖的管理，使其更易使用；优化了部分 NPK 包安装的提示信息及日志，提高 NPK 包管理的使用体验。具体参见*NPK 软件包管理* (page 76)。

> **Note**
>
> 注意：本次版本升级，变更了 NPK 包管理的配置，在 2024.02 版及之前版本中安装的 NPK 包在 2024.06 版 NucleiStudio 无法识别，用户需重新下载安装 NPK 包。

增加和优化部分编译选项

在 Properties 和 Nuclei Settings 页面内，在 Optimization Level 中新增 -Oz 选项；在 GNU RISC-V Cross C++ Linker 的 Libraries 页新增对 group libraries 的支持, 参见工程编译链接 *C* 库找不到符号报错 (page 281)。

优化调试模式切换

NucleiStudio 支持多种调试模式，如 OpenOCD、Jlink、Dlink、Custom 等，同时还有 Qemu 等仿真器，为了方便用户在多工程多种模式之间切换，优化了调试模式的切换，具体内容参见调试模式管理 (page 143)。

优化和完善 DLink Debug 调试

Nuclei DLink 是芯来科技基于 RV Link，并在 RV Link 的基础上做了许多功能增加后，所研发的 RISC-V 调试器，使之更适应于 Nuclei Studio 的应用场景。具体内容可以查看使用 *DLink* 调试运行项目 (page 182)。

集成 Terapines ZCC Lite 编译器

Terapines ZCC 是兆松科技研发的高性能 RISC-V 编译器。Nuclei Studio 2024.06 版中对 Terapines ZCC 进行支持，用户可以在 Nuclei Studio 中直接使用。具体参见 *Nuclei Studio* 中编译 *Hello World* 项目 (page 136)。

新增 LST View 工具

LST View 是一个 lst 文件查看器，可以方便用户查看 lst 格式的文件，并实现 *.lst 文件与源代码的联动，具体请参见 *LST View* (page 196)。

优化和完善 Gprof 功能

Gprof 是一个强大的性能分析工具，可以帮助开发者理解 C/C++ 程序的运行情况，通过 Gprof 可以获取到程序中各个函数的调用信息、调用次数、执行时间等，对优化程序、提升程序运行效率具有重要的意义。具体请参见 *Code Coverage* 和 *Profiling* 功能 (page 199)。

优化和完善 Gcov 功能

Gcov 是一个测试 C/C++ 代码覆盖率的工具，伴随 GCC 发布，配合 GCC 共同实现对 C/C++ 文件的语句覆盖、功能函数覆盖和分支覆盖测试。具体请参见 *Code Coverage* 和 *Profiling* 功能 (page 199)。

新增 Call Graph 功能

Call Graph 是分析函数调用关系图的工具，结合 Gprof 使用，便于开发者快速了解程序执行的过程及调用关系。具体请参见 *Code Coverage* 和 *Profiling* 功能 (page 199)。

新增 Nuclei Near Cycle Model 支持

Nuclei Near Cycle Model，它是由芯来科技自主研发的仿真测试和性能分析工具，可以帮助研发人员在项目初期进行一些必要的仿真测试和程序性能分析，具体请参见使用 *Nuclei Near Cycle Model* 仿真性能分析 (page 248) 。

2.2.3 2024.02.dev 版更新说明

本版本是开发版本（您下载到的链接内容随时可能会变更），本版本解决了 Nuclei Studio 2023.10 版中存在的缺陷，并优化了部分原有功能如 ETrace 特性，新增了一些功能如对 N100 的支持、Dlink 的支持等，更好为满足客户评估和更新使用。

升级 Eclipse CDT 版本

在 Nuclei Studio 2024.02.dev 版本中基础的 Eclipse CDT 版本，升级到了 Eclipse CDT 2023.12 版。

新增对 N100 的支持

配合 N100 核的发布，同步增加了对 N100 的配套支持。

新增批量转换 Gcc13 工程工具

在 2023.10 版 Nuclei Studio 中，升级 GCC 13 后，当有大量工程需要转换时，单个转换效率低，为方便开发者，提供了一个批量转换 GCC 13 工具。具体内容参见批量将工程转换成支持 *gcc 13* 的工程 (page 190) 。

优化和完善 Trace 功能

Nuclei Studio 中 Trace 功能升级，实现了在 OpenOCD 模式下对单核应用、SMP 多核应用、AMP 多核应用的支持，具体内容参见*Trace* 功能的使用 (page 220) ；在 Dlink 模式下，仅对单核应用支持。Trace 功能需要有对应 CPU IP 的支持，如需体验此功能，请与我们联系。

优化和完善 RVProf 功能

RVProf 是芯来科技基于 CPU cycle model 开发的性能分析工具，具体内容参见第*RVProf* 功能的使用 (page 233) 。此功能需要有相应的 NPK 软件包支持，如需体验此功能，请与我们联系。

新增对 DLlink Debug 的支持

Dlink 是芯来自主研发的调试解决方案，在本次版本中得到支持。此功能需要有相应的 Dlink 调试器的支持，如需体验此功能，请与我们联系。

2.2.4 2023.10 版更新说明

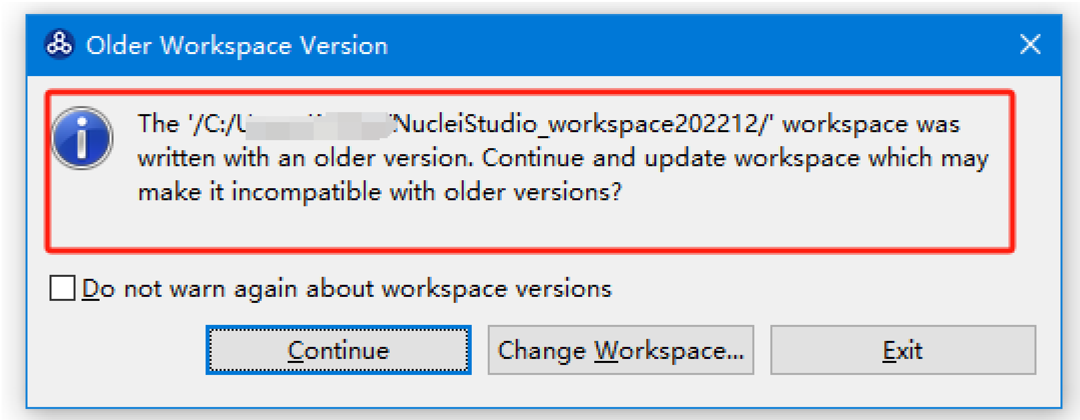
本版本是一次比较重大的版本升级，集成了 Nuclei 2023.10 版本的 Toolchain, QEMU, OpenOCD, 且 Eclipse CDT 版本进行了升级，GCC 版本也做了重大迭代，升级到了 GCC 13, NPK 部分也做了大量的新功能的增加以支持 GCC 或者 CLANG 的工程创建，并且增加很多新的 Configuration 字段类型，方便在 Project Wizard 中更灵活的进行工程配置。

升级 Eclipse CDT 版本

在 Nuclei Studio 2023.10 版本中基础的 Eclipse CDT 版本，升级到了 Eclipse CDT 2023.06 版; Eclipse CDT 2023-06 版本是 Eclipse 基金会 2023 年第二个季度同步版本，有 64 个参与项目，于 2023 年 6 月 14 日发布。

参考地址：Eclipse IDE for C/C++ Developers[1]

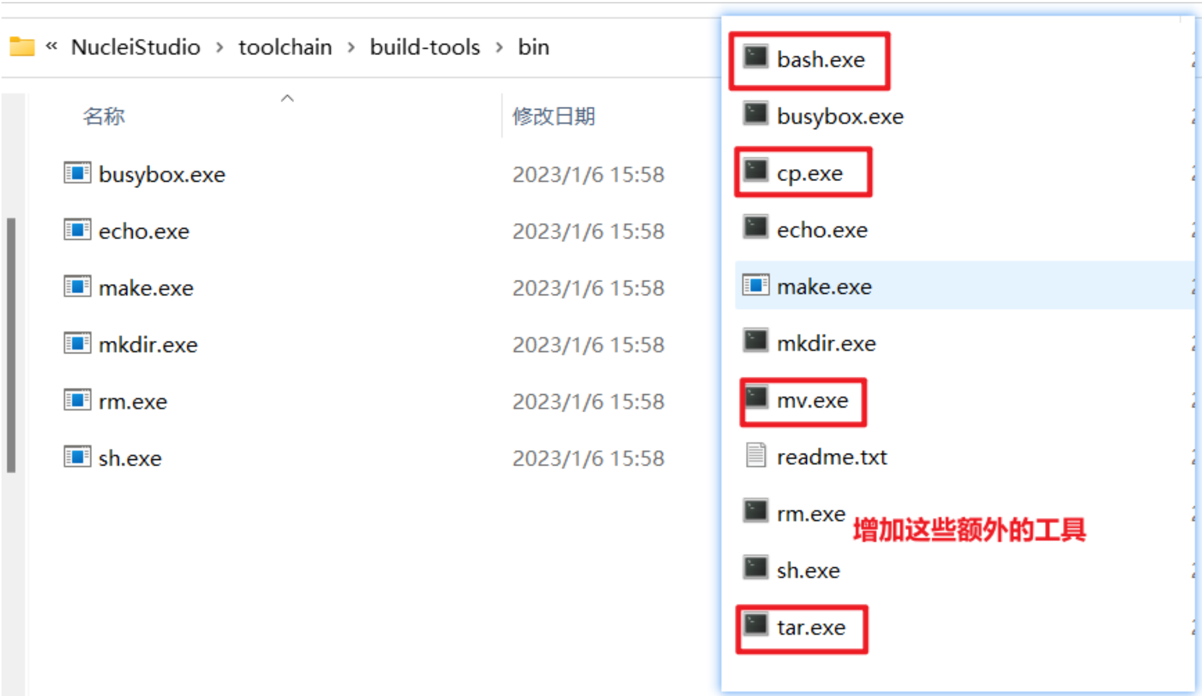
升级后打开之前版本创建的 workspace, 会弹出不兼容的警告，使用时可能会有异常，建议更换新的 workspace 目录。



升级 build-tools 版本

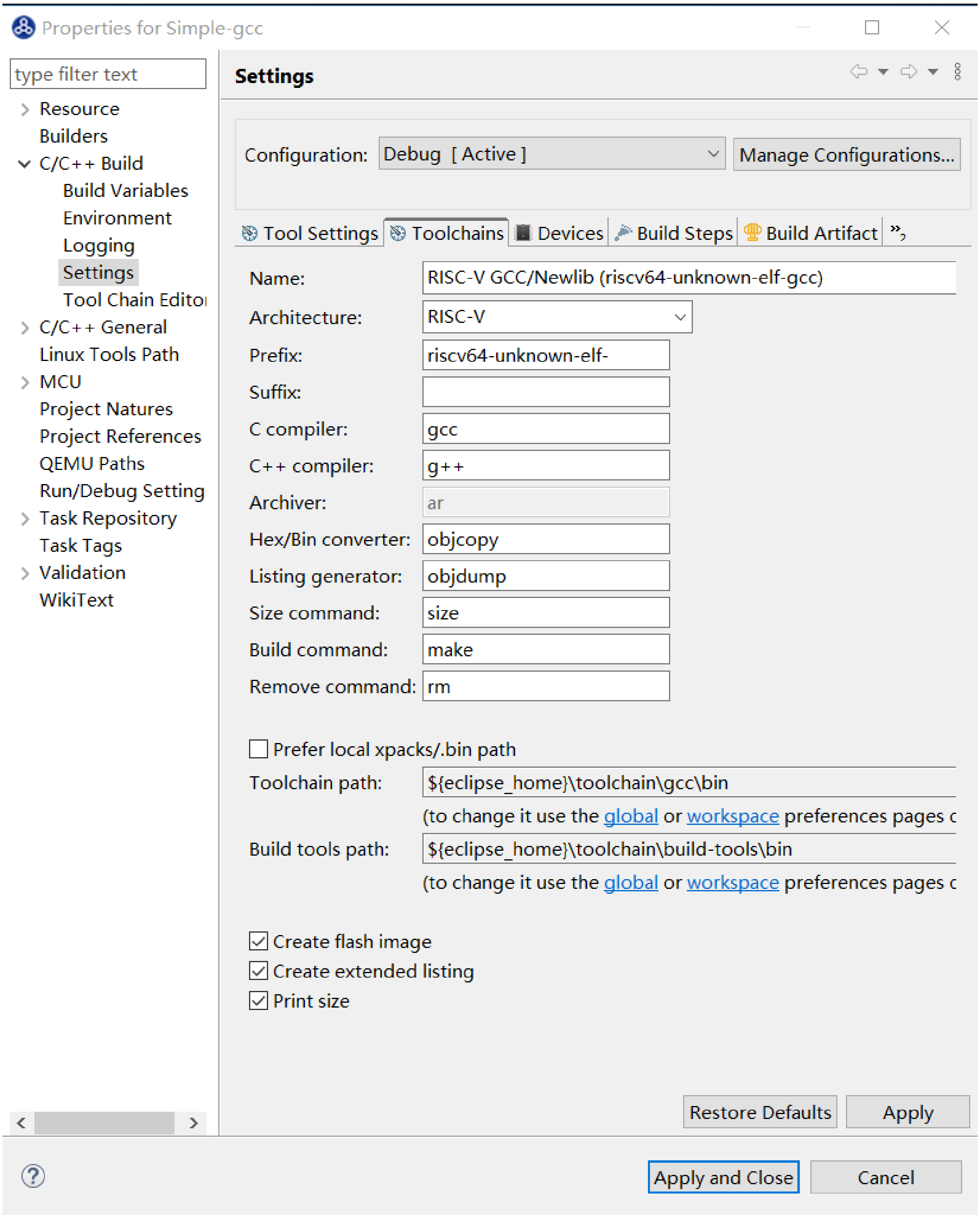
在 Nuclei Studio 2023.10 版将 toolchain 中的 build-tools 更新到 4.4 版本，并额外增加了 bash.exe、cp.exe、mv.exe、tar.exe 工具。

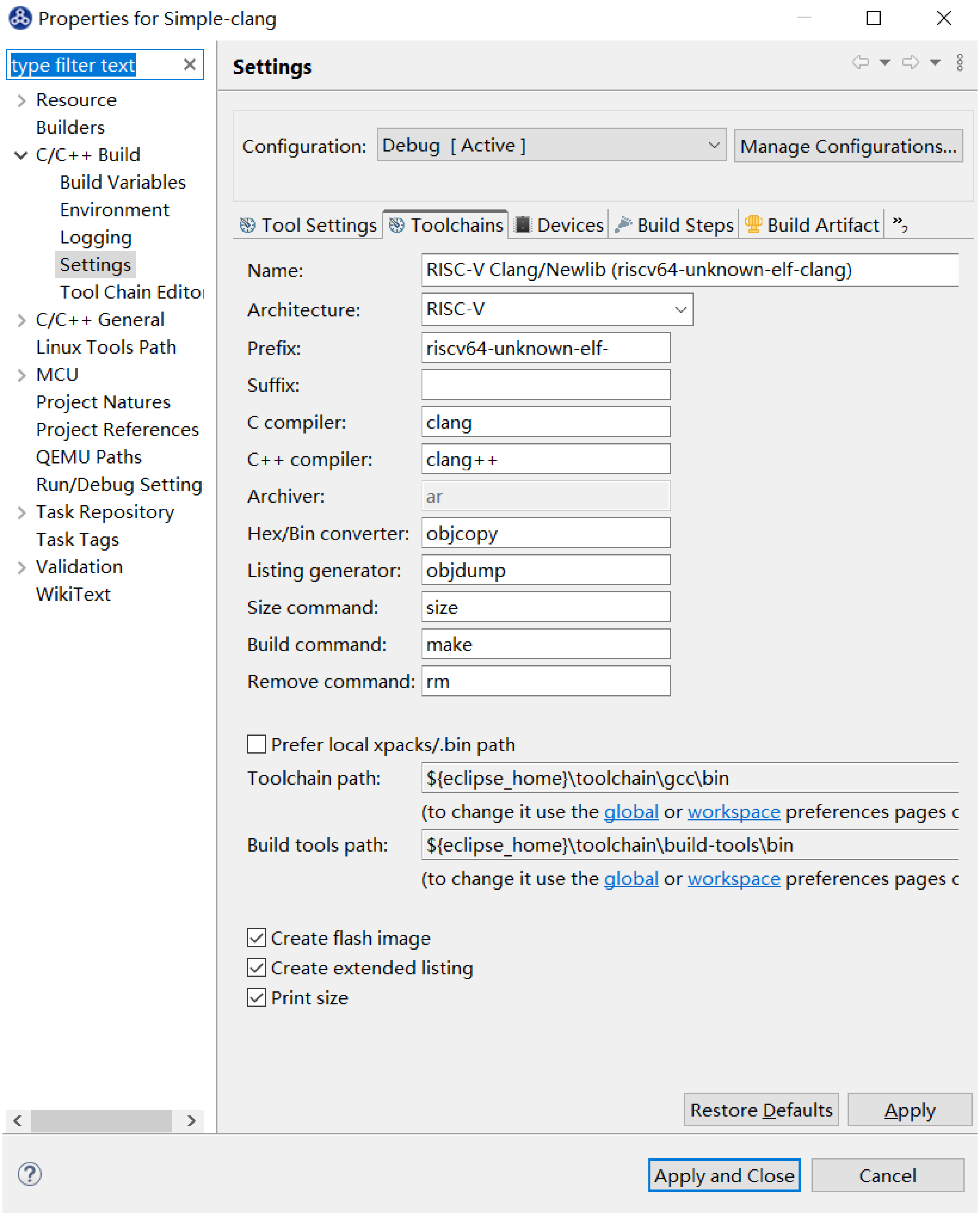
[1] https://www.eclipse.org/downloads/packages/release/2023-06/r/eclipse-ide-cc-developers



支持 GCC 13 和 Clang 17

在 Nuclei Studio 2023.10 版本中实现了对 GCC 13 的支持，相对于之前的 gcc10 版本 GCC 13 在对 RISC-V 指令扩展的支持更加完备，且在我们维护的版本中，支持完整的 RVV Intrinsic API v0.12 版本。同时 Nuclei Studio 2023.10 版本中也实现了对 Clang 17 的支持（参考地址：https://releases.llvm.org/17.0.1/docs/RISCVUsage.html）。当然，如果有用户依然想使用 GCC 10 时行项目开发，我们也保留了相关的配置，但是工具链并没有集成到 IDE 中，用户需要自行下载并放置在 gcc10 目录中，参见里面的 README.txt，并且我们也提供了老版本采用 gcc10 的 Nuclei Studio 创建的工程升级到 gcc13 工具链上，具体使用可以参考导入旧版本 *Nuclei Studio* 创建的工程 (page 188)。Nuclei RISC-V Toolchain 2023.10 更详细的说明，请参阅: https://github.com/riscv-mcu/riscv-gnu-toolchain/releases/tag/nuclei-2023.10


Properties for Simple-gcc
type filter text
Resource
Builders
C/C++ Build
Build Variables
Environment
Logging
Settings
Tool Chain Editor
C/C++ General
Linux Tools Path
MCU
Project Natures
Project References
QEMU Paths
Run/Debug Setting
Task Repository
Task Tags
Validation
WikiText
Settings
Configuration: Debug [Active]
Manage Configurations...
Tool Settings
Toolchains
Devices
Build Steps
Build Artifact
Name: RISC-V GCC/Newlib (riscv64-unknown-elf-gcc)
Architecture: RISC-V
Prefix: riscv64-unknown-elf-
Suffix:
C compiler: gcc
C++ compiler: g++
Archiver: ar
Hex/Bin converter: objcopy
Listing generator: objdump
Size command: size
Build command: make
Remove command: rm
Prefer local xpacks/.bin path
Toolchain path: ${eclipse_home}\toolchain\gcc\bin
(to change it use the global or workspace preferences pages
Build tools path: ${eclipse_home}\toolchain\build-tools\bin
(to change it use the global or workspace preferences pages
Create flash image
Create extended listing
Print size
Restore Defaults
Apply
Apply and Close
Cancel

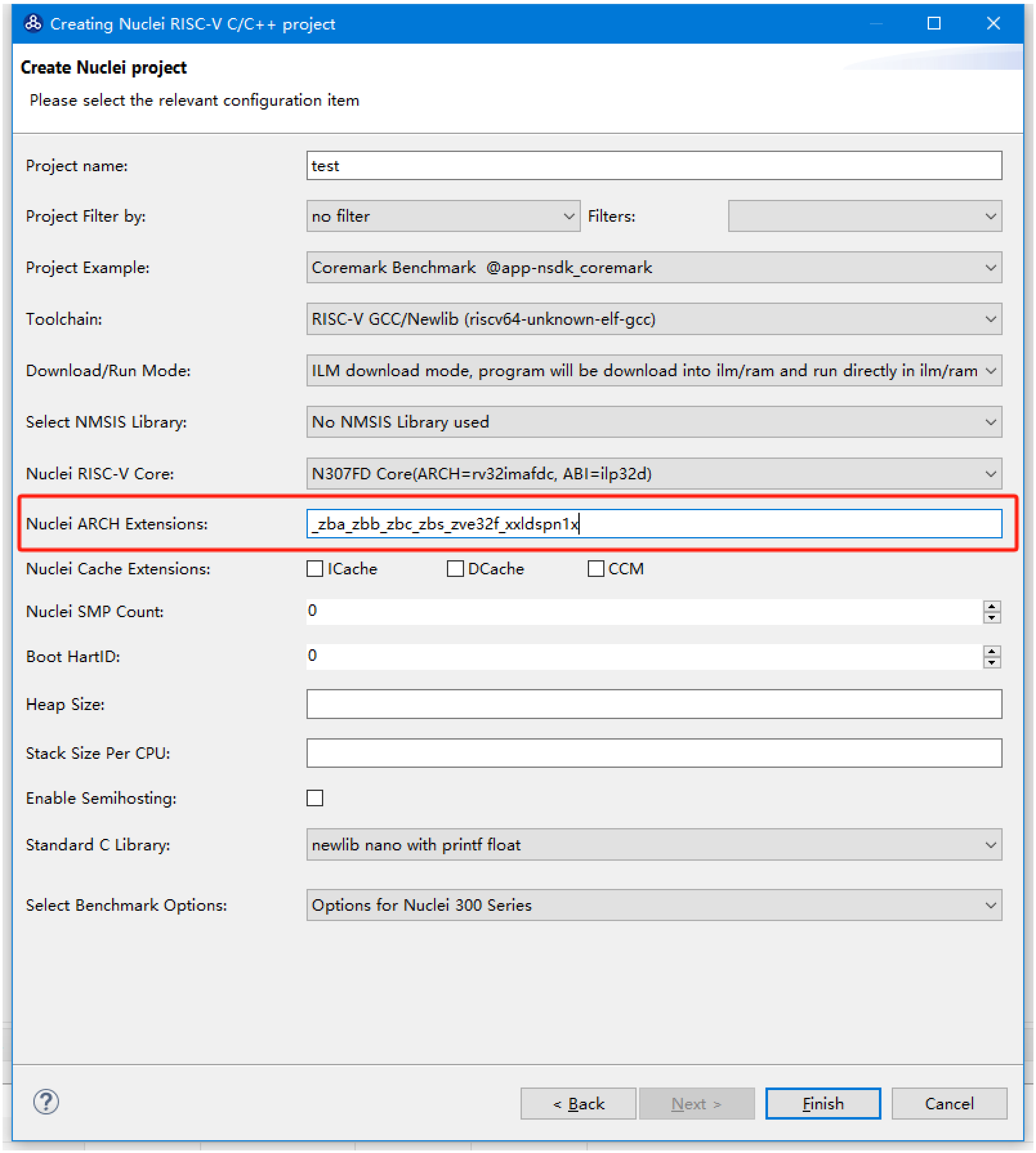

Properties for Simple-clang
type filter text
Resource
Builders
C/C++ Build
Build Variables
Environment
Logging
Settings
Tool Chain Editor
C/C++ General
Linux Tools Path
MCU
Project Natures
Project References
QEMU Paths
Run/Debug Setting
Task Repository
Task Tags
Validation
WikiText
Settings
Configuration: Debug [Active]
Manage Configurations...
Tool Settings
Toolchains
Devices
Build Steps
Build Artifact
Name: RISC-V Clang/Newlib (riscv64-unknown-elf-clang)
Architecture: RISC-V
Prefix: riscv64-unknown-elf-
Suffix:
C compiler: clang
C++ compiler: clang++
Archiver: ar
Hex/Bin converter: objcopy
Listing generator: objdump
Size command: size
Build command: make
Remove command: rm
Prefer local xpacks/.bin path
Toolchain path: ${eclipse_home}\toolchain\gcc\bin
(to change it use the global or workspace preferences pages c
Build tools path: ${eclipse_home}\toolchain\build-tools\bin
(to change it use the global or workspace preferences pages c
Create flash image
Create extended listing
Print size
Restore Defaults
Apply
Apply and Close
Cancel

RISC-V 指令扩展使用变更

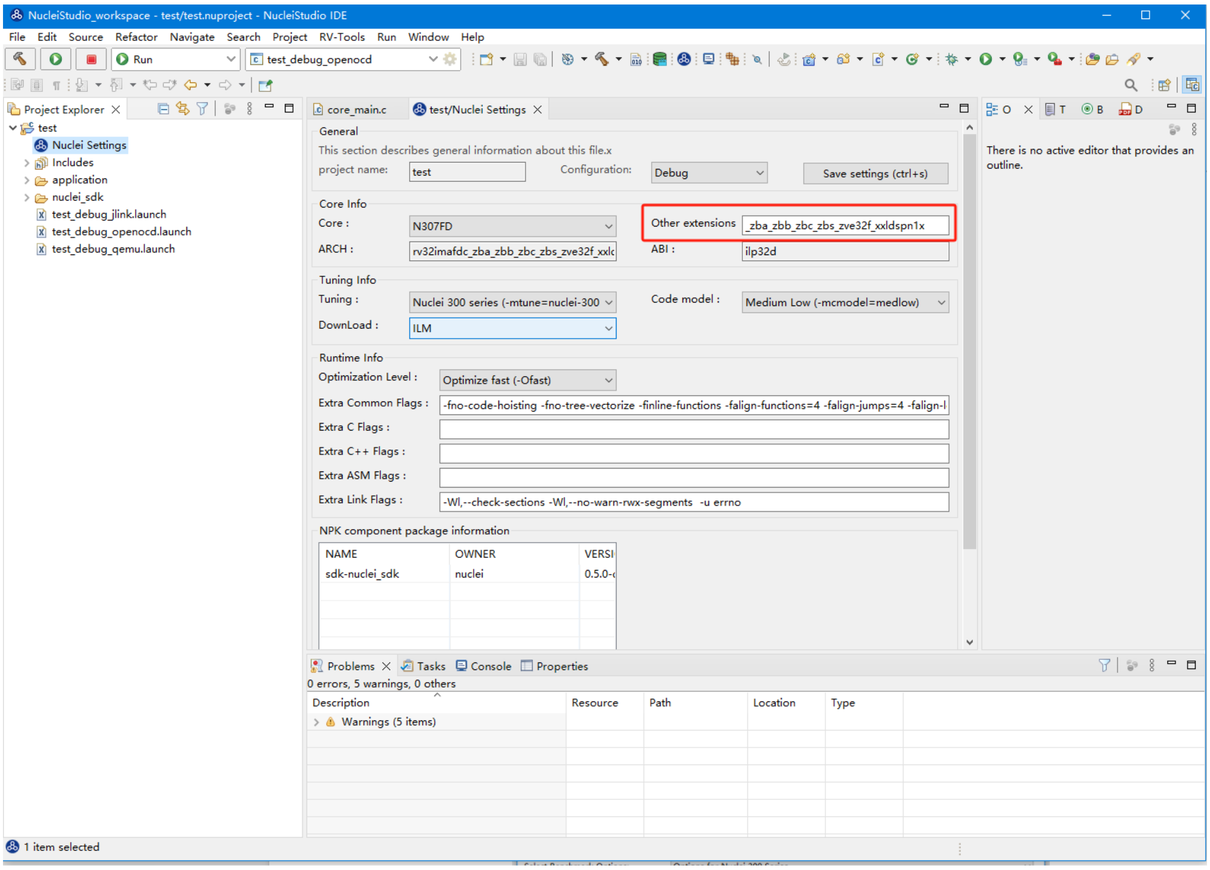
因 gcc 和 clang 的变更，在扩展的使用上，有了较大的变化。原来的 bpkv 扩展与新的规则对应关系如下，更详细的说明，请参阅https://doc.nucleisys.com/nuclei_sdk/develop/buildsystem.html#arch-ext

- `b` -> `_zba_zbb_zbc_zbs`
- `p` -> rv64: `_xxldsp`, rv32: `_xxldspn3x` for n300, `_xxldspn1x` for n900
- `k` -> `_zk_zks`
- `v` -> rv32f/d : `_zve32f`, rv64f: `_zve64f`, rv64fd: `v`

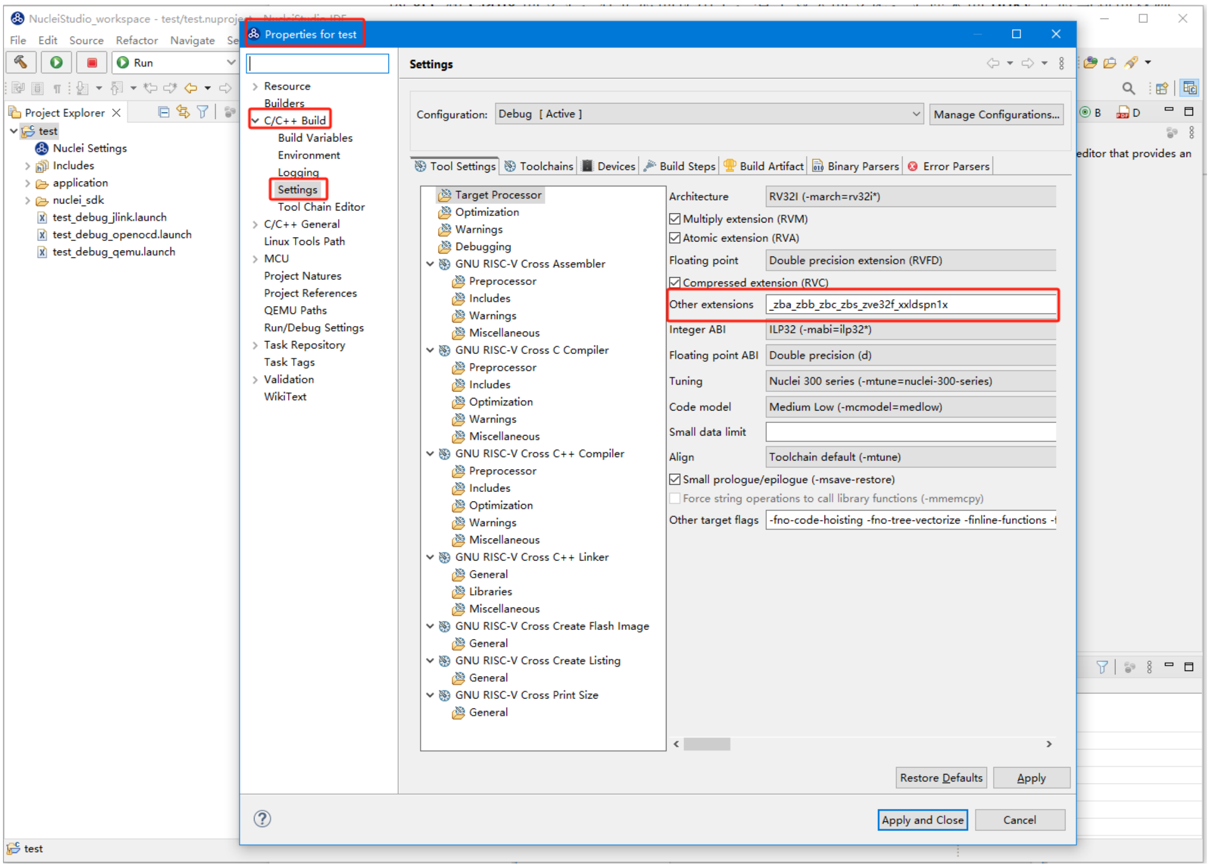
以 N307FD + B + V + Nuclei DSP with N1 extension 为例，创建一个使用扩展的应用，在创建工程的引导中，需要 Nuclei ARCH Extensions 中填入对的扩展字段，如需要使用 bpv 扩展，根据以上规则，需要填入 `_zba_zbb_zbc_zbs_zve32f_xxldspn1x` 。



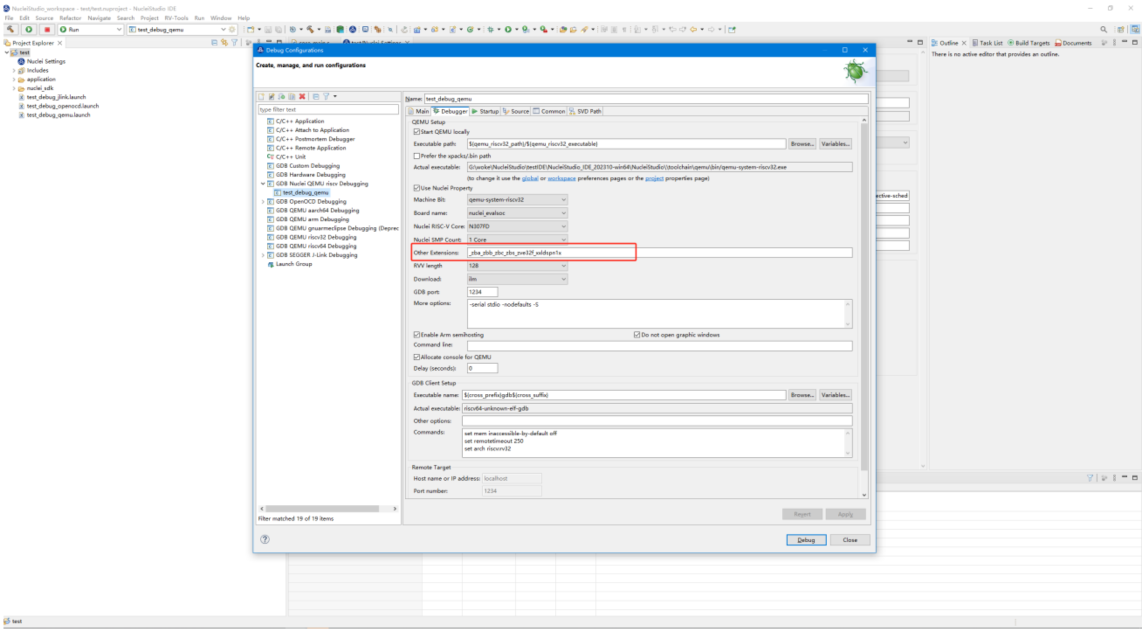
生成的工程中，可以看到在工程的 **Nuclei Settings** 。



同样的查看工程的属性，在 C/C++ Build->Settings->Target Processor 中也是有关于 RISC-V 指令扩展的配置项。

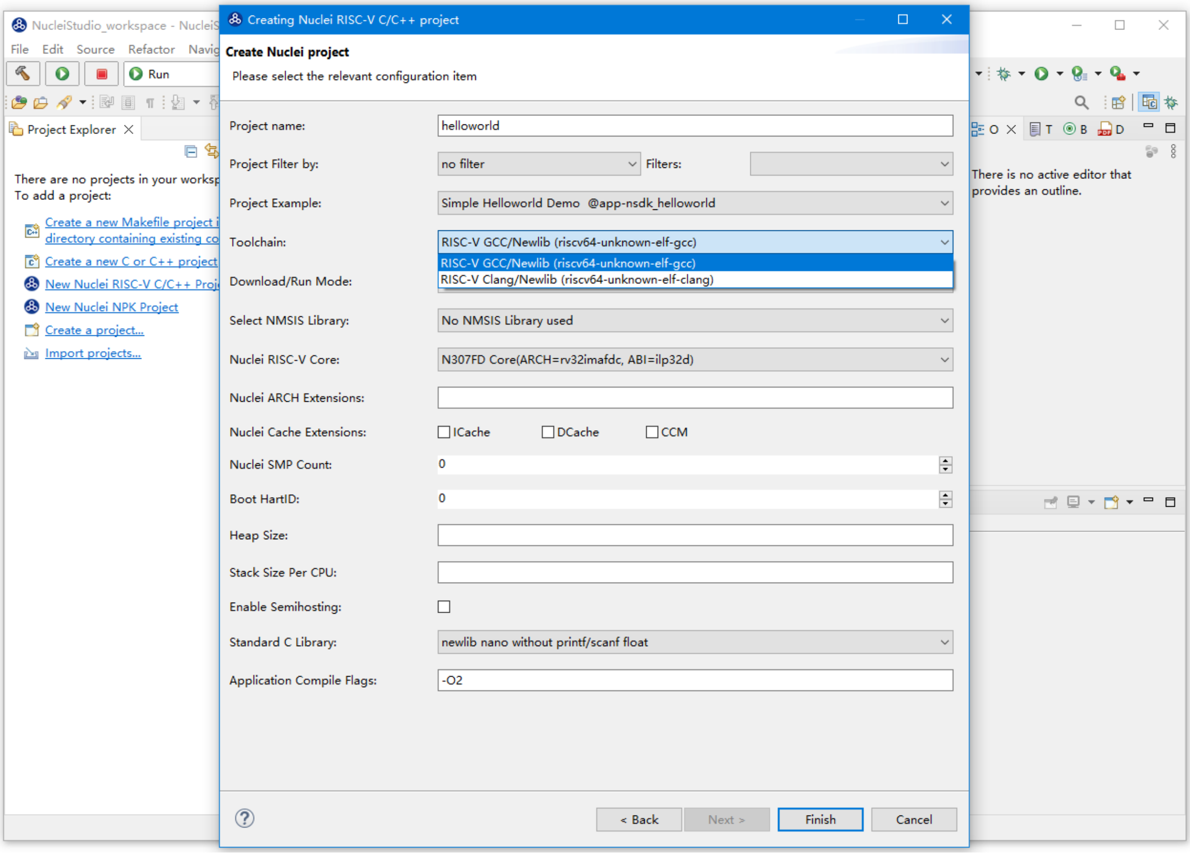


同时在 QEMU 的配置中也会有相对应的 RISC-V 指令扩展的配置项。



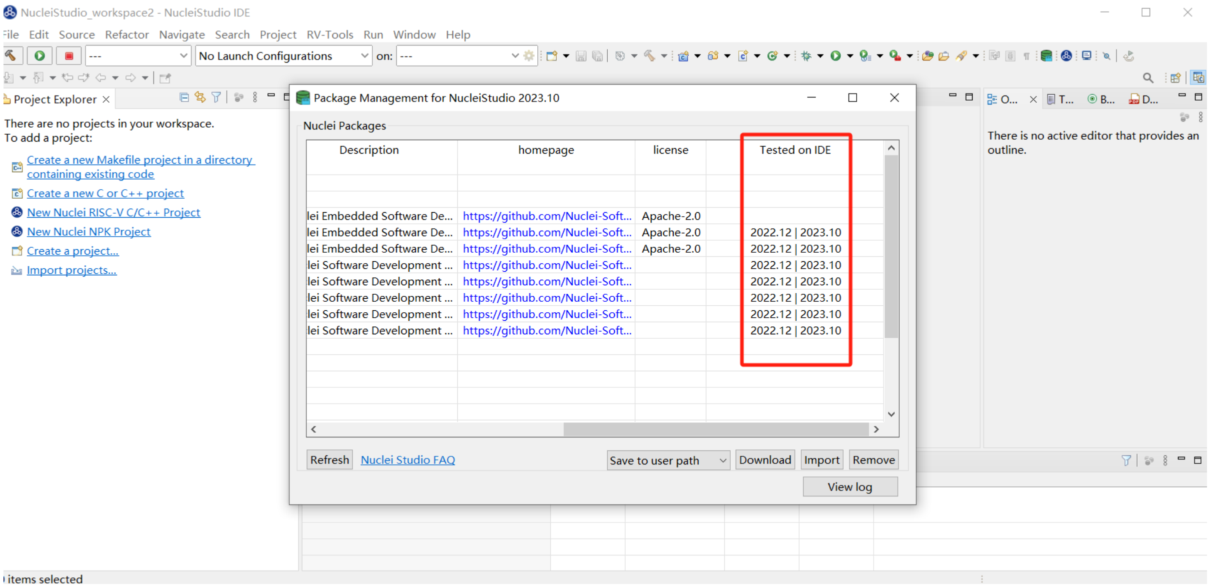
NPK 包的使用变更

为了支持 GCC 13 和 Clang 17，Nuclei SDK 包升级到了 0.5.0 版本，使用 SDK 包创建工程时，用户可以根据需要，选择创建一个 GCC 13 或者 Clang 17 的工程。因为版本变动较大，0.5.0 之前的 sdk 可能有部分功能在 Nuclei Studio 2023.10 版中使用异常，所我们提供了工具帮助您快速进行工程迁移和升级，请自行备份老版本的工程，具体可能参考导入旧版本 *Nuclei Studio* 创建的工程 (page 188)。



另外 Nuclei Studio 2023.10 中会对 npk 在线组件包做适配版本的校验（上传阶段需要填写测在什么版

本的 Nuclei Studio 上测试使用），不同的组件包所适配的 Nuclei Studio 版本号会在 Package Management 页面展示，在下载安装的时候如果版本不匹配，会给与提示，但是导入离线包不会有任何提示，请自行甄别是否被所使用的 Nuclei Studio IDE 版本所支持，具体如下。

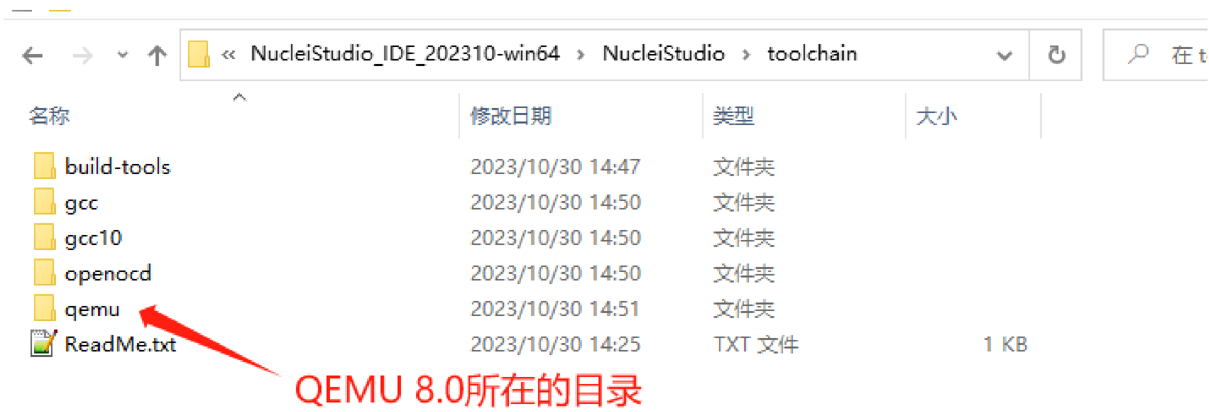


升级 OpenOCD

OpenOCD 版本升级至 2023.10 版，增加了一些额外的调试特性，例如查看 cpu 信息，etrace 实验性的支持。关于 OpenOCD 变更更详细的说明，请参阅：https://github.com/riscv-mcu/riscv-openocd/releases/tag/nuclei-2023.10

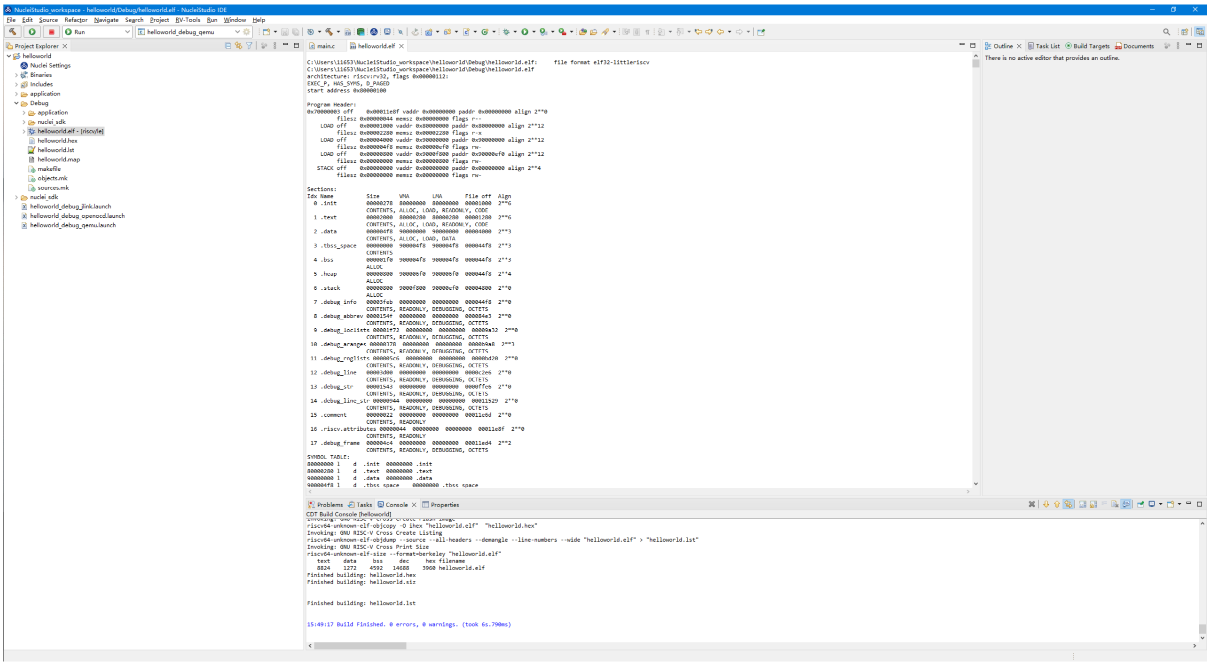
升级 QEMU

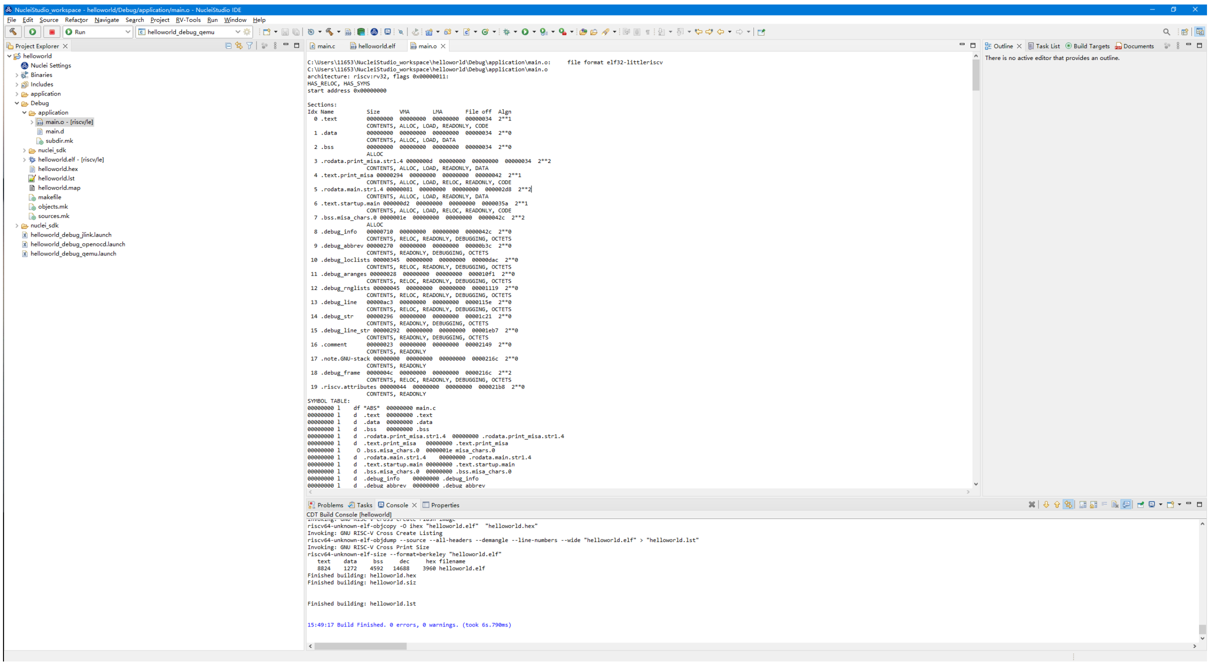
在 Nuclei Studio 2023.10 中集成 Nuclei QEMU 2023.10 版本，而 Nuclei QEMU 2023.10 基于 QEMU 8.0 进行二次开发（参考地址：https://wiki.qemu.org/ChangeLog/8.0）。本版本的 QEMU 和 2022.10 版本使用方面有比较大的变化，不再支持 gd32vf103_rvstar 这块开发板，转而只支持 Nuclei EvalSoC, 可以配置 Nuclei SDK/Nuclei Linux SDK 无缝使用。且支持的 machine 由 nuclei_n/nuclei_u 转而统一变为 nuclei_evalsoc。关于详细 Nuclei QEMU 更详细的说明，请参阅：https://github.com/riscv-mcu/qemu/releases/tag/nuclei-2023.10



新增了 elf 文件查看器

在 Nuclei Studio 2023.10 新增 elf 文件编辑器，方便用户查看编译后产生 `.elf`、`.o` 文件。





新增 Code Coverage 和 Profiling 功能

在 Nuclei Studio 2023.10 新增了对 Code Coverage 和 Profiling 功能的支持，具体参考*Code Coverage 和 Profiling 功能* (page 199) 。

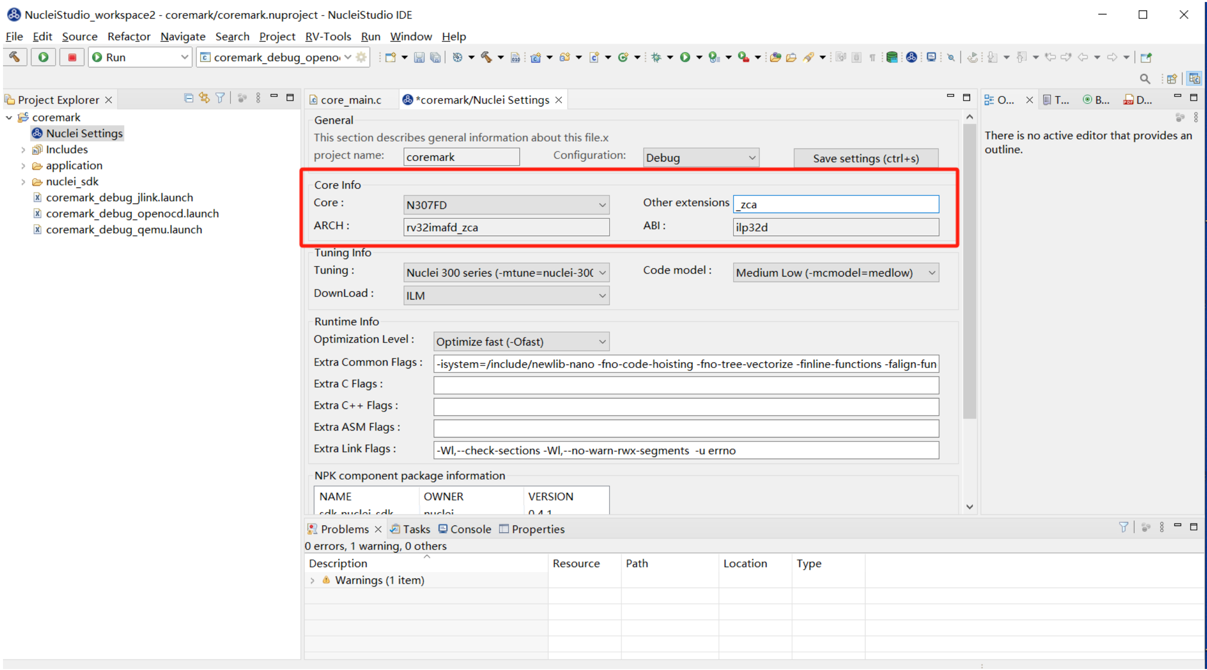
新增 trace 功能

在 Nuclei Studio 2023.10 实验性新增了 trace 功能，因使用此功能需要带有 Nuclei Trace IP 的 CPU，如需体验此功能，请与我们联系。

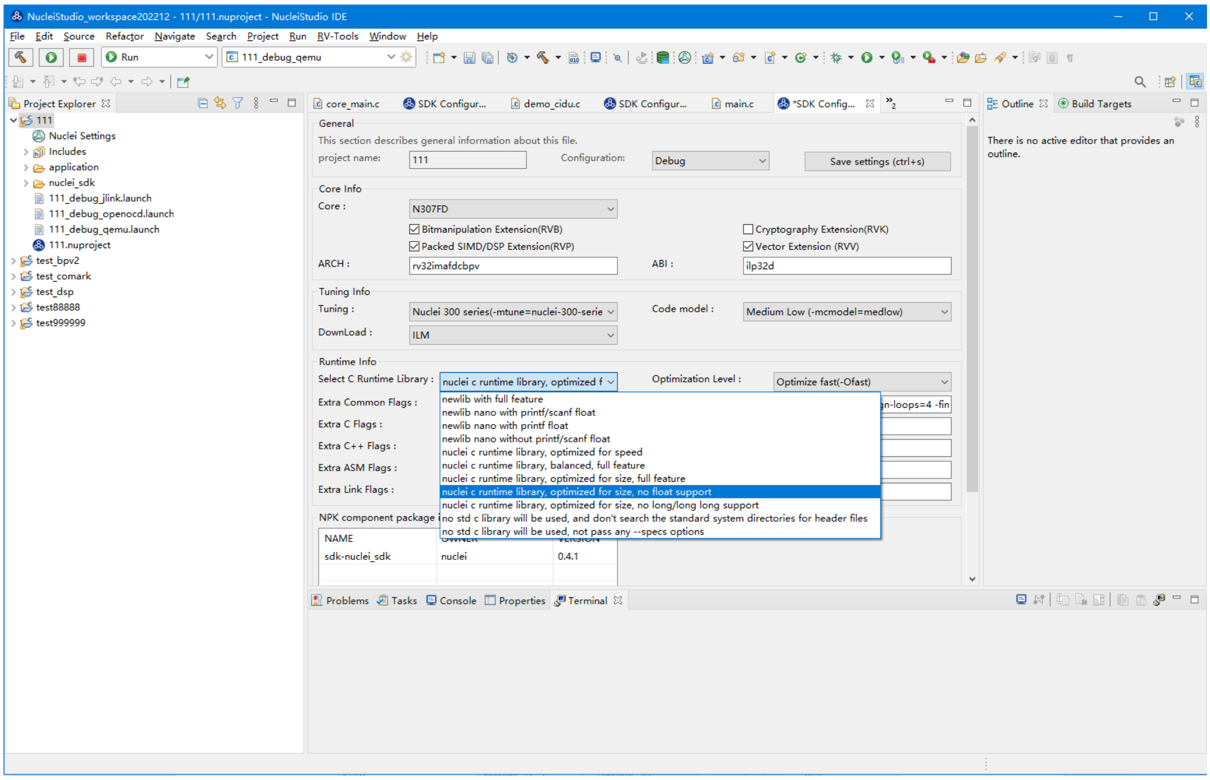
Nuclei Settings 功能优化

为了应对更个性化的配置，我们修改了 Nuclei Settings 部分功能。

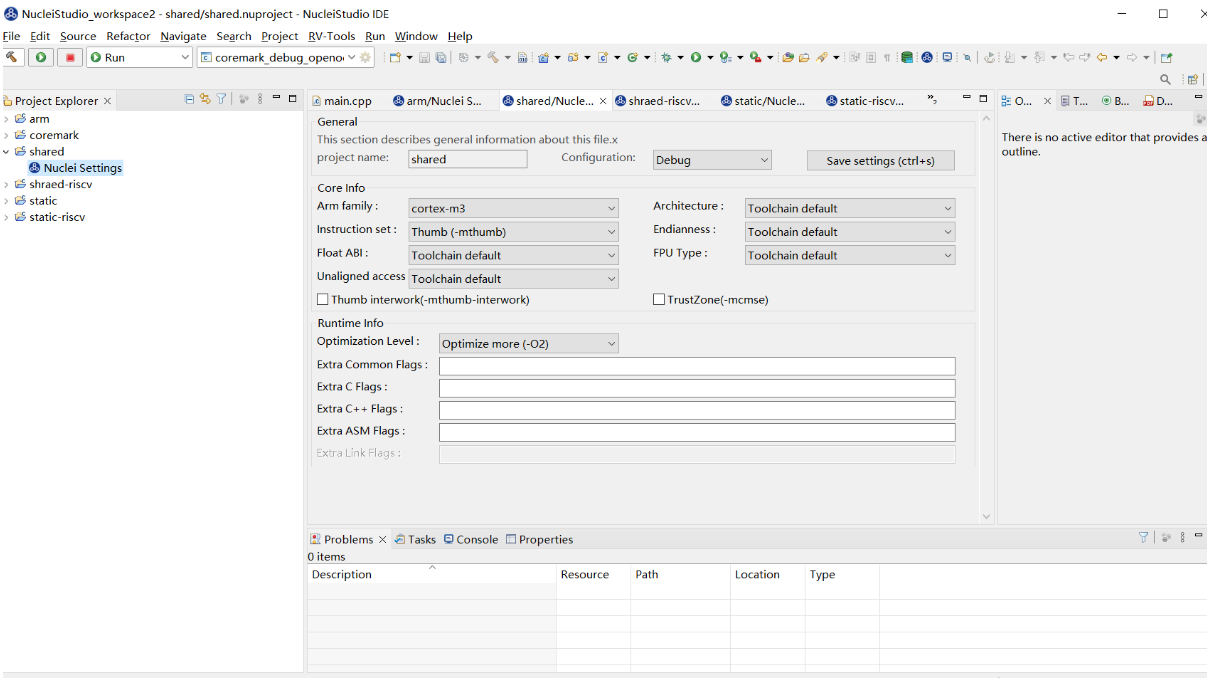
Nuclei Studio 2023.10 去掉了原来的 B/P/K/V 的单选框，换成 Other Extensions 输入框，用户可以根据自己的需求自定义填写。而关于 B/P/K/V 的使用，可以参考*RISC-V 指令扩展使用变更* (page 13) 。

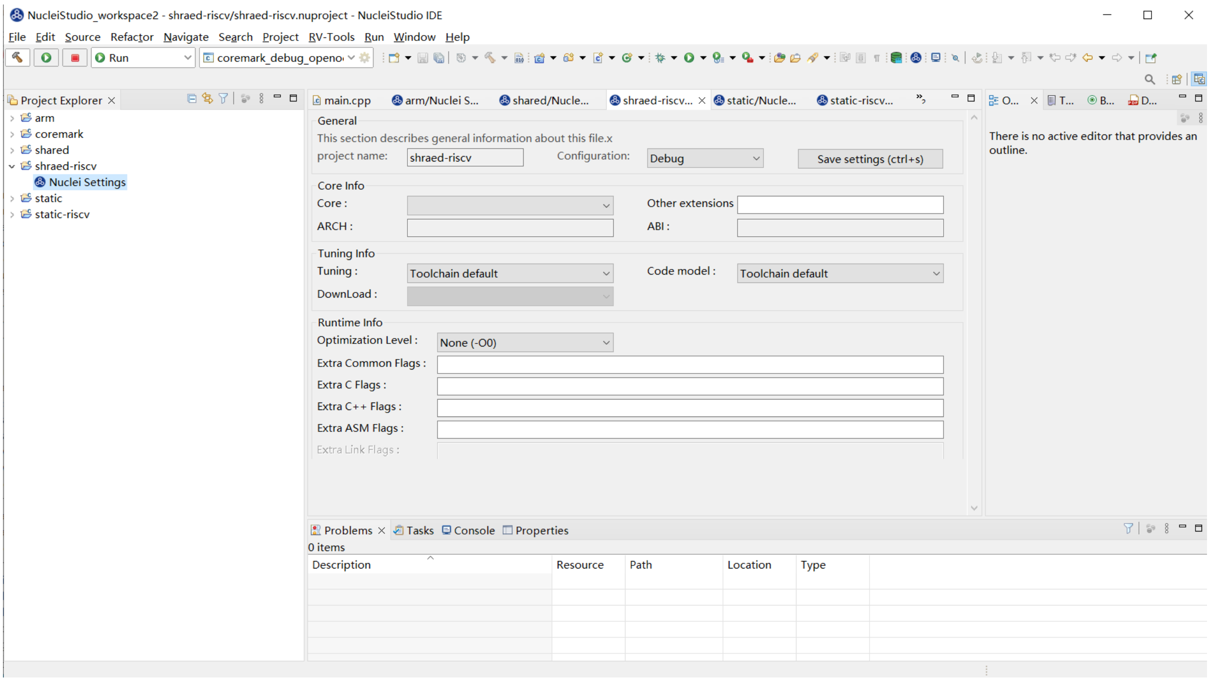


Nuclei Studio 2023.10 去掉了原来的 Select C Runtime Library 单选框，在项目中如果需要使用，可能过项目配置传入的 `--specs=` 选项，或者 Libraries 选项, 来实现。



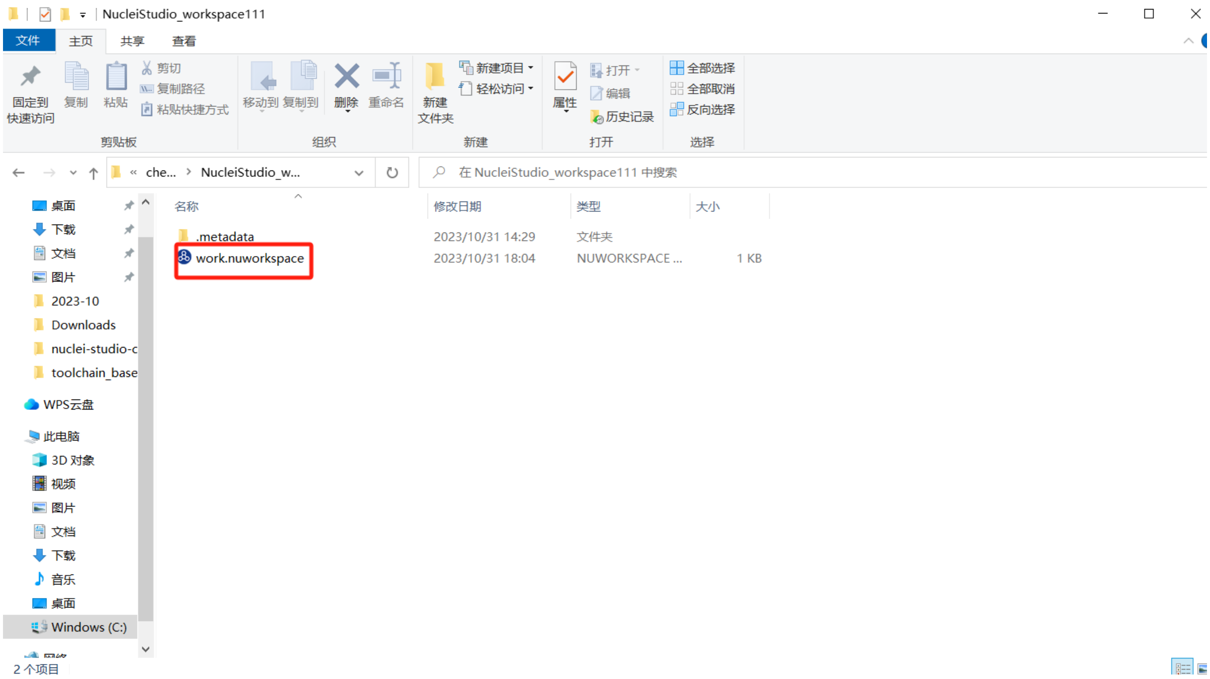
Nuclei Settings 增强了其通用性，使它不仅仅能对 Nuclei 的工程进行快速修改，也新增以对通用 riscv 和 arm 创建的 static 和 shared 的 library 工程的支持。下面为 shared 对应示图。





新增指定工作空间快速打开

类似双击项目下的 `*.nuproject` 文件可快速打开 Nuclei Studio 并导入该项目，现在 Nuclei Studio 会在使用过的工作空间目录下创建 `work.nuworkspace` 文件，双击该文件可以直接打开 Nuclei Studio，但该功能暂时只支持 windows 版本。这个功能需要解压 IDE 后，在 windows 上执行 `install.bat` 来设置文件关联。



2.2.5 2022.12 版更新说明

Nuclei Studio 自 2021.09 版后，将 IDE 与 SDK 完全分离，将采用全新的 Nuclei Package(NPK) 的包管理的方式进行模板工程的管理和使用，方便用户进行不同 SDK 的导入并且在 IDE 上创建示例工程并使用，针对 Nuclei SDK 和 HBird SDK 以及我们公司的 SoC IP 产品提供的 SDK，均可以打包成 Zip 包的方式以通过 Nuclei Package Management 方式进行导入使用。

2.3 Nuclei Studio 下载与安装

2.3.1 Nuclei Studio IDE 下载

为了方便用户快速上手使用，本文档推荐使用预先整理好的 Nuclei Studio IDE 软件压缩包。芯来公司已经将该软件压缩包上传至公司网站，具体地址为 https://www.nucleisys.com/download.php。

用户可以在芯来科技公司网站的“下载中心”，根据用户开发环境，下载对应 Windows 或 Linux 的 Nuclei Studio 压缩包（注意：芯来科技公司网站的下载中心，其内容会不断更新，用户请自行选择使用最新版本或继续使用当前版本）。

目前已在 Win 10 64 位系统，Ubuntu 18.04/20.04 和 Redhat7.6 64 位版本上验证测试，推荐使用以上版本的系统。

2.3.2 Nuclei Studio IDE 安装

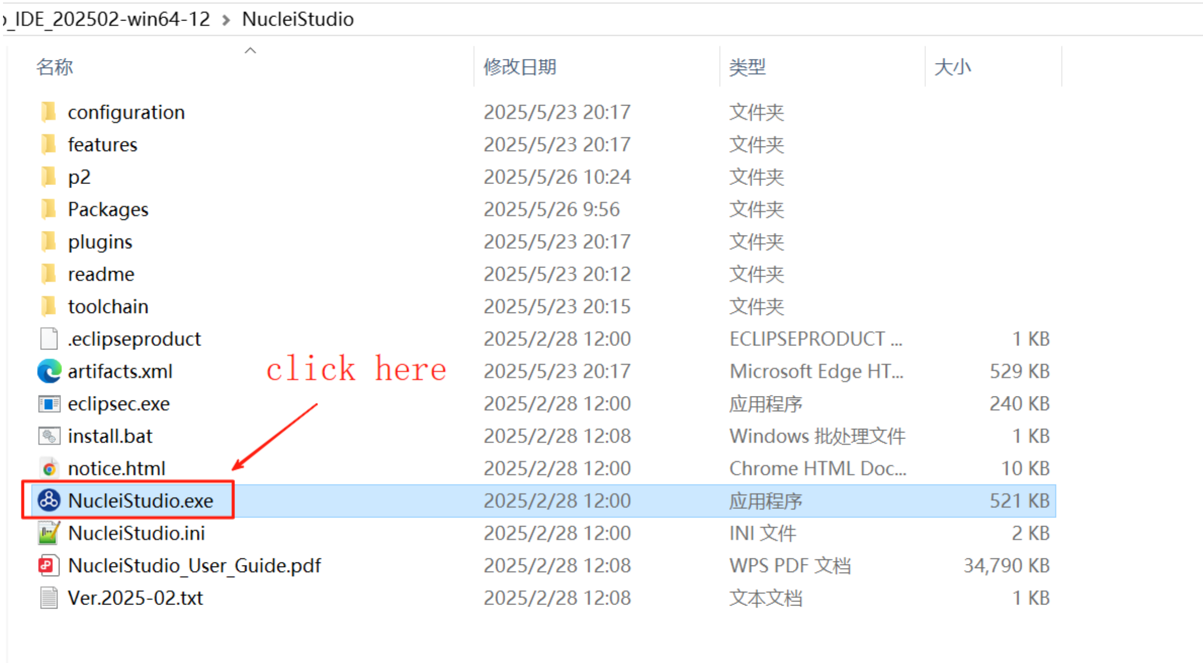
当完成 Nuclei Studio IDE 压缩软件包下载，解压后包含若干文件，分别介绍如下。

- Nuclei Studio 软件包
  - 该软件包中包含了 Nuclei Studio IDE 的软件。注意：具体版本以及文件名可能会不断更新。
- HBird_Driver.exe（2021.02 版本起不再提供）
  - 仅 **Windows** 版提供，此文件为芯来蜂鸟调试器的 USB 驱动安装文件。
  - 当在 Windows 环境下，使用该调试器时，需要安装此驱动使该 USB 设备能够被系统识别。
  - 由于 2021.02 版本中更新的 openocd 引入了免驱功能。
- SerialDebugging_Tool（2021.02 版本起不再提供）
  - 仅 **Windows** 版提供，此文件为“串口调试助手”软件。此软件可以用于后续软件示例调试时通过串口打印信息。

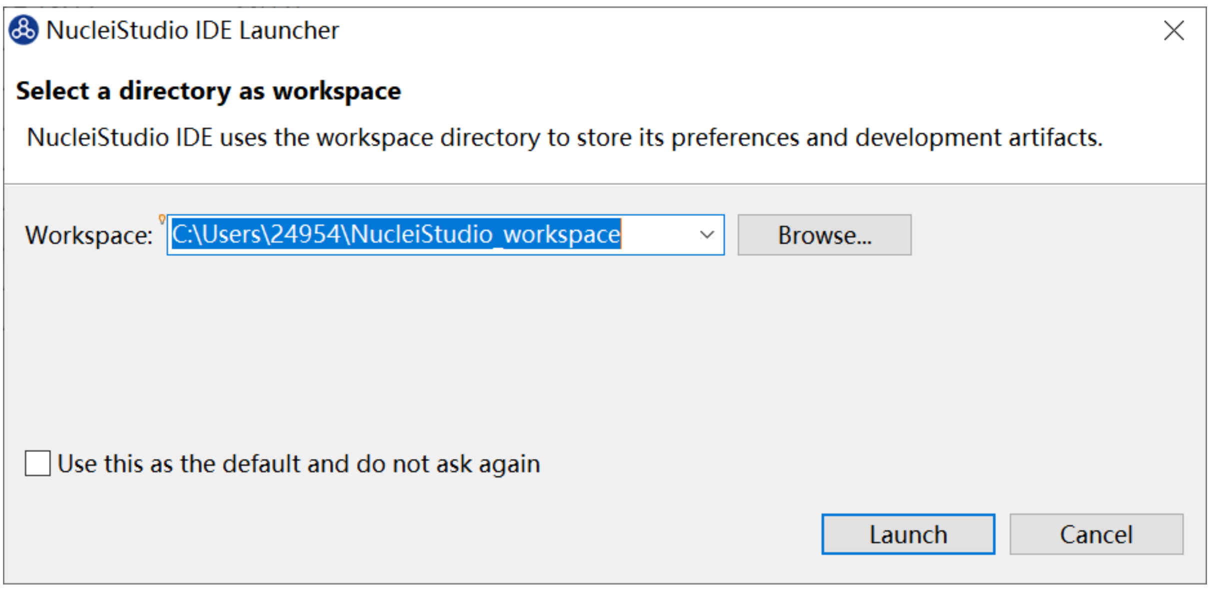
2.3.3 Nuclei Studio IDE 启动

启动 Nuclei Studio 的要点如下（windows 和 linux 均按照如下操作）：

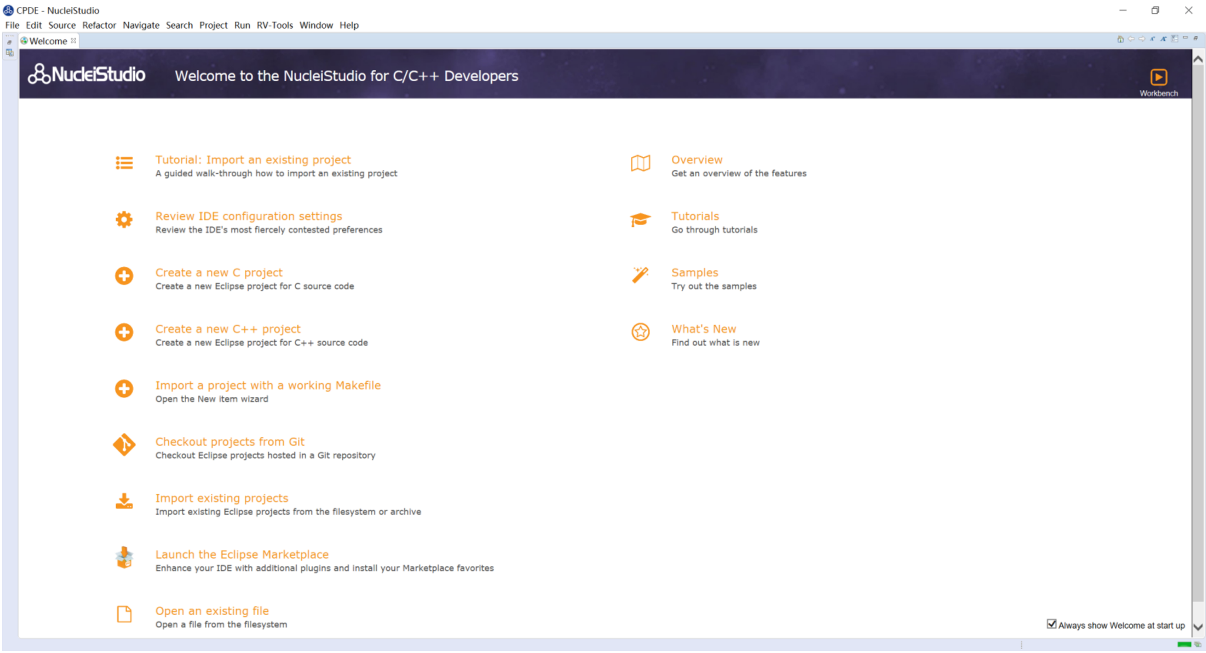
直接双击 Nuclei Studio IDE 文件包中 Nuclei Studio 文件夹下面的可执行文件，即可启动 Nuclei Studio。



第一次启动 Nuclei Studio 后，将会弹出对话框要求设置 Workspace 目录路径，该目录将用于存放后续创建的项目工程文件。



设置好 Workspace 目录之后，单击 "Launch" 按钮，将会启动 Nuclei Studio。第一次启动后的 Nuclei Studio。



Note

2021.02 版本 Nuclei Studio 默认关闭了 Launch Bar，请参照 10.10.1 开启 Nuclei Studio 中的 Launch Bar 功能，方便快速编译调试和下载。

2.4 Nuclei Studio NPK 介绍

在芯来 RISC-V 嵌入式软件开发里面引入组件 (Package) 的理念，主要是借鉴于 npm 包管理，开发出方便开发者开发，使用，分发设计好的软件组件，可以大大加快软件的迭代速度。

组件包主要由以下文件组成：

- 组件描述文件 (基于 YAML 语言)：`npk.yml`
- 组件相关代码以及说明文档

组件包会以 zip 包形式存在，组件包在导入使用时，需要被 IDE 或者其他工具进行包完整性以及可用性检查，才予以导入。

Note

- 我们提供了一个开源的 NPK 软件包检查工具，参见 https://github.com/Nuclei-Software/npk-checker
- 最好的 NPK 软件包参考项目可以参见 Nuclei SDK 以及其他线上的软件包 (搜索 npk.yml 文件) https://www.rvmcu.com/nucleistudio-npk.html
- 组件包后期会提供签名机制，在发布前先签名，然后导入软件包会验证签名，确保导入的开发包是合法身份发布，不导入不可信的开发包。

2.4.1 组件描述文件 (npk.yml)

组件主要分为以下几大类型，分别是：

- **csp:** Core Support Package, 例如 NMSIS
- **ssp:** SoC Support Package, 例如 gd32vf103 的 SoC 的支持包
- **bsp:** Board Support Package，例如 rvstar 开发板板级支持的代码包
- **osp:** RTOS Support Package，例如各类 RTOS 支持包
- **app:** Application Package，例如各类上层应用
- **mwp:** Middleware Package，各类第三方中间件，例如语音识别，算法库之类的
- **sdk:** Software Development Kit，一组预设定好的软件开发包，一般情况下里面会包含了 CSP, SSP, BSP, APP 类型的包，OSP 和 MWP 类型的包可选加入
- **bdp:** Bundle Package, 一组 package，目前仅对 app 有效

以下是比较特殊的类型，主要用于工具包和模版包

- **tool:** Tool Package，各类工具组件包，可放入其他需要引用或参考的文件
- **tpp:** Template Package, 模板类型，可以用于创建 csp/ssp/bsp/osp/app/mwp/sdk 的模板工程, 该类型比较特殊，描述文件名称为 **npk_template.yml**

描述文件通用定义

组件描述文件示例以及详细说明参见如下：

```
Package Base Information
--------------------------
name: your_package_name                    # <MUST> 组件包名称 ID，不要有空格，符合 C 语言命
名规范，英文名称，唯一名称 ID，用于 dependency 管理
                                           # name 命名规则：唯一性，全英文，不支持特殊字符
                                           # csp 类型 package: csp-xxxx
                                           # ssp 类型 package: ssp-xxxx
                                           # bsp 类型 package: bsp-xxxx
                                           # osp 类型 package: osp-xxxx
                                           # app 类型 package: app-xxxx
                                           # mwp 类型 package: mwp-xxxx
                                           # sdk 类型 package: sdk-xxxx
                                           # tpp 类型 package: tpp-xxxx
                                           # tool 类型 package: tool-xxxx
description: 简要描述软件包的功能           # <MUST> 一句话描述清楚包的主要特性与功能, English␣
↪only, 20 字符以内
details: 详细描述软件包的功能               # <MUST> 描述清楚包的主要特性与功能, English only
version: 1.2.3                             # <OPTIONAL> 可选，如不填，默认为空，建议采用
SEMVER2.0 版本号管理，只能数字打头, 例如 1.2.3
                                           # <MUST> 针对 sdk 类型的 package 必须包含一个
version tag
type: ssp                                  # <MUST> 包类型， 可选类型有 csp, ssp, bsp,␣
↪osp, app, mwp, sdk, tpp, tool
os: win64                                  # <MUST> OS 类型，例如 win32、win64、lin64、
lin32, 但目前组件包上传页面只支持 win64 和 lin64, 该字段只存在 tool 类型 package
category:                                  # <OPTIONAL> 包分类，按照上面的包类型进行二次分类，
主要是针对 package 进行分类
keywords:                                  # <MUST> 包关键字，主要用于索引与搜索
  - soc
  - risc-v
  - nuclei
license: Apache-2.0                        # <OPTIONAL> 对应包代码所用的许可证，可选列表␣
↪GPL, GPLv2, MIT, Apache License v2, BSP 等
```

(continues on next page)

(continued from previous page)

```
owner: nuclei                                   # <MUST>    组件包的拥有者，便于后期进行权限查找匹
配
                                                # 后期在整合到网站以后，用户在发布新的开发包的时候，
需要先注册一个唯一的 owner 名称
                                                # 然后申请一个 package name，确保 name 可用的情
况下才可以发布新的该 name 的包，同时限制单用户发布的包个数。
contributors:                                   # <OPTIONAL> 组件包的代码贡献者列表
    - author_name, author_email
homepage:                                       # <OPTIONAL> 组件包的主页, 如果有的话，必须是链接

## Package Information
-----------------------

packinfo:                                       # <MUST> Only for bsp/ssp type, optional␣
↪for other types 用于描述 SoC 层面的一些信息
  core_vendor: Nuclei                           # <MUST> only for ssp, 处理器内核的供应商英文名
称，如果是基于芯来的处理器内核，这里填写 Nuclei
  vendor: Nuclei                                # <MUST> For ssp/bsp, <OPTIONAL> for others
                                                # For ssp, SoC 或者芯片的供应商英文名称，填写对应
的厂商的名称，例如 GigaDevice
                                                # For bsp, 表示板子的提供商英文名称
                                                # For others, OPTIONAL, 表示该 package 提供商
  name: demosoc                                 # <MUST> 这里 name 和 package name 作用不一样,
↪ 用来显示具体的包名称
                                                # For ssp, 表示本组件包中 SoC 的具体英文名称，可
以是一个系列名称，例如 GD32VF103
                                                # For bsp, 板子的名称
                                                # For others, 表示该 package 显示名称，中英文均
可，不写的话，默认使用根字段 name
  doc:                                          # <OPTIONAL>, 不填的话，这里为空，不做展示
    website:                                    # <OPTIONAL>, Package 对应的网站
    sch:                                        # <OPTIONAL> only for bsp, 表示板子的电路图
    datasheet:                                  # <OPTIONAL>, package datasheet 本地地址或者是
网址
    usermanual:                                 # <OPTIONAL>, Package User Manual
    extra:                                      # <OPTIONAL>, 额外的一些资料，列表形式 (URI +␣
↪DESCRIPTION)
      - uri:                                    # <OPTIONAL>, file path or web link
        description:                            # <MUST> when uri is defined, description␣
↪for file


## Package Dependency
----------------------

dependencies:                                   # <OPTIONAL> 列出依赖的组件包列表 owner/
↪name:version
  - name: csp-nsdk_nmsis                        # <MUST> when defined        # 1. 针对 sdk
类型的包，内部有一个隐形的依赖，依赖有且仅有一个 bsp 类型的包，
    owner: nuclei # <OPTIONAL> if not defined, it will use the owner definiton␣
↪above. 用于依赖特定所有者的 package，  owner/name:version，  最终查找的包按照这样来找的
      version: # <OPTIONAL> when defined empty, use default as version number #   ␣
↪并且会自动查找当前路径下所有的 npk.yml 文件，并作为 sdk 包中一部分
                                                # 2. 除了 sdk 类型的包之外的其他的包，如果不是放在
sdk 包下面的目录，则依赖于
                                                #    sdk，则需要显式加上 dependency
                                                # 3. bsp 类型的 package 肯定会依赖一个 ssp 类型
的 package
                                                # 4. 依赖包也可以带上版本号，支持版本号条件比对，如
果不带版本号，则优先选择不带版本号的包，其次最新的包
                                                # 参考 https://docs.platformio.org/en/
```

(continues on next page)

(continued from previous page)

```
↪latest/librarymanager/config.html#version
                                          # https://docs.npmjs.com/about-semantic-
↪versioning
                                          # 1.2.3 - an exact version number. Use␣
↪only this exact version
                                          # ^1.2.3 - any compatible version (exact␣
↪version for 1.x.x versions
                                          # ~1.2.3 - any version with the same major␣
↪and minor versions, and an equal or greater patch version
                                          # >1.2.3 - any version greater than 1.2.3.␣
↪>=, <, and <= are also possible
                                          # >0.1.0,!=0.2.0,<0.3.0 - any version␣
↪greater than 0.1.0, not equal to 0.2.0 and less than 0.3.0
                                          #    例如 version: master, version: 1.2.3,␣
↪version: >0.1.0
                                          # 依赖关系处理规则
                                          # 1. sdk，app 类型的包可以依赖多个 ssp、bsp 类型
的包，但是最终只会根据 project wizard 选择具体使用到 package
                                          # 2. app/bsp/ssp/csp/osp/mwp 类型的包可以依赖
sdk，如果依赖了 sdk 类型的包，则表述该包隶属于 sdk 下
                                          # 3. bsp 类型的包只能依赖一个 ssp/csp 类型的包，
ssp 类型的包只能依赖一个 csp 类型的包
                                          # 4. sdk 类型的包可以依赖多个 app/bsp/ssp/csp/
↪osp/mwp 类型，但是这种依赖只是建立从属关系，表示该 sdk 包含了这些包
                                          # 5. 一个 sdk 类型的包不可以依赖另一个 sdk 包，但
是 app/bsp/ssp/csp/osp/mwp 却可以依赖多个 sdk 类型的包

## Package Configurations
--------------------------

configuration:                            # <OPTIONAL> 关于包配置的一些选项，用于 Project␣
↪Wizard 创建以及内部参数设置
                                          # 其中 sdk 类型暂不支持 configuration 参数定义
                                          # configuration 定义的配置可以互相覆盖
                                          # 覆盖规则为 app > mwp  > osp > bsp > ssp >␣
↪csp
  nuclei_core:                            # <OPTIONAL> 一个配置选项，类型为 choice
    default: n307fd                       # <MUST> 如果这个配置定义了，针对 choice 类型，默
认值选择必须是 choices 里面列举的
    type: choice                          # <MUST> 配置类型，可选有 choice, list,␣
↪checkbox, multicheckbox，text
    global: true                          # <OPTIONAL> 可选为 true 或者 false，默认为
true
    description: Nuclei RISC-V Core       # <MUST> 该配置项的描述，20 字以内
    choices:                              # <MUST> 当配置项 type == choice 时
      - name: n201                        # <MUST> item 中必须包含 name 和 description
        arch: rv32iac                     # <OPTIONAL> 仅用于表示 RISC-V CORE 中的 ARCH
信息，不建议随意使用
        abi: ilp32                        # <OPTIONAL> 仅用于表示 RISC-V CORE 中的 ABI
信息，不建议随意使用
        tune:                             # <OPTIONAL> 仅用于表示 RISC-V CORE 中的 TUNE
信息，不建议随意使用
        info:                             # <OPTIONAL> 用于自定义 key-value 数据的访问，例
如 ${nuclei_core.info.key1} 返回的是 value1
          - name: key1                    # <MUST> key in pair with value, 字符串类型，
不包含任意空格
            value: value1                 # <MUST> value in pair with key，字符串类型
          - name: key2                    # <MUST>
            value: value2                 # <MUST>
        description: N201 Core(ARCH=rv32iac, ABI=ilp32)     # <MUST> 描述这个 item 具
体含义
```

(continues on next page)

(continued from previous page)

```
                                                # 除了 name, description 之外，可能会定义其他
items, 名称不定
                                                # 例如在这里就定义了 arch 和 abi
      - name: n201e                             # 另一个 item
        arch: rv32eac
        abi: ilp32e
        description: N201E Core(ARCH=rv32eac, ABI=ilp32e)
  extra_flags:                                         # <OPTIONAL> 一个配置选项, 当前这个为
text 类型
    value:                                             # <MUST>, 仅接受英文字符串
    description: Extra compiler flags                  # <MUST> 该配置项的描述，30 字以内
  dsp_present:                                         # <OPTIONAL> 一个配置选项, 当前这个为
checkbox 类型
    default: 0                                         # <MUST>, 默认为 0，可选 0 或者 1
    type: checkbox                                     # <MUST>, 配置类型，当前为 checkbox
    global: true                                       # <OPTIONAL> 可选为 true 或者 false，默
认为 true
    description: P-Extension(DSP) Present              # <MUST> 该配置项的描述，30 字以内
  libraries:                                           # <OPTIONAL> 一个配置选项，当前这个为
multicheckbox 类型
    default: [dsp, nn]                                 # <MUST> 默认值, 为 choices 里面的组合
    type: multicheckbox                                # <MUST>, 配置类型，当前为
multicheckbox
    global: true                                       # <OPTIONAL> 可选为 true 或者 false，默
认为 true
    description: Libraries Used                        # <MUST> 该配置项的描述，30 字以内
    choices:                                           # <MUST> 当配置项 type ==␣
↪multicheckbox 时
      - name: dsp                                      # <MUST> item 中必须包含 name 和
description
        description: DSP Library                       # <MUST> 描述这个 item 具体含义
                                                       # 除了 name, description 之外，可能会定
义其他 items, 名称不定
      - name: nn
        description: NN Library
      - name: ai
        description: AI Library

## Source Code Management
-------------------------

codemanage:                                         # <MUST> 这个为必选项
  installdir: demosoc                               # <MUST> 希望代码安装的目录名称，仅限英文，满
足 C 语言命名格式
                                                    # 针对 sdk 类型的 package, 会被安装到<sdk_
↪installdir>, 如果 installdir 未定义，默认为 SDK，如果没有任何 sdk 类型的 package 被引用，
sdk_installdir 也被默认设置为 SDK
                                                    # 针对 csp 类型的 package，会被安装到<sdk_
↪installdir>/<csp_installdir> 目录下，TBD
                                                    # 针对 ssp 类型的 package, 会被安装到<sdk_
↪installdir>/SoC/<ssp_installdir>/Common 下面
                                                    # 针对 bsp 类型的 package, 会被安装到<sdk_
↪installdir>/SoC/<ssp_installdir>/Board/<bsp_installdir> 下面，如果不依赖于任何 ssp 类
型，则安装到<sdk_installdir>/BSP/<bsp_installdir>
                                                    # 针对 osp 类型的 package, 会被安装到<sdk_
↪installdir>/OS/<osp_installdir> 下面
                                                    # 针对 app 类型的 package, 会被安装到<app_
↪installdir>/目录下， 如果 installdir 未定义，默认为 application
                                                    # 针对 mwp 类型的 package, 会被安装到<sdk_
↪installdir>/Components/<mwp_installdir> 目录下
  copyfiles:                                        # <MUST> 待拷贝的文件或者文件夹，支持 glob␣
```

(continues on next page)

(continued from previous page)

```
↪pattern 匹配，这里是指所有的目录或者文件
    - path: ["Source/", "Include/", "demosoc.svd"]          # <MUST> 待拷贝的文件或者文件
夹的路径列表，支持 glob pattern 匹配
    - path: ["DSP_Source", "DSP_Include"]
      condition: $( ${dsp_present} == 1 )                   # <OPTIONAL> 这里的 if 是一
个固定的标识符，如果出现则表示要做判定，判定的方式如下
                                                            # 如 dsp_present 是在
configuration 里面定义的，根据 wizard 或者其他 package 选定而定
  incdirs:                                                  # <OPTIONAL> 需要加入头文件
目录列表
    - path: ["Include/"]                                    # <OPTIONAL> 需要加入头文件
的目录
  libdirs:                                                  # <OPTIONAL> 可选的 lib 库
所在目录
      - path:
  ldlibs:                                                   # <OPTIONAL> 可选的需要链接
的库
      - libs:

## Set Configuration for other packages
----------------------------------------

setconfig:                                                  # <OPTIONAL> 这个用于设置其
他 Package 的选项

# 以下选项是覆盖关系，规则 app > mwp  > osp > bsp > ssp > csp
  - config: nmsislibarch
    value : ${nuclei_core.arch}p                            # <OPTIONAL> 直接设置
Configuration 里面的选项
    condition: $( ${dsp_present} == 1 )                     # <OPTIONAL> 根据这里 dsp_
↪present 来判断是否设置 nmsislibarch 值
  - config: nmsislibarch
    value: ${nuclei_core.arch}
    condition: $( ${dsp_present} == 0 )

## Build Configuration
-----------------------

buildconfig:                                   # <OPTIONAL> 编译选项的配置
                                               # 目前编译选项会将 package 中定义的所有的拼
接在一起或者覆盖
                                               # 以下选项是覆盖方式，app > mwp  > osp >␣
↪bsp > ssp > csp
                                               # type 是一个特殊字段，用于标识特定的编译器，␣
↪目前支持 gcc
                                               # cross_prefix, prebuild_steps,␣
↪postbuild_steps, description
                                               # 其余选项是拼接的
  - type: gcc                                  # <OPTIONAL> 目前只有 gcc，预留其他接口
    description: Nuclei GNU Toolchain          # <MUST> For ssp
      cross_prefix: riscv-nuclei-elf-          # <OPTIONAL> 如果不写或者留空，就自动按照系
统里面提供的工具链来定
    common_flags:                              # <OPTIONAL> 通用的编译选项，将会添加到
cflags, asmflags, cxxflags 上
      - flags: -g -fno-common -ffunction-sections -fdata-sections
      - flags: -march=${nuclei_core.arch} -mabi=${nuclei_core.abi} -mcmodel=medany
    ldflags:                                   # <OPTIONAL> 链接选项列表，留空表示没有任何
选项
      - flags: -nostartfiles --specs=nosys.specs
      - flags: --specs=nano.specs
        condition: $(${newlib} != "normal")
```

(continues on next page)

(continued from previous page)

```
      - flags: -u _printf_float
        condition: $(${newlib} != "nano_with_printfloat")
      - flags: -u _isatty -u _write -u _sbrk -u _read -u _close -u _fstat -u _lseek
    linkscript:                                  # <MUST> 链接脚本的定义，必须在 bsp/ssp 中
定义
      - script: "Source/GCC/gcc_demosoc_${.download}.ld"
          condition: $(check pattern)            # <OPTIONAL> 进行条件判断
    cflags:                                      # <OPTIONAL> C 编译选项，留空表示没有任何选
项
      - flags: -O3
    asmflags:                                    # <OPTIONAL> ASM 编译选项，留空表示没有任何
选项
      - flags: -O2
    cxxflags:                                    # <OPTIONAL> CXX 编译选项，留空表示没有任何
选项
      - flags: -O1
    common_defines:                              # <OPTIONAL> 通用的宏定义
      - defines: __RISCV_FEATURE_DSP=1
        condition: $(${dsp_present} == 1)
      - defines: DOWNLOAD_MODE_STRING=\"flashxip\"
    cdefines:                                    # <OPTIONAL> C 的宏定义
      - defines: __RISCV_FEATURE_DSP=1
        condition: $(${dsp_present} == 1)
      - defines: DOWNLOAD_MODE_STRING=\"flashxip\"
    asmdefines:                                  # <OPTIONAL> ASM 的宏定义
      - defines: __RISCV_FEATURE_DSP=1
        condition: $(${dsp_present} == 1)
      - defines: DOWNLOAD_MODE_STRING=\"flashxip\"
    cxxdefines:                                  # <OPTIONAL> CXX 的宏定义
      - defines: __RISCV_FEATURE_DSP=1
        condition: $(${dsp_present} == 1)
      - defines: DOWNLOAD_MODE_STRING=\"flashxip\"
    prebuild_steps:                              # <OPTIONAL> 编译前执行的命令
      command:                                   # <OPTIONAL> 执行的命令行
      description:                               # <OPTIONAL> 执行的命令的描述
    postbuild_steps:                             # <OPTIONAL> 编译完成后执行的命令
      command:                                   # <OPTIONAL> 执行的命令行
      description:                               # <OPTIONAL> 执行的命令的描述

## Debug Configuration
------------------------

debugconfig:                                # <MUST> For bsp type, optional for app/
↪ssp type
# 目前 Debug 选项会将 package 中定义的所有的拼接在一起或者覆盖
# type 是一个特殊的字段，用于描述特定的调试器，目前支持 openocd, qemu
# 以下字段是覆盖关系：app > mwp  > osp > bsp > ssp > csp
# description, svd
# configs 字段下面的 key, value 是合并关系，如果对应的 key 存在就覆盖，覆盖规则同上，如果不存在
就合并
  - type: openocd    # <MUST> 选择的工具
    description: Nuclei OpenOCD # <MUST> For bsp type
    svd: gd32vf103.svd   # <OPTIONAL> 可选的 SVD 文件
    configs:
      - key: config   # openocd 配置文件
        value: "openocd_gd32vf103.cfg"

  - type: qemu
    description: Nuclei QEMU
    svd:
    configs:
```

(continues on next page)

(continued from previous page)

```
      - key: nuclei_core    # Nuclei RISC-V Core
        value: ${nuclei_core}
        condition:  # condition set nuclei_core key
      - key: download_mode  # Download mode
        value: ${download_mode}
      - key: riscv_arch     # RISCV ARCH
        value: ${nuclei_core.arch}
      - key: riscv_abi      # RISCV ABI
        value: ${nuclei_core.abi}
      - key: machine    # QEMU Machine
        value: gd32vf103v_rvstar

##Extended variable
## Only works on tool 类型
## 每个包存在一个包路径，引用为 npk 名称-版本号，例如${tool-cmlink-1.0.0}，
## 其他变量的引用为 npk 名称-版本号-变量名，例如 ${tool-cmlink-1.0.0-proxy}
environment:                # 扩展变量
  - key: proxy                # 变量名,
    value: bin/cmlink_gdbserver.exe         # 实际引用结果为 npk 文件父路径 +value，例如 C:\Users\jj\nuclei-pack-npk\NPKs\XinShengTech\Tool_Package\tool-cmlink\1.0.0\
↪cmlink\bin\cmlink_gdbserver.exe
    description: proxy location
    system: true     # 默认为 false，当 system 为 true 时，该变量引用时直接使用变量名，例如$
↪{proxy}

## Template File Management
## Only works on tpp 类型，该类型比较特殊，描述文件为 npk_template.yml，是基于 npk.yml 做的扩展
templatemanage:
  installdir: ${soc}
  files:
    build.mk.ftl: build.mk
    Common/npk.yml.ftl: Common/npk.yml
    Common/demosoc.svd.ftl: Common/${soc}.svd
    Common/Source/demosoc_common.c: Common/Source/${soc}_common.c
    Common/Source/system_demosoc.c.ftl: Common/Source/system_${soc}.c
    Common/Source/Drivers/demosoc_uart.c.ftl: Common/Source/Drivers/${soc}_uart.c
    Common/Source/GCC/intexc_demosoc.S.ftl: Common/Source/GCC/intexc_${soc}.S
    Common/Source/GCC/startup_demosoc.S.ftl: Common/Source/GCC/startup_${soc}.S
    Common/Source/Stubs: Common/Source/Stubs
    Common/Include/demosoc.h.ftl: Common/Include/${soc}.h
    Common/Include/demosoc_uart.h.ftl: Common/Include/${soc}_uart.h
    Common/Include/nuclei_sdk_soc.h.ftl: Common/Include/nuclei_sdk_soc.h
    Common/Include/system_demosoc.h.ftl: Common/Include/system_${soc}.h
    Board/nuclei_fpga_eval/openocd_demosoc.cfg: Board/${board}/openocd_${soc}.cfg
    Board/nuclei_fpga_eval/npk.yml.ftl: Board/${board}/npk.yml
    Board/nuclei_fpga_eval/Source/GCC/gcc_demosoc_ilm.ld.ftl.ftl: Board/${board}/
↪Source/GCC/gcc_${soc}_ilm.ld
    Board/nuclei_fpga_eval/Source/GCC/gcc_demosoc_flash.ld.ftl: Board/${board}/
↪Source/GCC/gcc_${soc}_flash.ld
    Board/nuclei_fpga_eval/Source/GCC/gcc_demosoc_flashxip.ld.ftl: Board/${board}/
↪Source/GCC/gcc_${soc}_flashxip.ld
    Board/nuclei_fpga_eval/Include/board_nuclei_fpga_eval.h.ftl: Board/${board}/
↪Include/board_${board}.h
    Board/nuclei_fpga_eval/Include/nuclei_sdk_hal.h.ftl: Board/${board}/Include/
↪nuclei_sdk_hal.h
```

内容约定

为了保证 `npk.yml` 文件的可读性与简约性，对 `npk.yml` 文件的存储制定如下约定：

- 各字段的存储顺序请保持与模板一致，数据与 DICT 按读入时顺序保存
- 各字段建议适当加上注释，尤其是那种需要解释的地方
- **MUST** 类型的字段需要按照上述注释描述的规则进行检查，如果不合规请报错提示，并不予导入
- 缩进建议采用 2 个空格字符
- 针对一些 **OPTIONAL** 的字段可以留空或者不写该字段
- 一级字段之间增加一行空行，二级及以下字段不使用空行，第一部分基础信息一级字段间不使用空行
- 字符串建议不使用引号，除特殊语法需要
- 所有的 `description`字段建议控制在 20 字符以内，方便排版展示，仅限英文
- 关于 yaml 里面多行的约定如下：https://yaml-multiline.info/

包导入规则

下面定义合法的包导入规则：

- 如果存在导入包中存在一些依赖的包 (带版本匹配)，并没有被导入，则不允许导入，并提示缺乏依赖的包，请导入该包。
  - 后续包管理联网了，则可以提示是否从网上下载依赖的包，或者手动导入 zip 包
- 如果删除包，并且该包被其他包依赖，则提示哪些包依赖于该包，询问是否删除，如果删除以后，则在包管理中显示缺少的包
  - 后续包管理联网了，支持点击按钮下载缺失的包，或者手动导入 zip 包
- 如果导入相同版本的包，则提示该包已经存在，是否替换
- 如果导入不同版本的包，则提示已经存在其他版本的包，是否继续导入
- 导入的包，按照定义的类型分类显示，显示包的版本，包的 name，owner，description, homepage, license

zip 包内容规范

下面定义合法的 zip 包的内容规则：

- 一个 zip 包中必须包含至少 1 个 npk 文件
- 包类型的判定：如果包内存在多个类型的 **npk** 文件，**npk** 类型判定条件如下
  - sdk > ssp > bsp > osp > mwp > csp > app
  - 如果判定出包的类型存在多个相同的 npk，则该包不合法，不允许导入，并提示
- 该类型的包不允许存在多个该类型的 npk 文件
- 如果是 `sdk` 类型的包，则必须包含至少一个 `ssp` 和依赖于该 `ssp` 的 `bsp` 文件，以及至少一个 `app` 类型的文件，允许存在其他类型的包
- 如果是其他类型的包，则里面包含的其他 npk，必须显式依赖于该包

包依赖关系处理

包依赖关系的处理涉及到如何能够将包拆分并形成合理的依赖关系，便于包的独立维护。这里对不同类型的包的依赖处理进行详细的分析。

依赖通过 `dependencies` 字段下的依赖列表来控制，支持依赖特定 owner 的某个 name，某个 version 版本的包，查找规则为 `owner/name:version` ，如果 owner 未定义，则默认为该 npk 文件中定义的 owner，如果 `version` 未定义，则优先在同组件包查找，否则取最新的包。

csp Core Support Package 依赖

csp 类型的包是处理器内核 CORE 支持的软件包，目前针对 Nuclei RISC-V 内核，我们主要推广 NMSIS 这样的开源软件支持包。

一般情况下，csp 类型的包是非常底层的包，这里不支持依赖 `ssp/bsp/mwp/rtos/app` 这样的类型的包。但是可以依赖 sdk 类型的包，表示该包属于依赖的 sdk 包的环境中。

ssp SoC Support Package 依赖

ssp 类型的包是 SoC 或者芯片的支持的软件包，例如 `gd32vf103` , `demosoc` 这样 SoC 的支持软件包。

ssp 软件包仅可以依赖 `csp/mwp/osp` 这样的软件包，如果依赖了这三种类型的软件包，则表示在工程创建的时候或者是代码引入的时候，这三类软件包需要导入代码。而如果依赖 **sdk** 类型的软件包，则表示该 **ssp** 类型的包属于依赖的 **sdk** 类型的软件包的环境。

理论上用户可以创建一个 ssp 软件包，不依赖任何 `csp/mwp/rtos` 的软件包，也不属于 sdk 类型的软件包。**osp** 类型软件包仅可以依赖一个。

bsp Board Support Package 依赖

bsp 类型的包是针对基于某款 SoC/芯片做的开发板而推出的软件支持包，例如 `gd32vf103-rvstar` 这款开发板的 bsp 软件包。

bsp 软件包仅可以依赖 `ssp/csp/mwp/osp` 这样的软件包, 如果依赖了这几种类型的软件包，则表示在工程创建的时候或者是代码引入的时候，这类软件包需要导入代码。而如果依赖 **sdk** 类型的软件包，则表示该 **ssp** 类型的包属于依赖的 **sdk** 类型的软件包的环境。

理论上用户可以创建一个 bsp 软件包，不依赖任何软件包。**osp** 类型软件包仅可以依赖一个。

osp OS Support Package 依赖

osp 类型的包是指特定的 RTOS 的软件支持包，例如 freertos，ucosii 之类的。

osp 类型的包仅可以依赖 `ssp/csp/mwp` 类型的软件包，如果依赖了这几种类型的软件包，则表示在工程创建的时候或者是代码引入的时候，这类软件包需要导入代码。而如果依赖 **sdk** 类型的软件包，则表示该 **ssp** 类型的包属于依赖的 **sdk** 类型的软件包的环境。

mwp Middleware Support Package 依赖

mwp 类型的软件包是指中间件类型的软件包，例如某个语音算法的库，某种物联网连接库如 `mqtt` ，`coap` 之类。

mwp 类型的包仅可以依赖 `bsp/ssp/csp/mwp/osp` 类型的软件包，但是不建议直接依赖 `bsp/ssp` ，在创建 middleware 的时候尽量保证其通用性，可以很好被集成到其他的软件中。

sdk Software Development Kit Package 依赖

sdk 类型的软件包是一类非常特殊的软件包，本身并不会有额外的代码引入，而是通过依赖其他类型的软件包而组织的一个特殊的包。如果是 sdk 类型的软件包，导入是会强制检查软件包目录下所有的 `npk.yml` 文件以查找其他的软件包并引入该 SDK 依赖中，无需 `npk.yml` 文件显示进行依赖，这种依赖关系并不会直接导致创建工程时的代码的导入，更多的是软件包的集合。

一个 sdk 类型的软件包可能会依赖多个 ssp，多个 bsp，多个 csp，多个 app，多个 mwp 和多个 osp。更详细的内容请参见 [构建 SDK 开发包](# 构建 SDK 开发包)

对于属于 sdk 类型的软件包的其他软件包里面的依赖，优先使用都属于统一 sdk 软件包内部的软件包。例如：

- sdk-nuclei-sdk 是一个 sdk 类型的软件包，内部包含了 csp-nsdk_nmsis, bsp-nsdk_nuclei_fpga_eval, ssp-nsdk_demosoc, ssp-nsdk_gd32vf103, osp-nsdk_freertos, osp-nsdk_ucosii 这些软件包
- 而外部也有 csp-nsdk_nmsis，osp-nsdk_freertos 这类的软件包，在工程创建阶段，在版本匹配满足的情况下，优先使用内部的 csp-nsdk_nmsis 和 osp-nsdk_freertos，只有在版本匹配不满足要求的情况下，才会使用
- 在工程创建完成后，用户可以手动升级特定的包到其他版本。

2.4.2 模块说明

从上述描述文件中可以看出，一个标准的 `npk.yml` 实际是上由几个大块组成的，而在实现应用中，我们并不一定会完全用到，一个合规的 `npk.yml` 文件，只要拥有基本的信息，就是可以正常给 Nuclei Studio 使用。

Package Base Information

这一块分信息，是 NPK 的基础的信息，很多关键的信息在这部分内容中需要描述清楚。其中着重说明几个字段。

- **name**

  必填，NPK 的名称 ID，不要有空格，符合 C 语言命名规范，英文名称，是唯一名称 ID。

- **version**

  选填，如不填，默认为空，建议采用 SEMVER2.0 版本号管理，只能数字打头, 例如 1.2.3

- **type**

  必填，可选类型值有 csp, ssp, bsp, osp, app, mwp, sdk, tpp, tool

- **os**

  选填，标明该 NPK 适用于什么类型的 Nuclei Studio，目前我们发行的 Nuclei Studio 有 win64 和 lin64 两个版本。OS 类型可以填 win32、win64、lin64、lin32, 但目前组件包上传页面只支持 win64 和 lin64, 该字段只存在 tool 类型 package

- **owner**

  必填，组件包的拥有者，该 ID 一般为认证开发者 ID，便于后期进行权限查找匹配。如果该 NPK 仅作本地测试，可以随意。

Package Information

packinfo 这一块分信息，主要是对 NPK 做一些说明，包括一些文档等信息，最终在 Nuclei Studio 中使用该 NPK 时，这部分信息，会在 New Project 的导引中显示。

Package Dependency

dependencies 描述的是 NPK 的依赖关系，为了实现 NPK 的复用性，减少 NPK 的维护成本，我们在设计时，是允许 NPK 实现依赖关系，一个 NPK 可以依赖 0 个以上的 NPK，所以在这里 dependencies 是以组对象出现，每个依赖对象内需要明确 NPK 的 **name owner version**

Package Configurations

Configuration 字段是个非常特殊的字段，主要用于提供一些可配置项，以满足在工程创建时的交互场景。

不同包里面的 configuration 字段的下的二级字段名称可以一样，如果使用一样的名称则具备一样的含义，如果定义了一样的名称则按照如下的规则进行覆盖。

覆盖规则为：app > mwp > osp > bsp > ssp > csp

Configuration 对象组会包含多个对象，而每个对象有固定的结构。

- **XXX(变量名)**

  变量名可以随意，简合 c++ 的命名规范即可。在后面部分会以 ${XXX} 或 ${XXX.XX} 的方式引用。

- **default**

  默认值，可选项。

- **type**

  这个变量的类型，为了支持更丰富的 UI 体验，我们在 NPK 中定义了很多的 UI 组件类型，具体请参看后面章节。

- **global**

  标明此字段是否在工程创建时显示在引导页面中。

- **tips**

  对该变量的说明信息，主要用于 UI 的 tips 事件。

- **hints**

  对该变量的说明信息，如值的示例等，主要用于 UI 的 hints 事件。

- **description**

  此变量的 NPK 中的说明描述。

- **UI 组件信息**

  支持的类型有 choice, list, checkbox, multicheckbox, text 等，具体信息参见

Source Code Management

codemanage 描述的是跟模板工程有关的内容，大多的时候，我们的 NPK 会包括很多复杂的功能，需要创建某个一个具体的工程的时候，我们又只需要一些具体的文件，同时需要配置这些文件的信息。**codemanage** 就是将这些信息描述出来，它包含以下关键字：

- **custom**

默认为 false, 当为 true 时，这个 installdir 就表示直接安装的目录

- **srcroot**

默认为 `.`，表示当前 `npk.yml` 所在目录，可以是相对路径, 例如 `../`，`../bsp` 等；需要注意的是，设置了这个以后，对应的 copyfiles/incdirs/libdirs 的路径的根目录均受到影响，就会使用新设置的和这个路径

- **installdir**

希望代码安装的目录名称，仅限英文，满足 C 语言命名格式

- 针对 sdk 类型的 package, 会被安装到 `<sdk_installdir>` , 如果 installdir 未定义，默认为 SDK，如果没有任何 sdk 类型的 package 被引用，sdk_installdir 也被默认设置为 SDK
- 针对 csp 类型的 package, 会被安装到 `<sdk_installdir>/<csp_installdir>` 目录下，TBD
- 针对 ssp 类型的 package, 会被安装到 `<sdk_installdir>/SoC/<ssp_installdir>/Common` 下面
- 针对 bsp 类型的 package, 会被安装到 `<sdk_installdir>/SoC/<ssp_installdir>/Board/<bsp_installdir>` 下面，如果不依赖于任何 ssp 类型，则安装到 `<sdk_installdir>/BSP/<bsp_installdir>`
- 针对 osp 类型的 package, 会被安装到 `<sdk_installdir>/OS/<osp_installdir>` 下面
- 针对 app 类型的 package, 会被安装到 `<app_installdir>/` 目录下，如果 installdir 未定义，默认为 application
- 针对 mwp 类型的 package, 会被安装到 `<sdk_installdir>/Components/<mwp_installdir>` 目录下

> **Note**
>
> 2023.05.26 新增 `copyfiles/incdirs/libdirs` 均支持 `../../` 这样的相对上级目录，但是安装或者设置路径的时候，均设置到 `<installdir>` 下面
>
> 例如: `path: ["../common/"]` 就拷贝上一级目录的 `common` , 并放在 `<installdir>/common` 下面，
>
> 如果下面有 `common` 这个目录，则创建 `R1L_common` , 如果是 `../../common` , 则创建 `R2L_common` ，
>
> 这种方案不考虑了，直接创建同名目录，同名文件直接覆盖，建议采用 `srcroot: ..` 来解决问题对应的 `incdirs/libdirs`
>
> 如果遇到这种相对路径，也需要以最终安装到路径以及文件名为准

- **copyfiles**

待拷贝的文件或者文件夹，这里是指所有的目录或者文件，支持 `../`、`*`、`*.*`，给结合 srcroot 一起使用，

- **incdirs**

必填，加入头文件目录列表，Nuclei Studio 会进行补全，最终的路径是相对于工程根目录的路径。

- **libdirs**

可选，lib 库所在目录，Nuclei Studio 会进行补全，最终的路径是相对于工程根目录的路径。

- **ldlibs**

  可选的需要链接的库，Nuclei Studio 会进行补全，最终的路径是相对于工程根目录的路径。

Set Configuration for other packages

setconfig 用来设置 NPK 的其他选项，遵循覆盖规则 app > mwp > osp > bsp > ssp > csp。

setconfig 是一个对象组，可以无限扩展，每个对象中有三个字段来描述一个对象。

- **config**

  变量名，遵循 C++ 命名规范，一般变量 XXX, 在其它部分以 ${XXX} 的方式引用

  该变量名不唯一，可以通过条件进行判定生效，也遵循覆盖规则 app > mwp > osp > bsp > ssp > csp，进行自动覆盖。

- **value**

  变量的值

- **condition**

  变量的条件，只有条件生效时，该变量的该值才会生效

Build Configuration

设置工程的编译工具和编译选项的配置，它的关键字包含以下几个固定字段。

- **type**

  支持的编译工具的类型，值一般为 gcc、clang、common。目前只支持 gcc、clang 两种，因为编译选项的配置有一些是相同的，为了提高代码的复用性，我们又添加 common 类型。

- **description**

  对此编译工具的说明。

- **toolchain_name**

  重要字段，编译工具名字。

- **cross_prefix**

  重要字段，编译工具的前缀。

- **unflags**

  在 buildconfig section 中的 `common_flags/cflags/asmflags/ldflags/cxxflags` 中生效，用于删掉之前已经定义的 flags。

- **undefines**

  在 buildconfig section 中的 `common_defines/cdefines/asmdefines/cxxdefines` 中生效，用于删掉之前已经定义的 defines。（字符串完全匹配，则生效）

- **common_flags**

  用的编译选项，将会添加到 cflags, asmflags, cxxflags 上，留空表示没有任何选项。

- **ldflags**

  链接选项列表，留空表示没有任何选项。

- **linkscript**

  链接脚本的定义，必须在 bsp/ssp 中定义，留空表示没有任何选项。

- **cflags**

  C 编译选项，留空表示没有任何选项。

- **asmflags**

  ASM 编译选项，留空表示没有任何选项。

- **cxxflags**

  CXX 编译选项，留空表示没有任何选项。

- **common_defines**

  通用的宏定义，留空表示没有任何选项。

- **cdefines**

  C 的宏定义，留空表示没有任何选项。

- **asmdefines**

  ASM 的宏定义，留空表示没有任何选项。

- **cxxdefines**

  CXX 的宏定义，留空表示没有任何选项。

- **prebuild_steps**

  - **command**

    编译前执行的命令，留空表示没有任何选项。

  - **description**

    编译前执行的命令的说明，留空表示没有任何选项。

- **postbuild_steps**

  - **command**

    编译后执行的命令，留空表示没有任何选项。

  - **description**

    编译后执行的命令的说明，留空表示没有任何选项。

Debug Configuration

设置工程的 Debug 类型及相关参数的配置，它的关键字包含以下几个固定字段，可以不用配，如果配置了，在工程生成的时候，Nuclei Studio 会根据这里面的内容，生成了个 launch 文件，同时可以根据相关内容进行工程的 Debug。

- **type**

  Debug 类型, 目前支持 GDB Custom 、GDB SEGGER J-Link、GDB OpenOCD、GDB Nuclei QEMU、Nuclei RVProf

- **description**

  对支持的 Custom Jlink OpenOCD Qemu RVProf 的说明

- **configs**

  对应的 Debug 类型的参数，所有的参数，都是以 key-value 的方式出现，因为每中 Debug 类型所需参数不同，对应的情况也不同，更详细的说明如下。

```yaml
debugconfig:
  - type: openocd
    description: Nuclei OpenOCD
    configs:
      - key: XXXX
        value: xxxx
```

GDB Custom 的 Debug 参数

Table 2.1: Arguments of GDB Custom Debug

Name	Reset Value	Description
doStartGdbCLient	true	Start locally
doStartGdbServer	true	Start GDB session
gdbClientOtherCommands		gdb Client Other Commands
gdbClientOtherOptions		gdb Client Other Options
gdbMode	Commands	Supported types: Commands, General, and Dli
gdbServerConnectionAddress		gdb Server Connection Address
gdbServerExecutable		gdb Server Executable
serverCheckFlag	Started by GNU MCU Eclipse	server Check Flag
gdbServerGdbPortNumber	3333	gdb Server Gdb Port Number
gdbServerOther		Config options
DEBUG_NAME	${cross_prefix}gdb${cross_suffix}	Executable path
ipAddress	localhost	Host name or IP address
portNumber	3333	
UPDATE_THREADLIST_ON_SUSPEND	false	Force thread list update on suspend
otherInitCommands		Initialization Commands
loadImage	true	Load executable
imageFileName		use File For Image name
imageOffset		Executable offset (hex)
useFileForImage	false	Use file for Image
useProjBinaryForImage	true	
loadSymbols	true	Load symbols
symbolsFileName		
symbolsOffset		Symbols offset (hex)
useProjBinaryForSymbols	true	Use project binany
useFileForSymbols	false	Use file for Symbols
doDebugInRam	true	Debug in RAM
otherRunCommands		Run/Restart Commands
setPcRegister	false	Set program counter at (hex)
pcRegister		
setResume	false	
setStopAt	true	Set breakpoint at
stopAt	main	
doContinue	true	Continue
svdPath		svd file path

GDB SEGGER J-Link 的 Debug 参数

Table 2.2: Arguments of GDB SEGGER J-Link Debug

Name	Reset Value	Description
arch		Nuclei ARCH
executable		OpenOCD Executable Name
tool_extrapreopts		Additional value, prepended to Config options
tool_extraopts		Additional value, replacing Config options
tool_extrapostopts		Additional value, appended to Config options
gdb_extraprecmds		Additional value, prepended to Commands
gdb_extracmds		Additional value, replacing Commands
gdb_extrapostcmds		Additional value, appended to Commands
doStartGdbServer	true	Start the j-Link GDB server locally
doConnectToRunning	false	Connect to running target

continues on next page

Table 2.2 – continued from previous page

gdbServerExecutable		Executable path
doGdbServerAllocateConsole	true	Allocate console for the GDB server
doGdbServerInitRegs	true	do Gdb Server Init Regs
doGdbServerLocalOnly	true	do Gdb Server Local Only
doGdbServerSilent	false	do Gdb Server Silent
doGdbServerVerifyDownload	true	do Gdb Server Verify Download
doStartGdbServer	true	Start the j-Link GDB server locally
gdbClientOtherCommands	set mem inaccessible-by-default off	gdb Client Other Commands
gdbServerConnection	usb	gdb Server Connection
gdbServerConnectionAddress		gdb Server Connection Address
gdbServerDebugInterface	jtag	gdb Server Debug Interface
gdbServerDeviceEndianness	little	gdb Server Device Endianness
gdbServerDeviceName		gdb Server Device Name
gdbServerLog		gdb Server Log path
gdbServerGdbPortNumber	2331	gdb Server Gdb Port Number
gdbServerSwoPortNumber	2332	gdb Server SwoPort Number
gdbServerTelnetPortNumber	2333	gdb Server Telnet PortNumber
gdbServerOther		gdb ServerO ther
DEBUG_NAME	${cross_prefix}gdb${cross_suffix}	Executable path
gdbClientOtherOptions		gdb Client Other Options
ipAddress	localhost	
portNumber	2331	
gdbServerDeviceSpeed	auto	gdb Server Device Speed
doFirstReset	false	Initial Reset and Halt
firstResetType		
firstResetSpeed	1000	
enableFlashBreakpoints	true	Enable flash breakpoints
doGdbServerAllocateSemihostingConsole	true	Allocate console for semihosting and SWO
enableSemihosting	true	Enable semihosting console routed to
enableSemihostingIoclientTelnet	true	Telnet
enableSemihostingIoclientGdbClient	false	GDB client
enableSwo	true	Enable SWO
swoEnableTargetCpuFreq	0	SWO Cpu freq
swoEnableTargetSwoFreq	0	SWO freq
swoEnableTargetPortMask	0x1	SWO Port mask
otherInitCommands		other Init Commands
jtagDevice	GNU MCU J-Link	
loadImage	true	Load executable
imageFileName		
imageOffset		
useFileForImage	false	
useProjBinaryForImage	true	
loadSymbols	true	Load symbols
symbolsFileName		
symbolsOffset		
useFileForSymbols	false	
useProjBinaryForSymbols	true	
doDebugInRam	true	Debug in RAM
doSecondReset	true	Pre-run/Restart reset
secondResetType		Type (always executed at Restart)
otherRunCommands		
setPcRegister	false	Set program counter
pcRegister		Set program counter at (hex)
setStopAt	true	Set breakpoint
stopAt	main	Set breakpoint at
doContinue	true	Continue

continues on next page

Table 2.2 – continued from previous page

svdPath	svd file path

GDB OpenOCD 的 Debug 参数

Table 2.3: Arguments of GDB OpenOCD Debug

Name	Reset Value	Description
arch		Nuclei ARCH
executable		OpenOCD Executable Name
tool_extrapreopts		Additional value, prepended to Config options
tool_extraopts		Additional value, replacing Config options
tool_extrapostopts		Additional value, appended to Config options
gdb_extraprecmds		Additional value, prepended to Commands
gdb_extracmds		Additional value, replacing Commands
gdb_extrapostcmds		Additional value, appended to Commands
doStartGdbServer	true	Start OpenocD locally
gdbServerExecutable		Executable path
gdbServerGdbPortNumber	3333	GDB port
gdbServerTelnetPortNumber	4444	Telnet port
gdbServerTclPortNumber	6666	Tcl port
gdbClientOtherOptions		gdb Client Other Options
gdbServerOther		Config options
doGdbServerAllocateConsole	true	do GdbServer Allocate Console
doGdbServerAllocateTelnetConsole	false	do Gdb Server Allocate Telnet Console
gdbServerConnectionAddress		gdb Server Connection Address
doStartGdbCLient	true	do Start Gdb CLient
DEBUG_NAME	${cross_prefix}gdb${cross_suffix}	
gdbClientOtherCommands		Commands
ipAddress	localhost	Host name or lp address
portNumber	3333	Port number
UPDATE_THREADLIST_ON_SUSPEND	false	Force thread list update on suspend
doFirstReset	false	Initial Reset and Halt
firstResetType	init	
otherInitCommands		
enableSemihosting	false	Enable semihosting console routed to
loadImage	true	Load executable
useFileForImage	false	
imageFileName		
imageOffset		
symbolsFileName		
symbolsOffset		
loadSymbols	true	Load symbols
useFileForSymbols	false	
useProjBinaryForImage	true	
useProjBinaryForSymbols	true	
useRemoteTarget	true	
doDebugInRam	true	Debug in RAM
doSecondReset	true	Pre-run/Restart reset
secondResetType	halt	Type (always executed at Restart)
otherRunCommands		
setPcRegister	false	Set program counter
pcRegister		Set program counter at (hex)
setStopAt	true	Set breakpoint
stopAt	main	Set breakpoint at

continues on next page

Table 2.3 – continued from previous page

doContinue	true	Continue
svdPath		svd file path

GDB Nuclei QEMU 的参数

Table 2.4: Arguments of GDB Nuclei QEMU Debug

Name	Reset Value	Description
nuclei_archext		Nuclei archext
extra_board		Nuclei DevelopBoard
extra_cpu		Nuclei CPU Core
riscv_arch		Nuclei ARCH
executable		OpenOCD Executable Name
tool_extrapreopts		Additional value, prepended to Config options
tool_extraopts		Additional value, replacing Config options
tool_extrapostopts		Additional value, appended to Config options
gdb_extraprecmds		Additional value, prepended to Commands
gdb_extracmds		Additional value, replacing Commands
gdb_extrapostcmds		Additional value, appended to Commands
doStartGdbServer	true	Start OpenocD locally
gdbServerExecutable		Executable path
gdbMachineBit		Machine Bit
gdbServerBoardName		Board name
gdbCoreName		Nuclei RISC-V Core
gdbServerSMPCount	1	Nuclei SMP Count
gdbDownloadName		Download
gdbServerOther	-serial stdio -nodefaults -S	More options
otherExtensions		other Extensions
gdbServerGdbPortNumber	1234	GDB port
isGdbServerVerbose	false	Extra verbose
enableSemihosting	true	Enable Arm semihosting
disableGraphics	true	Do not open graphic windows
doGdbServerAllocateConsole	true	Allocate console for QEMU
DEBUG_NAME	${cross_prefix}gdb${cross_suffix}	Executable name
gdbClientOtherOptions		Other options
gdbClientOtherCommands		Commands
ipAddress	localhost	Host name or lp address
portNumber	1234	Port number
doFirstReset	false	Initial Reset and Halt
otherInitCommands		
loadSymbols	true	Load symbols
symbolsFileName		
symbolsOffset		
useFileForSymbols	false	
useProjBinaryForSymbols	true	
useRemoteTarget	true	
loadImage	true	Load executable
useFileForImage	false	
imageFileName		
imageOffset		
useProjBinaryForImage	true	
doDebugInRam	false	Debug in RAM
otherRunCommands		
doSecondReset	true	Pre-run/Restart reset

continues on next page

Table 2.4 – continued from previous page

setPcRegister	false	Set program counter
pcRegister		Set program counter at (hex)
setStopAt	true	Set breakpoint
stopAt	main	Set breakpoint at
doContinue	true	Continue
svdPath		SVD file path

Nuclei RVProf 的参数

Table 2.5: Arguments of Nuclei RVProf

Name	Reset Value	Description
cycleModelExecutable	${cyclemodel_path}/${cyclemodel_executable}	cycleModel Executable
cycleModelExecutableTimeOut	20	cycleModelExecutable TimeOut
cycleModelExecutableProcessorCores	4	cycleModel Executable Processor Cores
cycleModelOther		cycleModel Other
docycleModelAllocateConsole	true	
docycleModelAllocateTelnetConsole	false	
RVProfExecutable	${rvprof_path}/${rvprof_executable}	RVProf Executable
RVProfExecutableTimeOut	20	RVProf Executable TimeOut
RVProfOther		RVProf Other
RVProfPortNumber	5000	RVProfPort Number
doRVProfAllocateConsole	true	
doRVProfAllocateTelnetConsole	false	

Extended variable

environment 是应用于 tool 类型的 NPK 包中的配置，当用户想要通过 NPK 来共享一个工具（tools 类型，如 cycleModel），可以使用 environment 配置。定义 **environment** 后，Nuclei Studio 会自动生成几个全局变量，这些变量可以在其他的 NPK 中以 `${xxx-1.0.0-XXX}` 的方式引用。

Note

- 每个包存在一个包路径，引用为 npk 名称-版本号，例如 `${tool-cyclemodel-1.0.0}`
- 其他变量的引用为 npk 名称-版本号-变量名，例如 `${tool-cyclemodel-1.0.0-cyclemodel_path}`,`${tool-cyclemodel-1.0.0-cyclemodel_executable}`
- 当变量的 system 值为 true 时，会额外生成一个不带版本号的变量（取最高版本），例如 `${tool-cyclemodel-cyclemodel_executable}`

```yaml
name: tool-cyclemodel
owner: nuclei
os:
version: 1.0.0
description: Nuclei Tools cyclemodel
details: Nuclei Tools cyclemodel
type: tool
keywords:
```

(continues on next page)

(continued from previous page)

```
  - tool
  - cyclemodel
license: Apache-2.0
homepage:

## 扩展变量  tool-cyclemodel-1.0.0 与 tool-cyclemodel-1.0.0-proxy
environment:
  - key: cyclemodel_path
    value: bin
    description: cyclemodel path
    system: true
  - key: cyclemodel_executable
    value: bin/n300_best_config_cymodel_latest
    description: cyclemodel executable
    system: true


## 这是另一个 NPK 中的代码，演示了如何使用 tool-cyclemodel
debugconfig:
  - type: rvprof
    description: Nuclei RVProf
    configs:
      - key: ncycm_path
        value: ${tool-cyclemodel-1.0.0-cyclemodel_executable}
      - key: rvprof_path
        value: ${tool-rvprof-1.0.0-rvprof_executable}
```

templatemanage

内部使用的配置，这里不做详细说明。

2.4.3 NPK 中的 UI 组件

NPK 中提供了丰富的 UI 组件，这些组件的字段里面都会有 default, description, global 这些子字段，这些字段均具备含义。

default 表示默认值，description 表示该选项的含义，global 表示这个选项是否在工程创建时显示 (true)，或者仅仅内部传参使用 (false)。

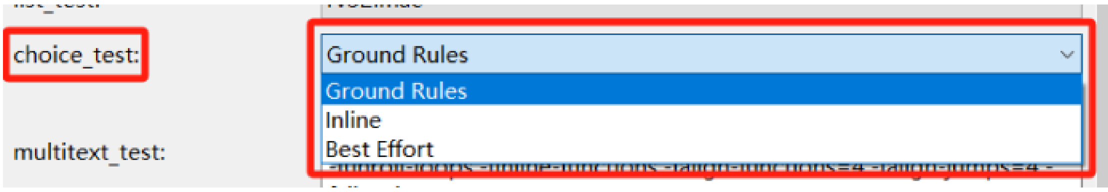
- **Choice 单项选择框**

```
choice_test:
  default_value: ground
  type: choice
  description: choice_test
  choices:
    - name: ground
      description: Ground Rules
      info:
        - name: app_commonflags
          value: >-
            -O3 -flto -fno-inline -funroll-loops -Wno-implicit -mexplicit-relocs
            -fno-builtin-printf -fno-common -falign-functions=4 -falign-jumps=4 -
↪falign-loops=4
    - name: inline
      description: Inline
      info:
        - name: app_commonflags
          value: >-
```

(continues on next page)

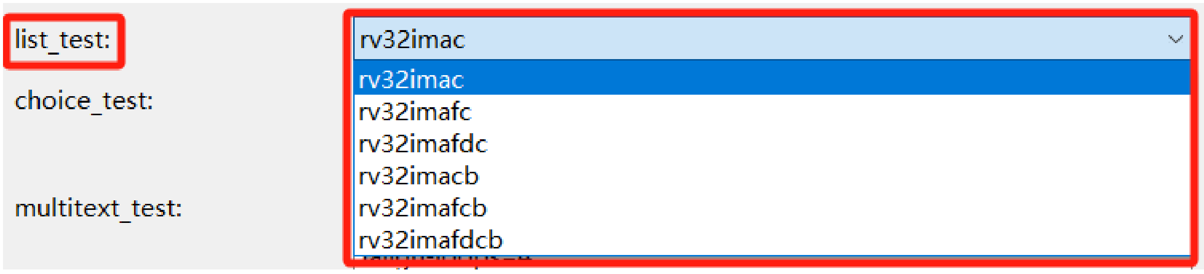
(continued from previous page)

```
            -O3 -flto -finline -funroll-loops -Wno-implicit -mexplicit-relocs -fno-
↪builtin-printf
            -fno-common -falign-functions=4 -falign-jumps=4 -falign-loops=4 -
↪finline-functions
    - name: best
      description: Best Effort
      info:
        - name: app_commonflags
          value: >-
            -Ofast -flto -fwhole-program -finline -funroll-loops -Wno-implicit -
↪mexplicit-relocs
            -fno-builtin-printf -fno-common -falign-functions=4 -falign-jumps=4 -
↪falign-loops=4
            -finline-functions
```



- **list** 单项选择框

```
list_test:
  default_value: rv32imac
  type: list
  global: true
  description: list_test
  value: >-
    [rv32imac,rv32imafc,rv32imafdc,rv32imacb,rv32imafcb,rv32imafdcb]
```



- **checkbox** 单项勾选框

```
checkbox_test:
  default_value: 0
  type: checkbox
  global: true
  description: checkbox_test
```

- **multicheckbox** 穿梭选择框

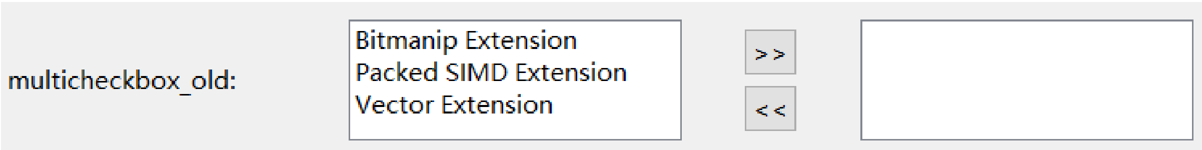
  下面提供 2 种写法

```
multicheckbox_old:
  default_value: []
```

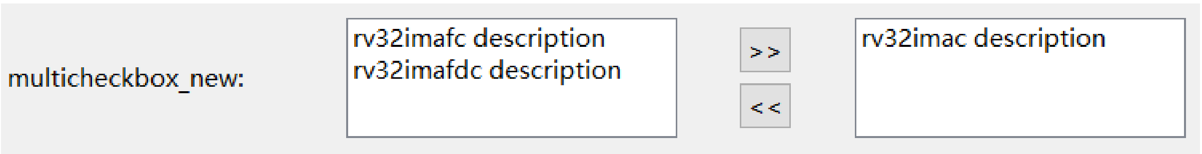
(continues on next page)

(continued from previous page)

```yaml
  type: multicheckbox
  global: true
  description: multicheckbox_old
  choices:
    - name: b
      description: Bitmanip Extension
    - name: p
      description: Packed SIMD Extension
    - name: v
      description: Vector Extension
```

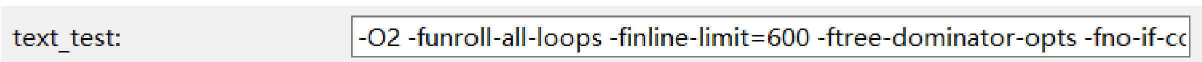



```yaml
multicheckbox_new:
  default_value: rv32imac
  type: multicheckbox
  global: true
  description: multicheckbox_new
  param:
    name: ["rv32imac","rv32imafc","rv32imafdc"]
    description: ["${name} description","${name} description","${name} description"]
```



- **text** 单行文本框

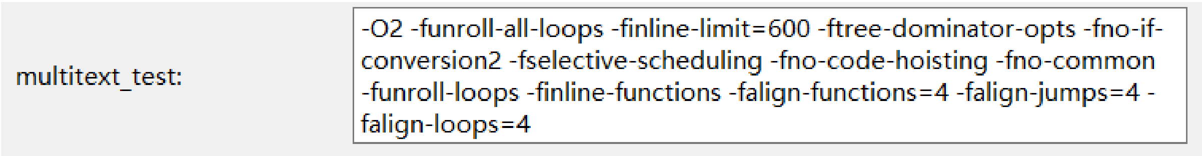
```yaml
text_test:
  value: >-
    -O2 -funroll-all-loops -finline-limit=600 -ftree-dominator-opts
    -fno-if-conversion2 -fselective-scheduling -fno-code-hoisting
    -fno-common -funroll-loops -finline-functions -falign-functions=4
    -falign-jumps=4 -falign-loops=4
  type: text
  description: text_test
```



- **multitext** 多行文本框

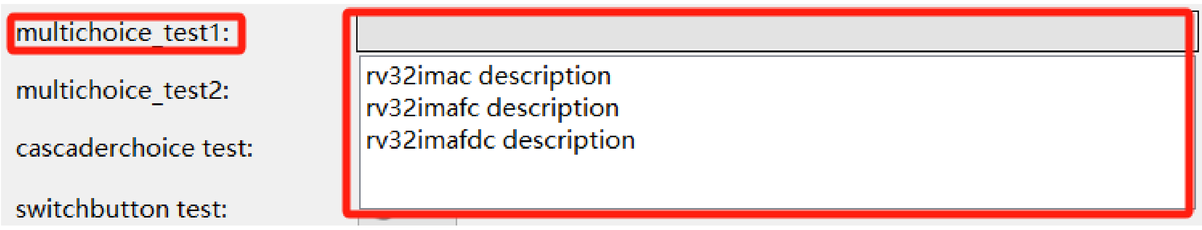
```yaml
multitext_test:
  value: >-
    -O2 -funroll-all-loops -finline-limit=600 -ftree-dominator-opts
    -fno-if-conversion2 -fselective-scheduling -fno-code-hoisting
    -fno-common -funroll-loops -finline-functions -falign-functions=4
    -falign-jumps=4 -falign-loops=4
  type: multitext
  description: multitext_test
```

- **multichoice** 多选下拉框

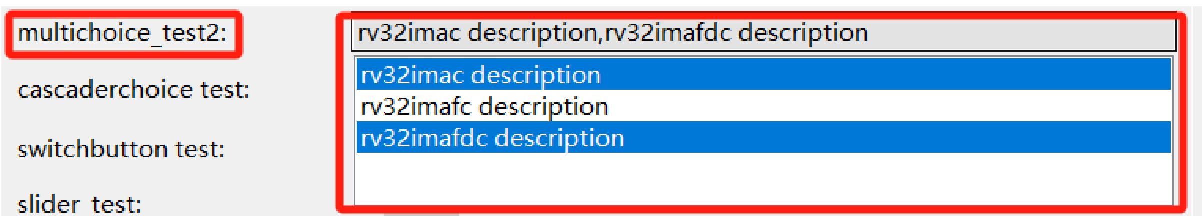


下面提供 2 种写法

```
multichoice_test1:
  default_value: []
  type: multichoice
  global: true
  description: multichoice_test1
  param:
    name: ["rv32imac","rv32imafc","rv32imafdc"]
    description: ["${name} description","${name} description","${name} description"]
```




```
multichoice_test2:
  default_value: >-
    [rv32imac,rv32imafdc]
  type: multichoice
  global: true
  description: multichoice_test2
  choices:
    - name: rv32imac      #
      description: ${name} description
    - name: rv32imafc
      description: ${name} description
    - name: rv32imafdc
      description: ${name} description
```



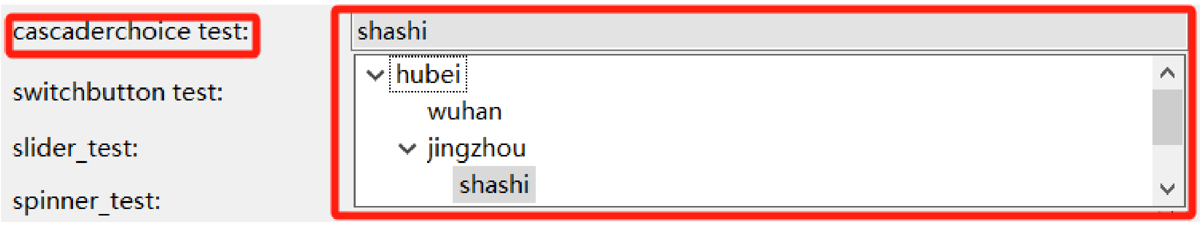
- **cascaderchoice** 级联选择框

```
cascaderchoice_test:
  default_value: >-
    [hubei,jingzhou,shashi]
  type: cascaderchoice
  global: true
  description: cascaderchoice test
  cascader_param:
    - hubei:
        - wuhan
```

(continues on next page)

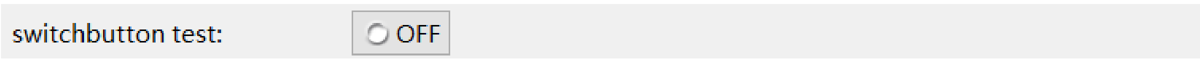
(continued from previous page)

```yaml
        - jingzhou:
            - shashi
            - jianli
    - hunan:
        - changsha
        - guangdong
```



- **switchbutton** 开关

```yaml
switchbutton_test:
  default_value: 0
  type: switchbutton
  global: true
  description: switchbutton test
```



- **slider** 数字选择框

```yaml
slider_test:
  default_value: 0
  type: slider
  description: slider_test
  param:
    range: >-
      [0,100,1]
```

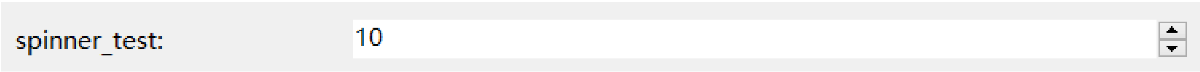
- **spinner** 数字选择框

```yaml
spinner_test:
  default_value: 10
  type: spinner
  description: spinner_test
  param:
    range: >-
      [-100,100,2]
```

- **multispinner** 多数字选择框

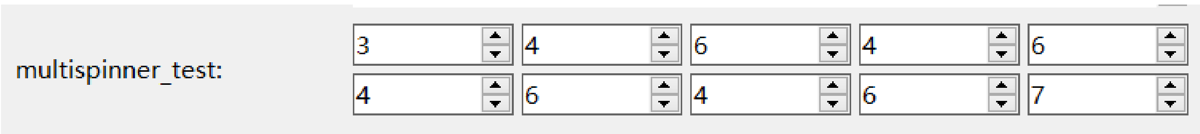
```yaml
multispinner_test:
  default_value: >-
    [3,4,6,4,6,4,6,4,6,7]
  type: multispinner
  global: true
  description: multispinner_test
```

(continues on next page)



(continued from previous page)

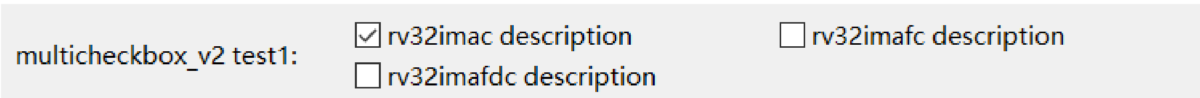
```
  param:
    range: >-
      [-100,100,1],[-100,100,2],[-100,100,3],[-100,100,3],[-100,100,3],[-100,100,3],[-100,100,3],[-100,100,3],[-100,100,3],[-100,100,4]
```



- **multicheckbox_v2** 多项勾选框

  下面提供 2 种写法

```
multicheckbox_v2_test1:
default_value: >-
  [rv32imac]
type: multicheckbox_v2
global: true
description: multicheckbox_v2 test1
param:
  name: ["rv32imac","rv32imafc","rv32imafdc"]
  description: ["${name} description","${name} description","${name} description"]
```

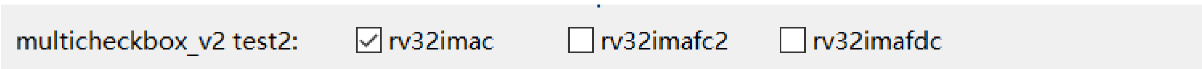


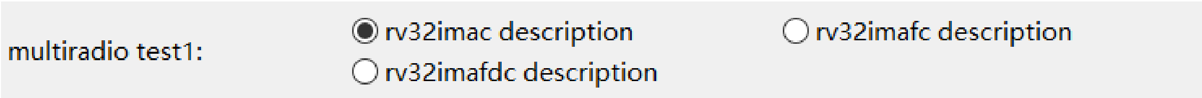

```
multicheckbox_v2_test2:
  default_value: >-
    [rv32imac]
  type: multicheckbox_v2
  global: true
  description: multicheckbox_v2 test2
  choices:
    - name: rv32imac
      description: rv32imac
    - name: rv32imafc
      description: rv32imafc2
    - name: rv32imafdc
      description: rv32imafdc
```

- **multiradio** 单选框

  下面提供 2 种写法

```
multiradio_test1:
  default_value: rv32imac
  type: multiradio
  global: true
  description: multiradio test1
  param:
    name: ["rv32imac","rv32imafc","rv32imafdc"]
    description: ["${name} description","${name} description","${name} description"]
```






```yaml
multiradio_test2:
  default_value: rv32imac
  type: multiradio
  global: true
  description: multiradio test2
  choices:
    - name: rv32imac
      description: rv32imac
    - name: rv32imafc
      description: rv32imafc2
    - name: rv32imafdc
      description: rv32imafdc
```

2.4.4 NPK 的语法

YAML 语言

NPK 的描述文件 npk.yml，是以 YAML 语言来编写，所以它支持标准的 YAML 语法，获取更多 YAML 相关的信息，可以参考：https://yaml.org/

变量定义

NPK 的描述语言中，允许用户自义定一个变量，并在有依赖关系的 NPK 中的任意位置使用

> **Note**
>
> - 例子 NPK 中定义了一个变量 app_commonflags，在与该 NPK 有依赖关系的任意 npk.yml 文件内的任意位置，我们可以通过 `${app_commonflags}` 来使用 app_commonflags 的值。
> - 例子 NPK 中定义了一个 list 对象 nuclei_core, 在与该 NPK 有依赖关系的任意 npk.yml 文件内的任意位置，我们可以通过 `${nuclei_core.arch}` 来使用 nuclei_core 对象中的 arch 值；通过 `${nuclei_core.abi}` 来使用 nuclei_core 对象中的 abi 值；通过 `${nuclei_core.tune}` 来使用 nuclei_core 对象中的 tune 值。

```yaml
configuration:
    app_commonflags:
        value:
        type: text
        description: Application Compile Flags

    nuclei_core:
        default_value: n201
        type: choice
        global: true
        description: Nuclei RISC-V Core
```

(continues on next page)

(continued from previous page)

```
        choices:
          - name: n200
            arch: rv32imc
            abi: ilp32
            cmodel: medlow
            tune: nuclei-200-series
            description: N200 Core(ARCH=rv32imc, ABI=ilp32)
          - name: n201
            arch: rv32iac
            abi: ilp32
            cmodel: medlow
            tune: nuclei-200-series
            description: N201 Core(ARCH=rv32iac, ABI=ilp32)
## Set Configuration for other packages
setconfig:
  - config: nmsislibarch
    value: ${nuclei_core.arch}
## Build Configuration
buildconfig:
  - type: gcc
    common_flags: # flags need to be combined together across all packages
      - flags: ${app_commonflags}
```

关键字

为了更好的描述 NPK，我们定义了一些字段，以描述出各种关系，其中大部分字段如其字面意义，这里重点介绍以下几个关键字。

- **condition**

condition 在 npk.yml 中，使用很频繁，是自定义的一个关键字，用来处理逻辑关系，类似 **if**，具体的使用如下。

```
ldflags:
  - flags: --specs=nosys.specs
    condition: $( ${stdclib} == "newlib_full" )
  - flags: --specs=nano.specs --specs=nosys.specs -u _printf_float -u _scanf_float
    condition: $( ${stdclib} == "newlib_fast" )
  - flags: --specs=nano.specs --specs=nosys.specs -u _printf_float
    condition: $( ${stdclib} == "newlib_small" )
  - flags: --specs=nano.specs --specs=nosys.specs
    condition: $( ${stdclib} == "newlib_nano" )
  - flags: --specs=${stdclib}.specs
    condition: $( startswith(${stdclib}, "libncrt") )

# 上述描术中 flags 的值，由 condition 决定，当在不同的场景时，flags 的值会不同，
# 又因为 flags 是一个数组类型，所以上述例子中 flags 会有多个值，最终使用是，是 flags 的值拼接成的字符串。
```

- **dependencies**

dependencies 在 npk.yml 中，用来描述 NPK 的依赖关系。

在很多的时候，NPK 需要依赖特定 owner 的某个 name，某个 version 版本的包，查找规则为 `owner/name:version`，如果 owner 未定义，则默认为该 npk 文件中定义的 owner，如果 **version** 未定义，则优先在同组件包查找，否则取最新的包。如果所依赖的包找不到，则该 NPK 将无法使用。

> **Note**
>
> 例子中 NPK 依赖了三个 npk，如下：

- sdk-nuclei_sdk owner、version 未定义，则优先在同组件包查找，否则取最新的包
- tool-testmodel 明确了 owner 和 version
- tool-rvprof 明确了 owner 和 version

```yaml
## Package Dependency
dependencies:
  - name: sdk-nuclei_sdk
    version:
    owner:
  - name: tool-testmodel
    version: 1.0.0
    owner: nuclei
  - name: tool-rvprof
    version: 1.0.0
    owner: nuclei
```

自定函数

在 NPK（npk.yml）中，为了更好地满足各种不同的需求，我们特意定义了一些常用的函数。

- **upper**

  将字符串变大写

```
${linker_script} = "test"
$(upper("${linker_script}CD")) => TESTCD
```

- **lower**

  将字符串变小写

```
${linker_script} = "test"
$(lower("${linker_script}cd")) => testcd
```

- **contains**

  判断字符串中是否包含另一个字符串

```
${linker_script} = "test"
$(contains(${linker_script}, nmsis) ) => false
```

- **join**

  将字符串连接数组

```
$(join([a,b,c,v], '') => abcv
```

- **concat**

  连接字符串成为新一字符串

```
${linker_script} = "test"
$(concat(${linker_script}, v) => testv
```

- **strip**

  去掉字符串两端空格

```
${linker_script} = "   test   "
$(strip(${linker_script})) => test
```

- **startswith**

  判断字符串是否以 xxx 开头

```
${linker_script} = "testabcd"
$(startswith(${linker_script}, test)) => true
```

- **endswith**

  判断字符串是否以 xxx 结尾

```
${linker_script} = "testabcd"
$(endswith(${linker_script}, test))  => false
```

- **arithop**

  数学运算符, 支持 +、-、*、/、%、?(三元运算) 等常用运算符，不支持 ++、

```
$(arithop(${linker_script}+22) > 1000)
$(arithop(${linker_script}+22))
$(arithop(${linker_script}>22?1:0))
```

- **npack/npack_installdir**

  npack 是否包含指定的 npk

  npack_installdir 包含的 npk 的路径

```
# a.yml
    name: mwp-a
    owner: nuclei
    copyfiles:
        - path: ["common", "abc.ld", "src/openocd.cfg", "inc"]

# b.yml

    name: mwp-b
    owner: nuclei
    copyfiles:
        - path: ["common", "111.ld", "src", "inc"]
    debugconfig:
      - type: openocd
        description: Nuclei OpenOCD
        configs:
          - key: config
            value: "$(npack_installdir(mwp-a))/src/openocd.cfg"
            condition: $( npack(nuclei:mwp-a) )

# 这段描述，是当 b.yml 如果依赖 a.yml 时，就可以将 config 的值，设置为 a.yml 的目录下的/src/
↪openocd.cfg
```

- **list_get**

  获取数组元素中指定脚标的值

```
${nuclei_cache}=[ic,dc,ccm]
$(list_get(${nuclei_cache},0)) -> ic
```

- **list_set**

  修改数组元素中指定脚标的值

```
${nuclei_cache}=[ic,dc,ccm]
$(list_set(${nuclei_cache},1,aa)) -> [ic,aa,ccm]
```

- **list_del**

删除数组元素中指定脚标的值

```
${nuclei_cache}=[ic,dc,ccm]
$(list_del(${nuclei_cache},1)) -> [ic,ccm]
```

- **list_add**

  在数组元素指定脚标插入值

```
${nuclei_cache}=[ic,dc,ccm]
$(list_add(${nuclei_cache},2,aa)) -> [ic,dc,aa,ccm]
```

- **list_size**

  获取数组元素 list 的长度

```
${nuclei_cache}=[ic,dc,ccm]
$(list_size(${nuclei_cache})) -> 3
```

- **list_sub**

  从 list 中指定的位置开始，截取指定长度的 list

```
${nuclei_cache}=[ic,dc,ccm]
$(list_sub(${nuclei_cache},1,2)) -> [dc]
$(list_sub(${nuclei_cache},0,2)) -> [ic,dc]
$(list_sub(${nuclei_cache},1,)) -> [dc,ccm]
$(list_sub(${nuclei_cache},,2)) -> [ic,dc]
```

- **subst**

  对字符串内部的指定字符串进行替换，第三个参数可为空

```
subst(libncrt_small,lib,) ==> ncrt_small
subst(libncrt_small,lib,ext) ==> extncrt_small
```

2.5 Nuclei Studio NPK 创建与共享

Nuclei Studio 2022.04 版中，提供了一个非常重要的功能，该功能主要通过提供的各种初始模板，方便开发者去创建自己的 NPK 组件包，并可以通过平台将自己的 NPK 组件包贡献出来，供其他开发者使用。

2.5.1 开发 NPK 组件包

认证开发者

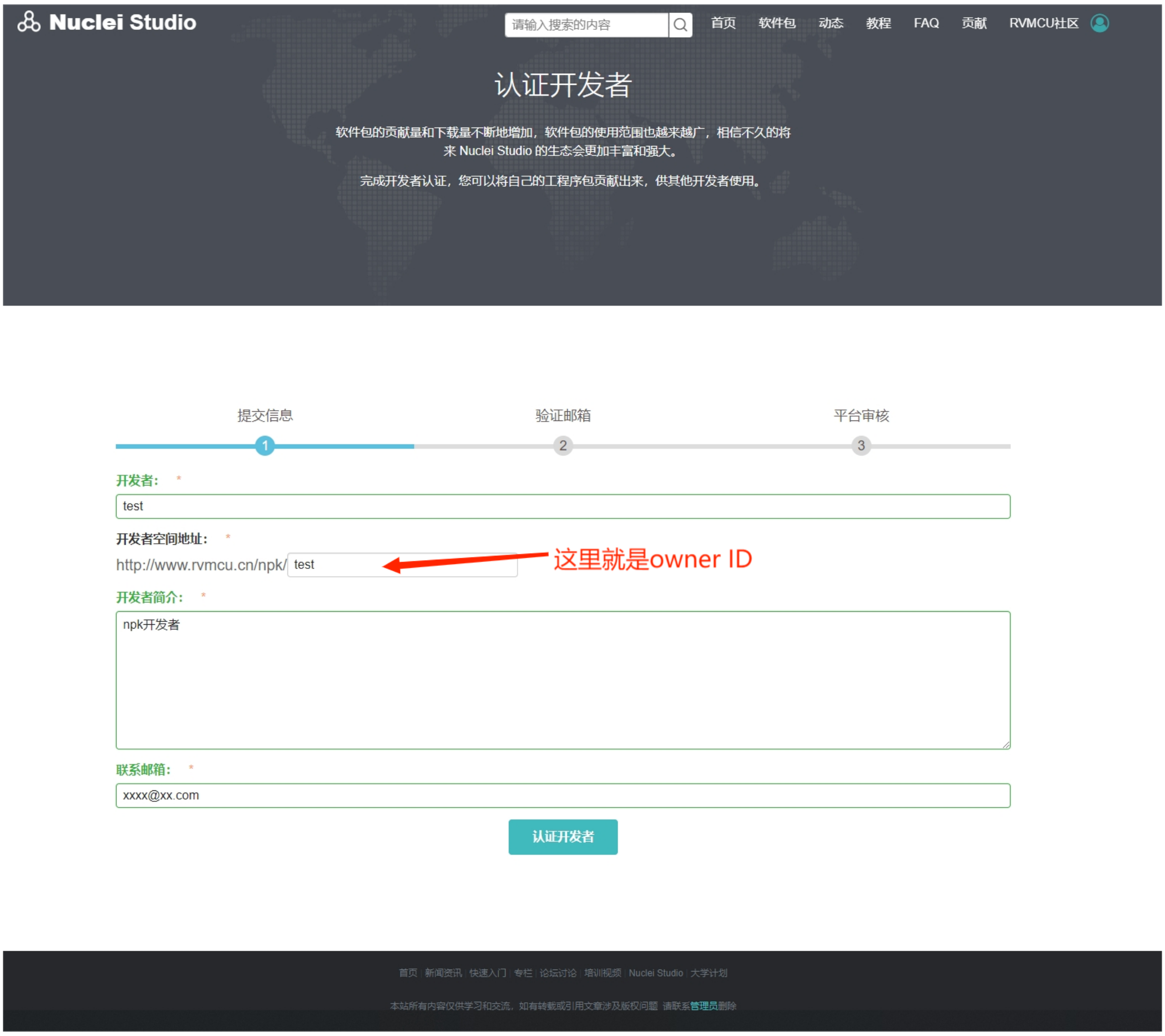
在创建 NPK 组件包前，先需要认证一个开发者帐号，获取一个唯一的 owner ID, 这在 owner ID 对应的就是前文中介绍的 owner，它能方便我们对 NPK 进行管理。

首先，登陆芯来科技的 RVMCU 社区的网站[2]，登陆成功后，依次进入 Nuclei Studio-> 贡献页面，就可以看到认证按钮。

[2] https://www.rvmcu.com/user-login.html

或者直接访问 认证地址[3] ，按提示分别填写相关信息，其中 开发者对应的开发者空间地址的后缀将会是您的 owner ID 。

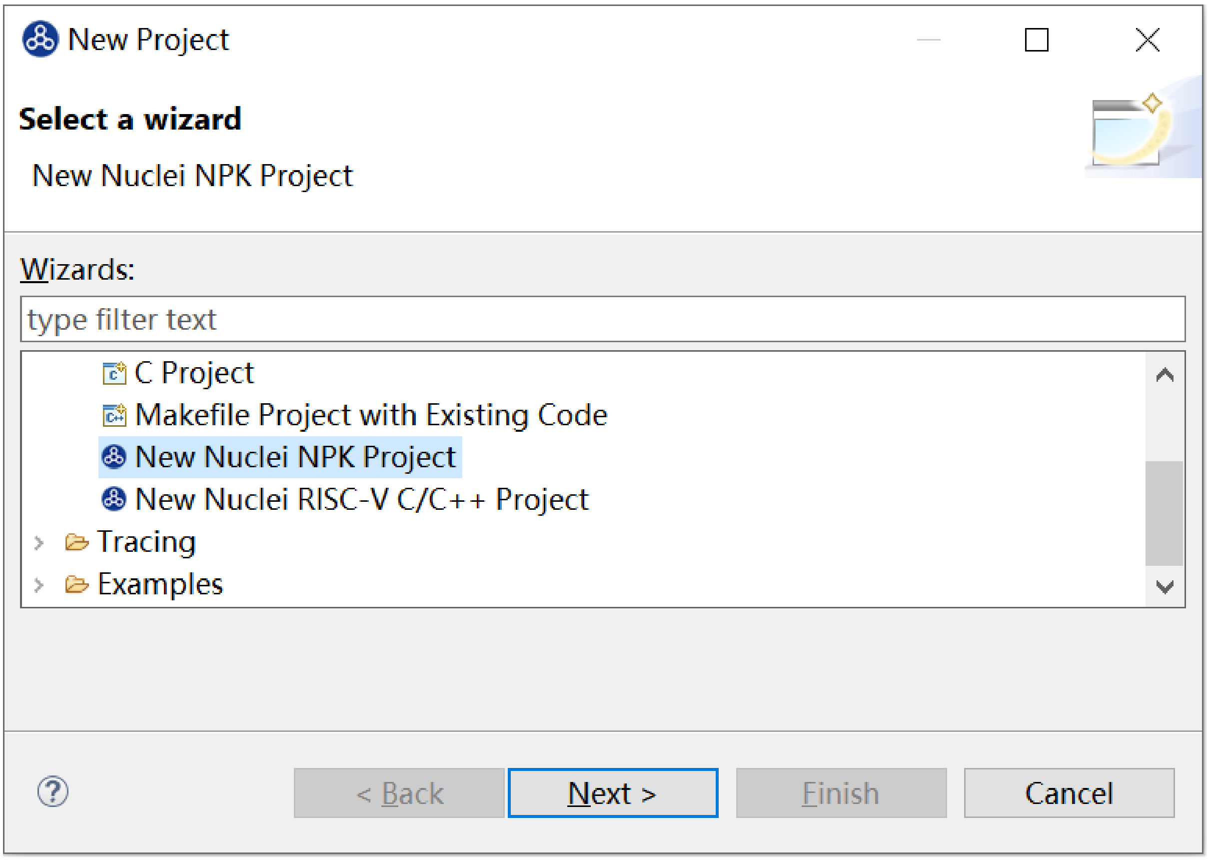
[3] https://www.rvmcu.com/nucleistudio-developer.html



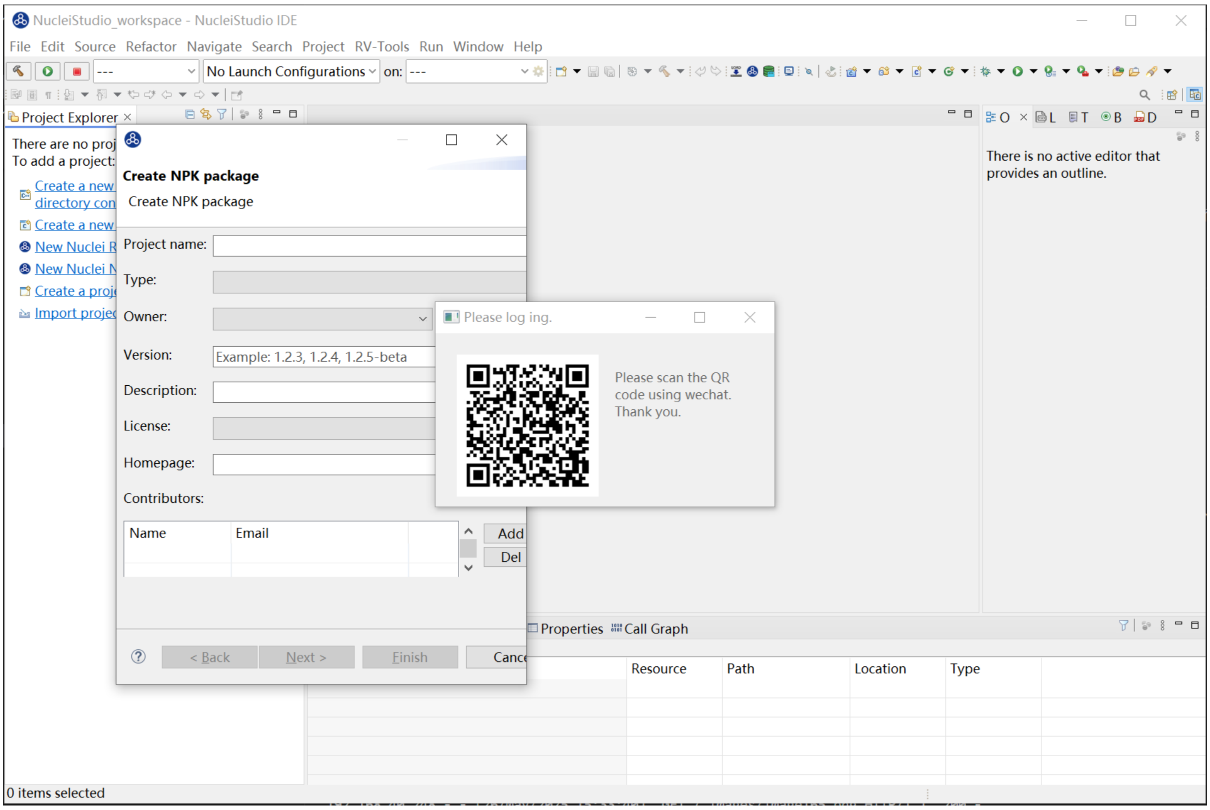
填写完相关内容后提交，系统审核通过后，即可完成认证。

创建 NPK 组件包

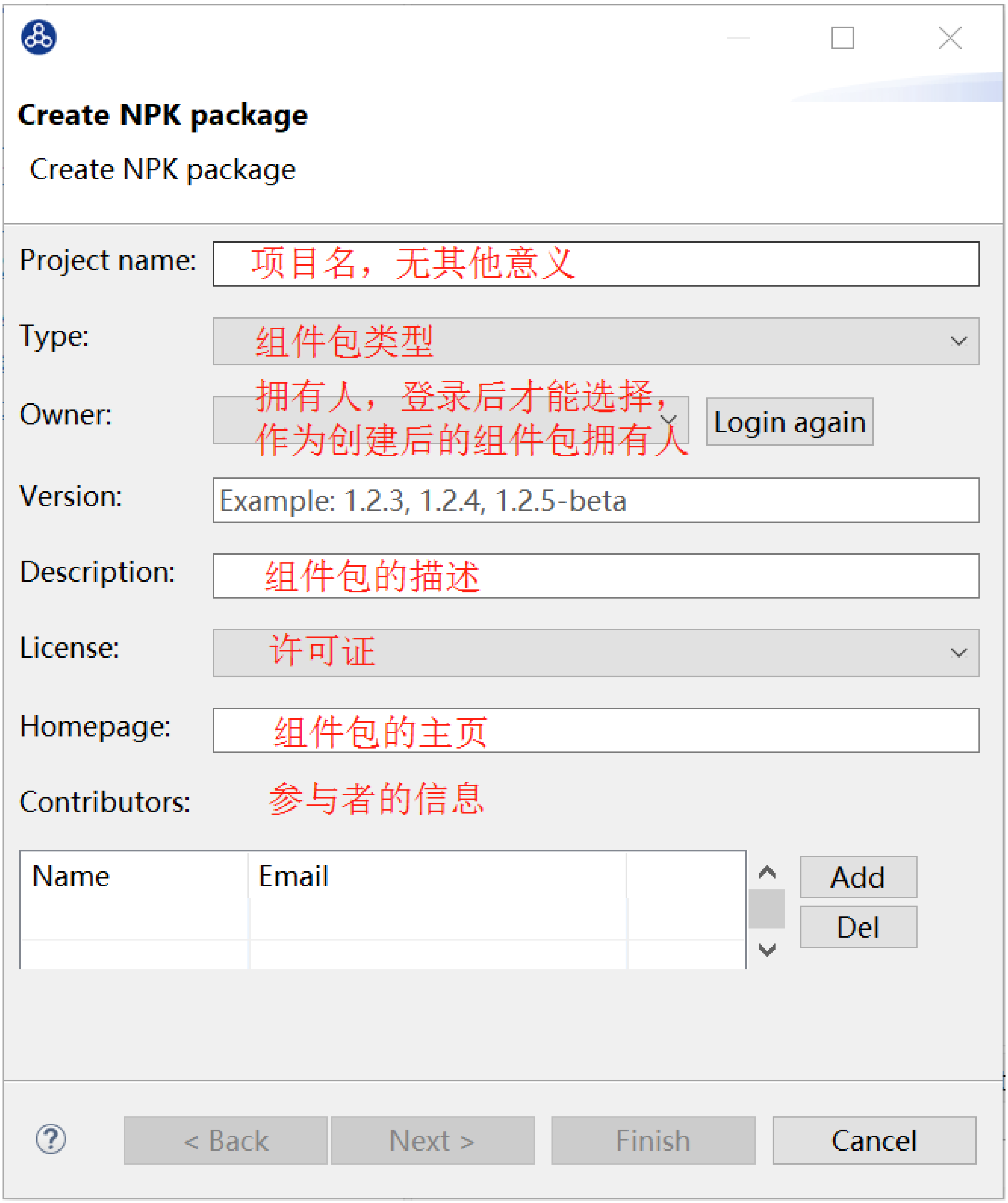
完成开发者认证后，在 Nuclei Studio 中新建一个 Nuclei Studio 组件工程, 在菜单栏中，选择 `File-> New-> Project-> C/C++-> New Nuclei NPK Project` 。



首次使用需要登录开发者帐号，且登录后 7 天有效，超过时间后需要再次登录，登录需要使用微信扫描二维码，见下图。登录成功后，owner 可选框会列出绑定的相关开发者账号，其他的根据提示输入相关信息。需要变更账号时，可点击 login again 重新登录。



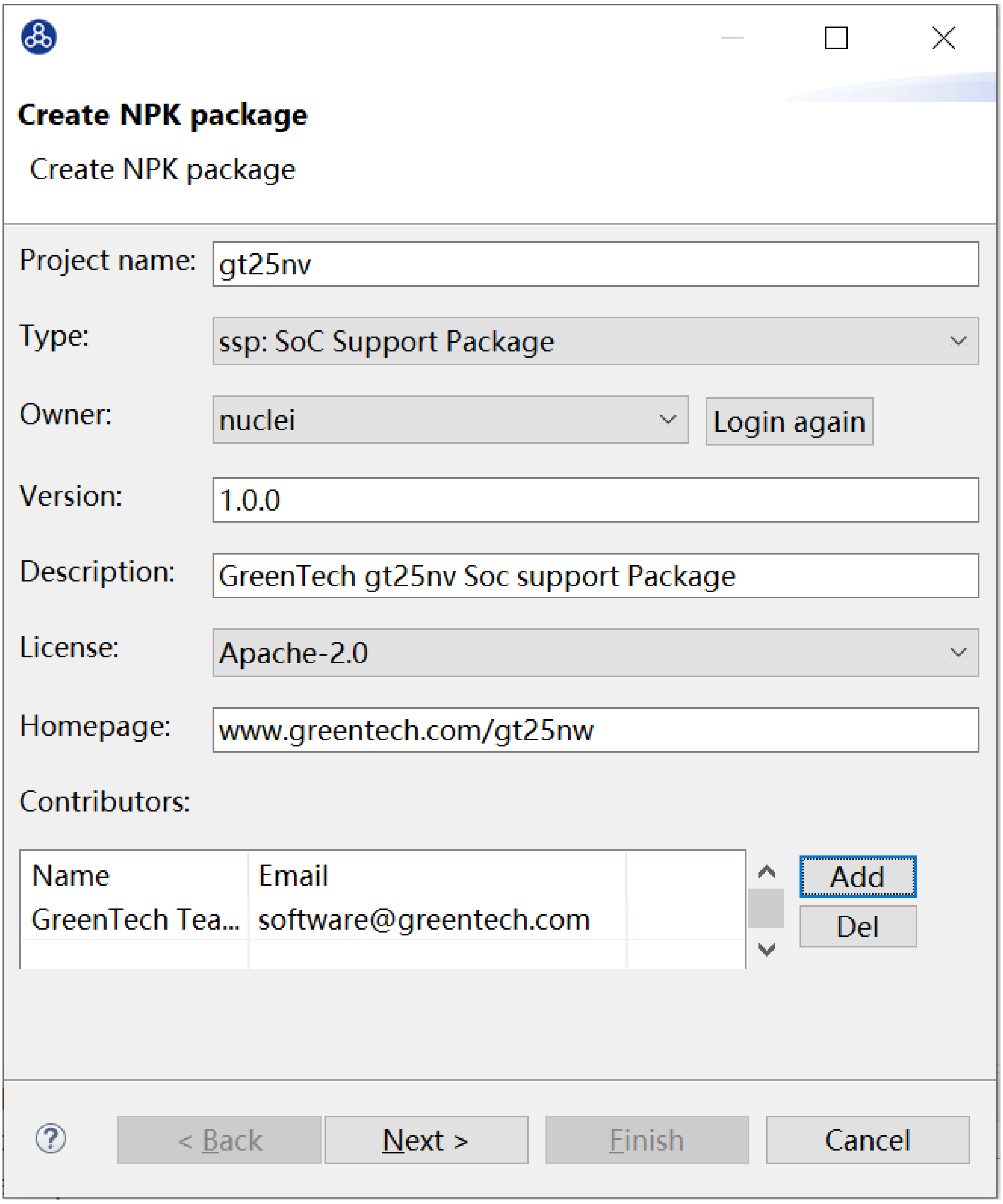
开发者帐号登陆成功后，依据工程向导依次填入工程名、工程类型等等相关信息。



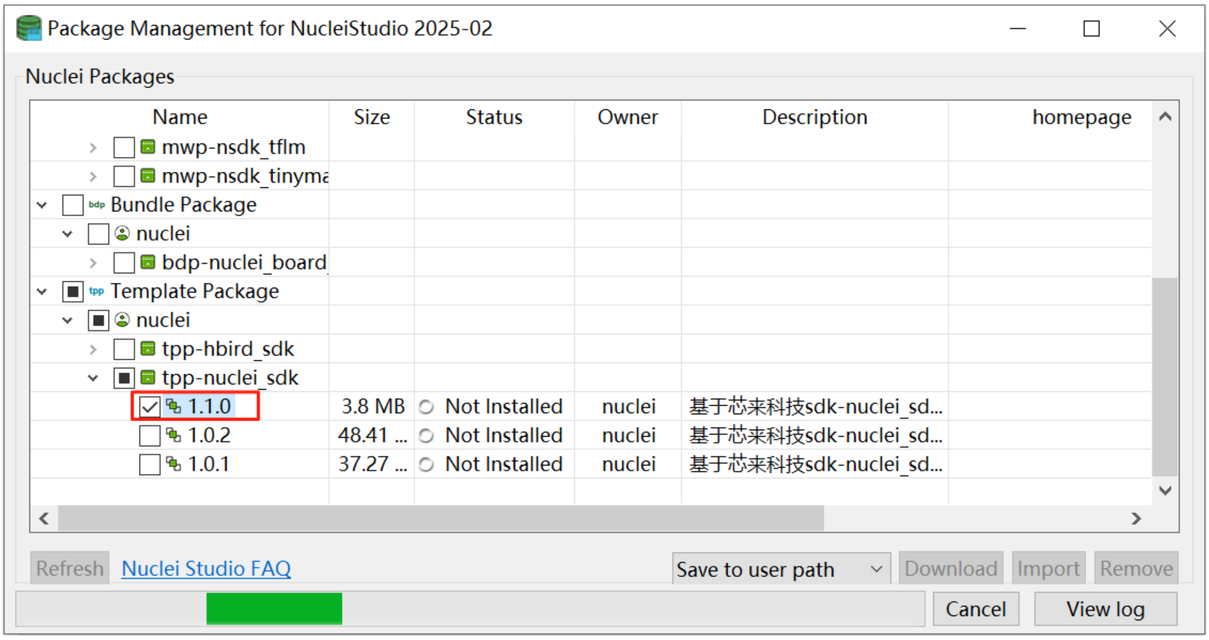
选择 Type 时，无对应模板时，会跳出对应提示，点击确定，进入 Nuclei Package Management 页面，根据需要下载 Template Package 的对应模板，或点击下图右下角 Import 自行导入。

下面以 ssp 类型模板为例，假设你的公司名称为 GreenTech，你的 SoC 名称为 gt25nv，适配的开发板为 gt25nv_fpga_eval，采用了我们的 n307FD 处理器 (rv32imafdc) 配置，并且配置了 dsp 特性，并且提供了 ilm, flash,flashxip 三种下载方式。

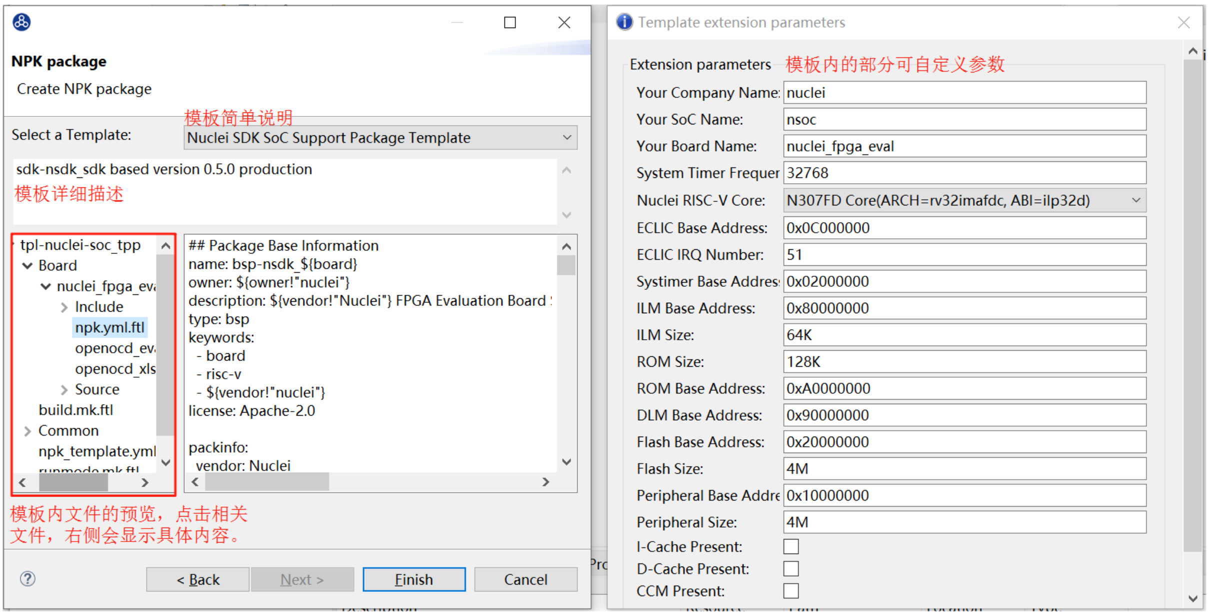
Type 选择 ssp: `Soc Support Package`



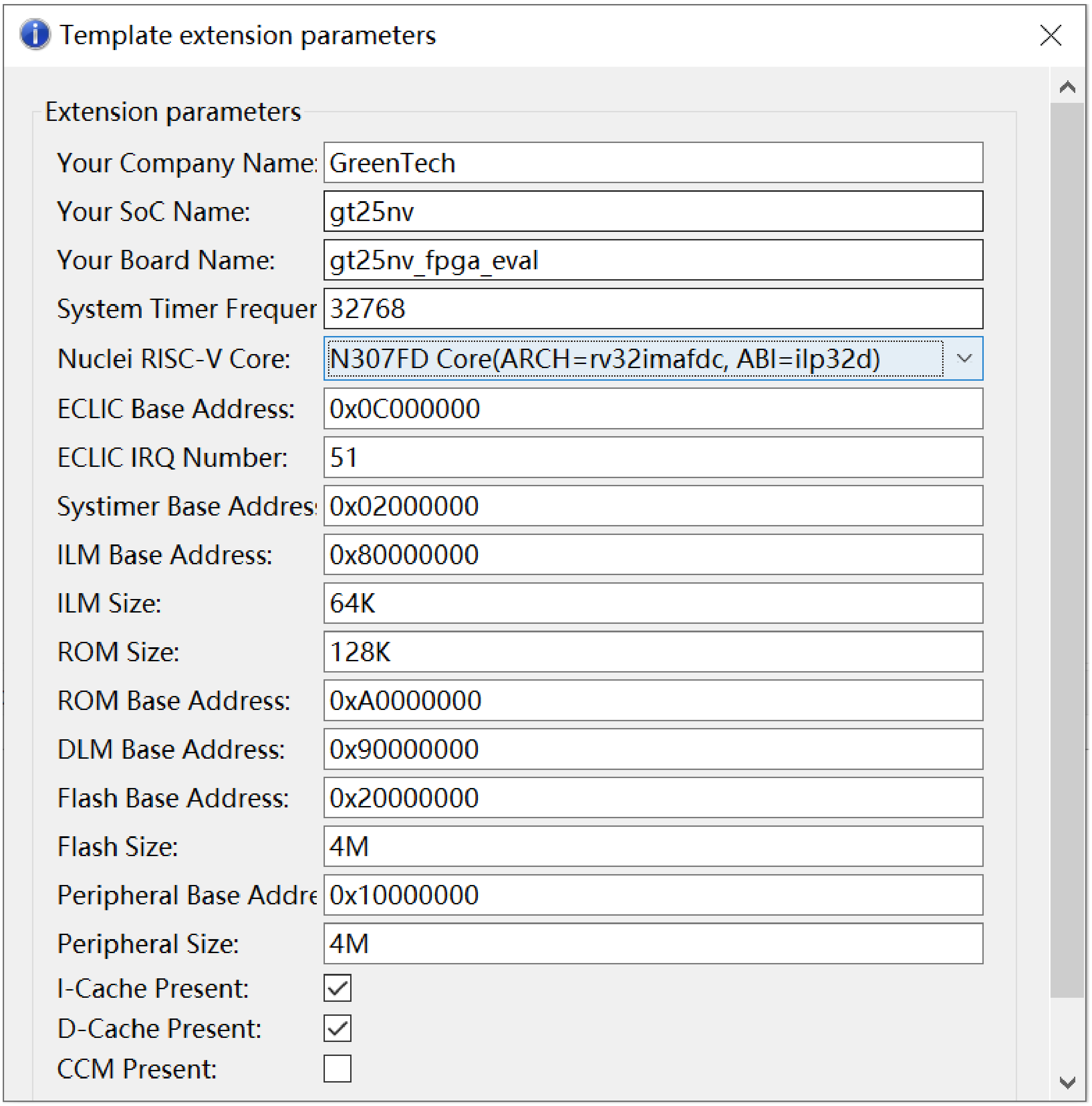
在选择 Type 时，如果没有相关的模板，会进入 Nuclei Package Management 页面，在该页面选择 tpp-nuclei_sdk 的 1.1.0 版本，点击 Download 下载，下载完成后关闭该页面



点击 Next，在 `Select a Template` 中选择刚才下载的模板 `tpp-nuclei_sdk`，左侧为模板描述和相关的文件预览，右侧为模板中部分可自定义的内容。

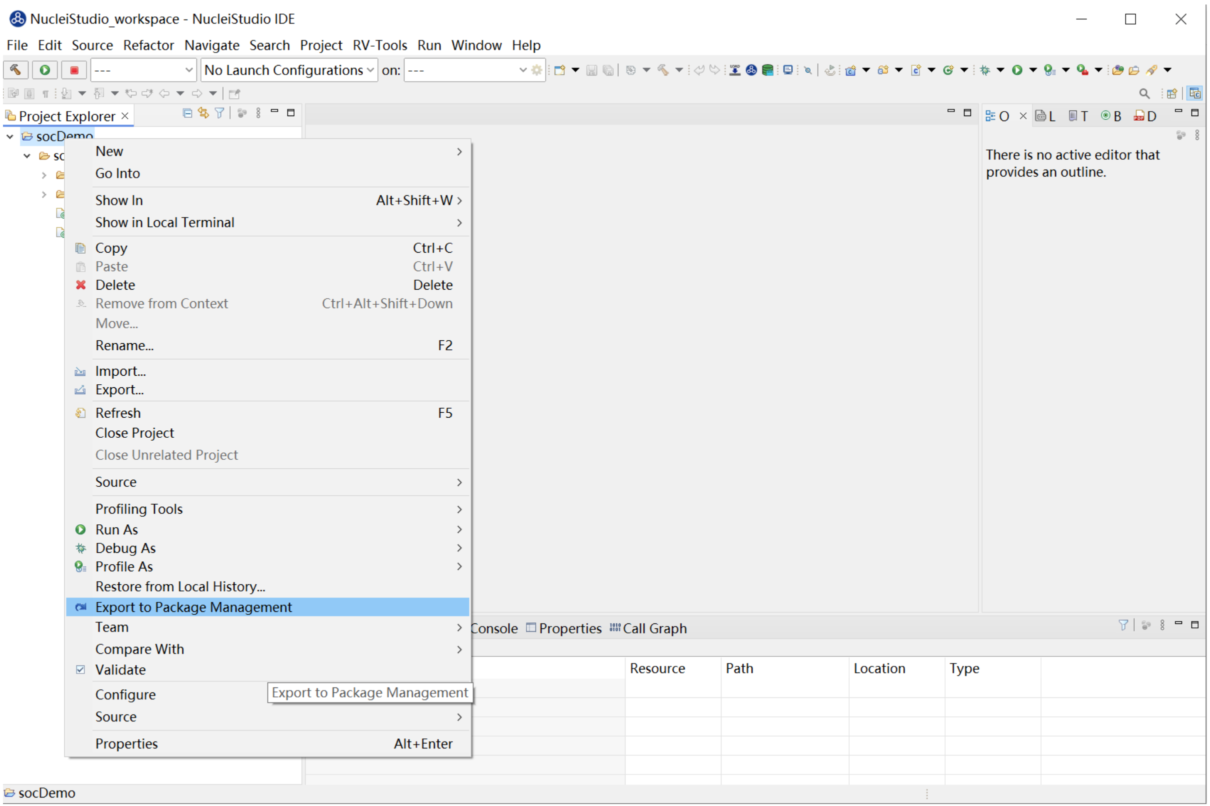


我们这里举例，公司名称为 `GreenTech`，SoC 名称为 `gt25nv`，适配的开发板为 `gt25nv_fpga_eval`，采用了我们的 n307FD 处理器（`rv32imafdc`）配置，并且配置了 dsp 特性，并且提供了 ilm，flash，flashxip 三种下载方式。经过修改后如图。

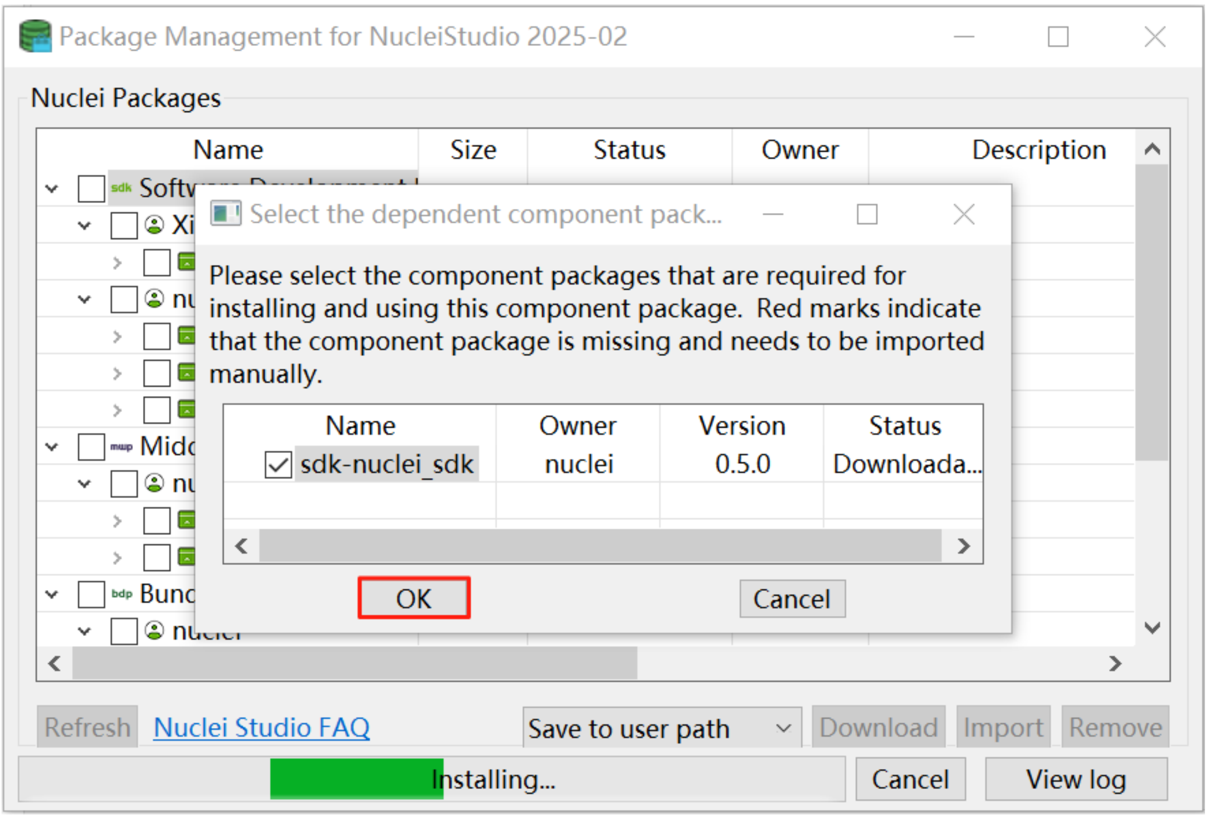


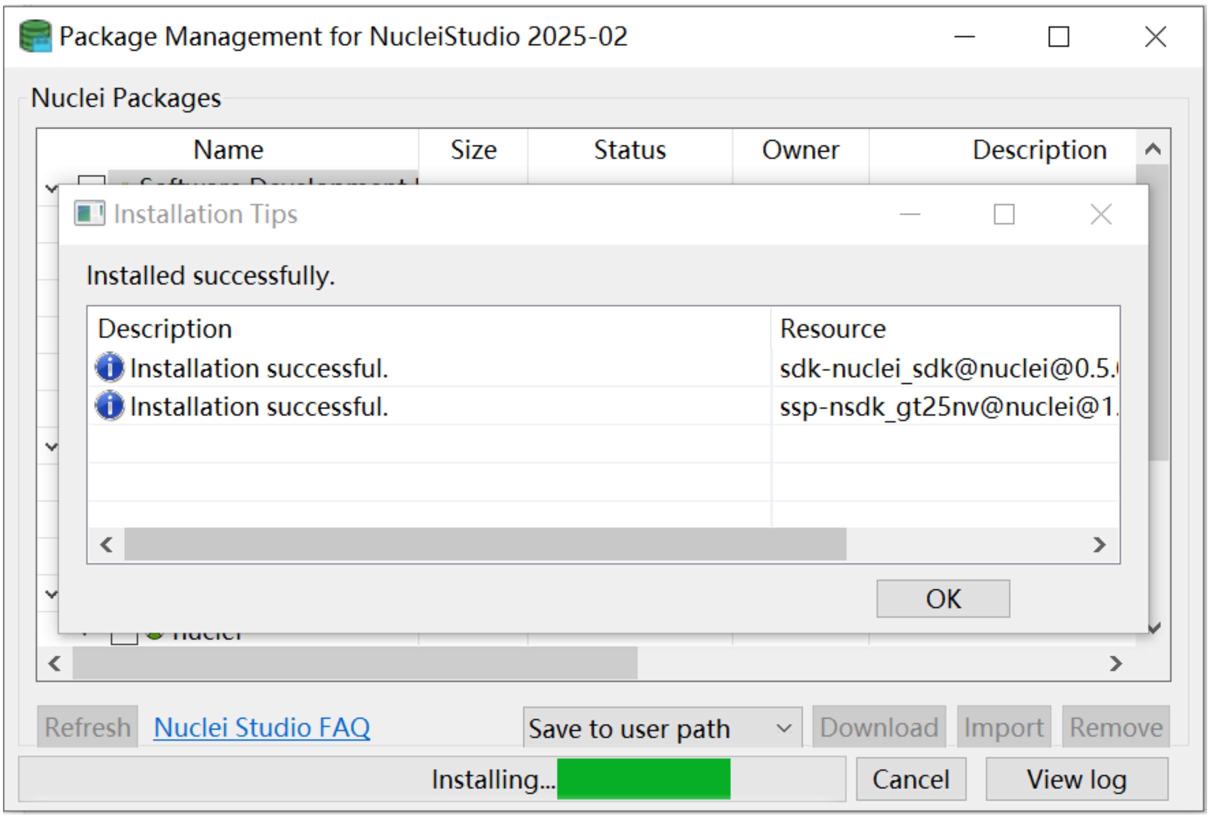
然后点击 Finish 即成功生成对应 NPK 组件包，再根据需要，可以打开目录查看并修改对应的文件和结构，修改完成后再导入该 NPK 组件包。

修改项目，根据需要自行修改。项目修改完成，导入 NPK 组件包。

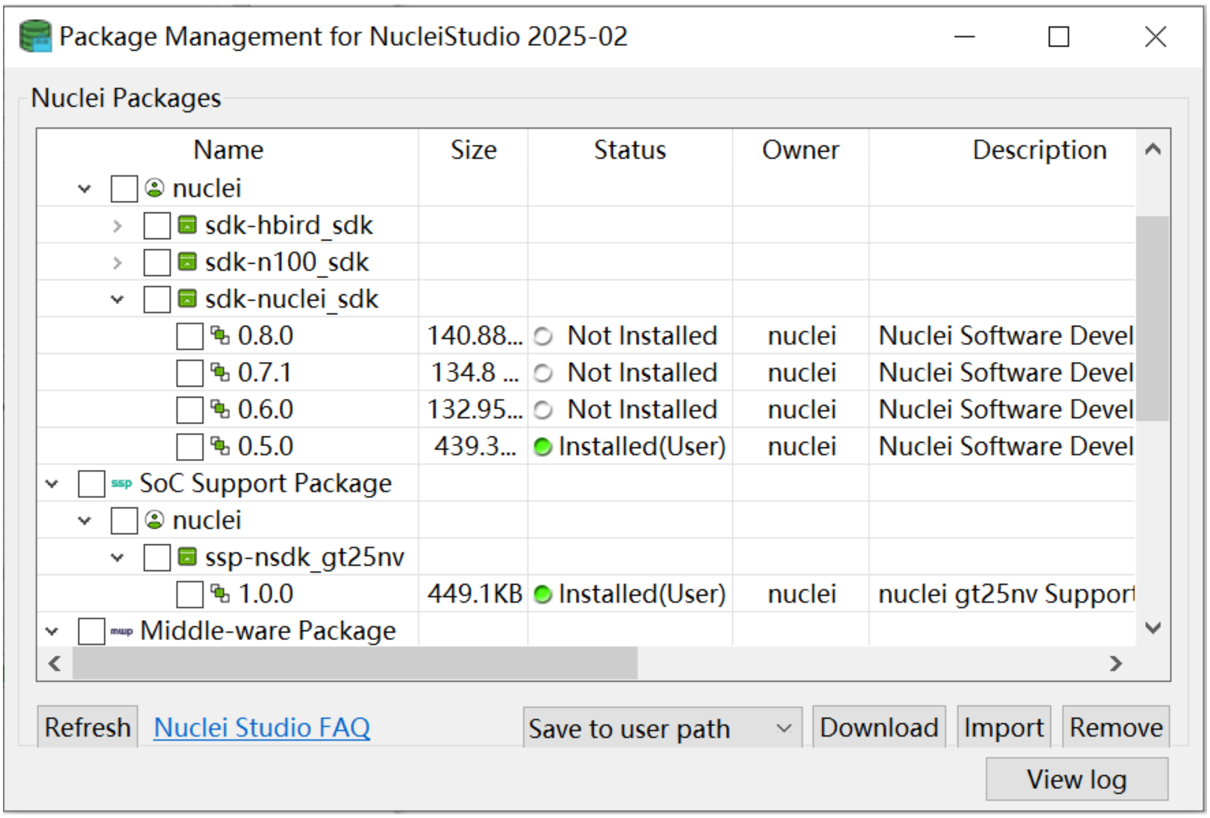


按照示例创建的 soc 在导入时，提示缺少依赖 sdk-nuclei_sdk，点击确定，进入 Nuclei Package Management 页面，根据提示，选中 sdk-nuclei_sdk，点击 Download。





下载完成后，NPK 组件包工程导入成功后结果如下图。



2.5.2 NPK 组件包的检测和问题处理

新创建组件包导入前的检测

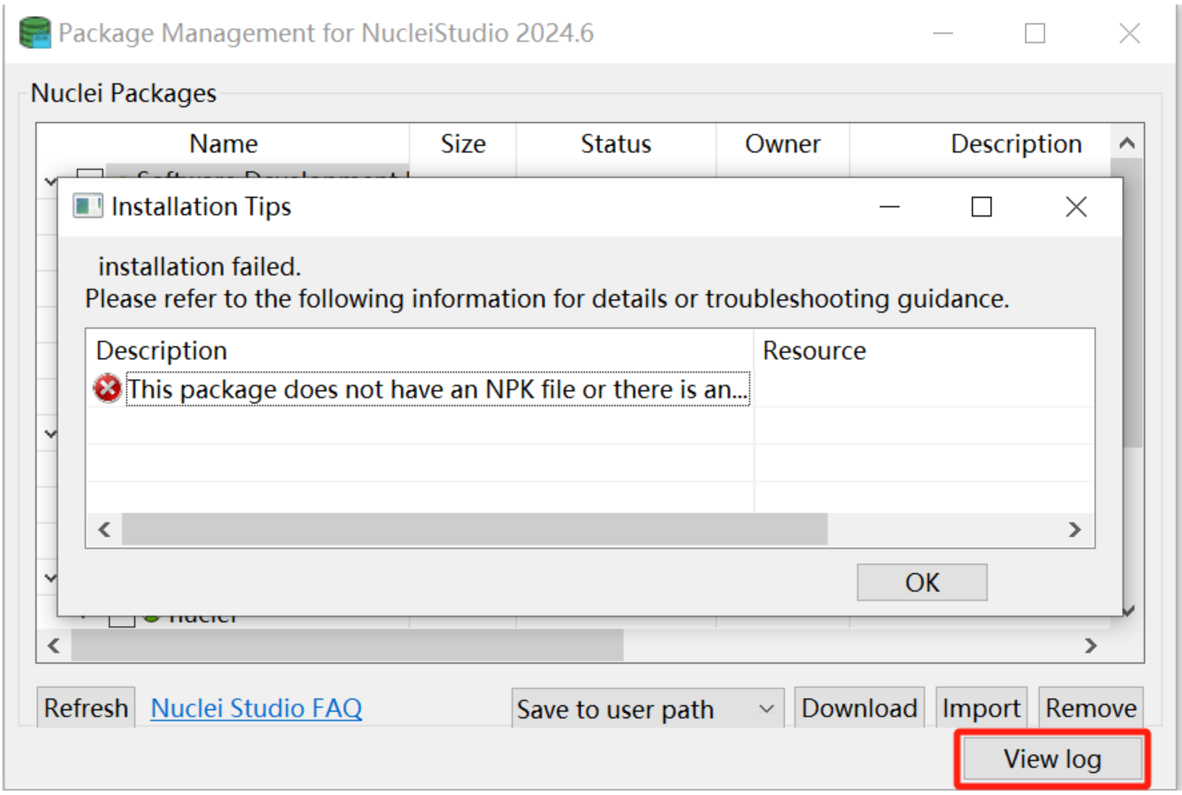
创建完成的软件包，在导入之前，可以先使用检查工具检测 npk 组件包的合法性，避免导入失败。其核心功能聚焦于代码风格、变量命名、函数结构以及依赖关系的合规性检查，确保代码质量。

关于该工具的内容请参见: 软件包检查工具[4]

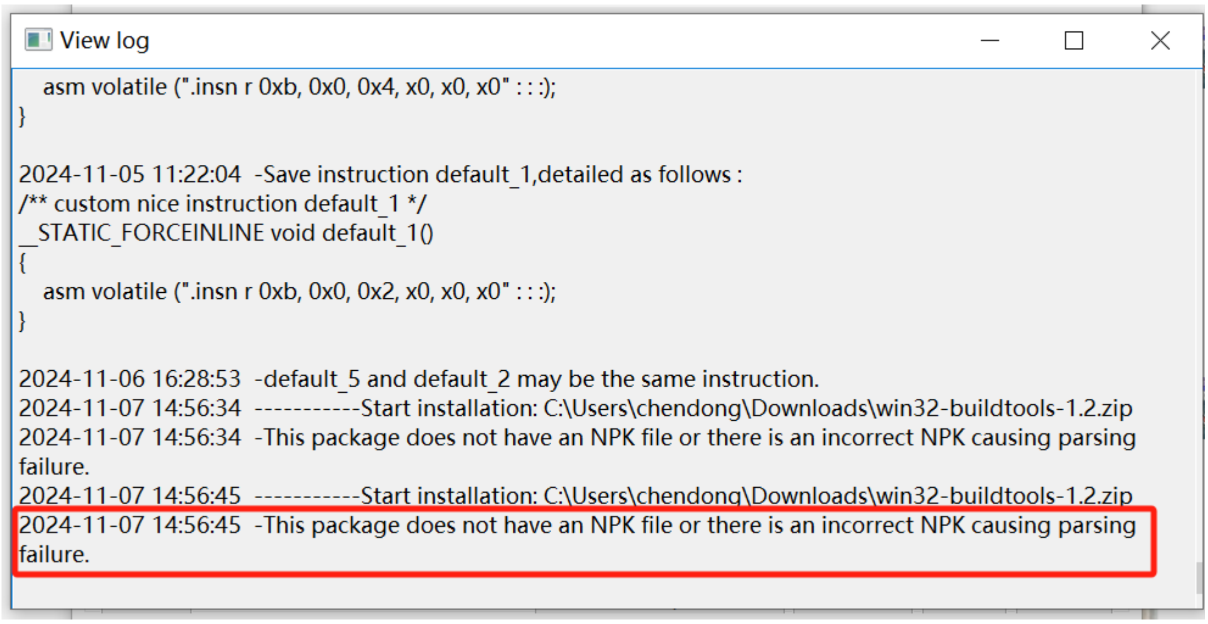
软件包导入失败的处理

在软件包导入失败的情况下，可以根据相关日志去判断问题所在，这里主要针对 npk 文件格式错误、依赖不满足等问题。

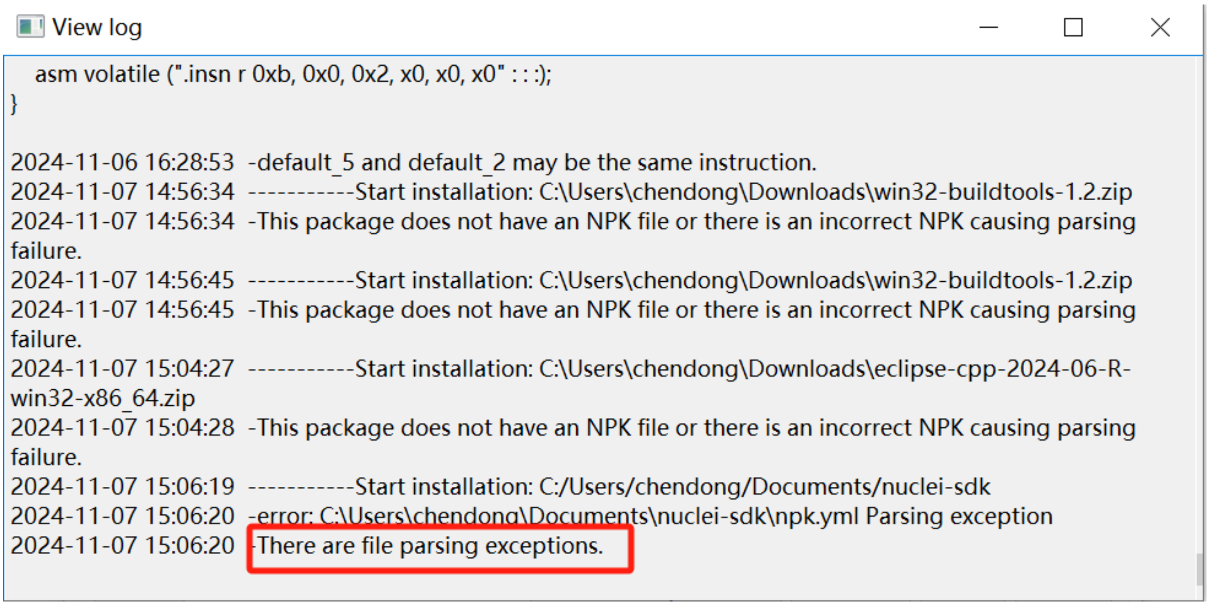
点击 Package Management 页面右下角的 View log，会弹出对应日志页面，简略的告知导入失败的原因。

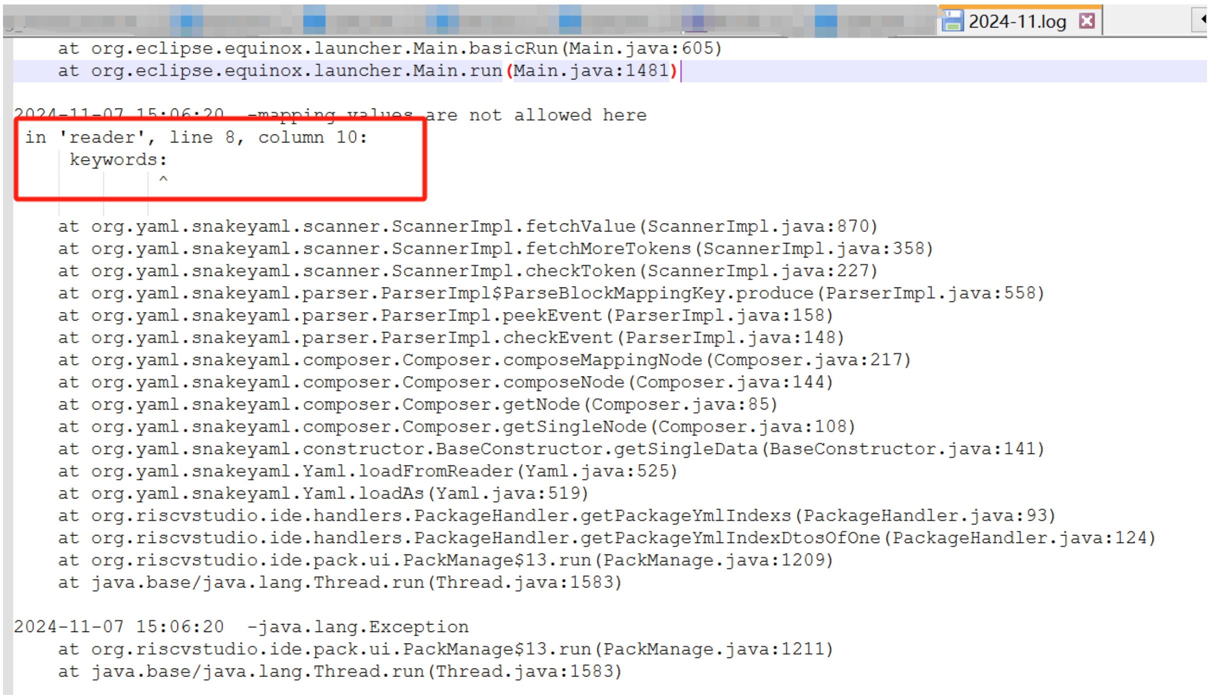


[4] https://github.com/Nuclei-Software/npk-checker



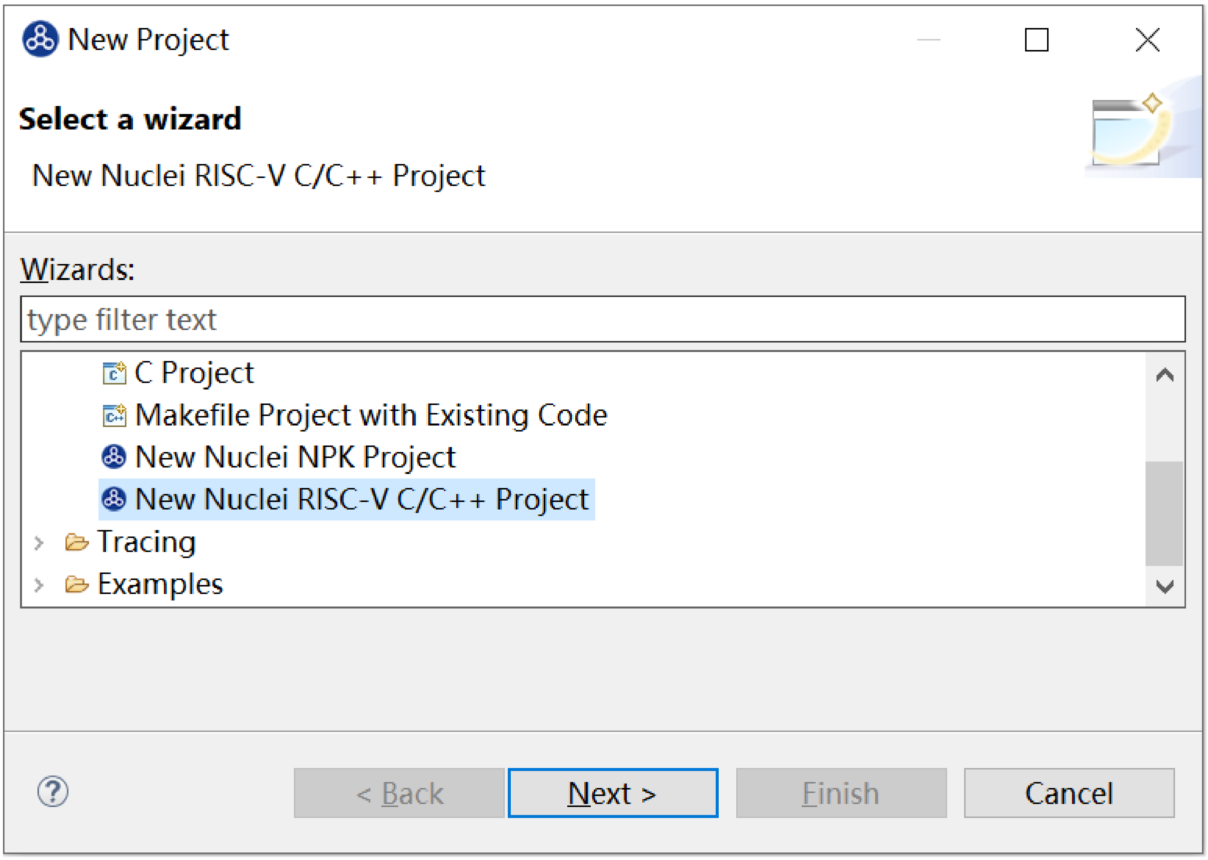
当日志中出现类似 There are file parsing exceptions 提示时，可以去对应的 < 用户目录>\nuclei-pack-npk-v2\CFGs\logs\sys\ 路径下看当天的日志文件，进一步了解相关 npk 文件的问题点，以便进行相应的修正。



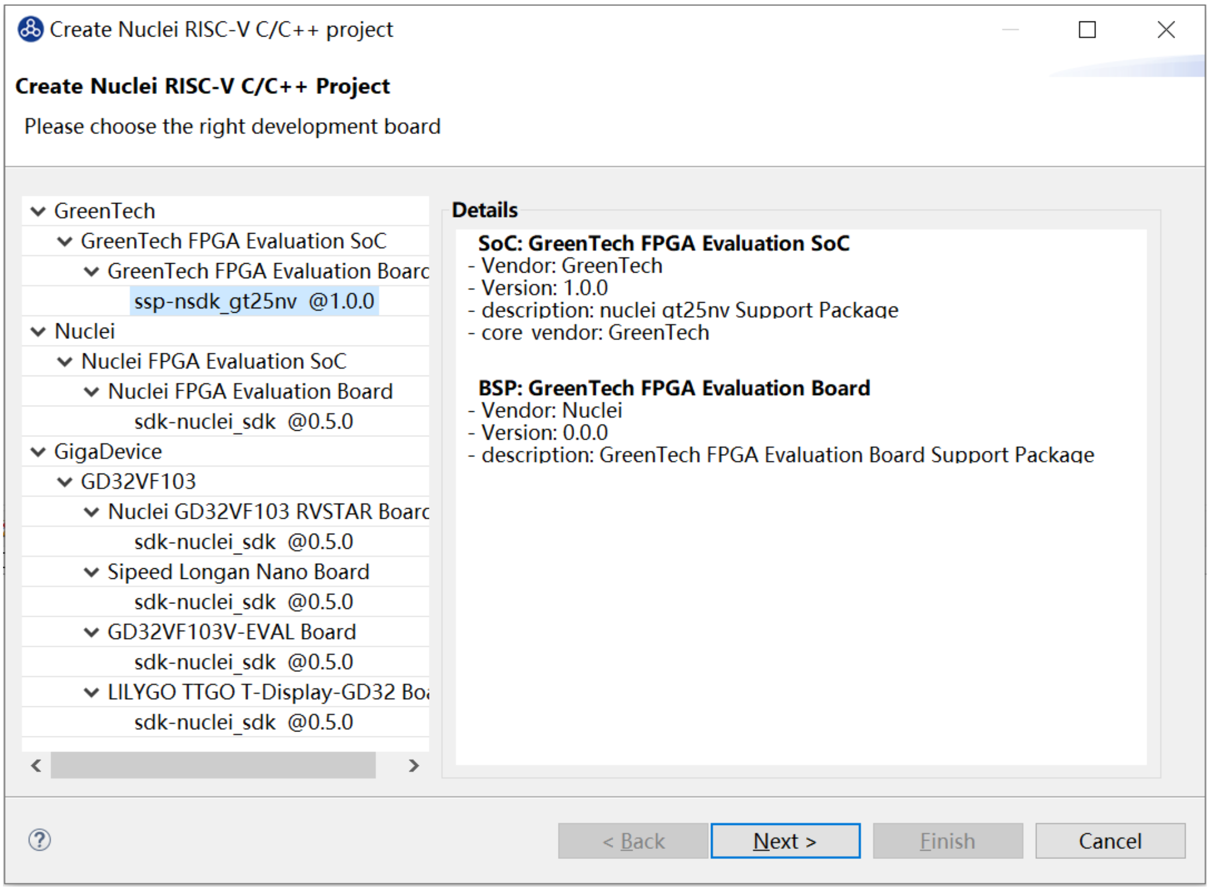


创建测试项目

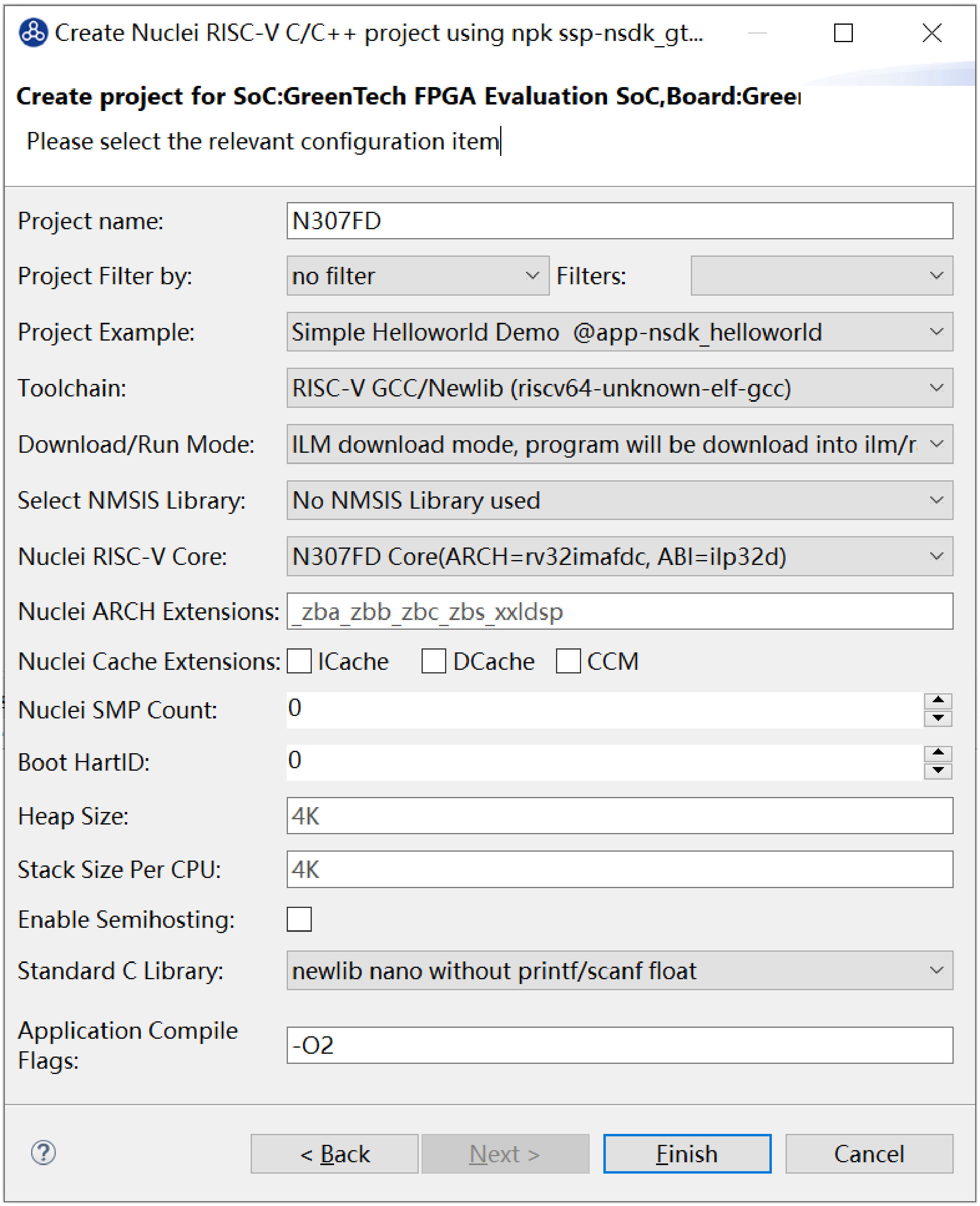
使用导入的 NPK 组件包工程创建一个工程，可以在菜单栏中，选择 `File-> New-> Project-> C/C++-> New Nuclei RISC-V C/C++ Project`。



点击 Next，选择上面创建的 soc 内所包含的 soc 和 board



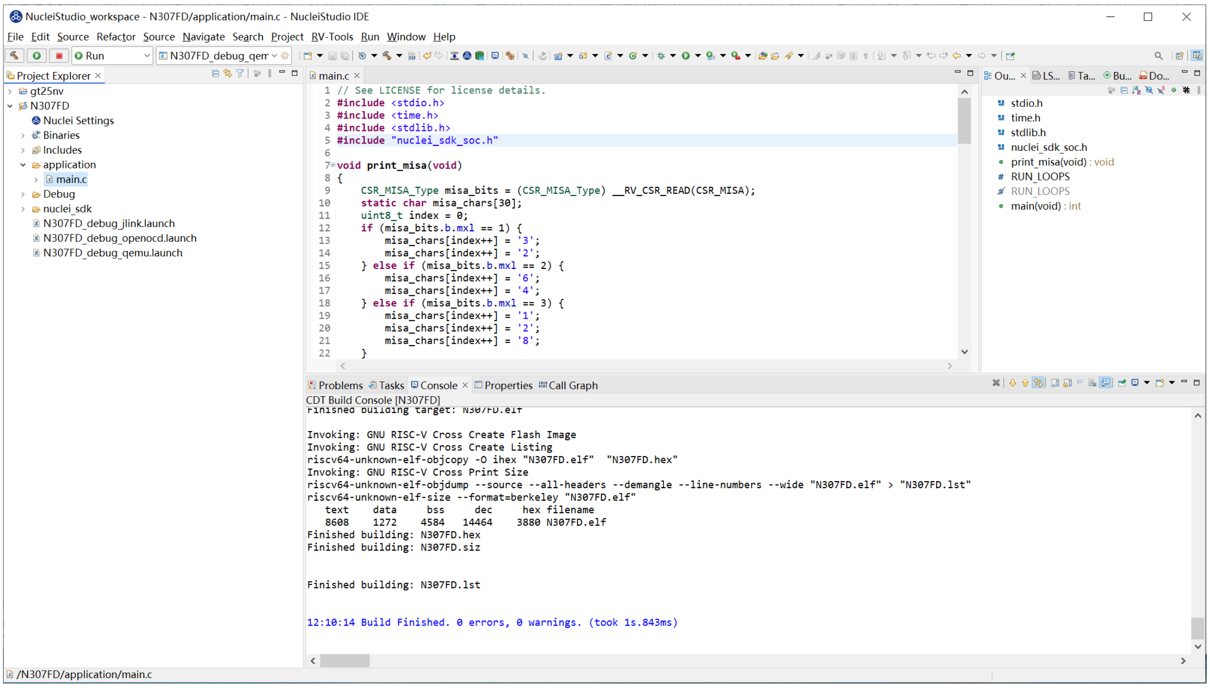
点击 Next，填入 Project Name，并选择 Project Example 为 `Simple Helloworld Demo`，点击 `Finish`，完成测试工程的创建。


Create Nuclei RISC-V C/C++ project using npk ssp-nsdk_gt...
Create project for SoC:GreenTech FPGA Evaluation SoC,Board:Greer
Please select the relevant configuration item
Project name:
N307FD
Project Filter by:
no filter
Filters:
Project Example:
Simple Helloworld Demo @app-nsdk_helloworld
Toolchain:
RISC-V GCC/Newlib (riscv64-unknown-elf-gcc)
Download/Run Mode:
ILM download mode, program will be download into ilm/r
Select NMSIS Library:
No NMSIS Library used
Nuclei RISC-V Core:
N307FD Core(ARCH=rv32imafdc, ABI=ilp32d)
Nuclei ARCH Extensions:
_zba_zbb_zbc_zbs_xxldsp
Nuclei Cache Extensions:
ICache
DCache
CCM
Nuclei SMP Count:
0
Boot HartID:
0
Heap Size:
4K
Stack Size Per CPU:
4K
Enable Semihosting:
Standard C Library:
newlib nano without printf/scanf float
Application Compile Flags:
-O2
< Back
Next >
Finish
Cancel

编译调试测试工程

上文步骤中创建的一个工程，就是根据开发者的 NPK 组件包创建出来的一个测试工程，开发者可以按一个正常的工程进行对应的编码、调式、运行等操作。

鼠标点击选中上一步生成的项目 N307FD，然后编译成功，后续运行等步骤略去，至此已成功创建了一个 NPK 组件包，并使用此 NPK 组件包进行了导入使用。

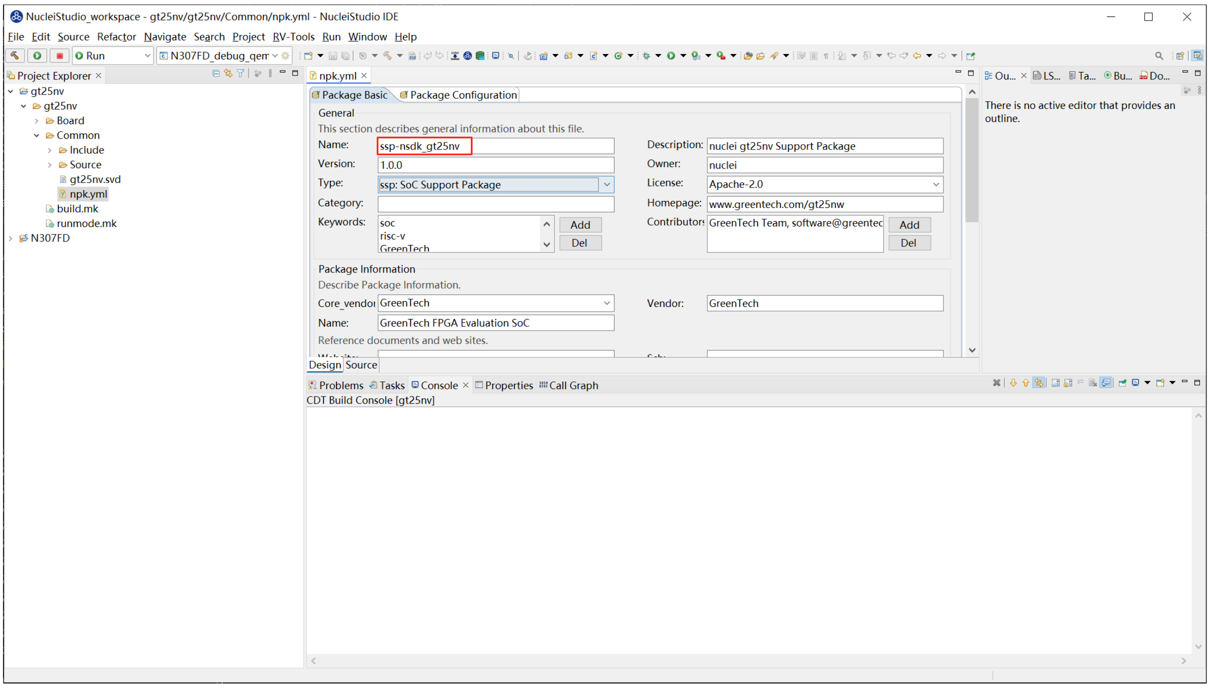


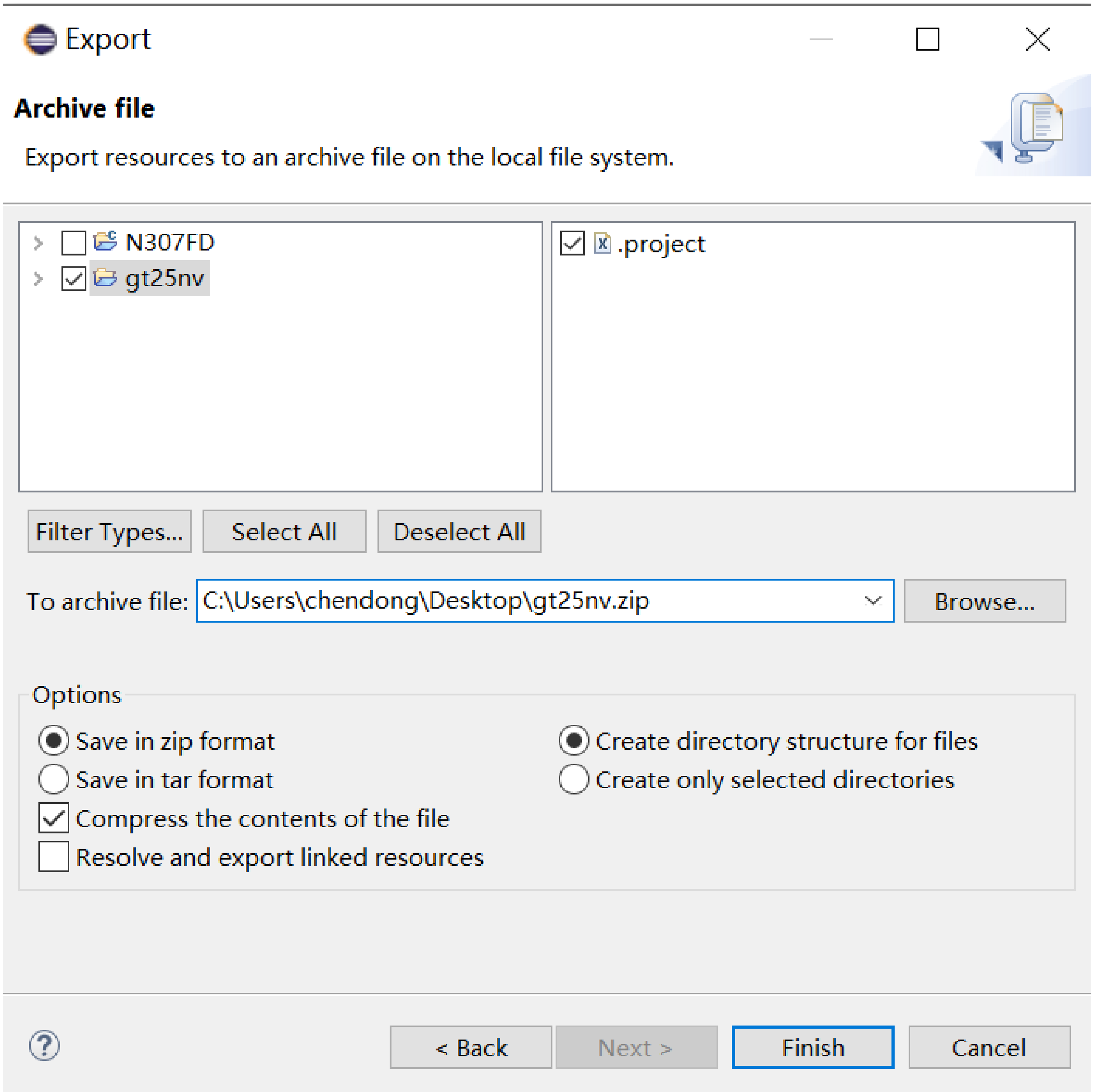
2.5.3 共享 NPK 组件包

NPK 组件包共享

经过测试通过后，可以将您创建的 NPK 组件包分享出去，首先需要将您的 NPK 组件包工程导出为一个 zip 包，具体操作如下。

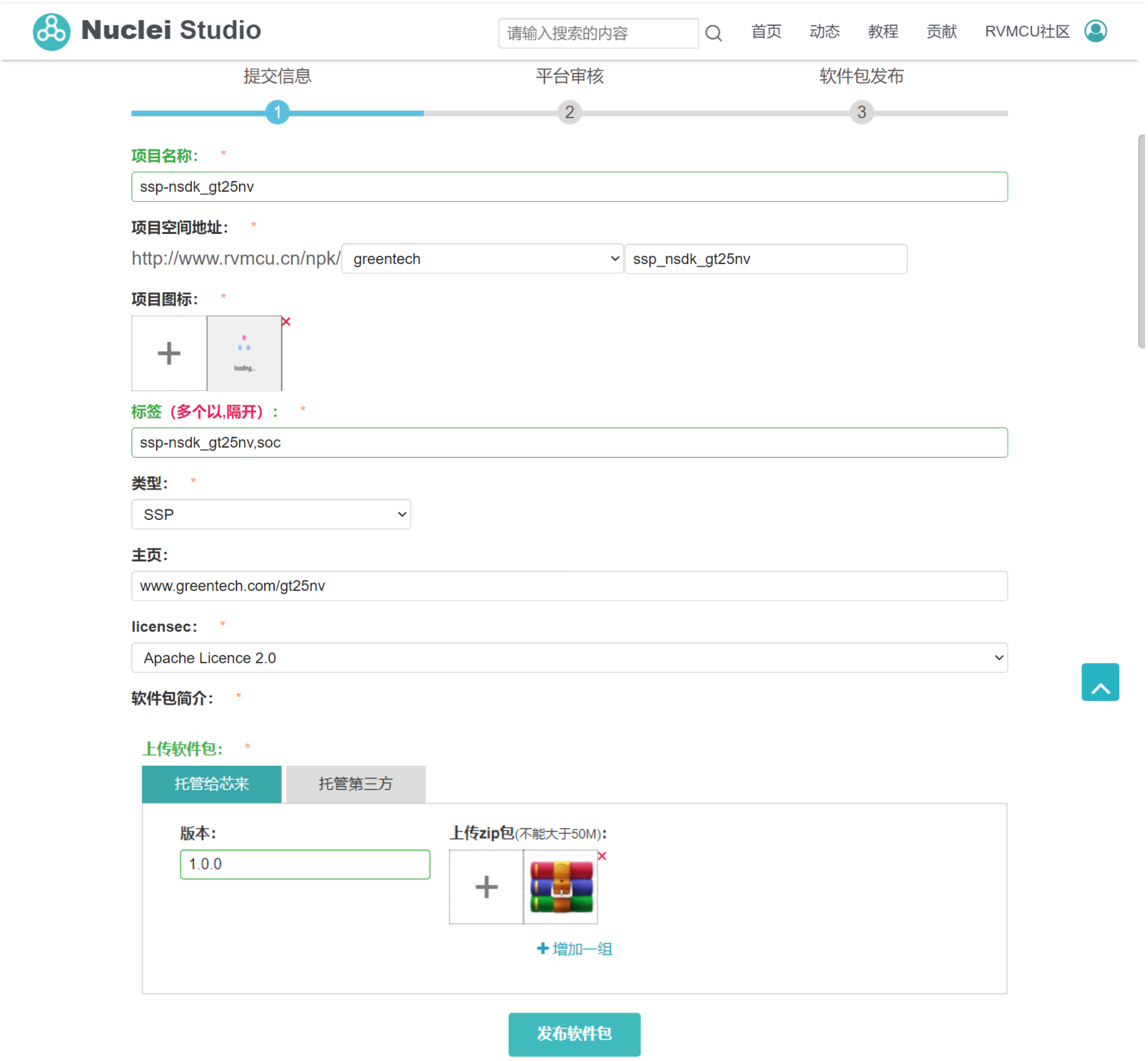
打开 NPK 组件包项目，双击最外层的 `npk.yml`，找到其 Name 为 `ssp-nsdk_gt25nv`，右键点击 NPK 组件包项目，点击 `Export`，选择 `Archive File`，选择需要导出的工程，然后根据提示指定导出 zip 文件存放的位置。


NucleiStudio_workspace - gt25nv/gt25nv/Common/npk.yml - NucleiStudio IDE
File Edit Source Refactor Navigate Search Project RV-Tools Run Window Help
Project Explorer
gt25nv
gt25nv
Board
Common
Include
Source
gt25nv.svd
npk.yml
build.mk
runmode.mk
N307FD
npk.yml
Package Basic
Package Configuration
General
This section describes general information about this file.
Name: ssp-nsdk_gt25nv
Version: 1.0.0
Type: ssp: SoC Support Package
Category:
Keywords: soc risc-v GreenTech
Add
Del
Description: nuclei gt25nv Support Package
Owner: nuclei
License: Apache-2.0
Homepage: www.greentech.com/gt25nw
Contributor: GreenTech Team, software@greentec
Package Information
Describe Package Information.
Core_vendor GreenTech
Vendor: GreenTech
Name: GreenTech FPGA Evaluation SoC
Reference documents and web sites.
Design Source
Problems Tasks Console Properties Call Graph
CDT Build Console [gt25nv]
There is no active editor that provides an outline.

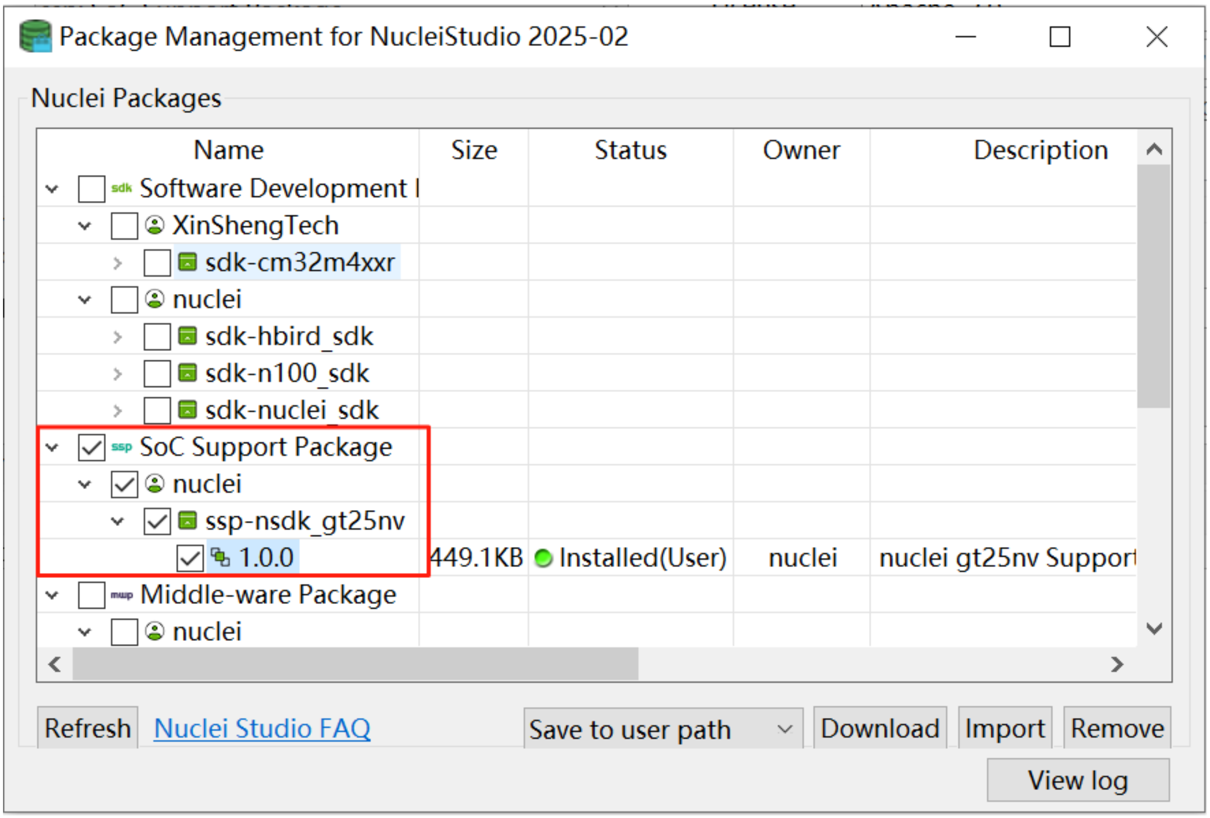


导出的 zip 包，可以通过 rvmcu 社区进行分享贡献。进入 社区分享页面[5]，依据提示信息，依次填写需要分享的 NPK 组件包的名称、所属类型、描述待信息，并上传刚导出的 zip 文件，信息提交后，待管理员审核通过后，该 NPK 组件包就成功贡献了，其他的开发者就可以通过 Nuclei Studio 的 `Nuclei Package Management` 页面找到您的 NPK 组件包，并下载使用。具体操作如下图

[5] https://www.rvmcu.com/nucleistudio-developer.html

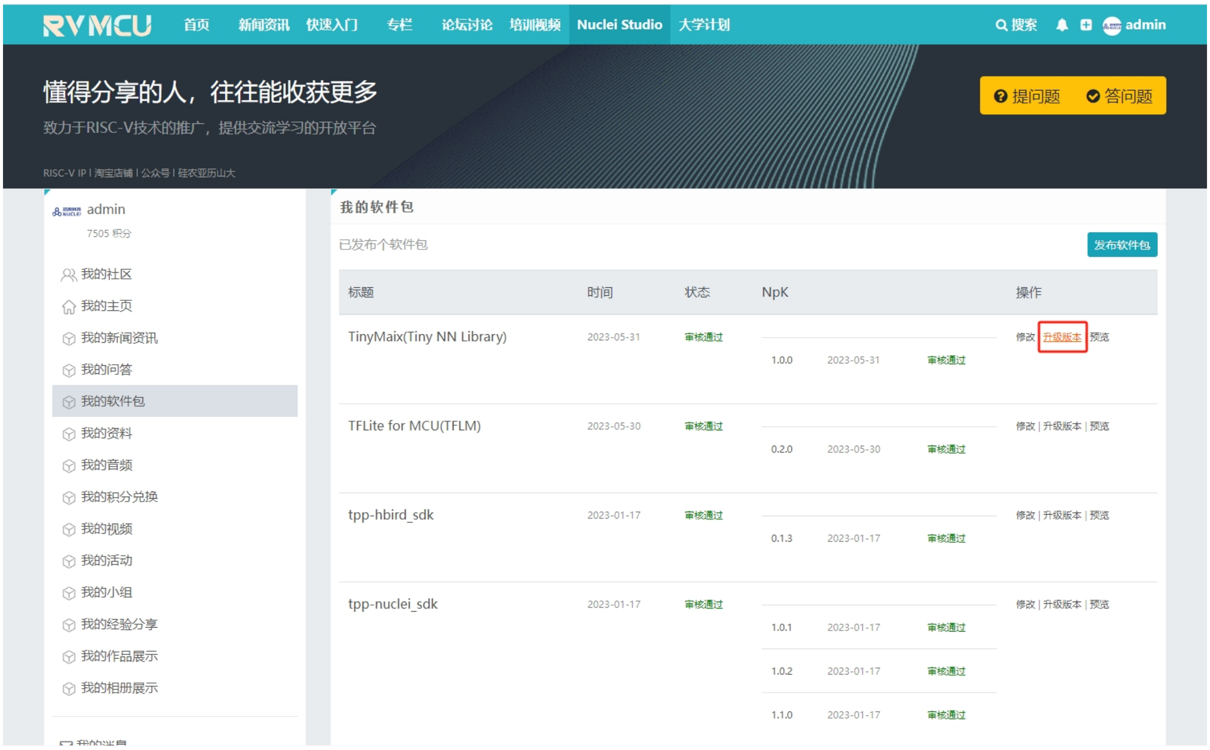

Nuclei Studio
请输入搜索的内容
首页
动态
教程
贡献
RVMCU社区
提交信息
平台审核
软件包发布
1
2
3
项目名称：
ssp-nsdk_gt25nv
项目空间地址：
http://www.rvmcu.cn/npk/
greentech
ssp_nsdk_gt25nv
项目图标：
标签（多个以,隔开）：
ssp-nsdk_gt25nv,soc
类型：
SSP
主页：
www.greentech.com/gt25nv
licensec：
Apache Licence 2.0
软件包简介：
上传软件包：
托管给芯来
托管第三方
版本：
1.0.0
上传zip包(不能大于50M)：
增加一组
发布软件包

分享的 npk 组件包通过审核后，在 Nuclei Studio 中打开 `Nuclei Package Management` 页面，然后点击 `Refresh`，刷新后即可找到刚分享的组件包。

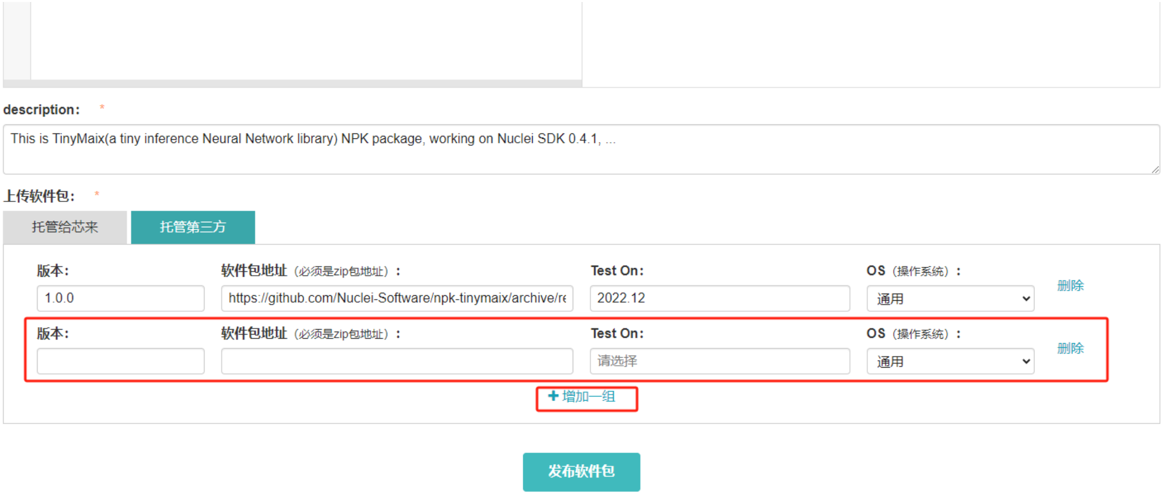


NPK 组件包升级

在 NPK 组件包共享后，如果有新的版本需要维护，在创建测试打包完成后，可以对原有的 NPK 组件包进行升级。共入 Nuclei Studio[6] 页面，找到管理组件包入口，然后进组件包管理页面，点击升级组件包，然后之前的步骤，上传 NPK 组件包，等待审核通过，则组件包升级完成。



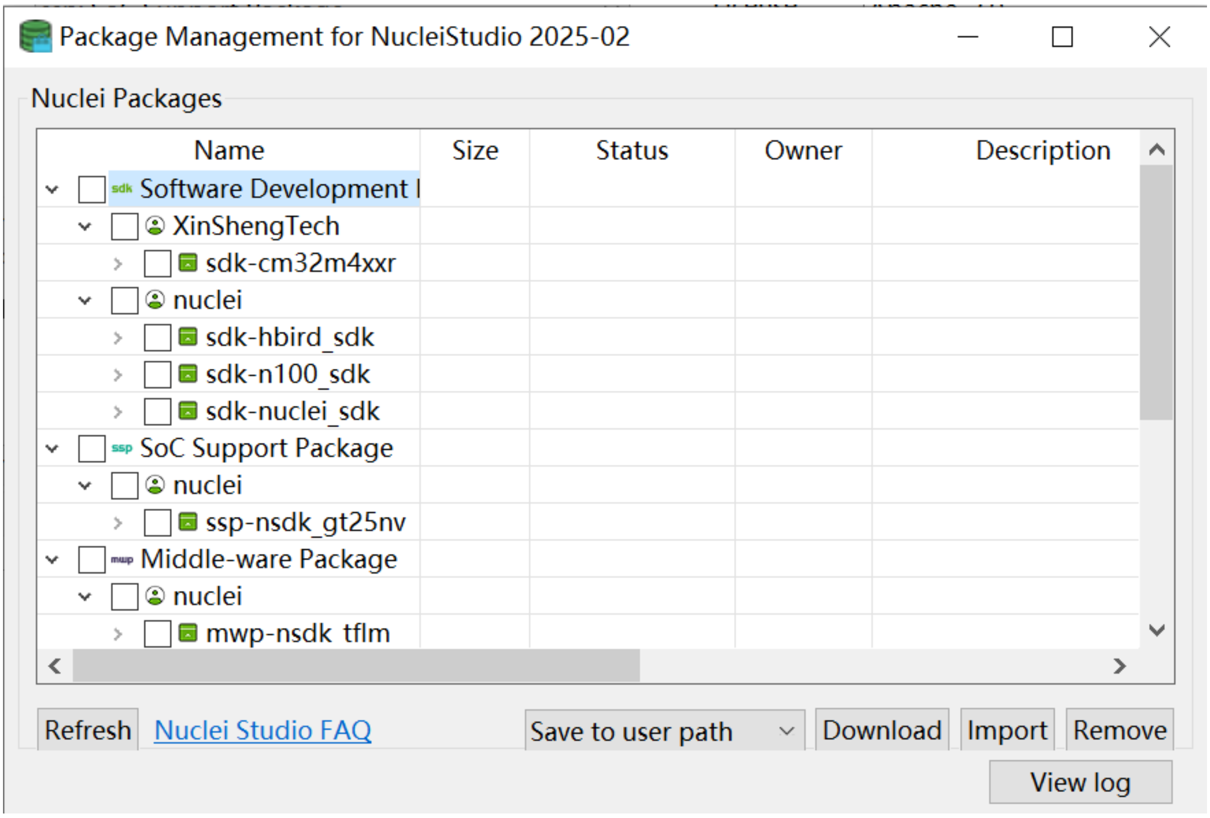
[6] https://www.rvmcu.com/nucleistudio.html



2.5.4 NPK 组件包在 Nuclei Studio 中的使用

NPK 组件包在 Nuclei Studio 中，丰富了其用户体验，通过 NPK 组件包我们可以定义各种不同的创建工程流程，也能很方便的将成熟的工程或者组件共享给其人。

我们所有贡献的 NPK 包，都在 Nuclei Studio 的 `NPK Package Managment` 中进行管理，用户可以在这里进行 NPK 的下载、导入、删除等操作。



2.6 Nuclei Studio NPK 应用

Nuclei Studio 中内建了对 Nuclei Package（NPK）功能的完整支持，方便开发者或者创建不同的软件开发包，并且通过 Nuclei Package Management 方式导入到 Nuclei Studio 中使用，如果是 CPU IP 客户，可以将自己的芯片 SDK 略作改造以支持 NPK，并导入到 Nuclei Studio 中使用，如果是 SoC IP 客户，可以将提供的 SDK 打包成 Zip 包，导入使用，如果是开发者，也可以依赖某个特定的 NPK，创建对应的 BSP 包或者 APP 包并提供给第三方使用，关于 NPK 功能的详细介绍，以及后续更新说明参见 https://github.com/Nuclei-Software/nuclei-sdk/wiki/Nuclei-Studio-NPK-Introduction 。通过在原有的 SDK 中引入 npk 功能，编写 npk.yml 可以达到 Project Wizard 功能的部分定制化。

开发者要使用 Nuclei Studio 进行工程的创建，需先将对应的 SDK NPK Zip 包安装到 IDE 中，方可根据不同的开发板快速新建不同的模板工程，并根据不同的模板添加需要的 SDK 源码，根据选项生成不同的编译链接选项设置。

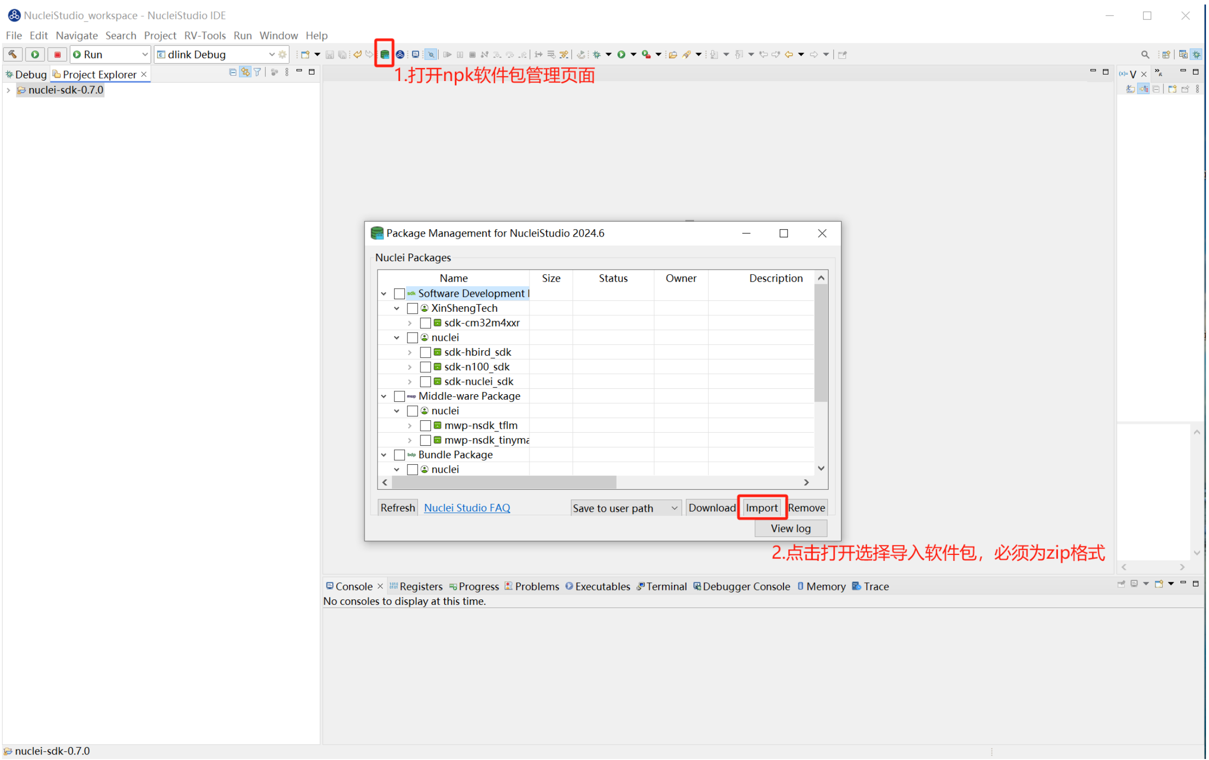
2.6.1 NPK 软件包管理

导入本地 NPK 软件包

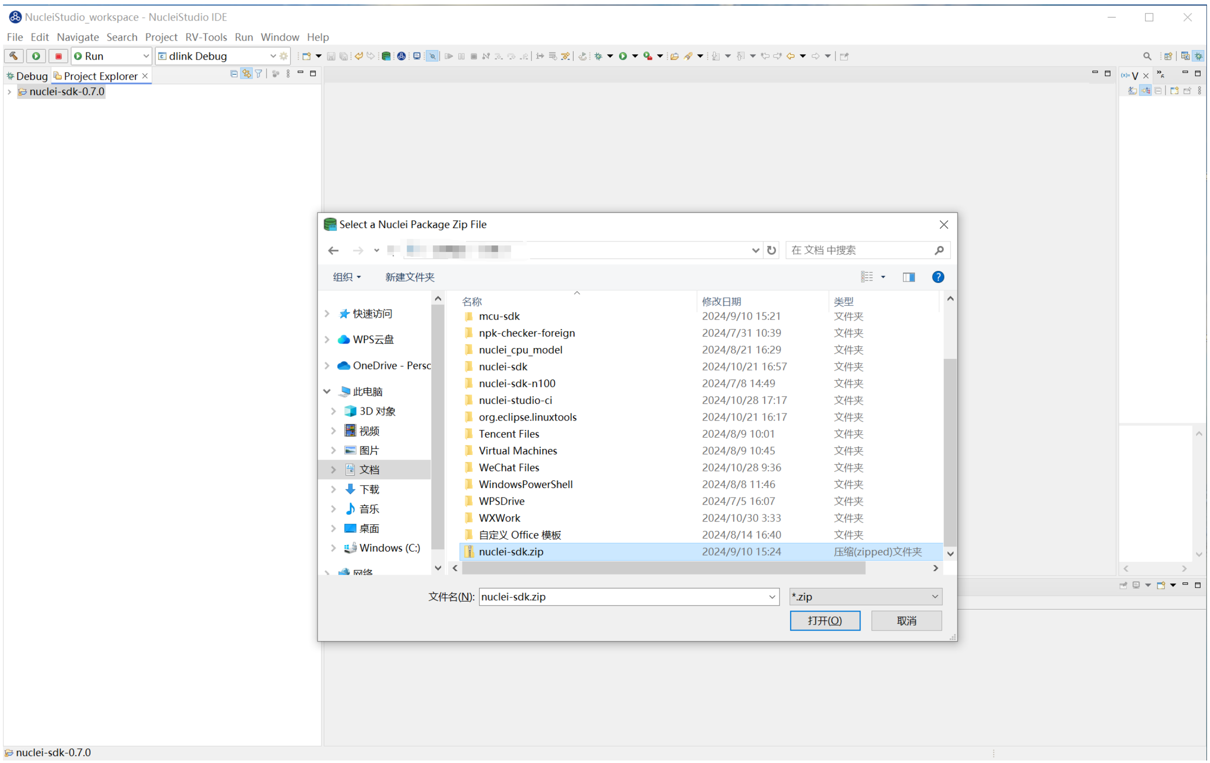
导入 zip 软件包

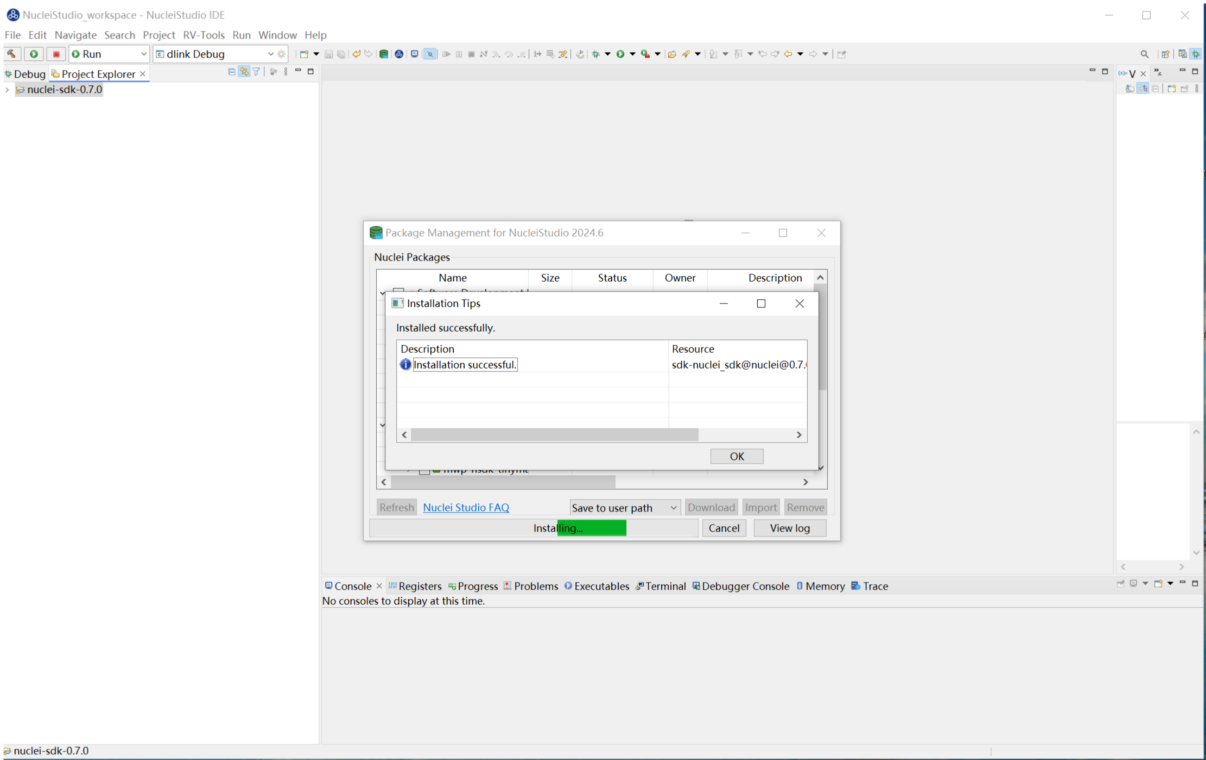
当用户拿到一个 zip 格式的软件包时，可以通过 Nuclei Package Management 导入一个 zip 格式的软件包，你可以按照以下的步骤操作：

首先，点击 Nuclei Studio IDE 工具栏上的 **Nuclei Package Management** 按钮，这将打开 npk 软件包管理页面。在这个页面上，你可以管理和浏览已安装的软件包，以及导入新的软件包。在这里，我们找到并点击 **Import** 选项。这个选项用于从本地文件系统选择 zip 类型的软件包。

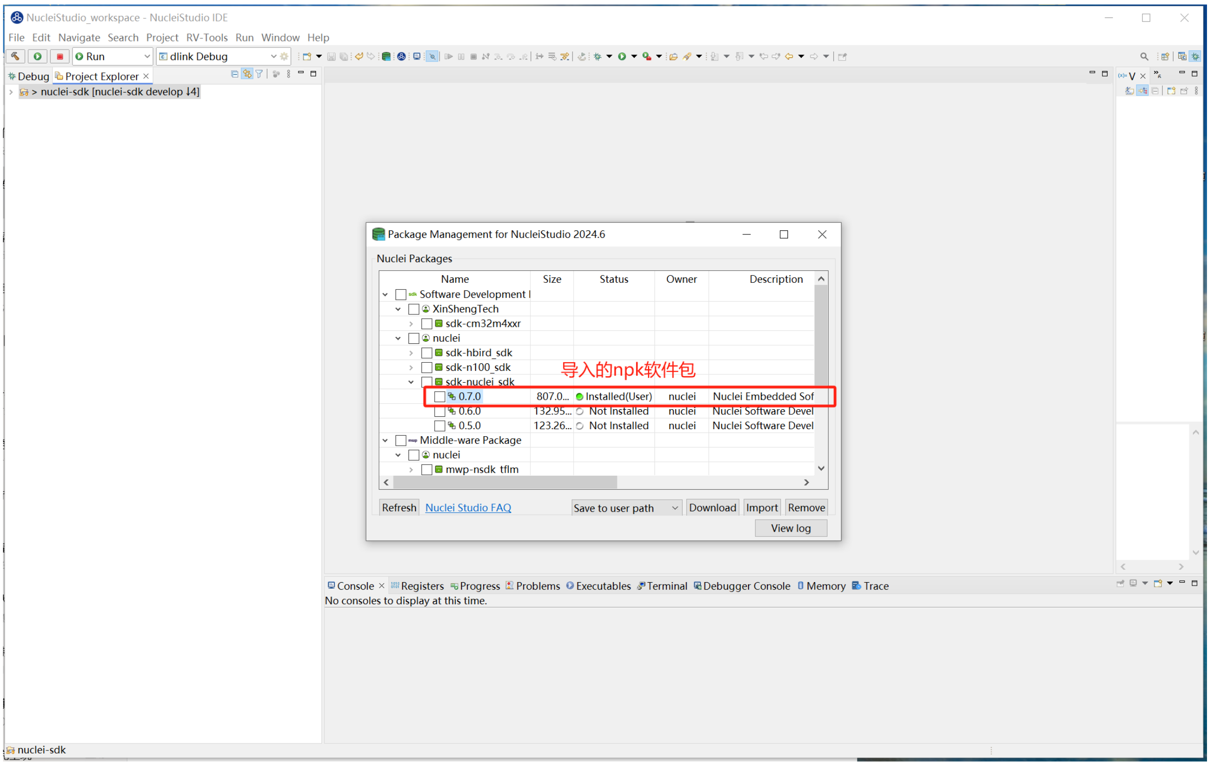


在点击 **Import** 后，系统会提示你选择一个要导入的软件包文件。此时，你需要浏览到你的 zip 格式软件包所在的目录，并选中它。确保选中的是一个有效的 zip 包, 点击 打开, 系统会开始处理并导入这个软件包。





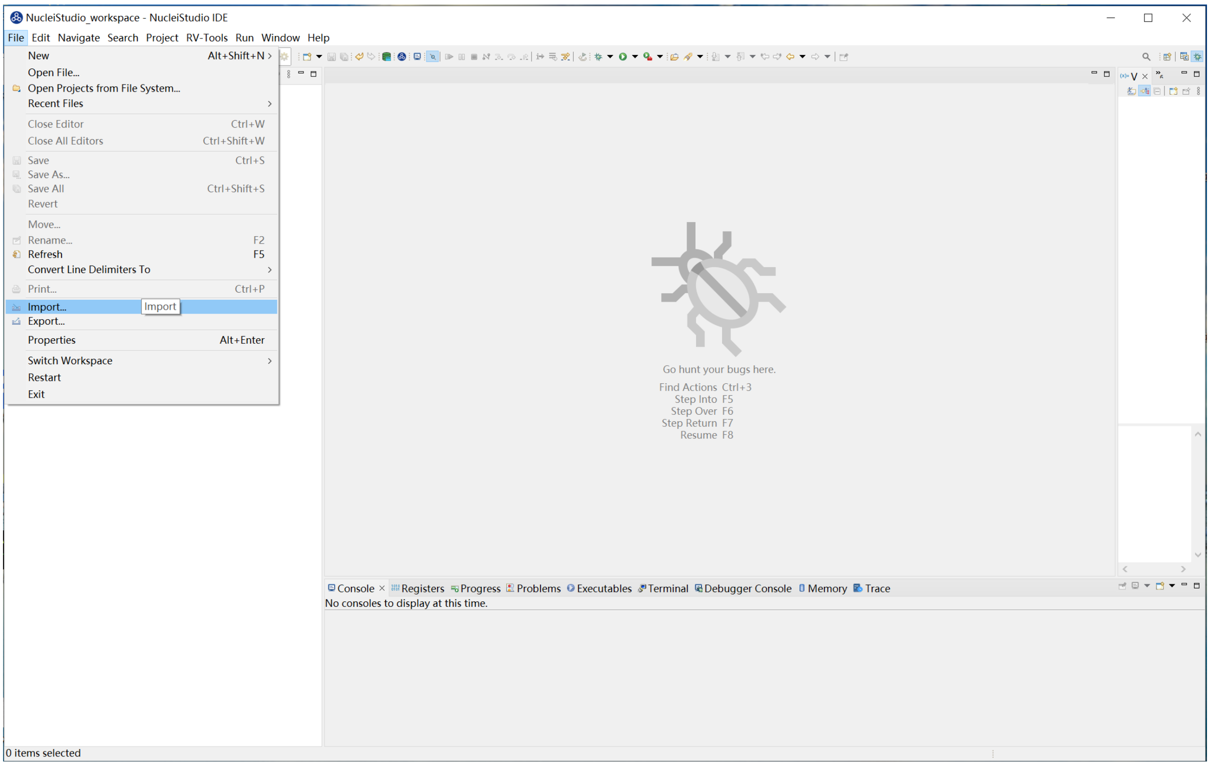
导入完成后，你应该能够在 Nuclei Studio IDE 的 Package Management 页面中看到新导入的软件包。



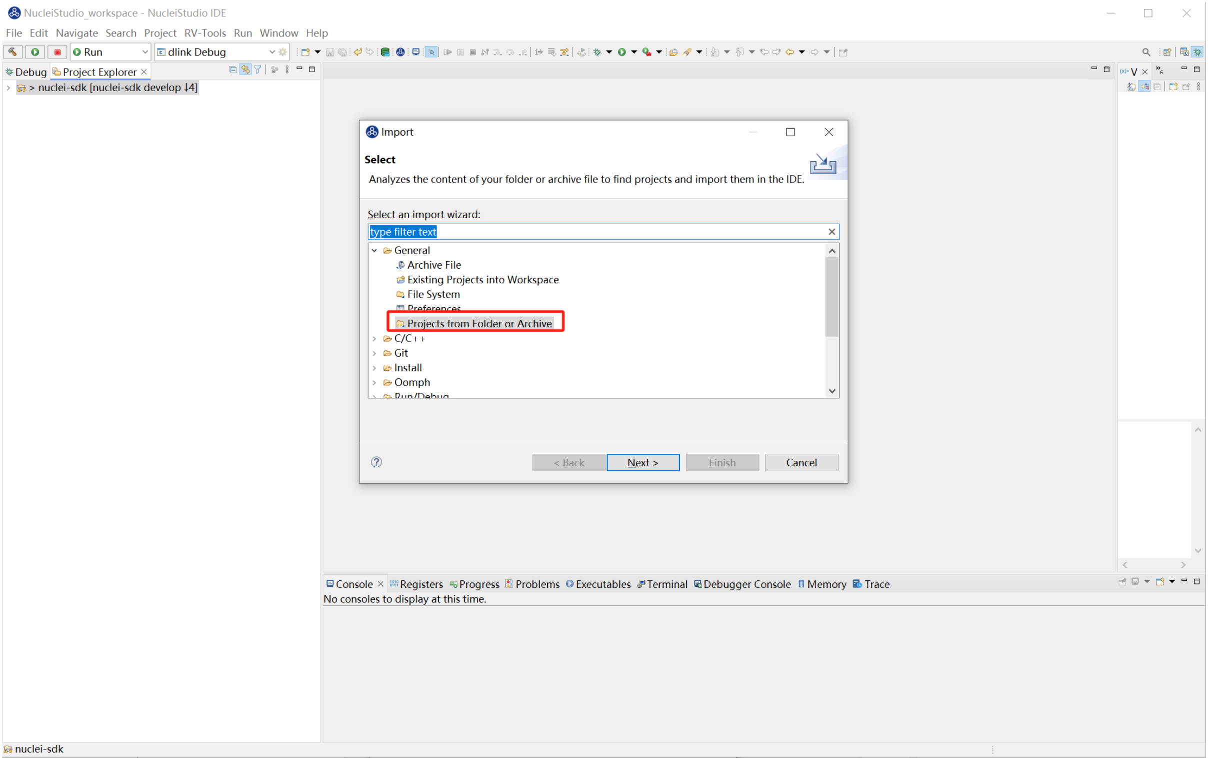
导入未压缩的 **npk** 软件包

当用户拿到一个未压缩的 npk 软件包工程时，如果想将它导入到 NucleiStudio 中，最简单的方式就是通过压缩软件，将其压缩成为一个.zip 压缩包，然后通过上述操作导入到 NucleiStudio 中，如果想一边修改这个 npk 软件包工程，修改完后快速导入到 NucleiStudio 中进行测试，可以通过如下操作实现。

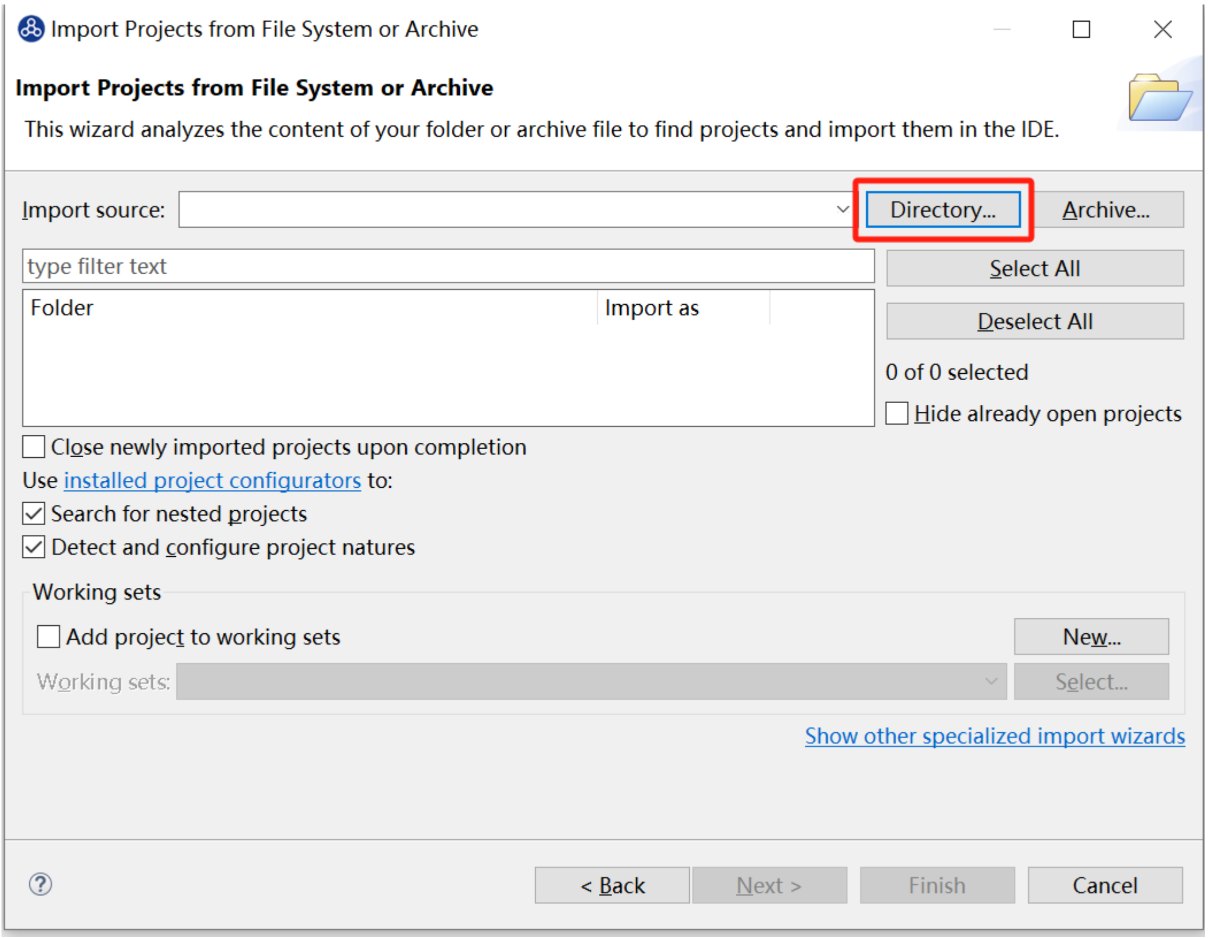
直接导入未压缩的软件包，先要将软件包导入到 Project Exlorer 视图中。点击菜单栏上的 File 选项，在下拉菜单中，找到并点击 **Import**…选项。这将打开导入向导，帮助你从本地文件系统或其他来源导入新的项目或文件。



然后选择 General 下的 **Projects from Folder or Archive** 选项，点击 Next 按钮。

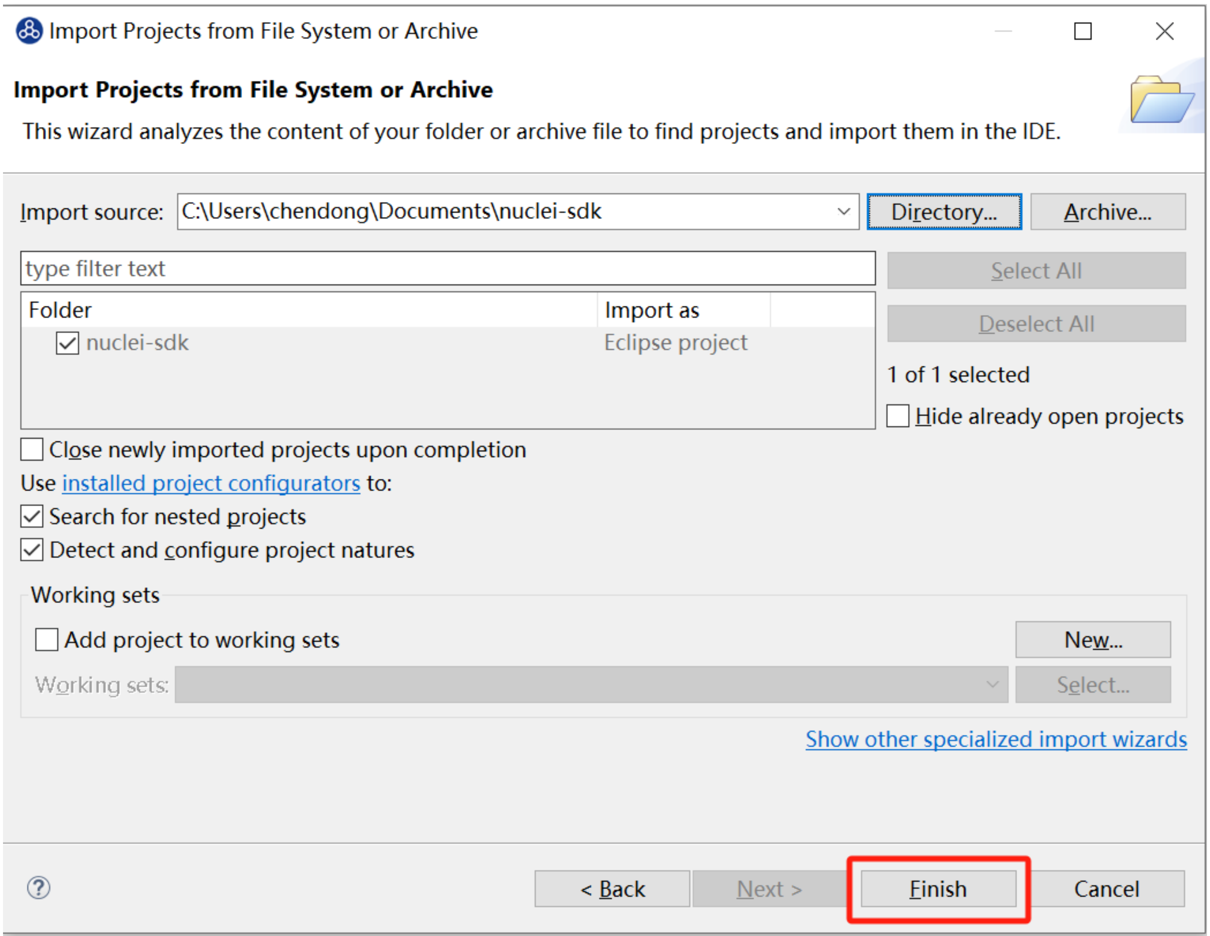


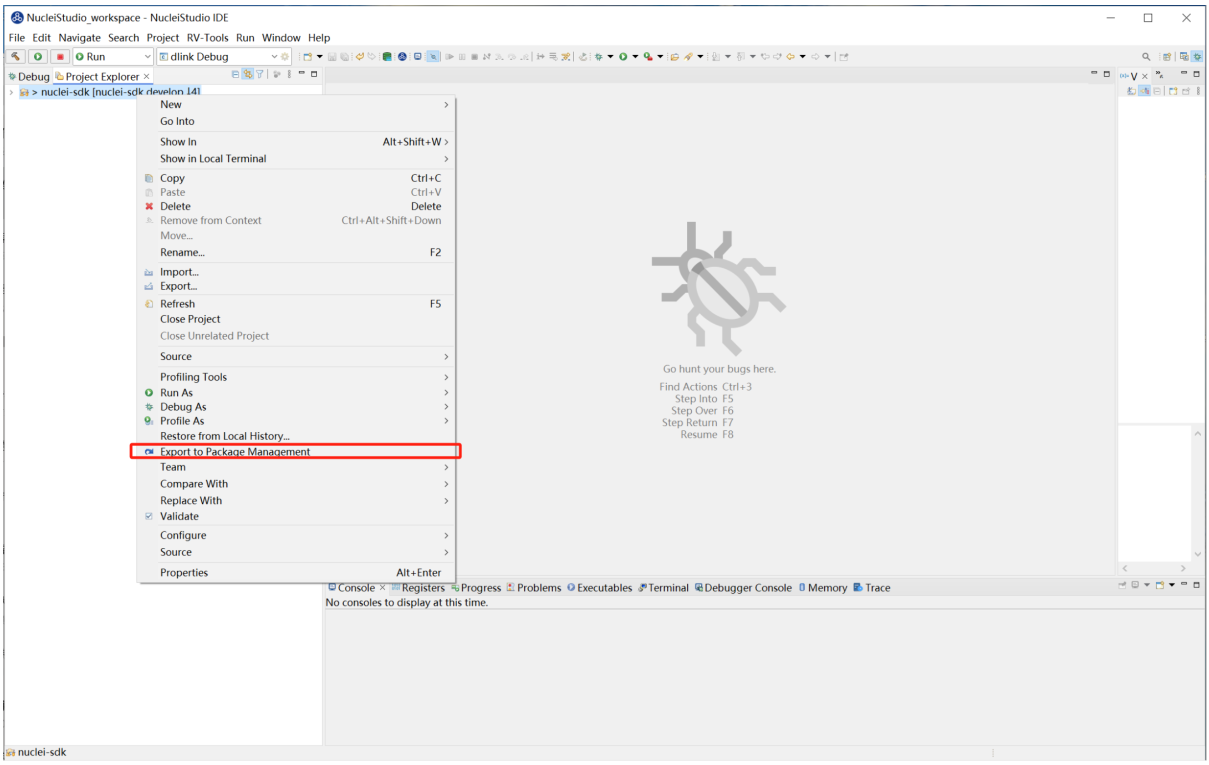
接下来，点击 **Directory**…，这里浏览到你的软件包所在的目录，并选择要导入的文件或文件夹。确保你选择了正确的路径和文件，然后点击 **Finish** 按钮。



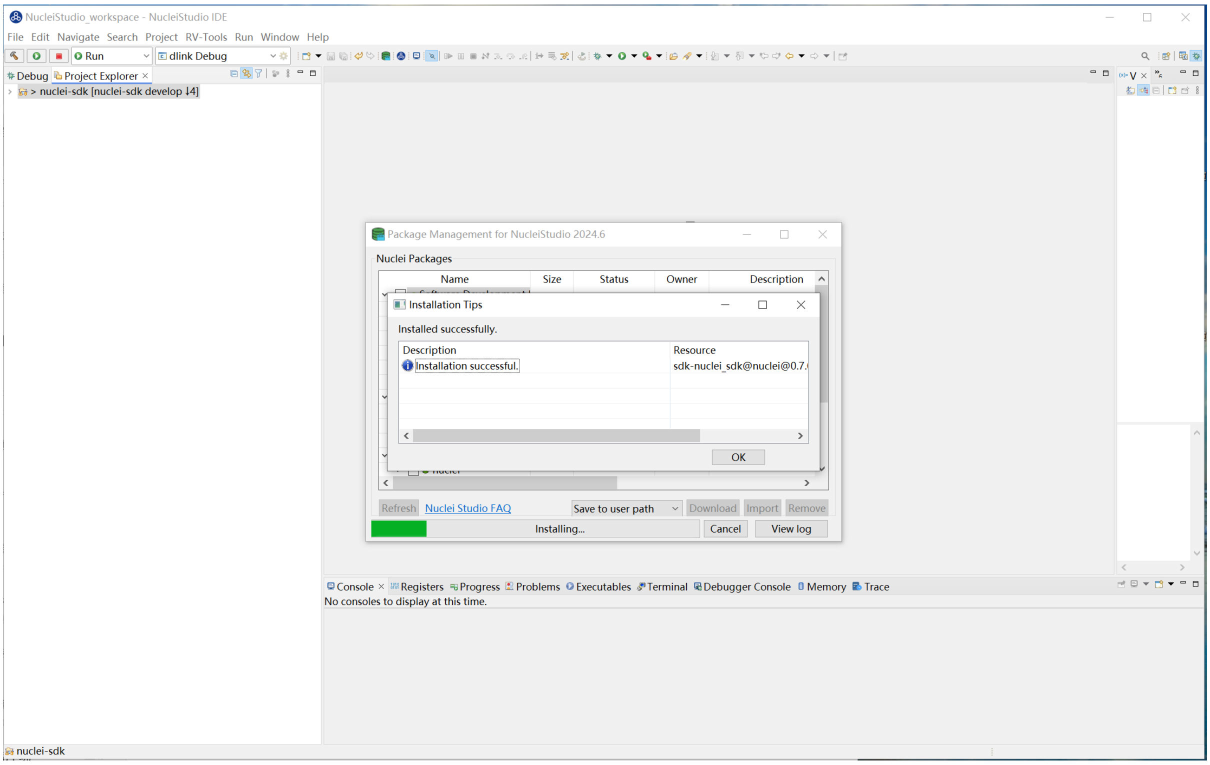
此时软件包已导入到 **Project Explorer** 视图中，右键点击这个文件夹（代表你的软件包），在弹出的

上下文菜单中选择 **Export to Package Management** 并点击，系统会开始处理并导入这个软件包。



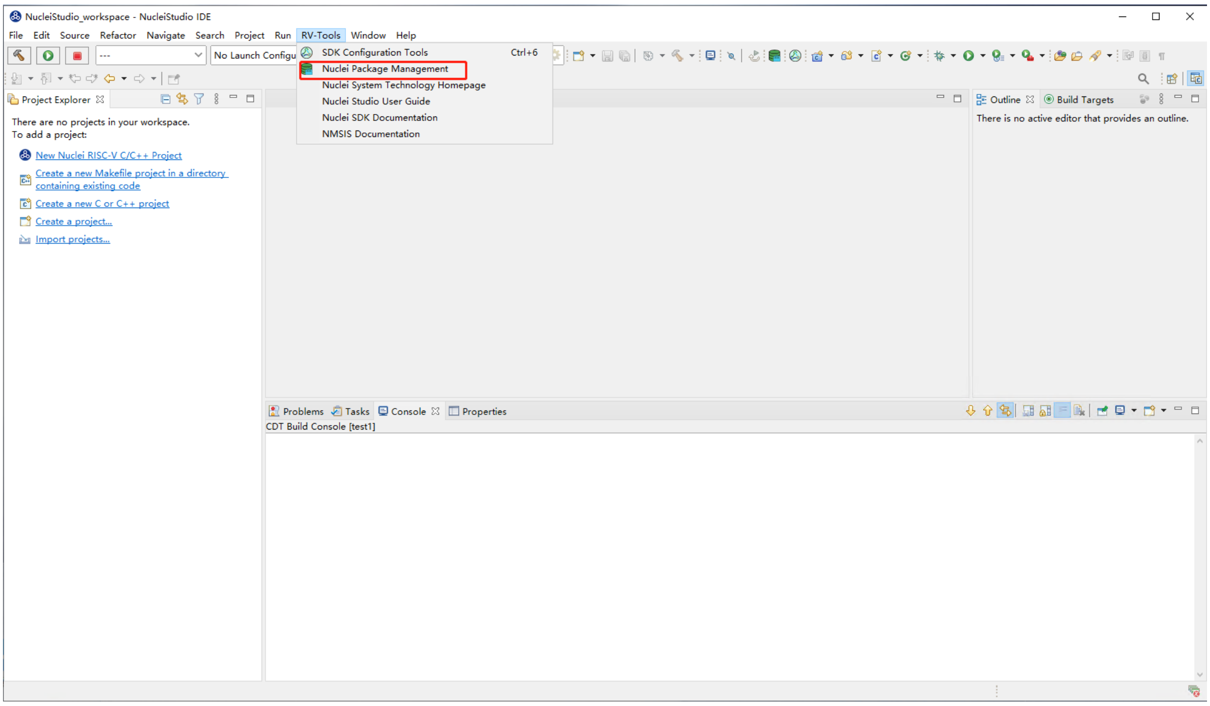


导入过程中，会自动打开 Package Management 页面。待导入完成后，你应该能够在 Nuclei Studio IDE 的 Package Management 页面中看到新导入的软件包。

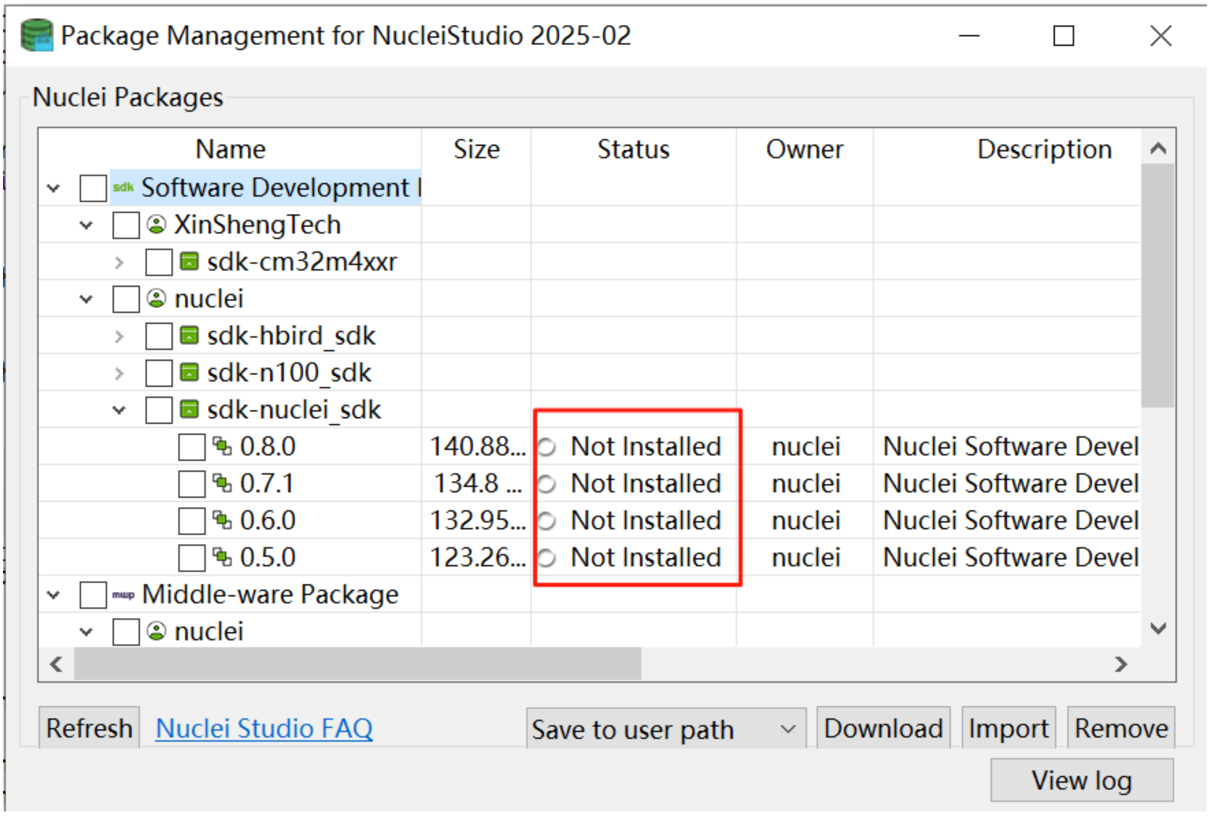


下载云端 NPK 软件包

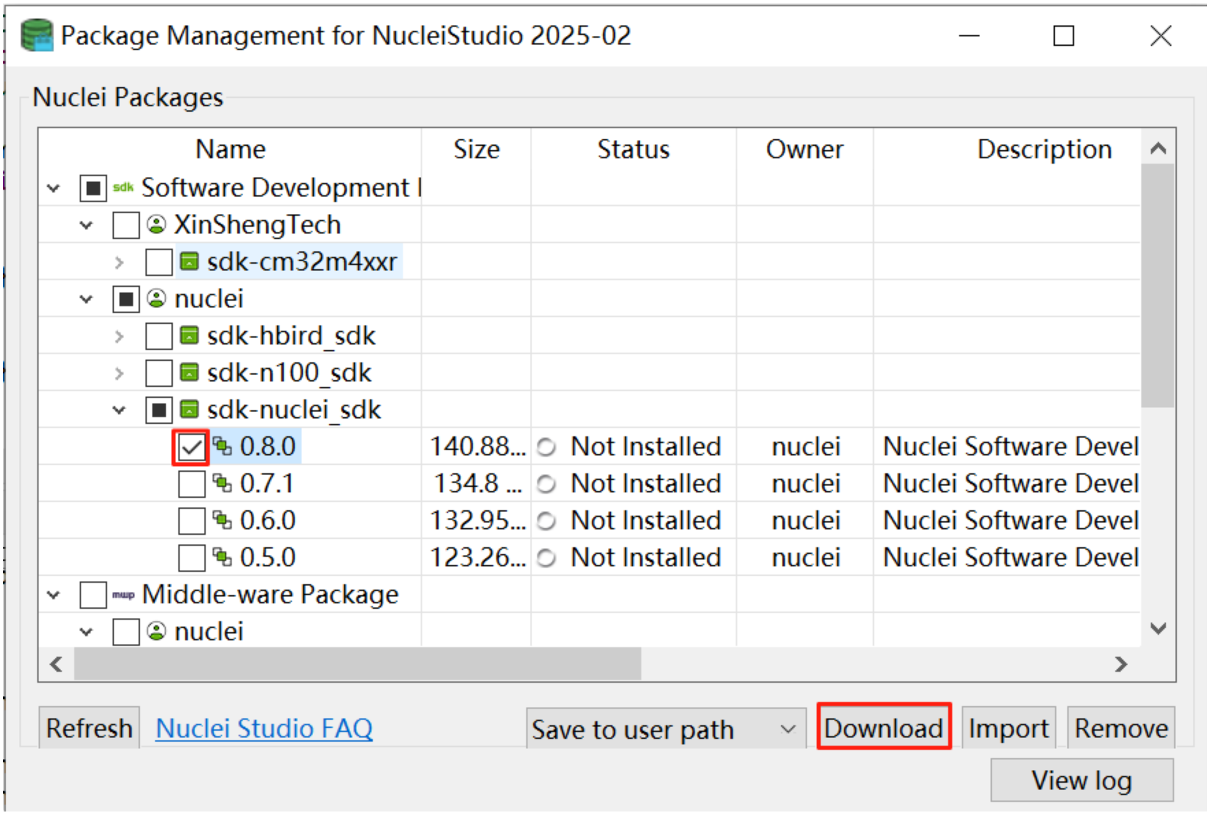
在 Nuclei Studio 中最大的更新，就是将 npk 云端化，用户直接在 Nuclei Studio 中就可以下查看到所有的 npk 并自行安装，在菜单栏选择 `RV-Tools-->Nuclei Package Management` 在弹出的 **Nuclei Package Management** 管理页进行 npk 管理。



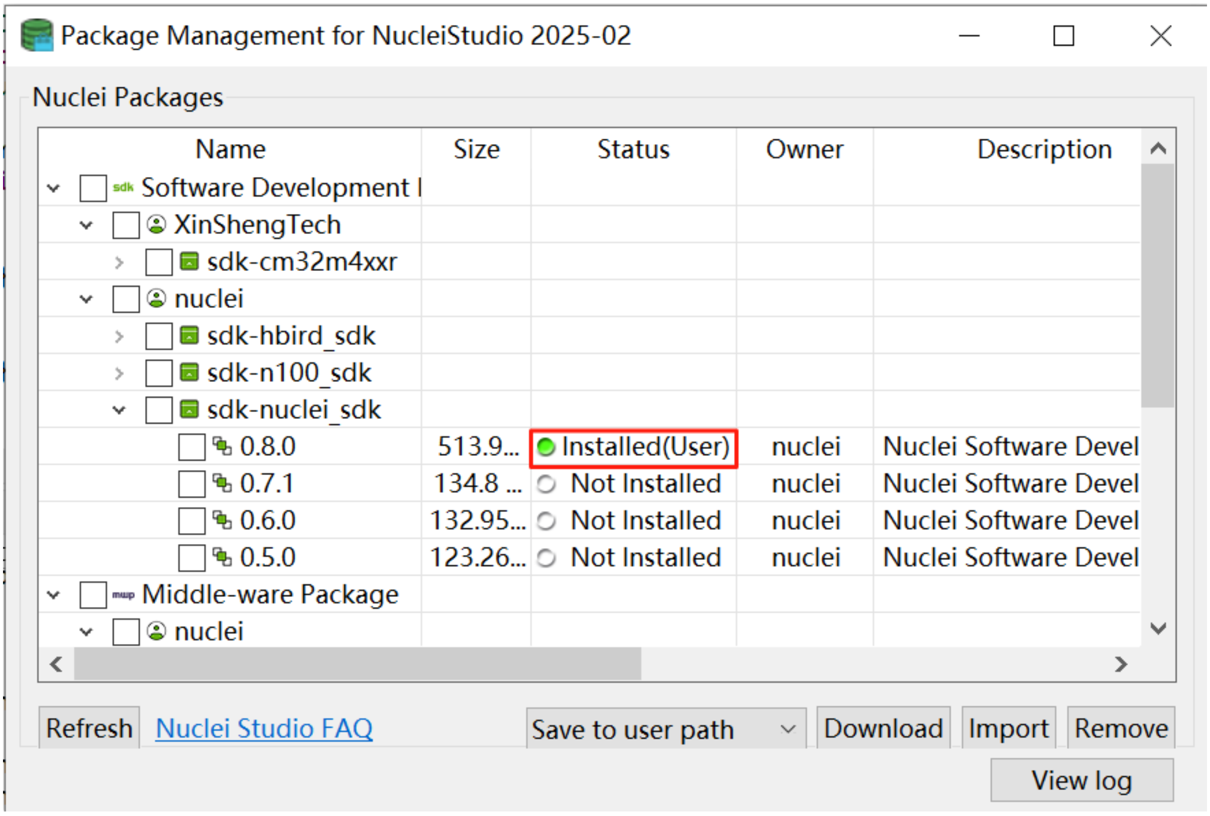
在 Nuclei Package Management 页面，先点击一下 Refresh 获取最新的 npk 信息，npk 安装前状态为 Not Installed。



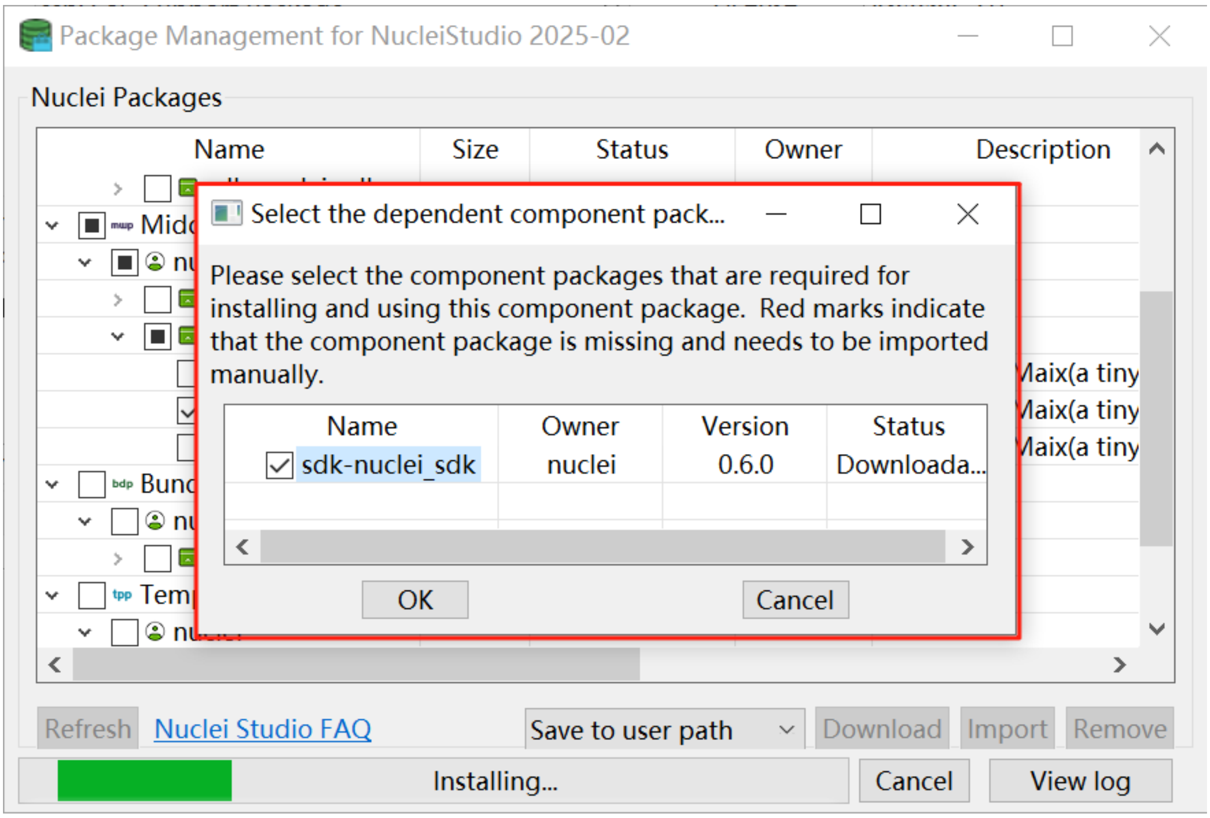
然后选中自己想要安装的 npk, 点击 Download 进行安装，本教程安装 sdk-nuclei_sdk 的 0.8.0 版本，请使用最新版本的 NucleiStudio 进行安装，最新版本支持 gcc14 工具链，之前的版本有支持 gcc13/gcc10。



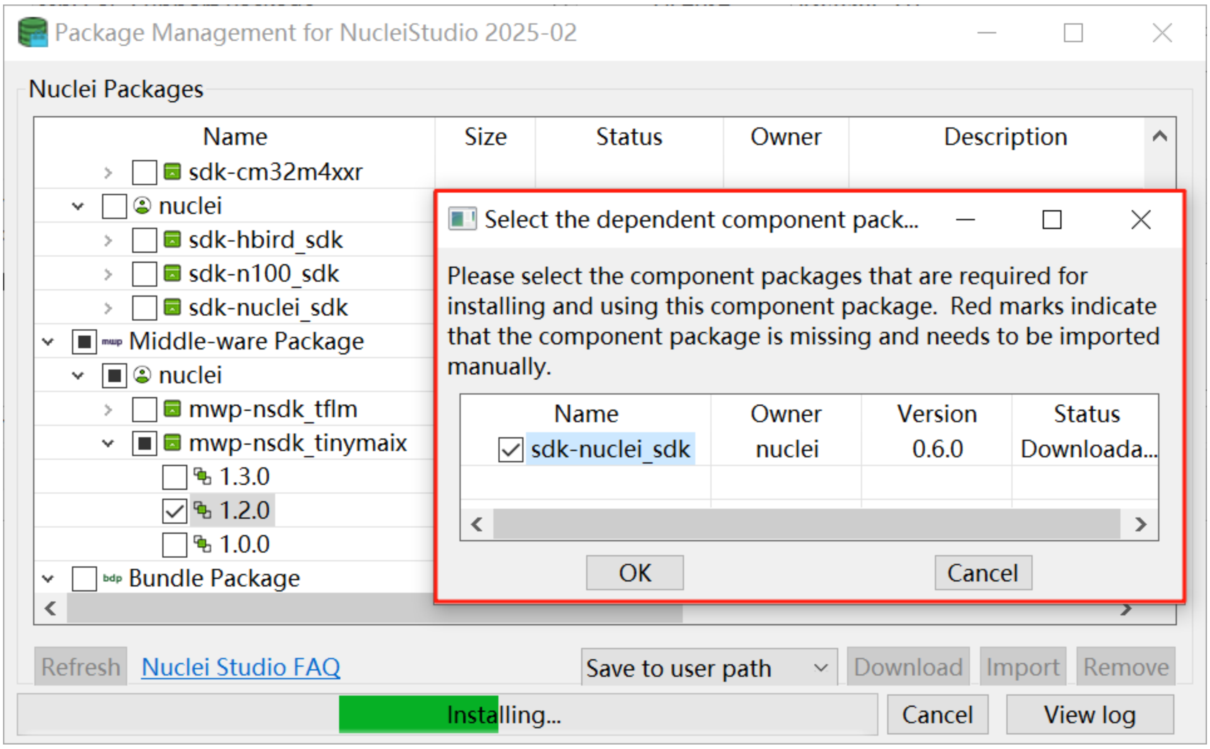
安装完成后 npk 状态变为 Installed。



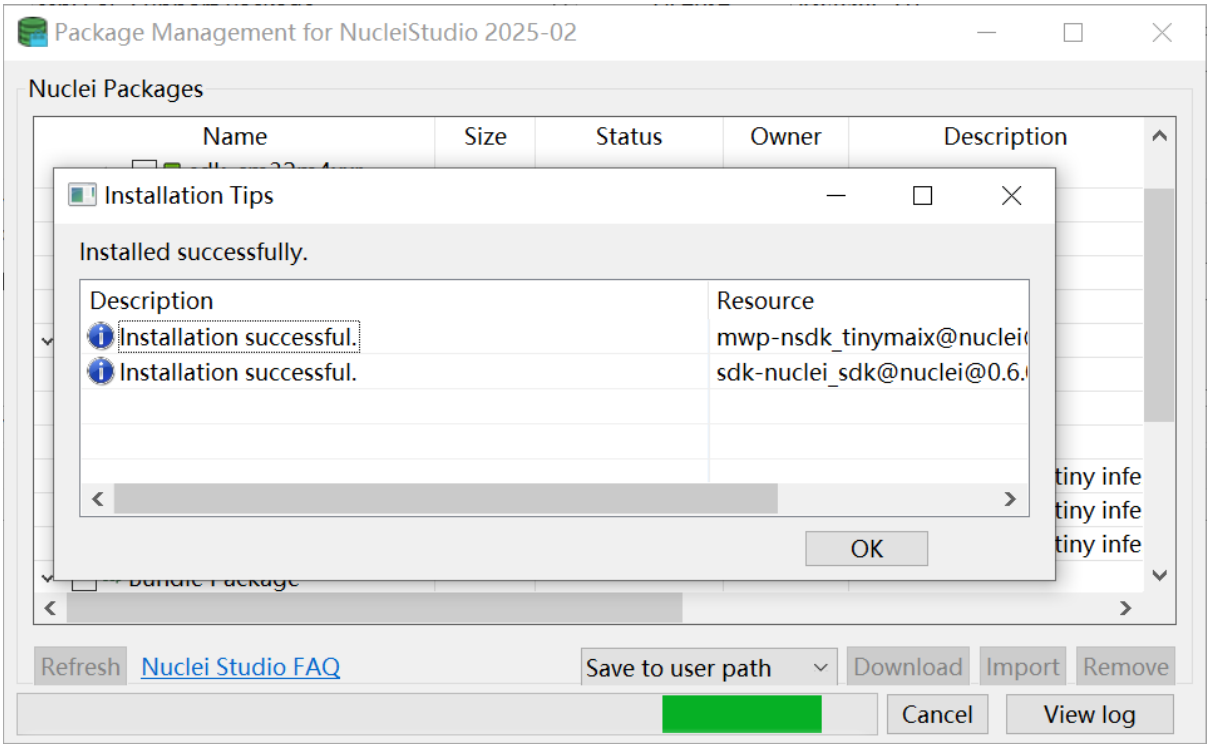
在 NucleiStudio 2024.06 及后续的版本中，升级了 Nuclei Package Management 管理页中关于依赖的管理，大大简化了使用者关于 NPK 包依赖的问题。



当用户在下载/安装某个 NPK 包时，NucleiStudio 会根据当前 NPK 包信息的依赖关系，再结合本地已安装的包信息及云端共享的 NPK 信息，给出其的所有依赖，并提示用户进行关联下载安装。



当 NPK 包安装完成后，会提示用户，此安共计安装了多少个 NPK 包，每个 NPK 包安装的后状态以及出错的原因。方便用户排查问题。极大的解决的之前版中用户因 NPK 包依赖的不正确而无法使用的问题。

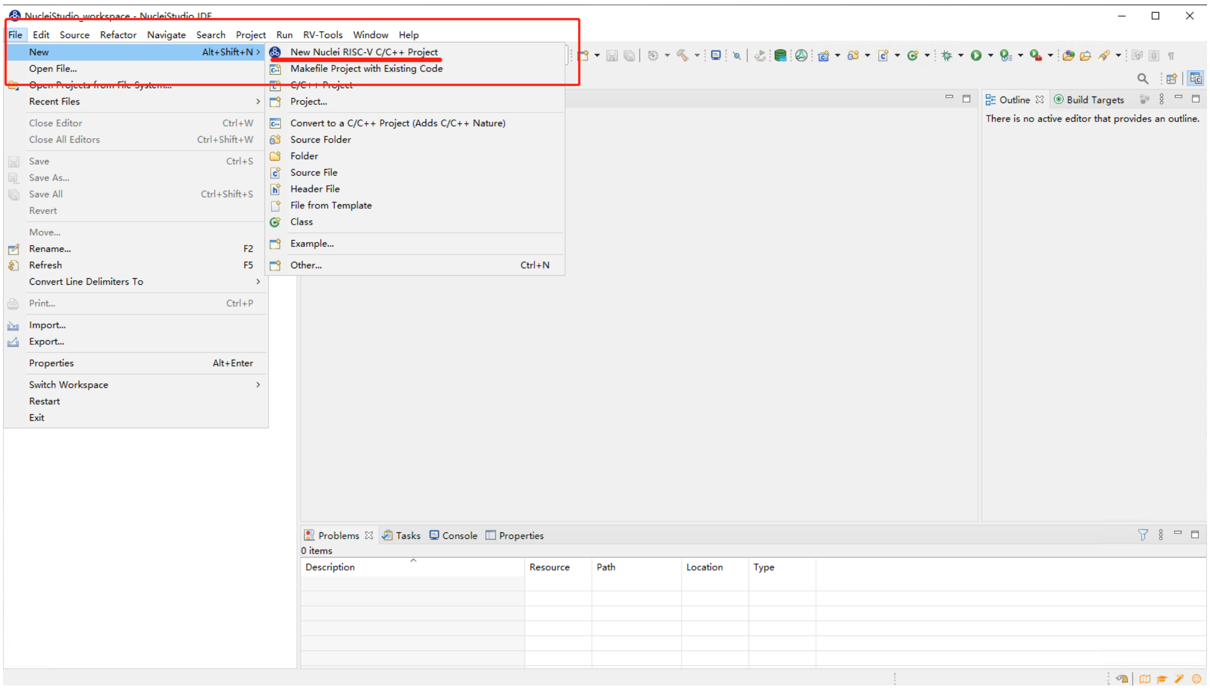


2.6.2 通过 NPK 创建工程

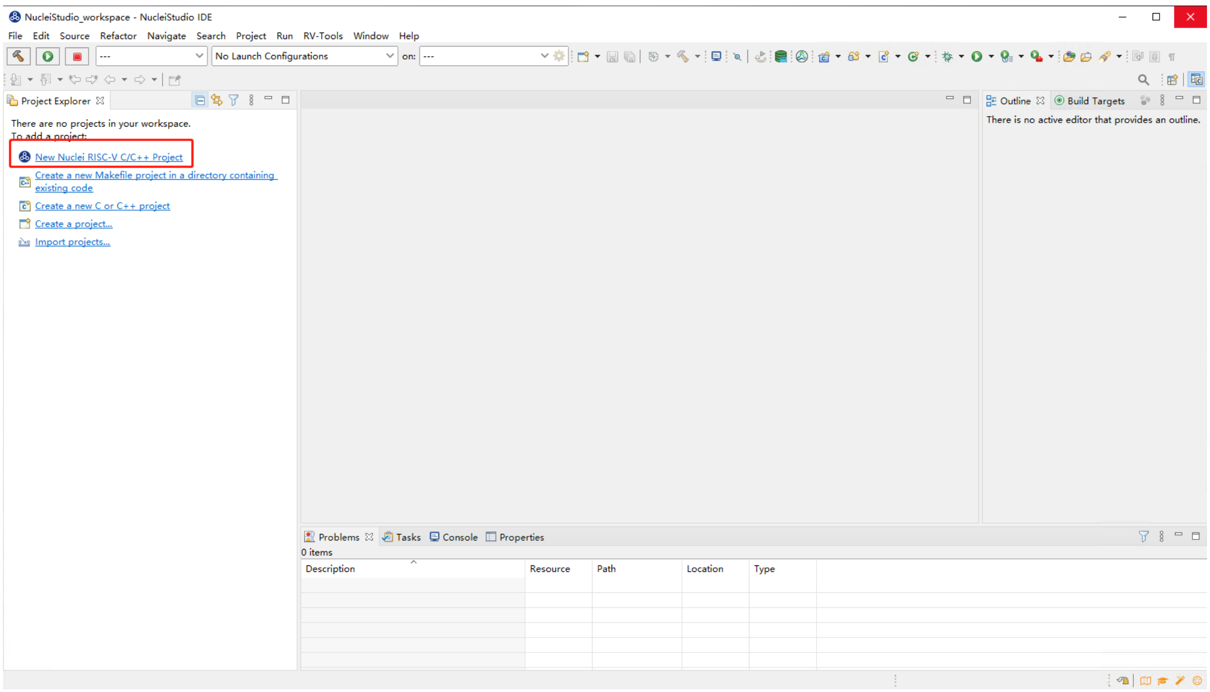
本章将在 RVSTAR 开发板上，以新建和修改 GD32VF103 的工程为例快速介绍 Nuclei Studio 功能，RVSTAR 开发板开发需要使用 nuclei_sdk 的 npk 包，详细的流程请参考之后的章节。

创建 NPK 示例工程

新建一个工程，可以在菜单栏中，选择 `File --> New --> New Nuclei RISC-V C/C++ Project` 。



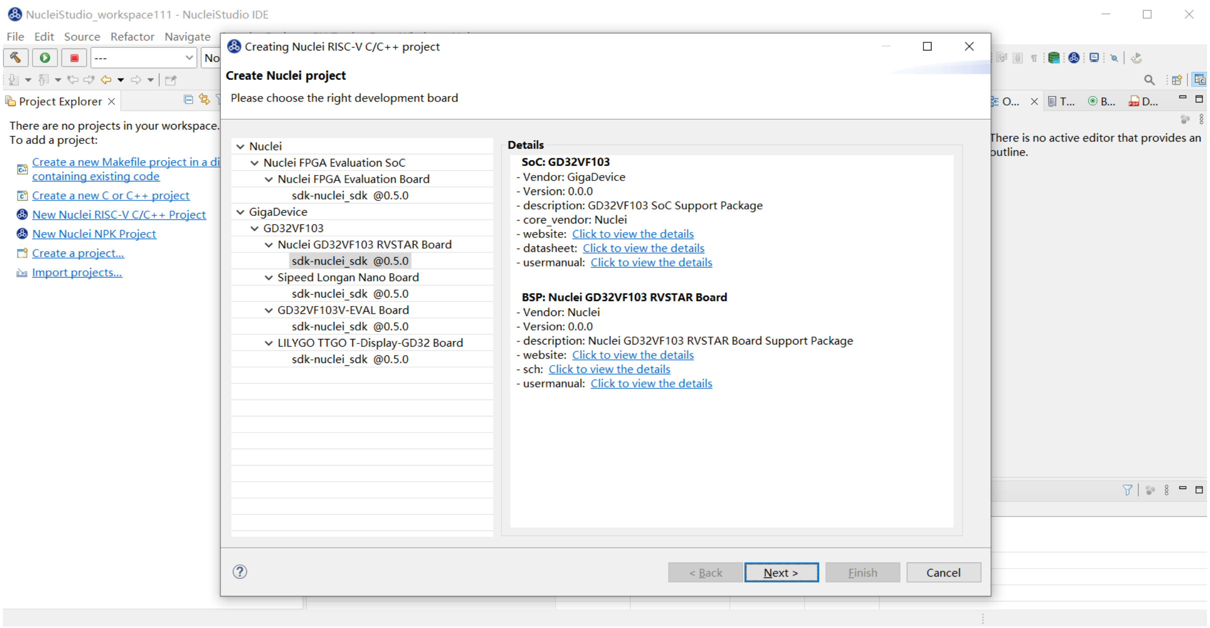
也可以在 Project Explorer 视图中选中 `New Nuclei RISC-V C/C++ Project` 。



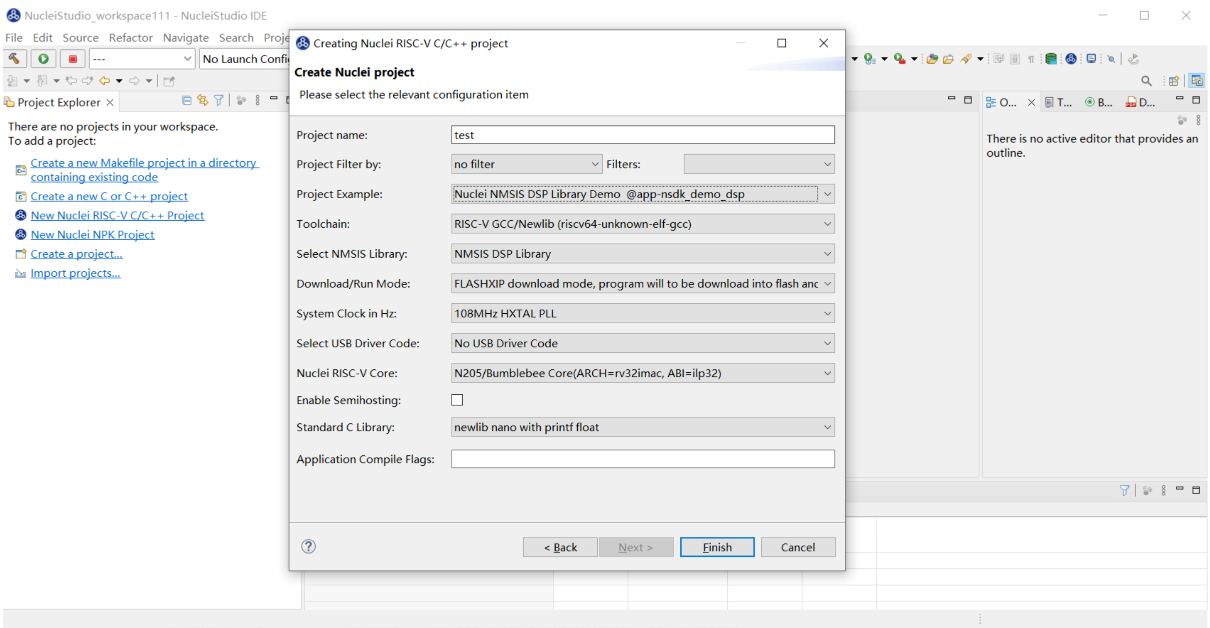
在弹出的窗口中可以不同的厂商提供的不同版本的 SDK，选种某一 Board 下的 SDK，看到相关 SoC 及 Board 的介绍。这里以 RVSTAR 开发板为例，所以这一项选择 `GigaDevice->GD32VF103->Nuclei`

GD32VF103 RVSTAR Board->sdk-nuclei_sdk@0.5.0 。点击 Next 进入下一步。

> **Note**
>
> 注意：这里的 sdk 版本号会随着版本迭代做相应的更新，并且也可能依赖特定版本的 Nuclei Studio 使用



进入具体的项目配置页如图所示，因为 RVSTAR 的内核是固定的 N205，其对应的 arch 和 abi 分别是 rv32imac 和 ilp32，所以 Core 选项不能修改。同样，RVSTAR 开发板仅支持一种 FLASHXIP 下载模式，所以 DOWNLOAD 这一选项也不能修改。点击 Finish 完成工程创建。在 2023.10 版本，增加了对 Arm 项目的支持。



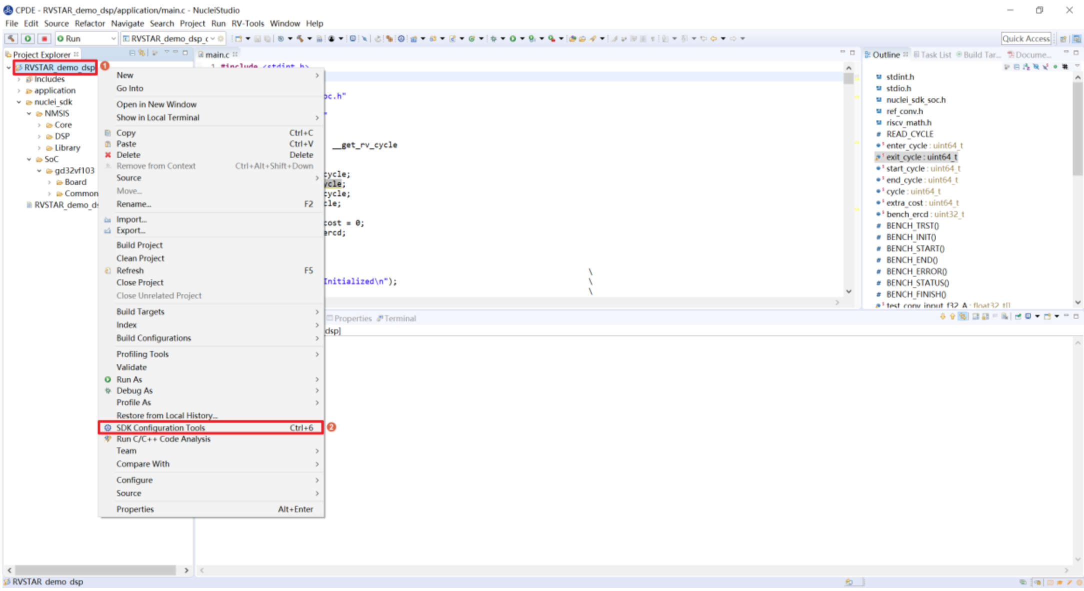
Nuclei Studio 可以根据不同的工程模板添加不同的 SDK 源码，例如 FreeRTOS 模板工程会添加对应的 OS 内容，Demo_dsp 模板工程可以添加 NMSIS 库文件。关于 NMSIS 详细信息请参考（https://doc.nucleisys.com/nmsis/index.html）。这里以 Demo_dsp 为例，Project Example 选择 Nuclei NMSIS DSP Library Demo 。因为使用 dsp 工程，需要添加 NMSIS 库，所以 Libraries 选择 NMSIS DSP Library 。

Nuclei Studio 可以根据新建工程时的选项自动设置工程的选项。这里选择使用浮点打印，所以 `NEWLIB` 选择 `newlib nano with printf float` 。之后一直选择 `Next` 直到 `Finish` 。

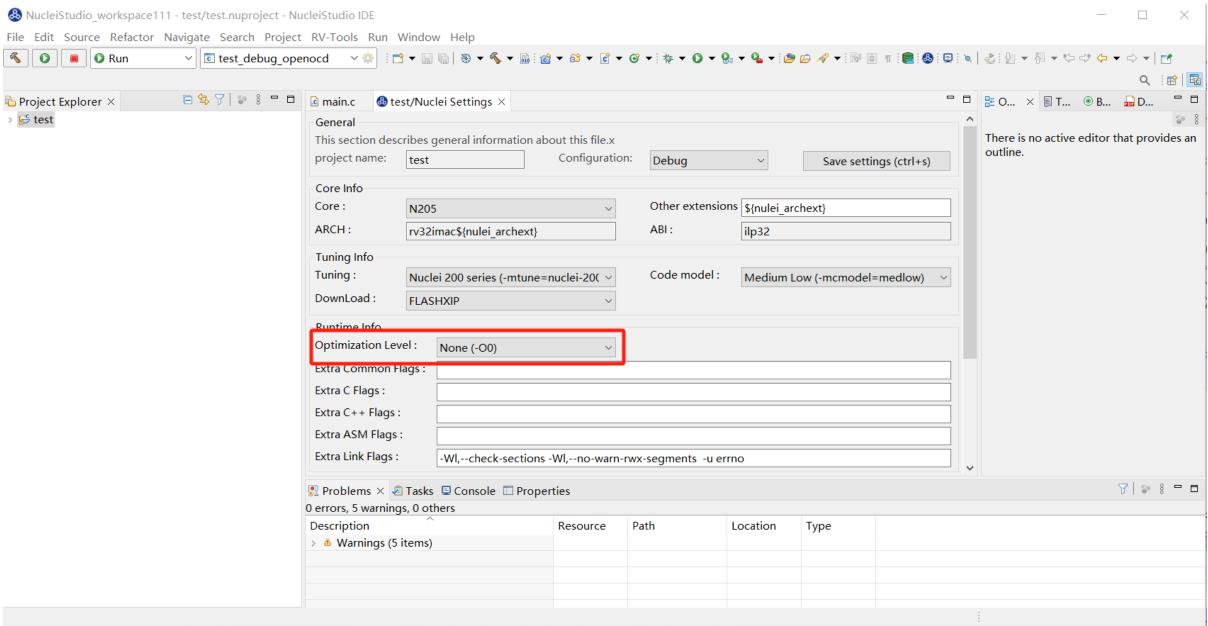
SDK Configuration Tools 更改工程配置

在 Nuclei Studio 可以快速修改工程的设置选项，提供了 `SDK Configuration Tools` 工具，Nuclei Studio IDE 2022.12 版后，对 `SDK Configuration Tools` 工具进行了重构，变更为用户体验更好的 Nuclei Settings 菜单。

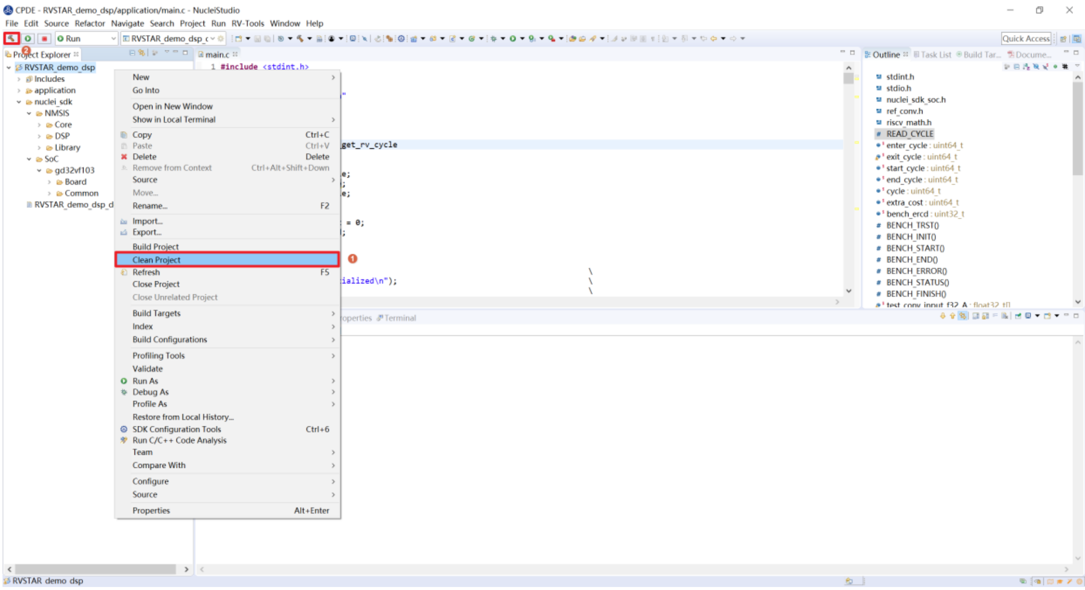
新建好的工程，单击要修改的工程名，右击打开右键菜单，选择 `SDK Configuration Tools` 打开设置选项工具。



如果要修改编译优化等级，修改 `Optimization Level` 为 `None (-O0)`，点击 `Save` 修改选项。

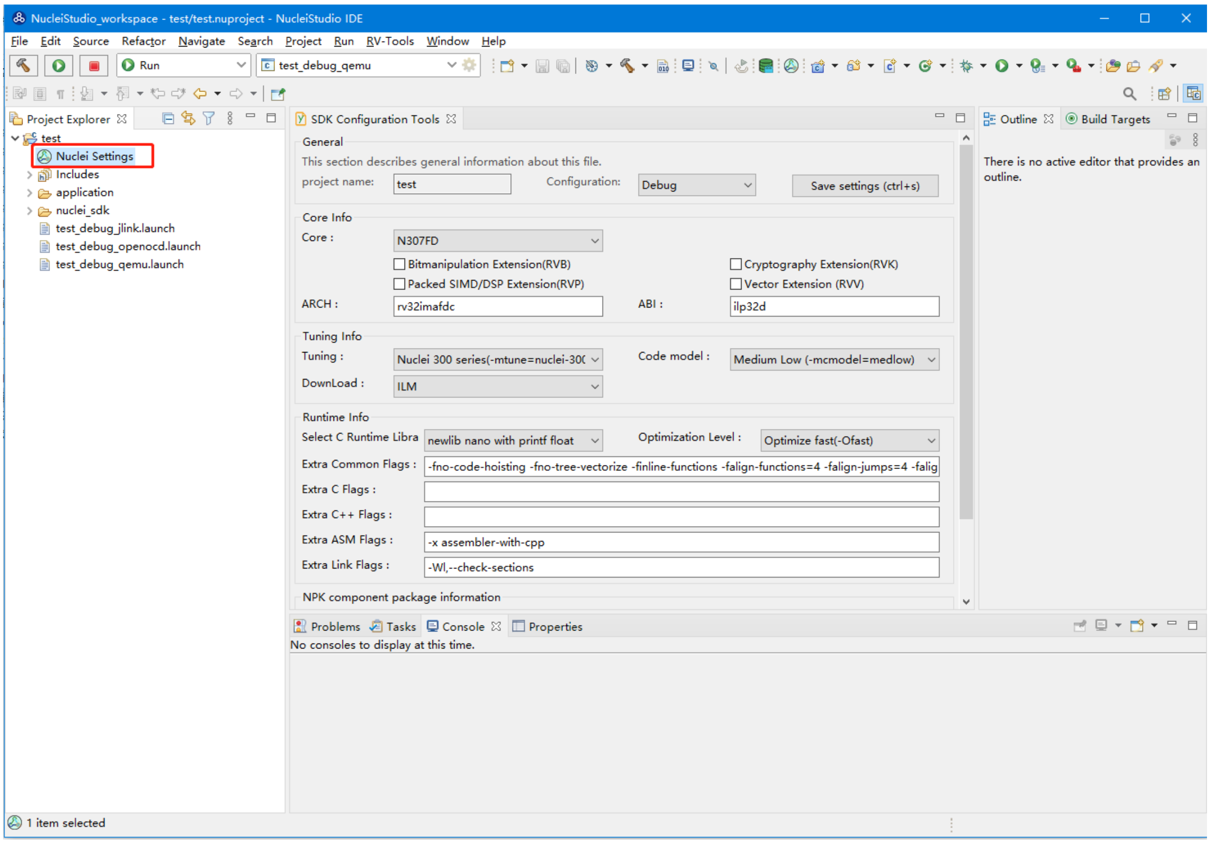


修改成功后在修改后的工程处右击打开右键菜单，选择 `clean` 清除一下工程，再点击锤子图标编译工程。



> **Note**
>
> - 注意： SDK Configuration Tools 修改编译配置后对调试配置（Debug Configurations）不生效，请手动修改对应的调试配置。
> - 注意：后续版本中，将不再维护 `SDK Configuration Tools` 功能，由 Nuclei Settings 菜单功能替代。

为了更好的用户使用体验，Nuclei Studio IDE 2022.12 版对 `SDK Configuration Tools` 进行了重构，新创建的工程中会多一个 Nuclei Settings 菜单，双击 Nuclei Settings 菜单，将打开工程配置工具其在功能上与 `SDK Configuration Tools` 无异，在 2023.10 版本及其后续版本，SDK Configuration Tools 将直接打开这个 Nuclei Settings 界面。



2.6.3 通过 NPK 导入工具

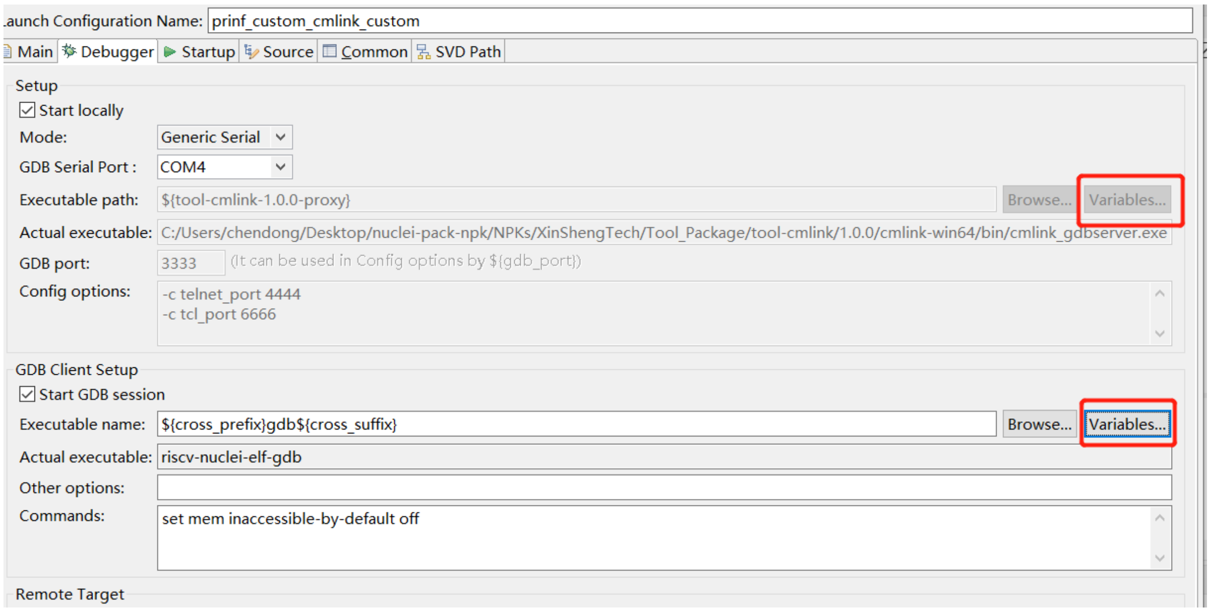
NPK 包除了可以导入 SDK, 还可以方便的导入各种工具包，来扩展 Nuclei Studio 的能力，2022.08 版本的 Nuclei Studio 增加 NPK Tools 的支持，为增加组件包的可扩展性，以及在编译和调试上使用更便捷，增加类型为 tool 的 npk 组件包。tool 组件包可包含 gcc,qemu,cmlink-gdb 等内容，以 zip 包的形式导入到 IDE 去使用。

以 tool-cmlink 包为例，一个工具包中有该工具的执行文件及 npk.yml，开发者在 npk.yml 文件中对该工具做了一些简单的描述，如工具包的开发者、版本、支持的操作系统、可执行文件的路径等，包结构和 npk.yml 内容如下示例。然后将工具包压缩成一个 zip 文件，可以参考导入本地 *NPK* 软件包 (page 76)的内容，将 npk tools 导入到 ide 中，或共享到www.rvmcu.com[7] 网站上。

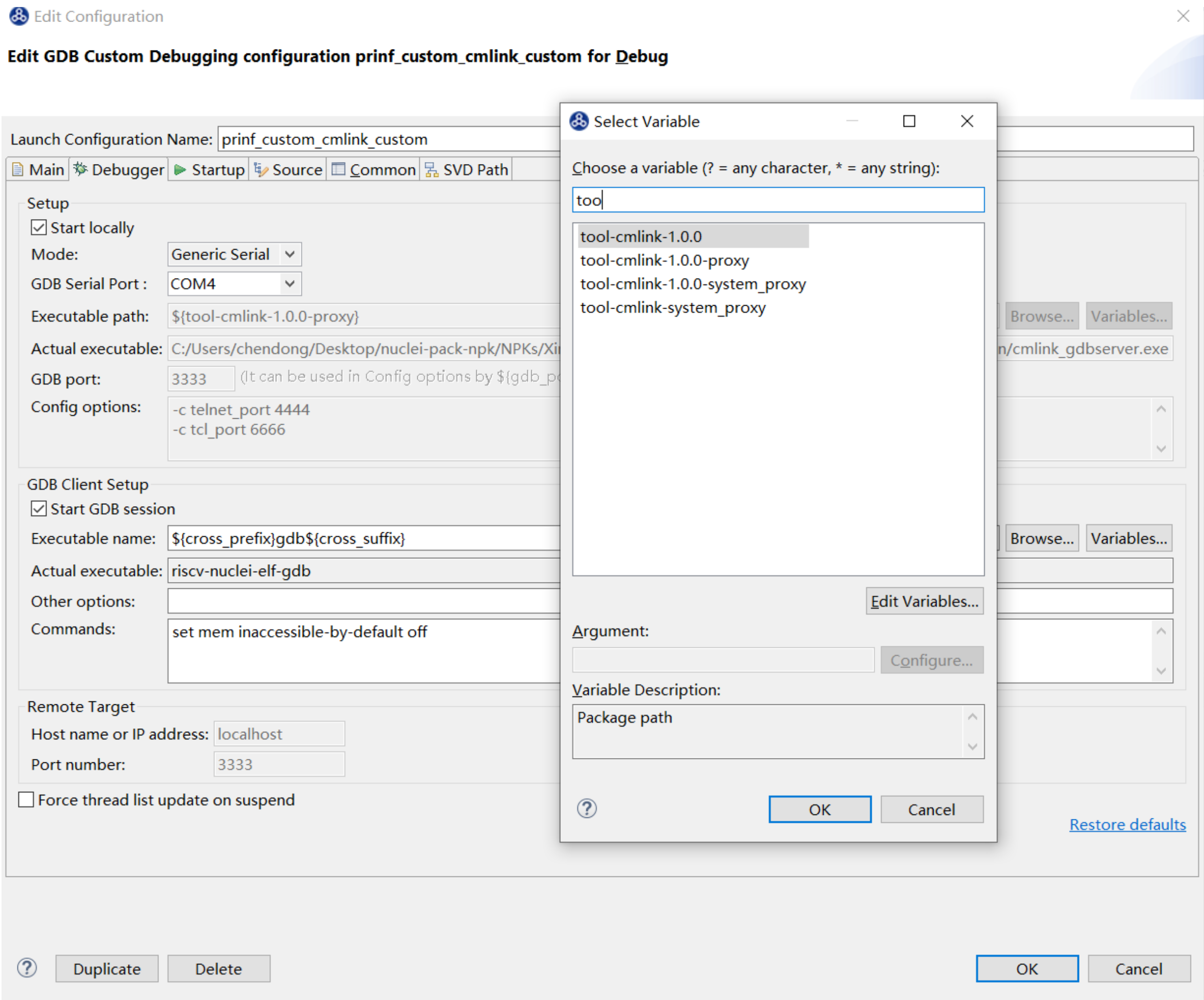
- `bin`
- `bin\cmlink_gdbserver.exe`
- `npk.yml`

[7] http://www.rvmcu.com

在 Nuclei Package Management 管理页中同样可以对 npk tools 进行管理，下载该组件包后，打开任意调试界面，点击 Variables 可以查看到该 npk tools 对应的参数，直接选中对应的参数就可以使用该工具了。



一般我们在 npk tool 中为该组件包扩展变量有 4 个，每个包存在一个包路径，引用为 npk 名称-版本号，例如 `${tool-cmlink-1.0.0}`，其他变量的引用为 npk 名称-版本号-变量名，例如 `${tool-cmlink-1.0.0-proxy}`，`${tool-cmlink-1.0.0-system_proxy}`，当变量的 system 值为 true 时，额外新增一个不带版本号的变量，取最高版本的该变量，例如 `${tool-cmlink-system_proxy}`。



2.7 Nuclei Studio 创建工程

这里以开发板为 Nuclei FPGA Evaluation Board，评估处理器内核为 N300FD(rv32imafdc) 为例，详细介绍 Nuclei Studio 中创建项目的常见方式。

在 Nuclei Studio IDE 创建项目可以有以下几种常见方式：

使用 **NPK** 模板工程自动创建项目：

这是最简单快捷的方式，目前模板项目功能依赖于 Nuclei Studio NPK 功能，在导入对应的 SDK NPK Zip 包，即可在 Nuclei Studio 上进行模板工程的创建。芯来科技提供了 Nuclei SDK、HBird SDK、SoC IP SDK 的 NPK Zip 包，均可导入到 IDE 中进行工程的创建和使用。

从已有项目直接导入创建新项目：

这是最常见的方式，譬如，用户 A 可以将已有项目的文件夹直接进行打包保存，然后进行分享传播，用户 B 可以在另外的电脑上直接导入该项目，从而以此为基础创建新的项目，在此基础上直接使用或者开发修改。

无模板手动创建项目：

这是最繁琐的方式，该方法除了创建项目之外，还需要手动设置各种选项和路径。由于该方式比较繁琐，所以在实际工作中较少使用，但是通过该方式的详细讲解，用户可以详细了解如何配置各中选项和路径。

基于已有的 **Makefile** 创建项目：

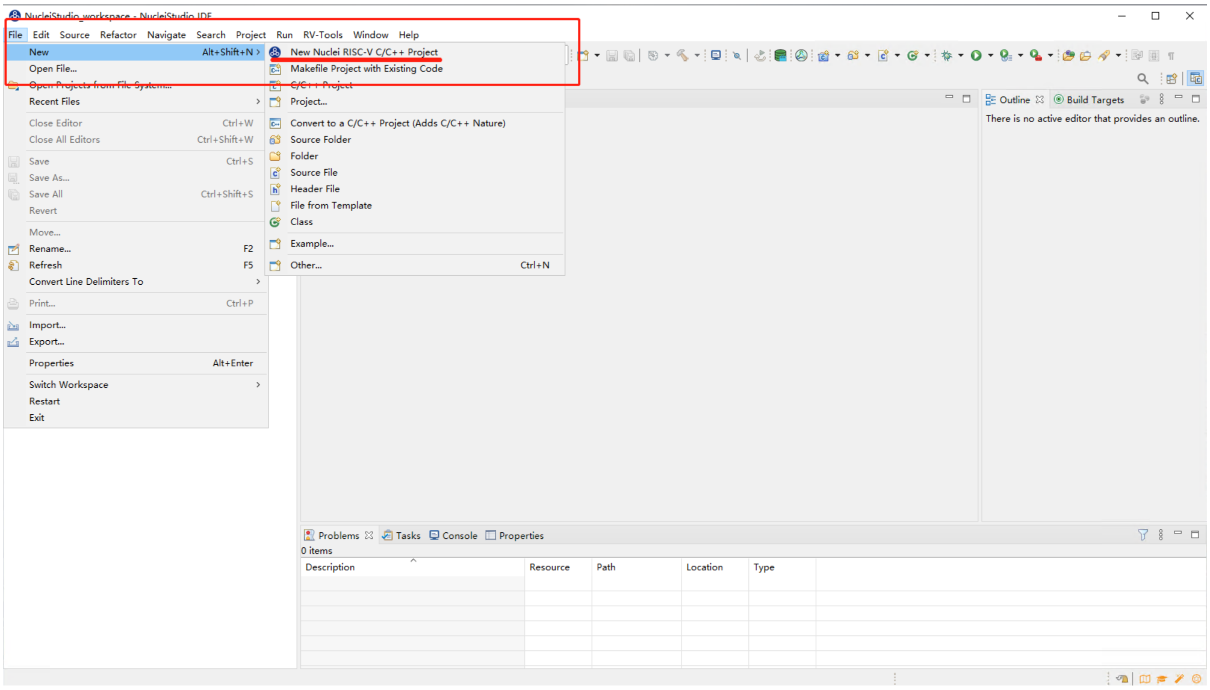
这种方式比较适合于已经采用 Makefile 或者其他编译工具的项目，提供一种在 IDE 中编译工程，清理工程，调试工程的方式。在不修改编译系统的基础上，提供良好的 IDE 调试环境。

下文将对这几种方式分别进行介绍。

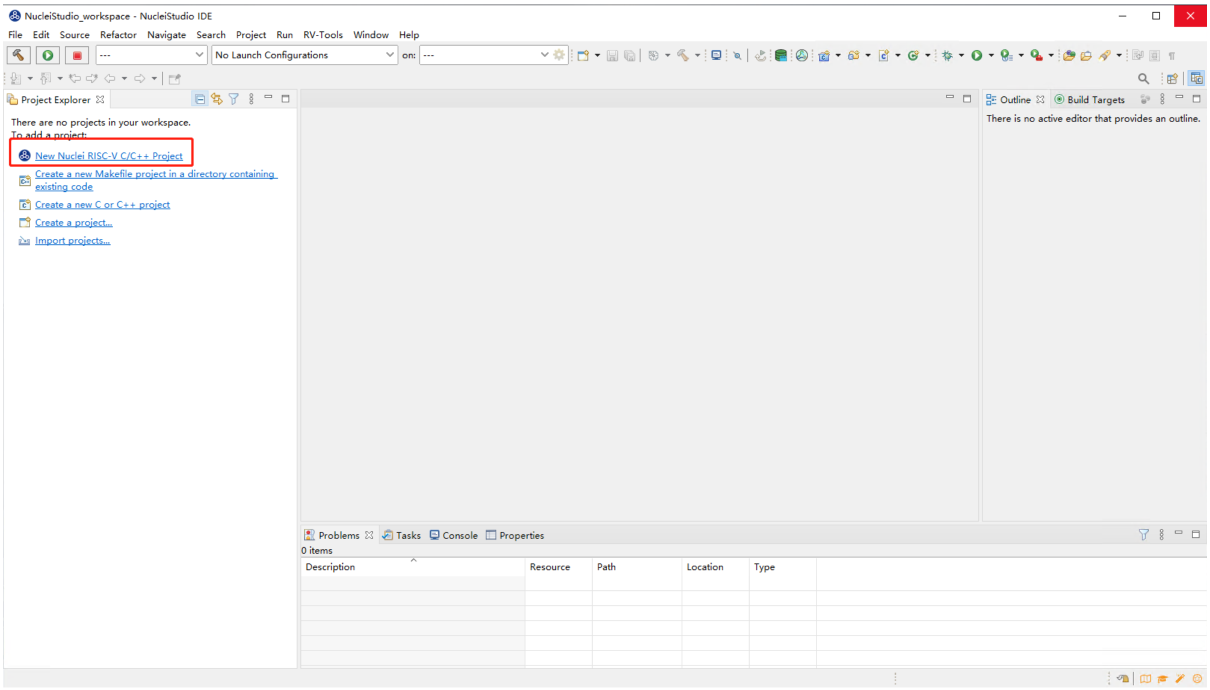
2.7.1 通过 NPK 模板工程自动创建项目

本节将介绍如何使用模板自动创建项目的方式，在 Nuclei Studio IDE 创建一个简单的 Hello World 项目，详细步骤如下。

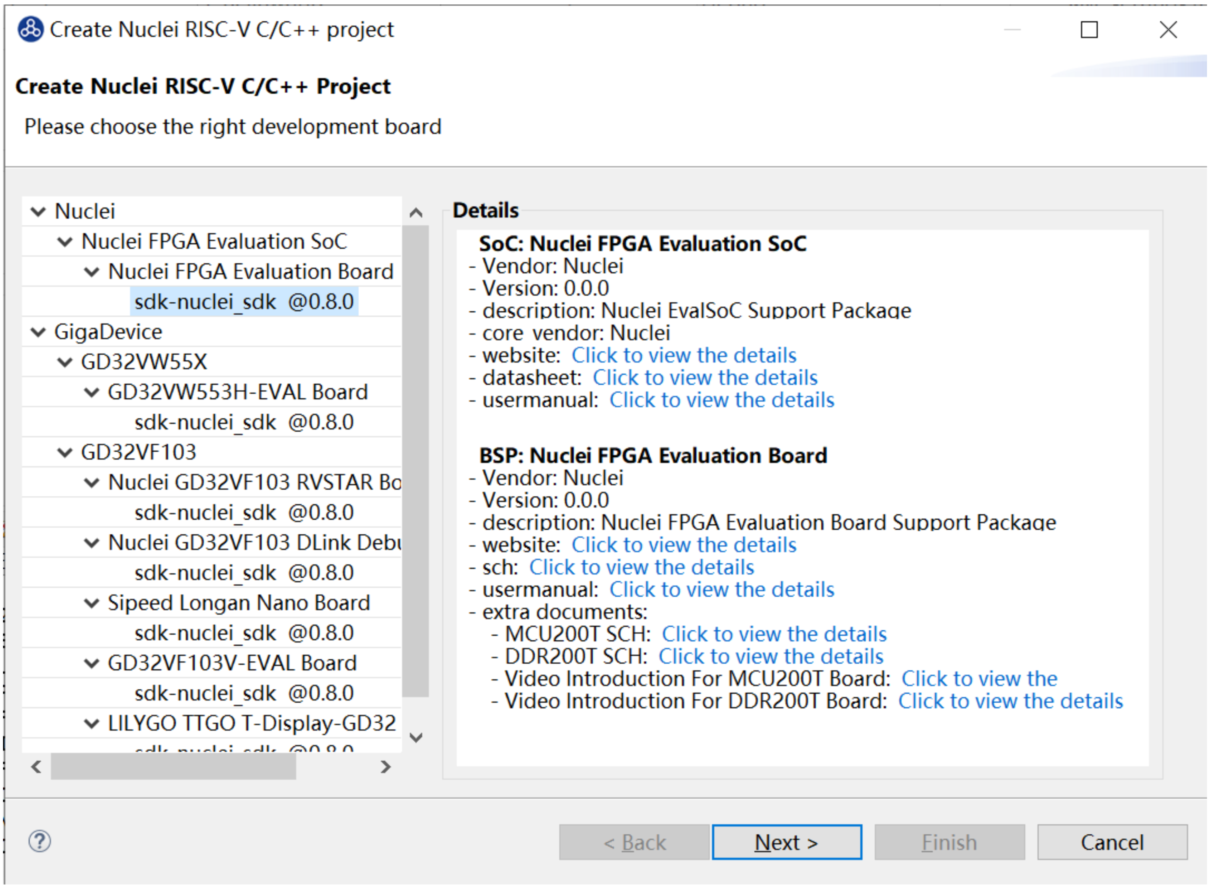
新建一个工程，可以在菜单栏中，选择 `File --> New --> New Nuclei RISC-V C/C++ Project`。



也可以在 Project Explorer 视图中选中 `New Nuclei RISC-V C/C++ Project`。



在弹出的窗口中选择项目类型，这里我们在 Nuclei FPGA 板，内核是 N300FD，SDK 为 nuclei_sdk@0.8.0 版本来做一个测试开发，选对对应的 Board 下的 SDK, 点击 `Next` 进入下一步。



在弹出的窗口中设定如下参数。

- Project name：项目命名。这里设置为 `1_helloworld`
- Project Example：选 `Simple Helloworld Demo`。
- Toolchains：我们使用 `RISC-V GCC/Newlib`。

> **Note**
>
> 注意：此页面是通过 NPK Configuration 字段自动解析并生成的页面，不同的 SDK 或者不同的开发板或者不同的例子都可能会有不同的选项页面，请注意。

我们的内核是 N300FD，所以 `Core` 选择 `N300FD`。

蜂鸟开发板支持三种下载模式，以下为每种下载模式的简介，这里我们选择 ILM 模式。

- **ILM**

ILM 下载模式程序将被直接下载在 MCU 的 ILM 中，并从 ILM 开始执行。ILM 由 SRAM 组成，会掉电丢失。

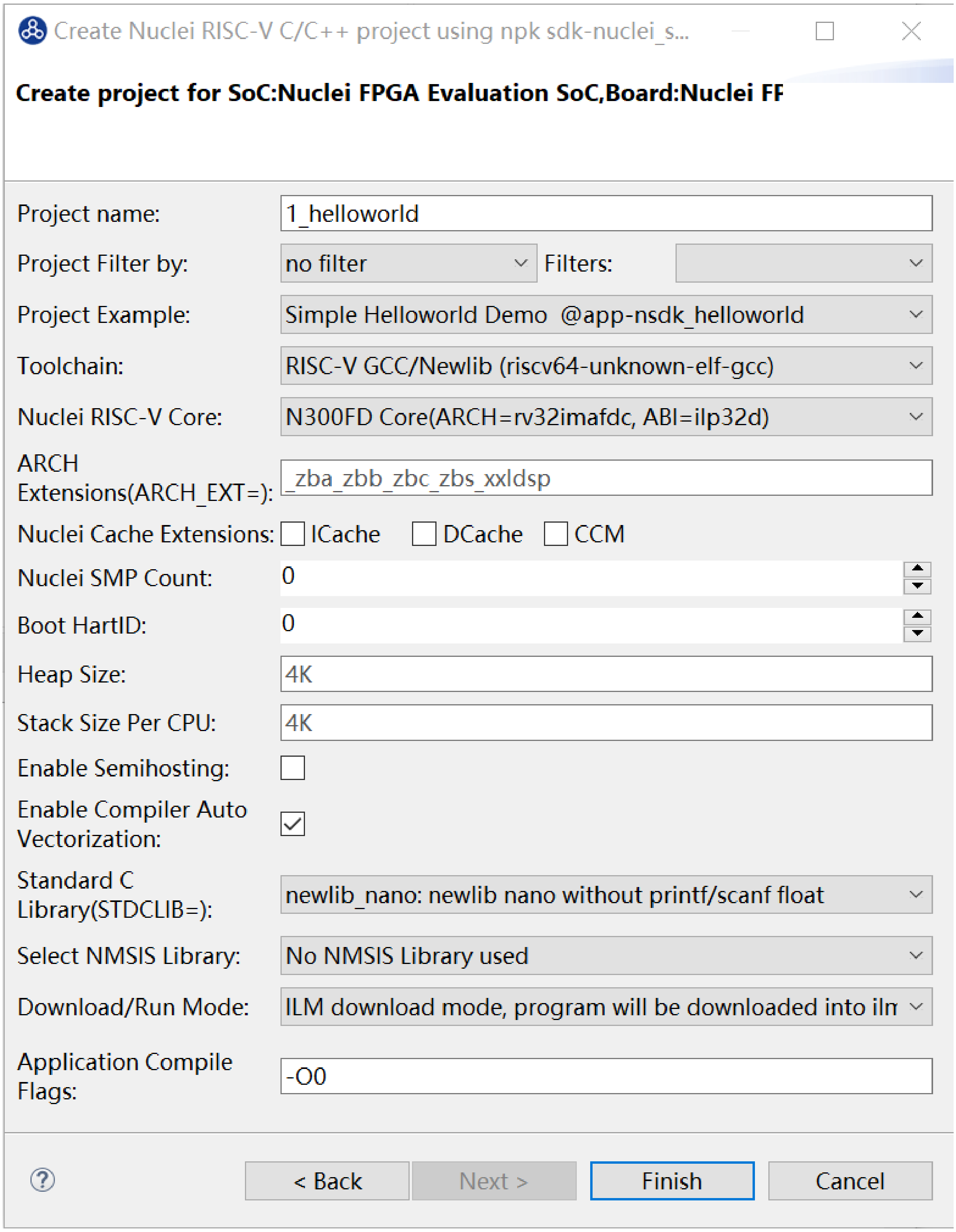
- **FLASH**

FLASH 下载模式程序代码段的物理地址约束 Flash 区间，将代码段的逻辑地址约束在 ILM 的地址区间，意味着程序将被直接下载在 MCU 的 Flash 中，但是上电后要通过引导程序将代码段搬运到 ILM 中，然后从 ILM 中开始执行。程序被烧写在 Flash 中，不会掉电丢失。

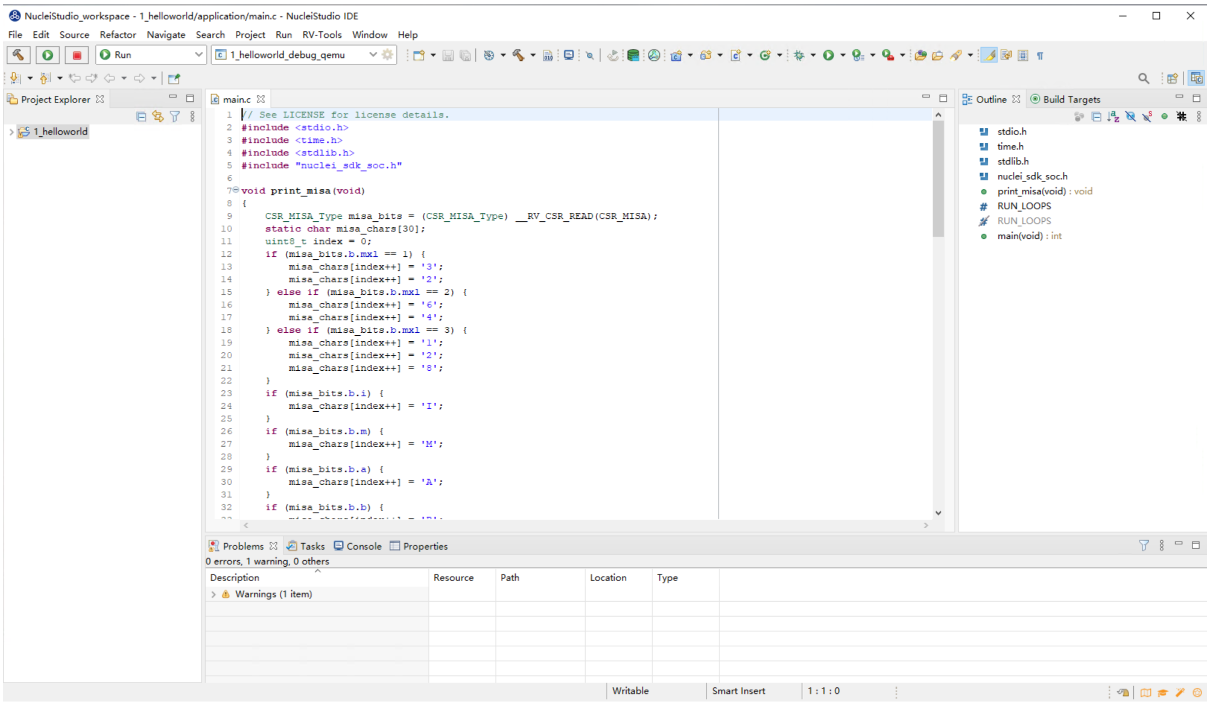
- **FLASHXIP**

FLASHXIP 下载模式程序代码段约束 Flash 区间，意味着程序将被直接下载在 MCU 的 Flash 中，并直接从 Flash 开始执行。程序被烧写在 Flash 中，不会掉电丢失。

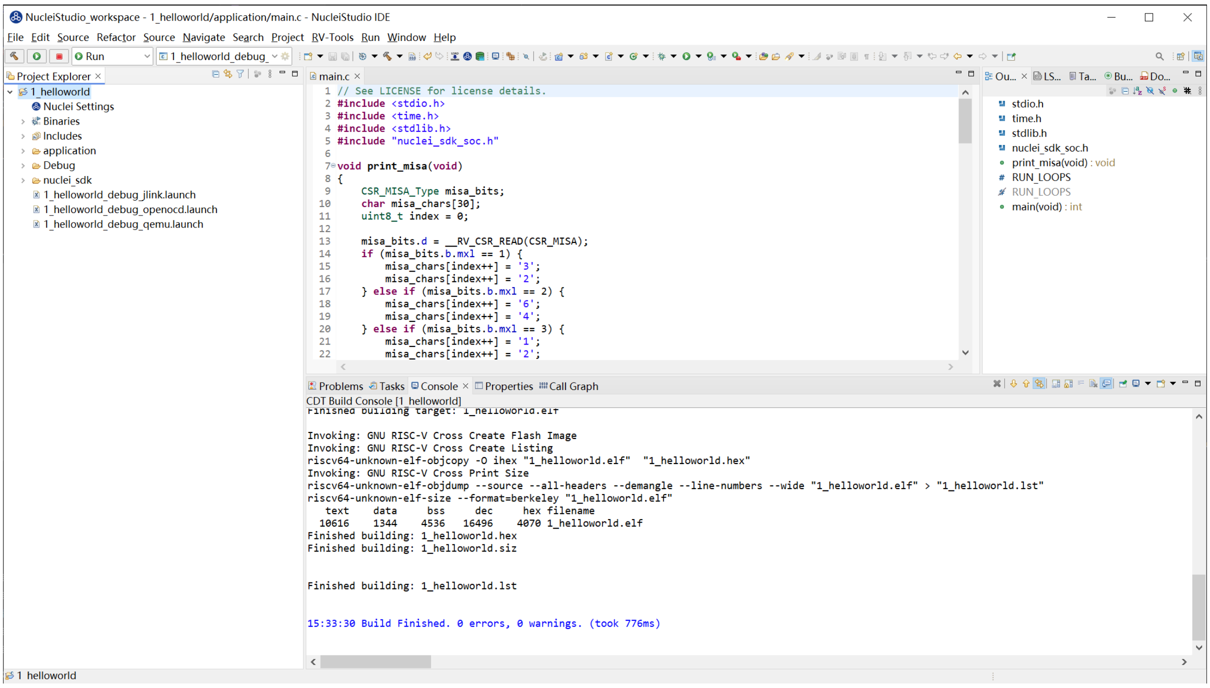
其他各项可以按需进行配置，点击 `Finish` 完成工程创建。



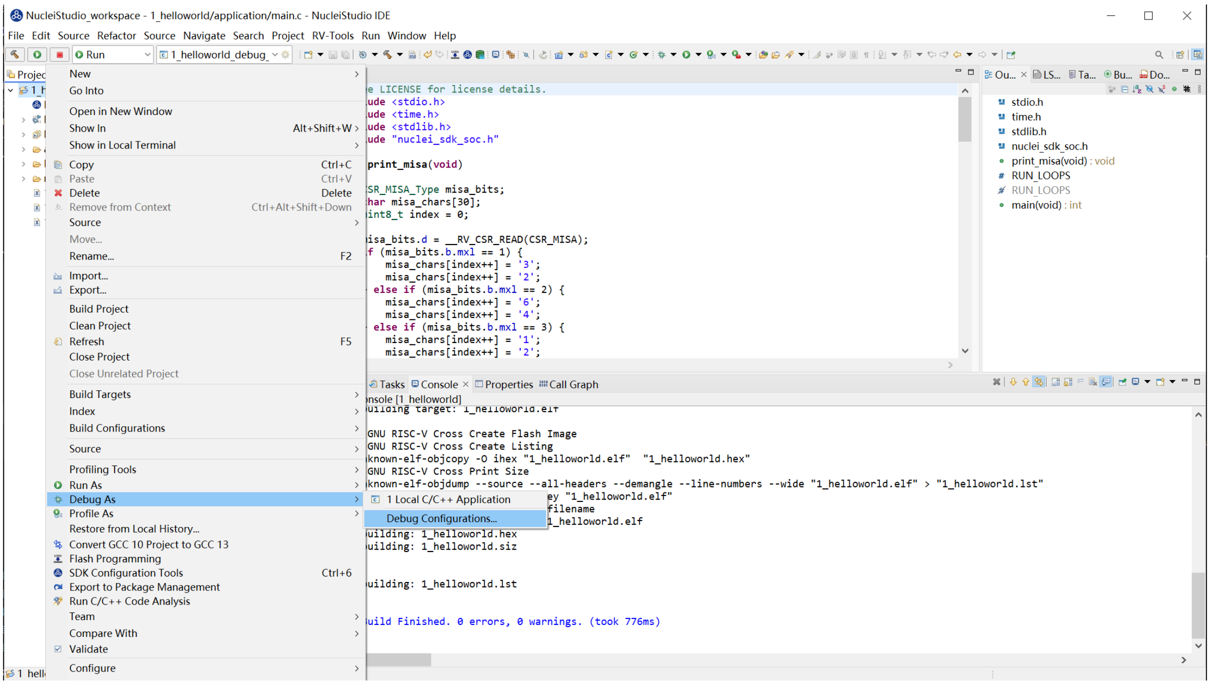
使用模板自动创建 Hello World 项目已经完成。



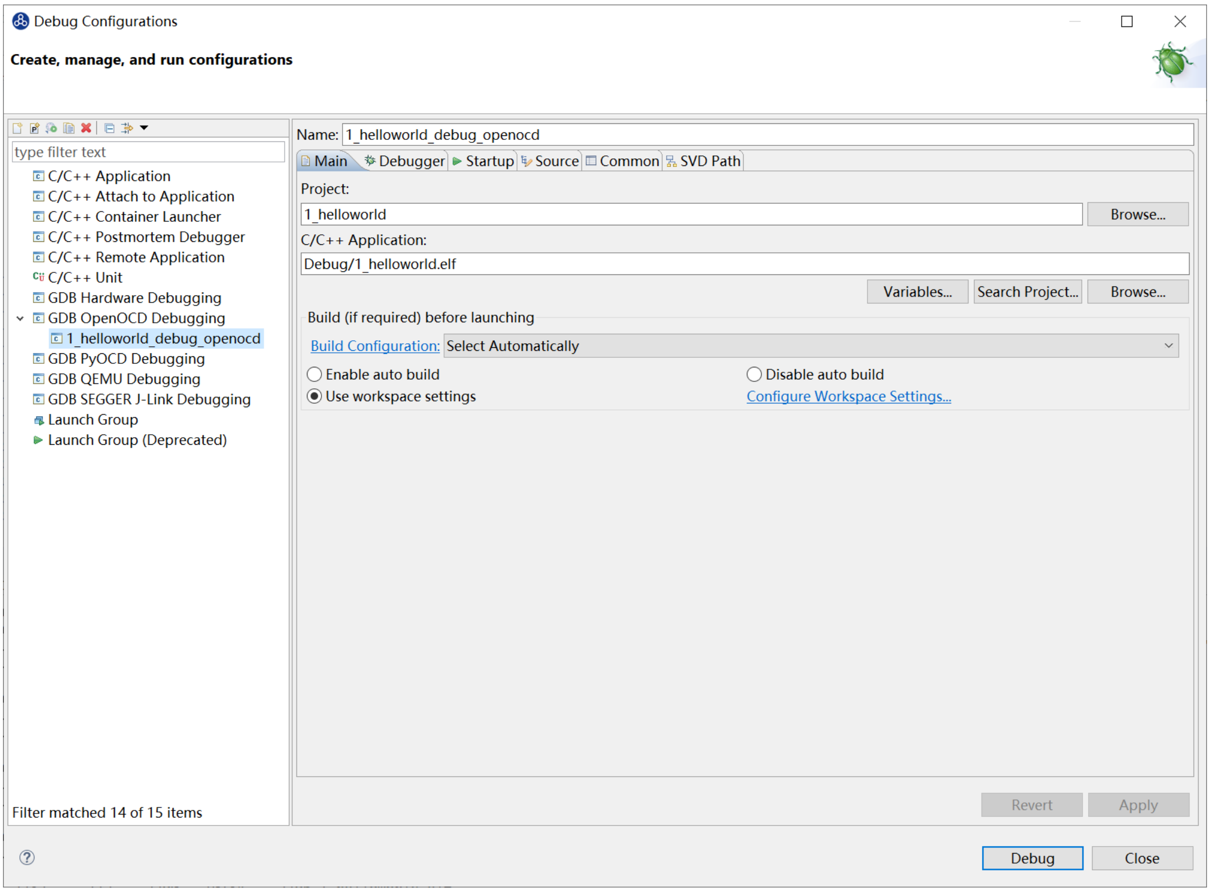
用户可以直接使用菜单栏 Project—> Build Project 或  按钮，来对该项目进行编译。



在 Hello World 项目自动生成过程中，其对应的 OpenOCD 配置已经同步完成。在项目编译完毕后，用户可以右键点击项目列表 Hello World，点击 Debug As —>Debug Configurations 开启调试配置面板进行查看。Debug 与 Run 使用相同的配置文件，所以也可通过 Run As —>Run Configurations 打开。



用于调试使用的配置文件 Hello World_Debug_OpenOCD 已经自动生成。关于使用芯来蜂鸟调试器结合 OpenOCD 进行下载和调试的方法，可以查看*使用蜂鸟调试器结合 OpenOCD 调试运行项目* (page 145) 进行详细了解。

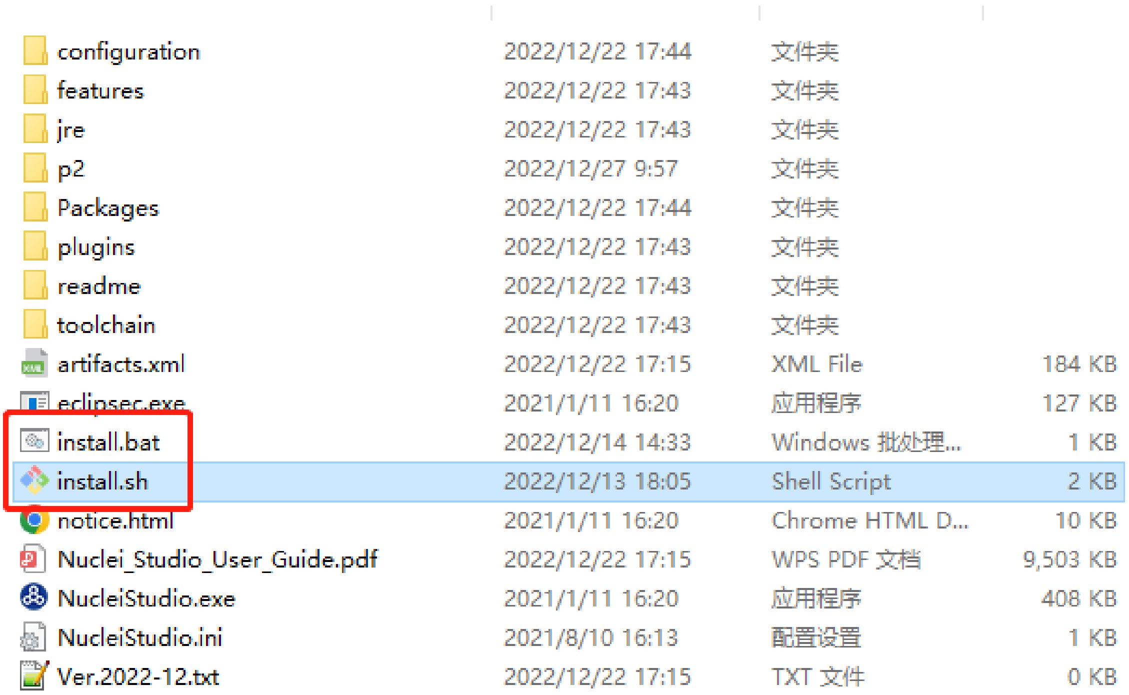


2.7.2 通过应用关联文件导入工程

本节将介绍如何通过 Nuclei Studio IDE 的关联文件，将一个 Hello World 项目导入到 IDE 中。Nuclei Studio IDE 2022.12 版中，新增了应用文件关联的支持，可以将 IDE 的启动路径注册到系统注册表中，然后通过特定的文件，可以实现 IDE 的启动、工程导入等功能，极大的方便了 Nuclei Studio IDE 用户在使用过程中，对工程的分享和快速导入。

将 Nuclei Studio IDE 写入到注册表

在 Nuclei Studio IDE 2022.12 及之后的版本，在安装包中多了两个文件 `install.bat/install.sh`，在 windows 系统下，双击 `install.bat`，因为这里需要写入注册表，所以需要一个用户授权，授权后安装成功；在 linux 系统下，需要在 shell 命令下执行 `install.sh` 文件。



`install.sh` 文件在运行后，有一个用户授权的界面，同意授权。

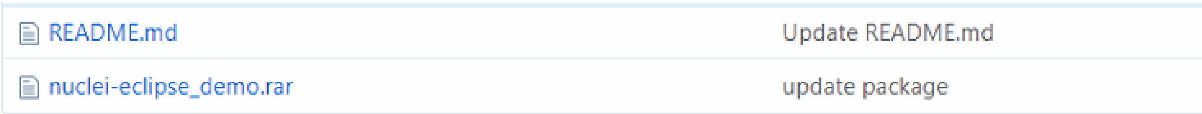
通过应用关联文件导入工程

使用 Nuclei Studio IDE 2022.12 及之后的版本创建工程 `test`，在工程中会有一应用关联文件 `test.nuproject`，如果 ide 的启动路径已写入注册表，双点 `test.nuproject` 文件，系统会自动启动 Nuclei Studio IDE 并将 test 工程导入到 IDE 中。

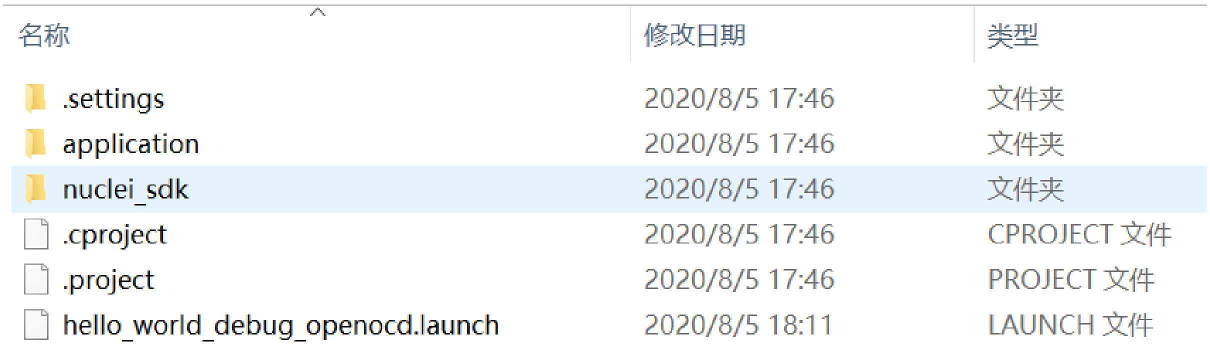
2.7.3 从已有项目直接导入创建新项目

本节将介绍如何使用 IDE 从已有项目直接导入创建新项目，本文以 N307 的项目包为例进行导入，项目包存放在（https://github.com/riscv-mcu/Nuclei-Studio_IDE-Project-Package）。如需其它项目包请与芯来科技联系。

在基于 Windows 的 Nuclei Studio IDE 开发环境中，如果用户使用 无模板手动创建工程，也需要加载此项目包中的 nuclei-sdk 文件夹，相关内容会在无模板手动创建项目 (page 103) 中具体介绍。



将 nuclei-eclipse_demo.rar 压缩包下载解压后，内容分别为：

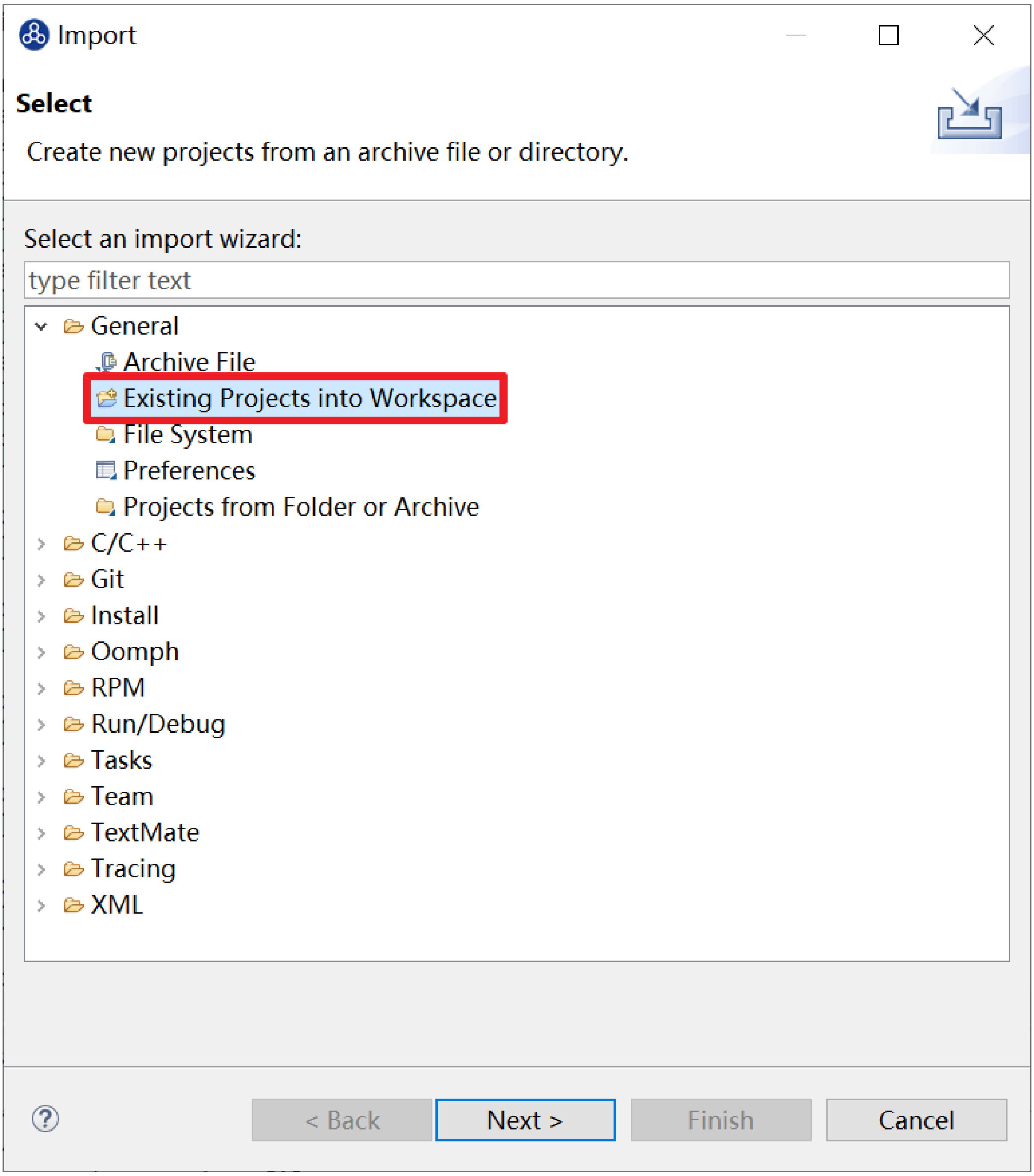


- 项目包的描述文件 `.setting`，`.project` 和 `.cproject`
- 项目包的 Debug 设置文件 `*.launch`
- nuclei_sdk 文件夹

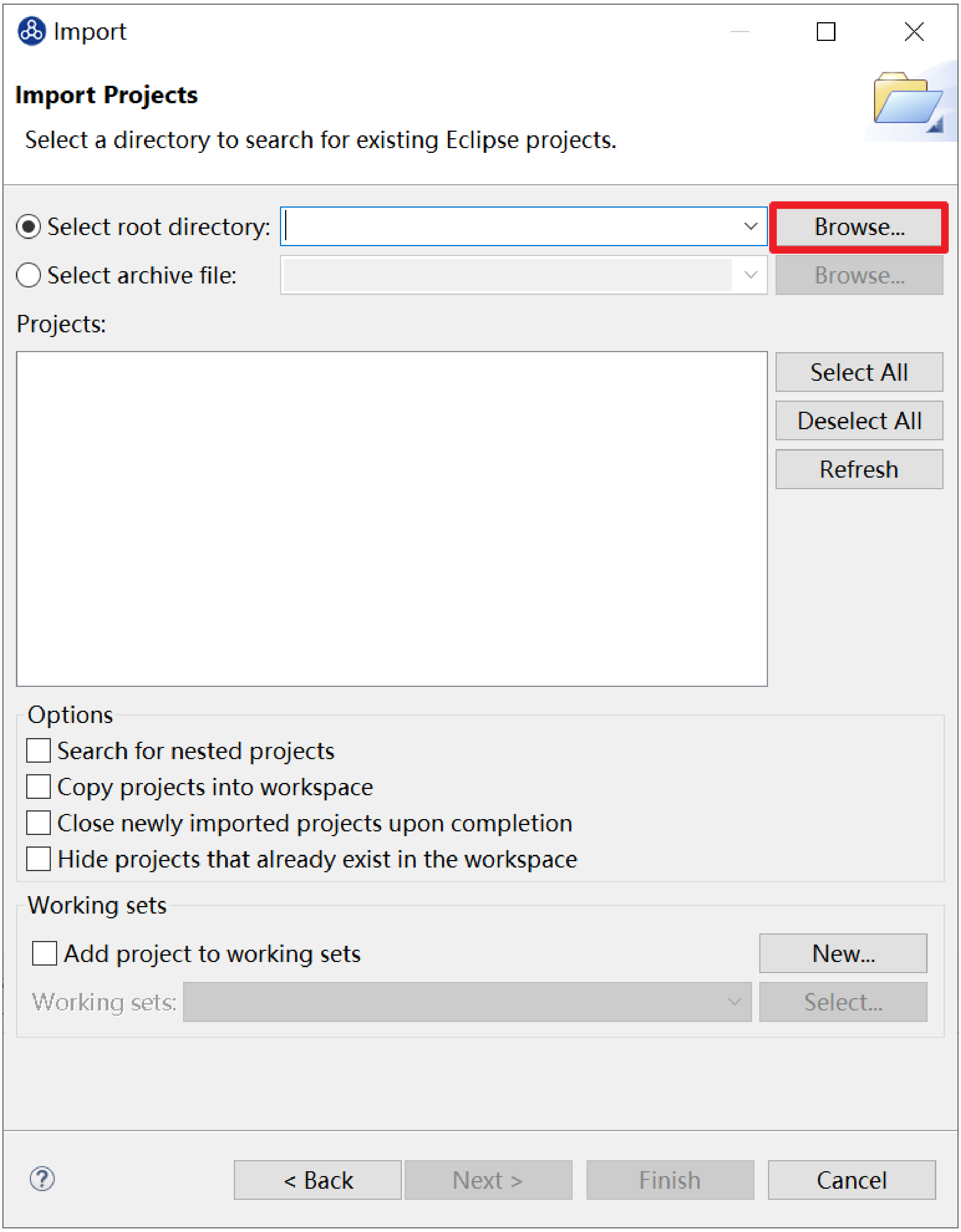
  该文件夹下存放部分 SDK 源代码。
- application 文件夹

  此文件夹包含 hello_world 样例程序的 main 函数源代码。

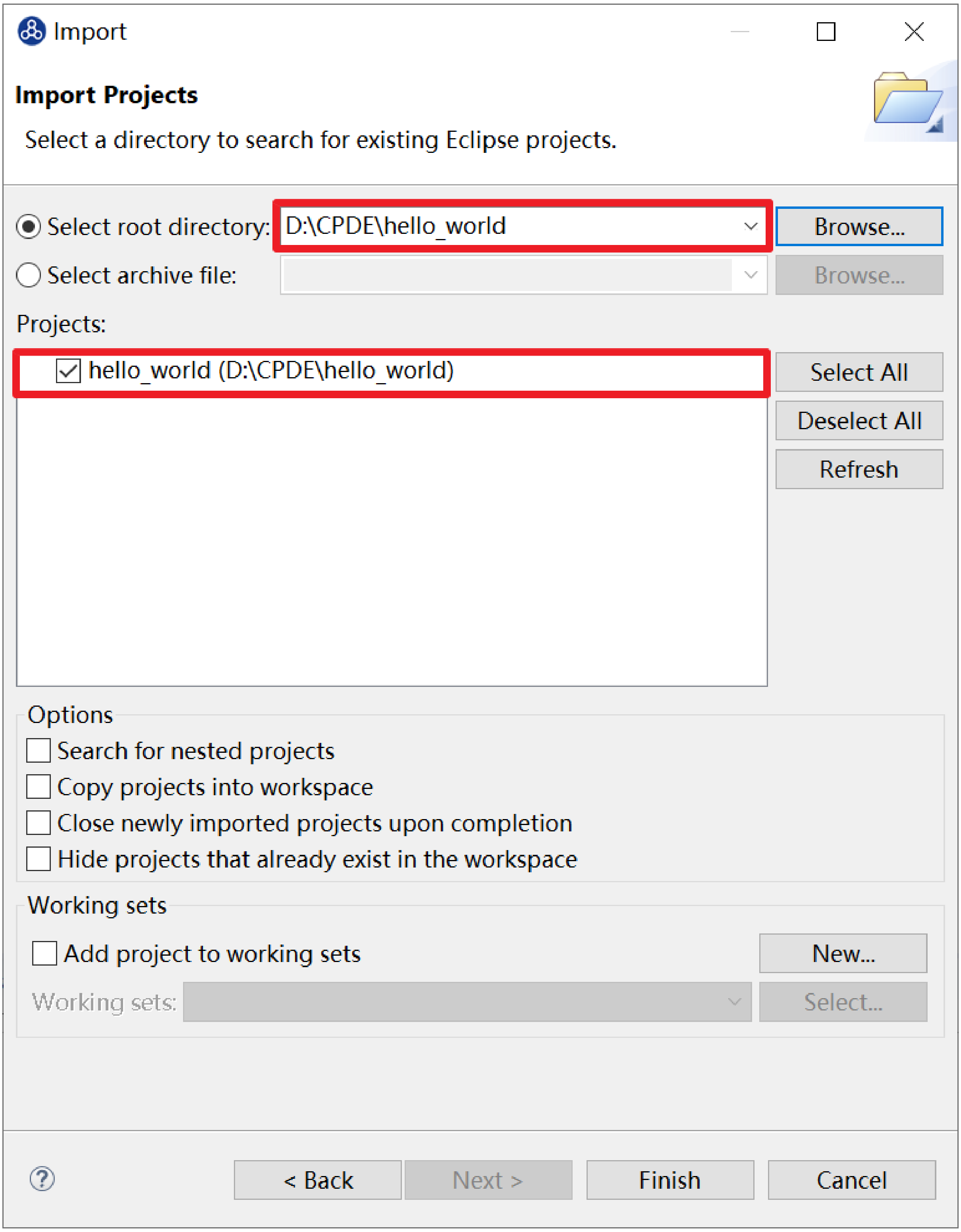
  下一步导入下载好的项目包，导入步骤如下：
- 在菜单栏中选择 `File—>import`。
- 如图所示，选择 `Existing Project into WorkSpace` 后，点击 `Next`。



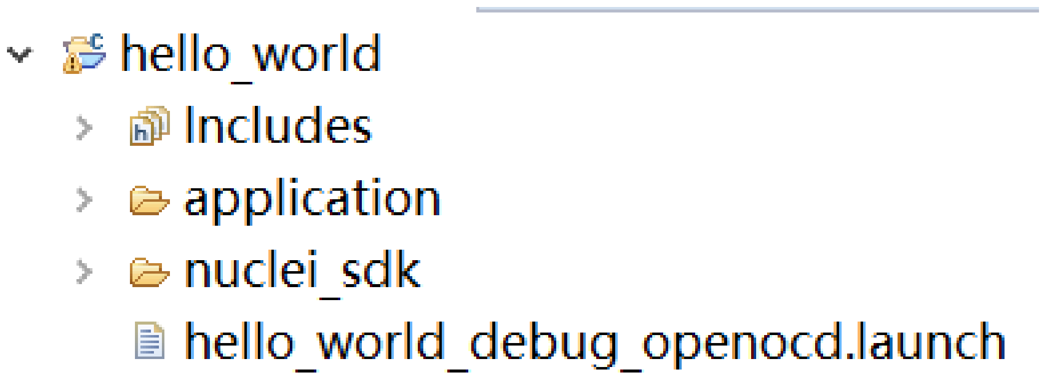
- 点击 `Browse`，选择需要导入的项目路径，如图所示。



- 需要的导入的项目成功被 IDE 识别，点击 `Finish` 。



- 在 IDE 的项目资源管理器中显示导入项目的目录结构如下图所示。已有项目默认为 N307 的编译选项，Nuclei SDK 仅包含 helloworld 使用到的文件。需要更多的 Nuclei SDK 源码请访问 Github（https://github.com/riscv-mcu/hbird-sdk）获取源码。



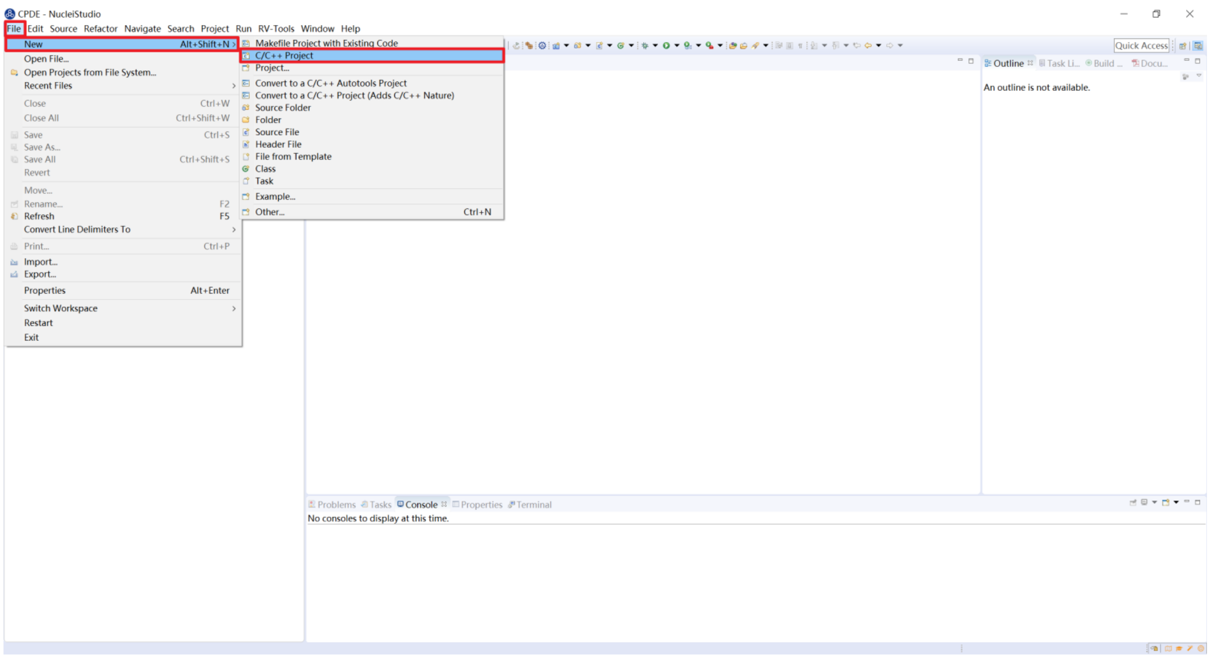
2.7.4 无模板手动创建项目

本节将介绍如何使用手动方式在 Nuclei Studio IDE 创建一个用户自定义的 Hello World 项目。开发板为 Nuclei FPGA Evaluation Board，内核为 N307。该方法除了创建项目之外，还需要手动设置各种选项和路径，详细步骤如下。

> **Note**
>
> 不建议使用，建议使用 NPK 模板的方式创建工程

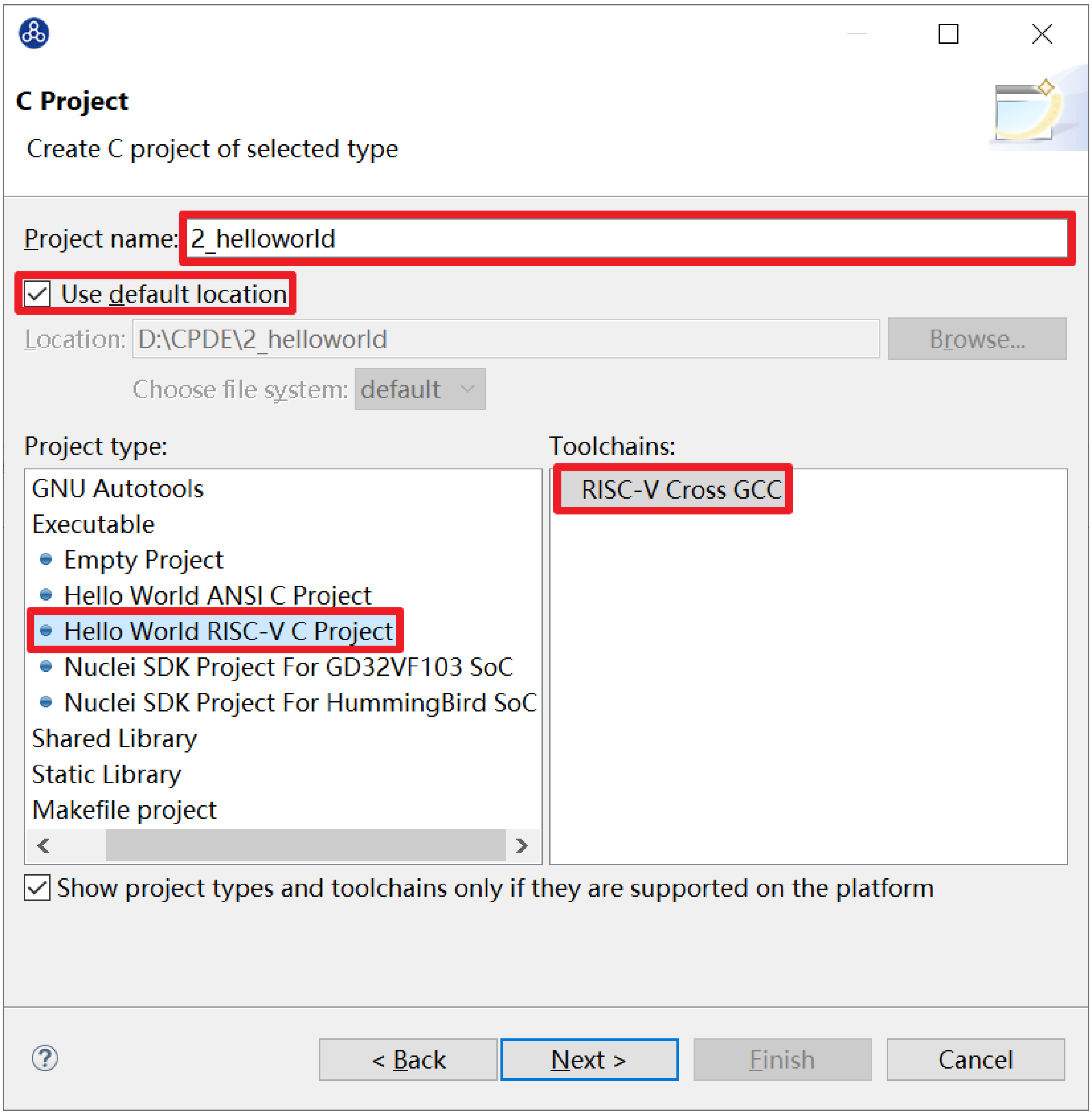
手动创建项目

在 Nuclei Studio 的主菜单栏中，依次选择 `File—> New —> C/C++ Project` 。

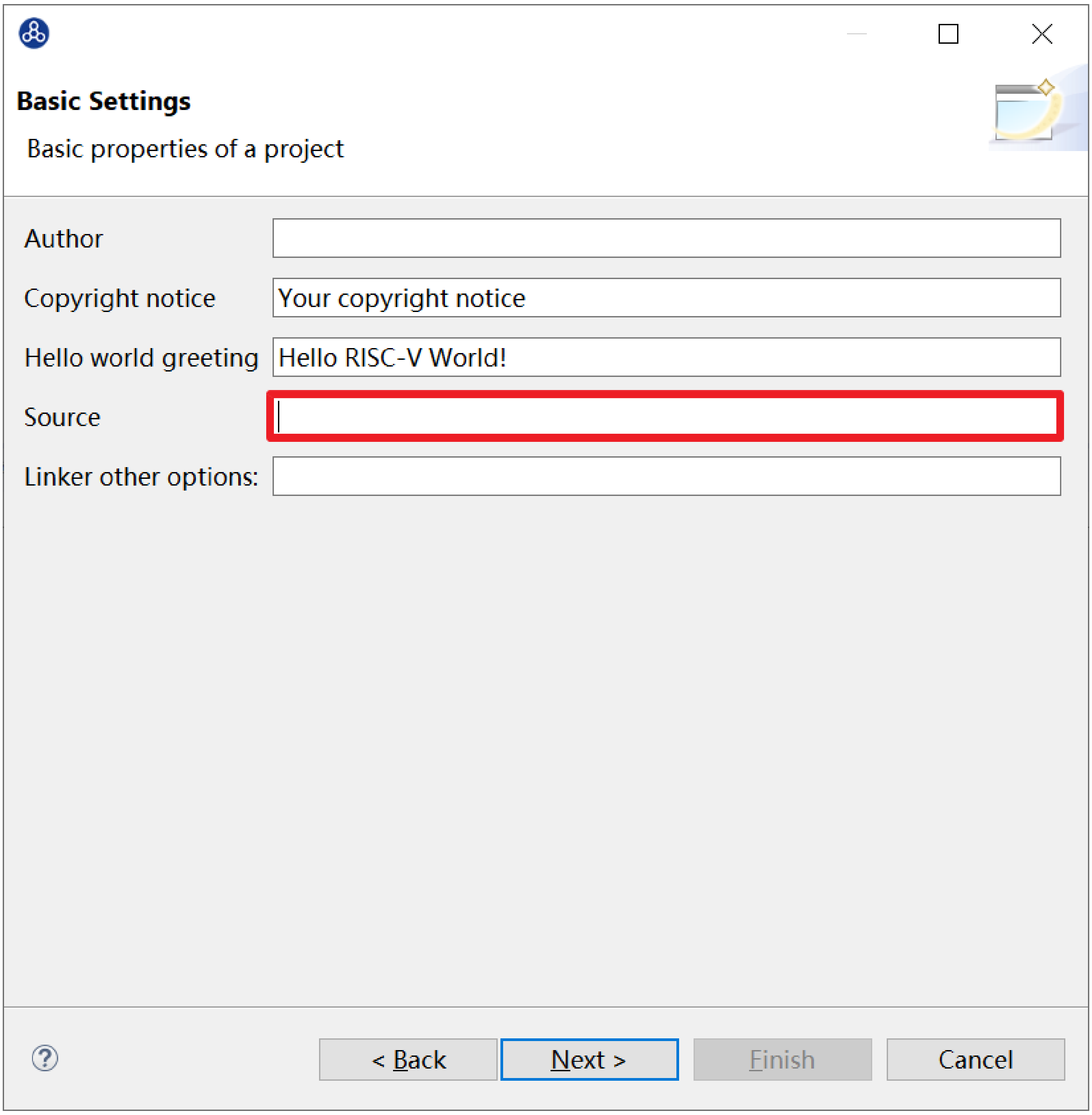


然后在弹出的窗口中设定如下参数。

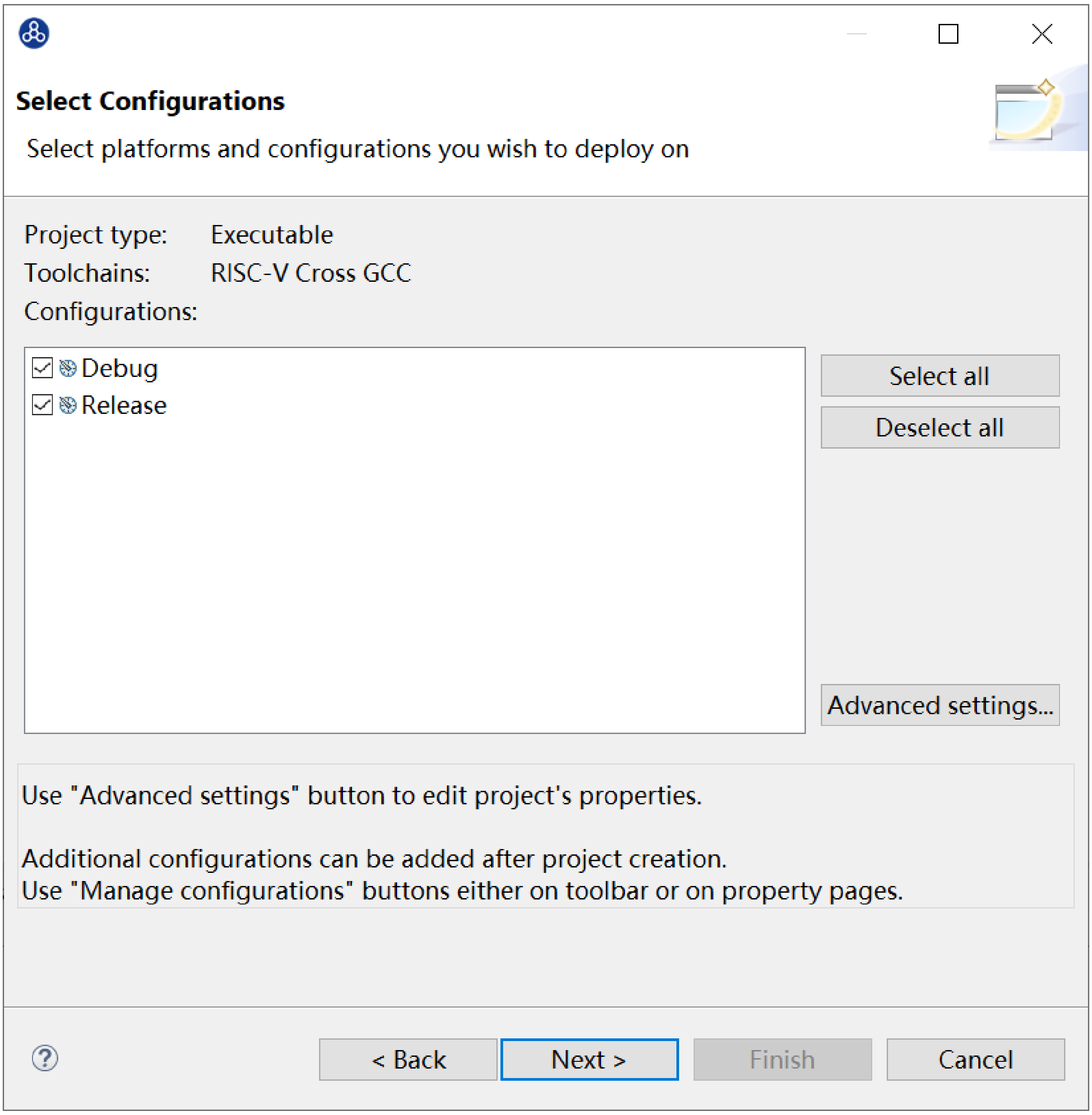
- Project name：项目命名。
- Use default location：如果勾选了此选项，则会使用默认 Workspace 文件夹存放此项目。
- Project type：选择 `Hello World RISC-V C Project` 。



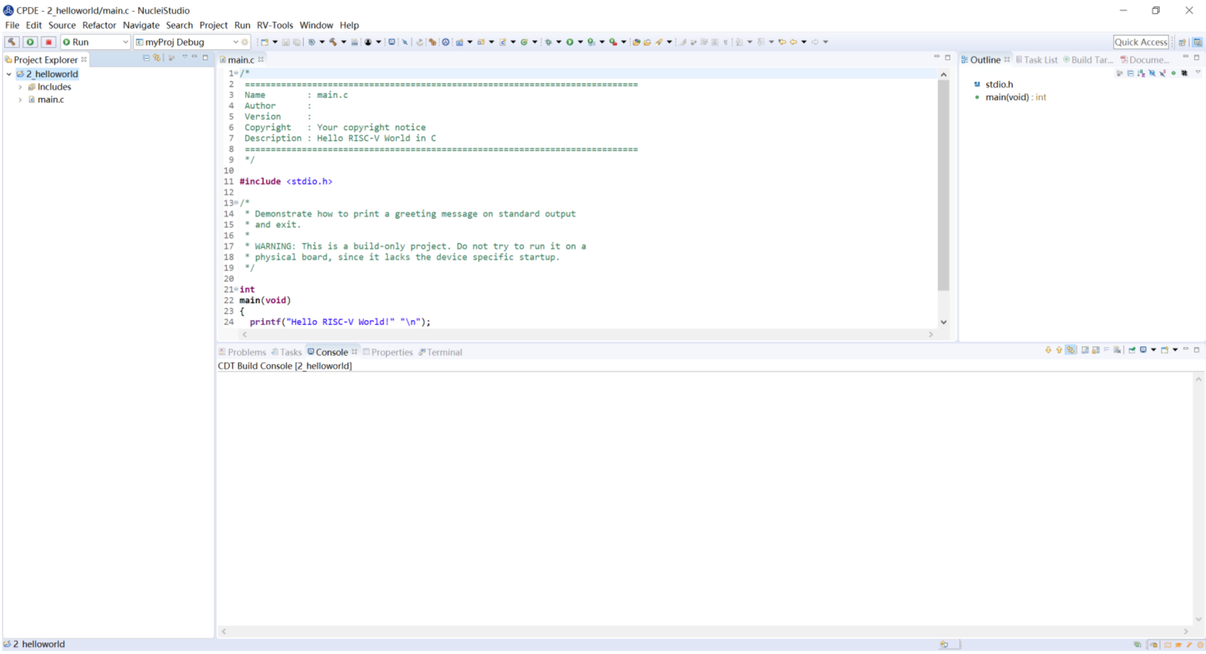
然后点击 Next 进入下一步，在弹出的窗口中设置 Hello World 项目的基本信息。确保 `Source` 选项内容为空，直接单击 `Next` 进入下一步。



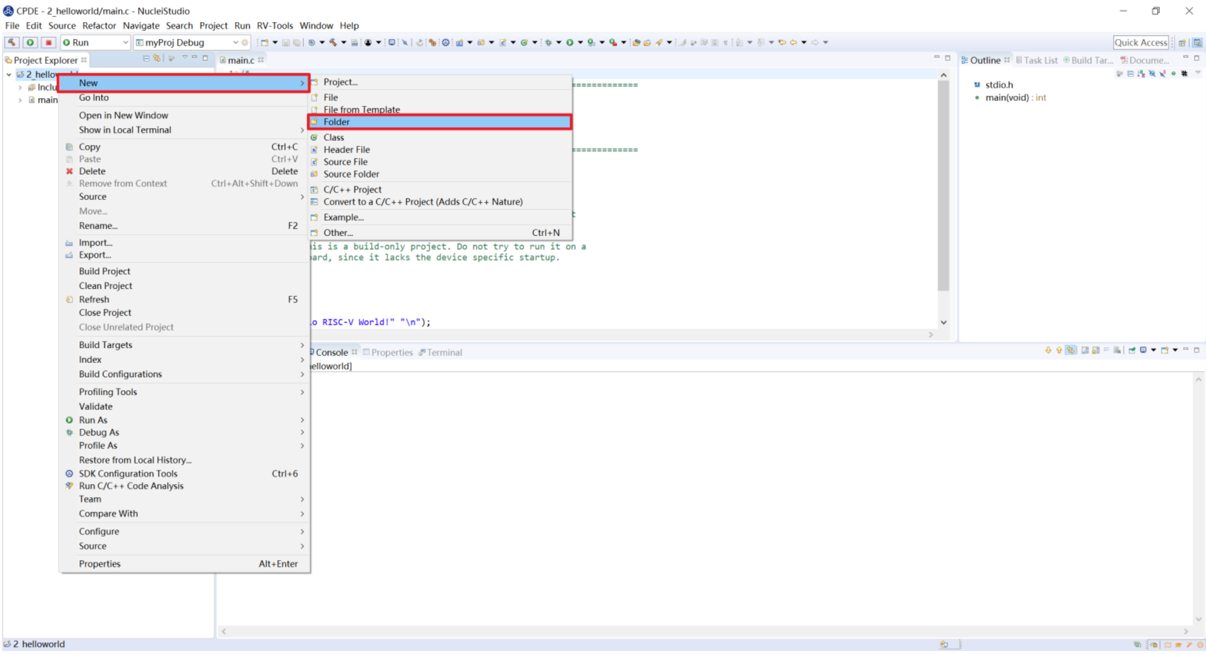
在弹出的窗口中设置项目的调试或者发布属性。该步骤可以使用默认信息不做任何修改，直接单击 Next 进入下一步。



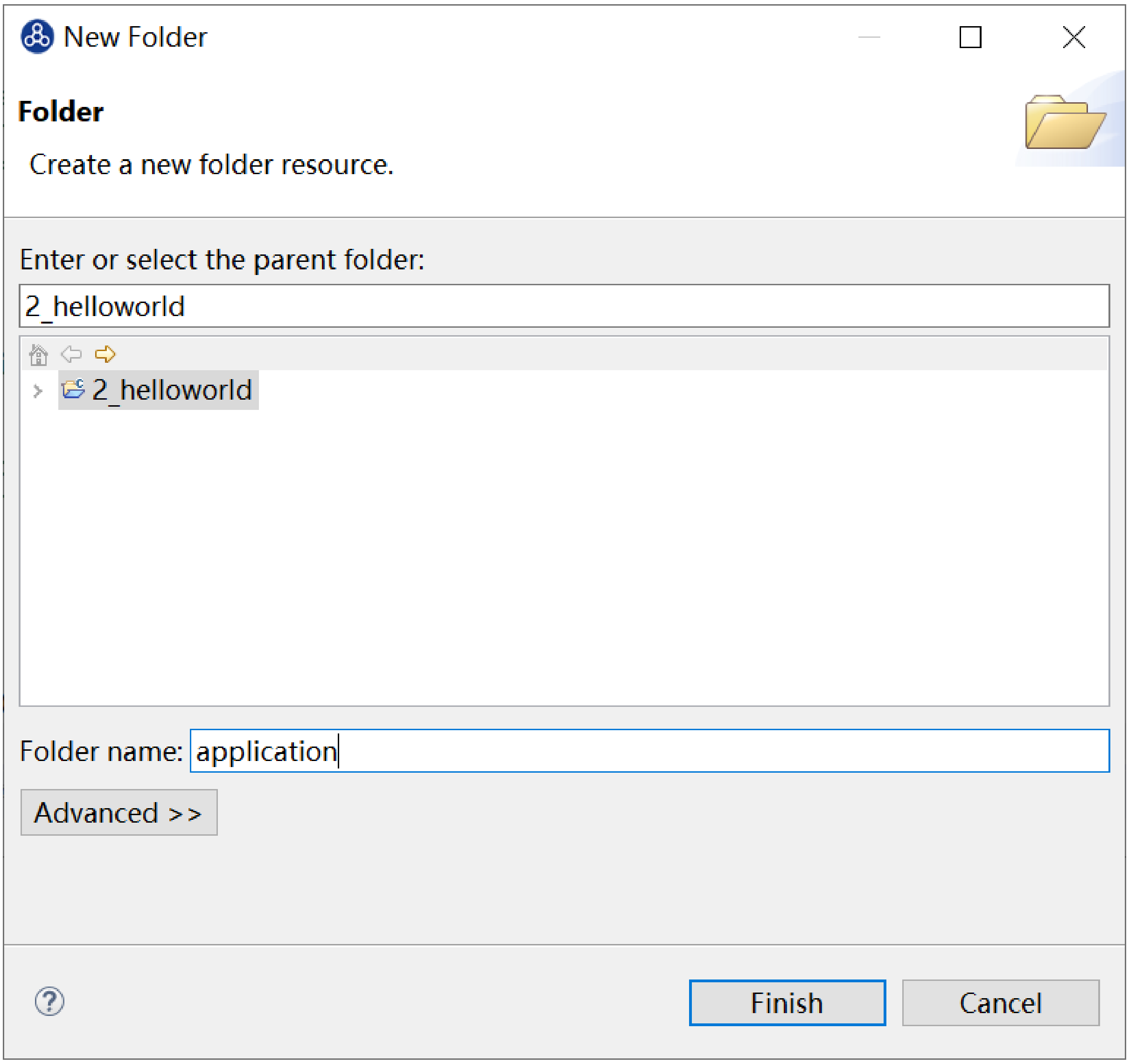
在弹出的窗口中设置项目所使用的 RISC-V 工具链。此处不要配置，直接选择 `Finish`，至此便完成了 HelloWorld 项目的创建。



创建完成，Hello World 项目的展示界面如下。



新建一个 application 文件夹。在工程处右击选择 `New —> Folder`，输入 application，点击 `Finish` 完成新建工程。将 main.c 拖入 application 文件夹完成文件分类。

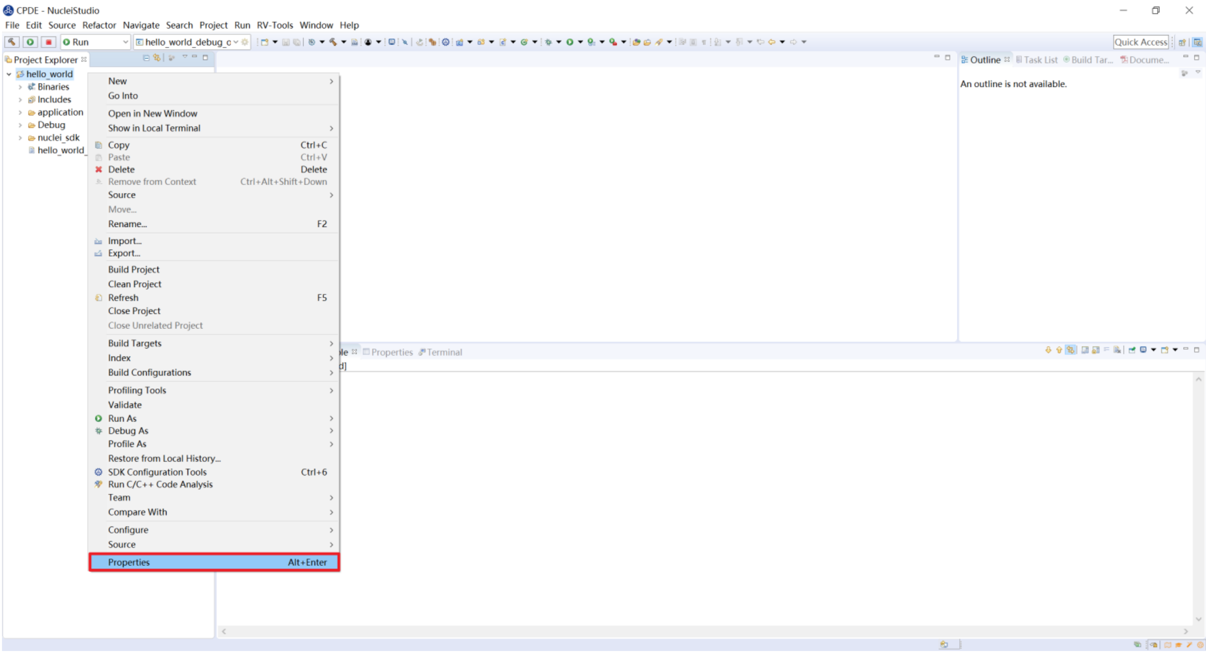


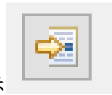
配置项目的 nuclei_sdk

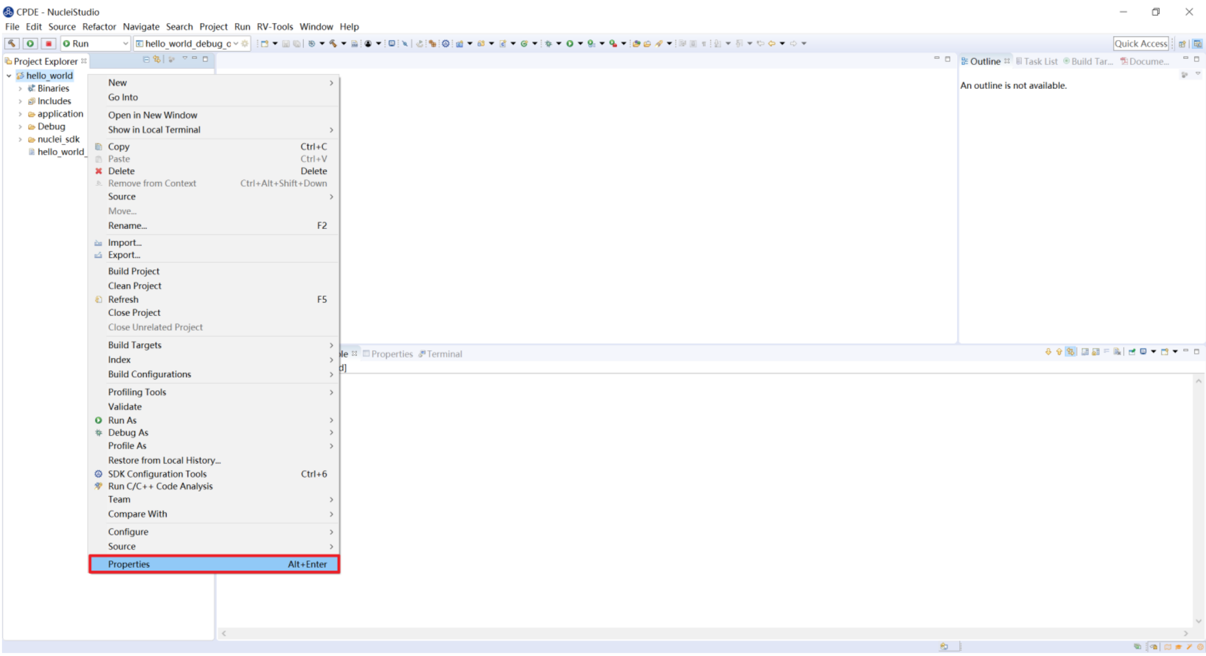
本节介绍如何将 nuclei_sdk 加入到项目中，SDK 的具体内容本文不做详细介绍，可以参考https://doc.nucleisys.com/nuclei_sdk/index.html。如果需要使用 SDK 的其他源文件，请到 Github 获取全部的 Nuclei SDK 源码（这里以 0.3.9 版本为例），链接如下：https://github.com/Nuclei-Software/nuclei-sdk/releases 。本节仅介绍将 nuclei_sdk 中 helloworld 需要的文件加入到项目的步骤，如果使用新版本的 SDK，对应的目录结构可能有所调整，请自行解决，具体步骤如下：

进入 Nuclei Studio 的 `2_helloworld` 项目，按照如下步骤添加 nuclei_sdk 源文件。

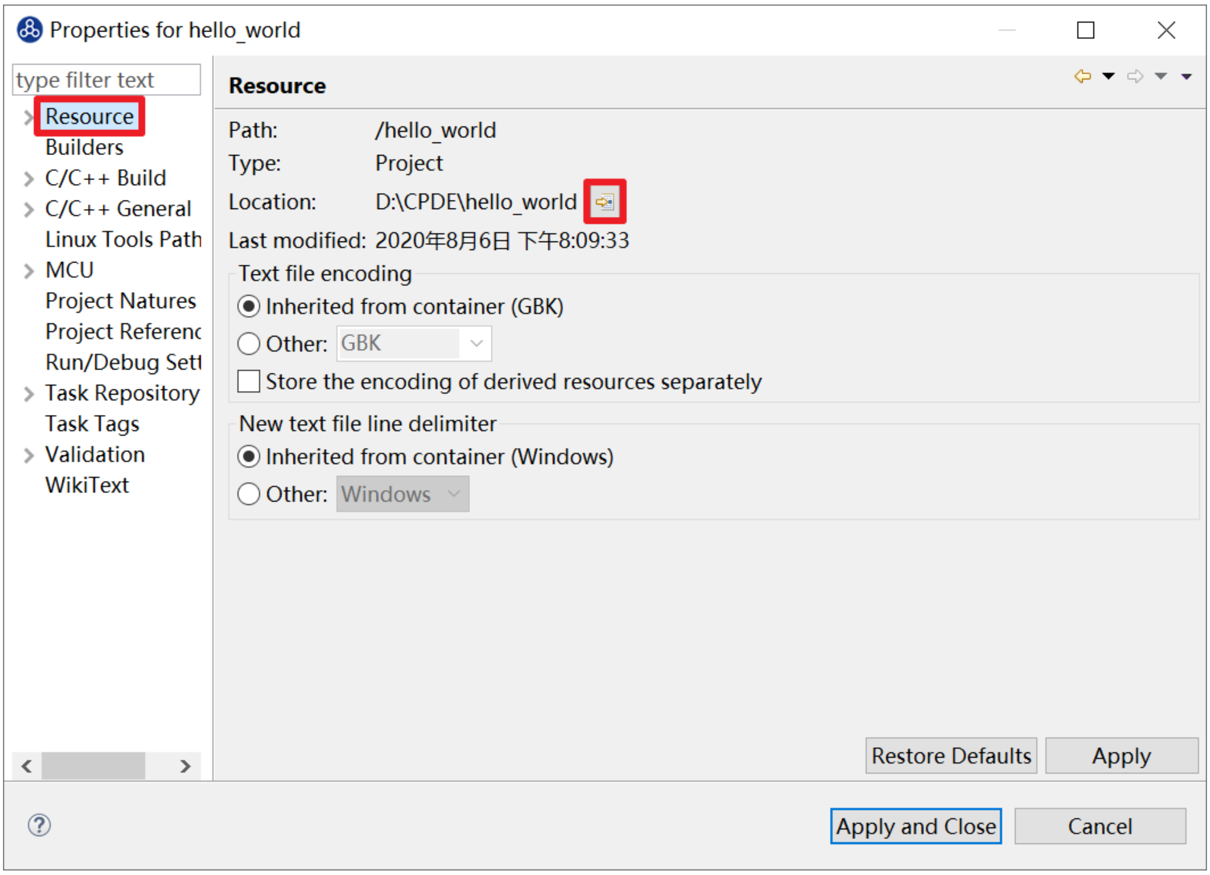
在 Project Explorer 栏中选中 `2_helloworld` 项目，单击鼠标右键，选择 `Properties` 打开工程设置页面。



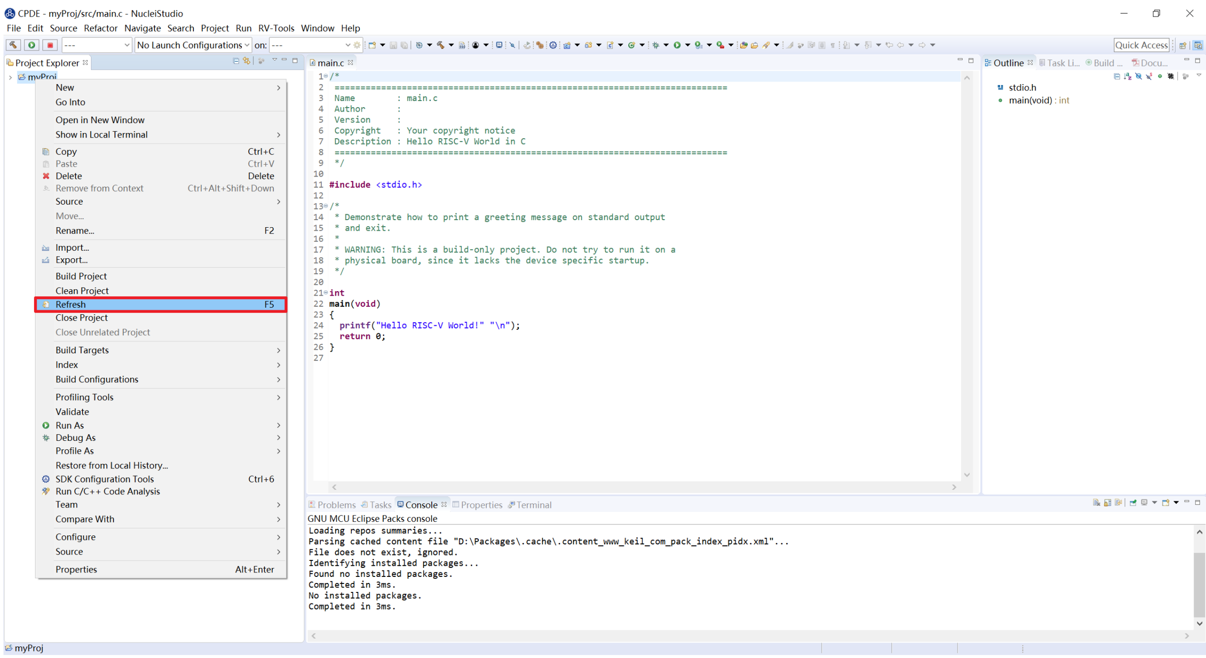
在弹出的窗口中单击 Resource，在右侧的 Location 栏目中单击其最右侧的箭头图标 ，则会弹出文件窗口进入 2_helloworld 项目的文件夹位置。



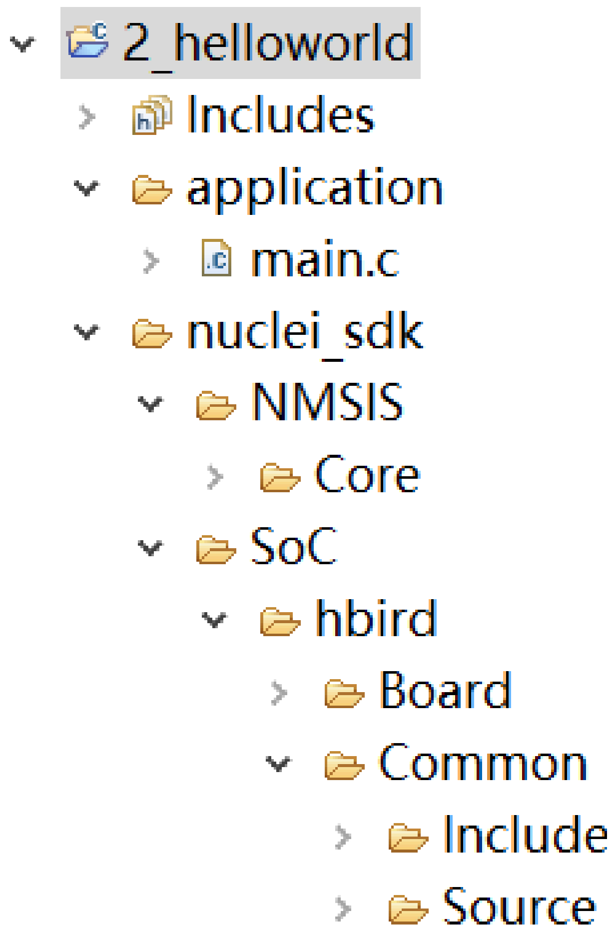
将 nuclei-eclipse_demo.rar 压缩包中的 nuclei_sdk 文件夹复制放于 2_helloworld 项目的目录下。



回到 Nuclei Studio，在 Project Explorer 栏中选中 `2_helloworld` 项目，单击鼠标右键，选择 `Refresh`
。



Refresh 之后 `2_helloworld` 项目的下便可以看到 nuclei_sdk 文件夹，至此便完成了 nuclei_sdk 源文件的导入。



配置项目的编译和链接选项

为了使项目源代码能够被正确编译，需要配置编译和链接选项。

> **Note**
>
> 注意：本节中设置的编译与链接选项均为 GCC 工具链的常用选项，与在 Linux 环境中使用时的同名选项含义一致，本节在此不做赘述介绍。

配置编译与连接选项的步骤如下：

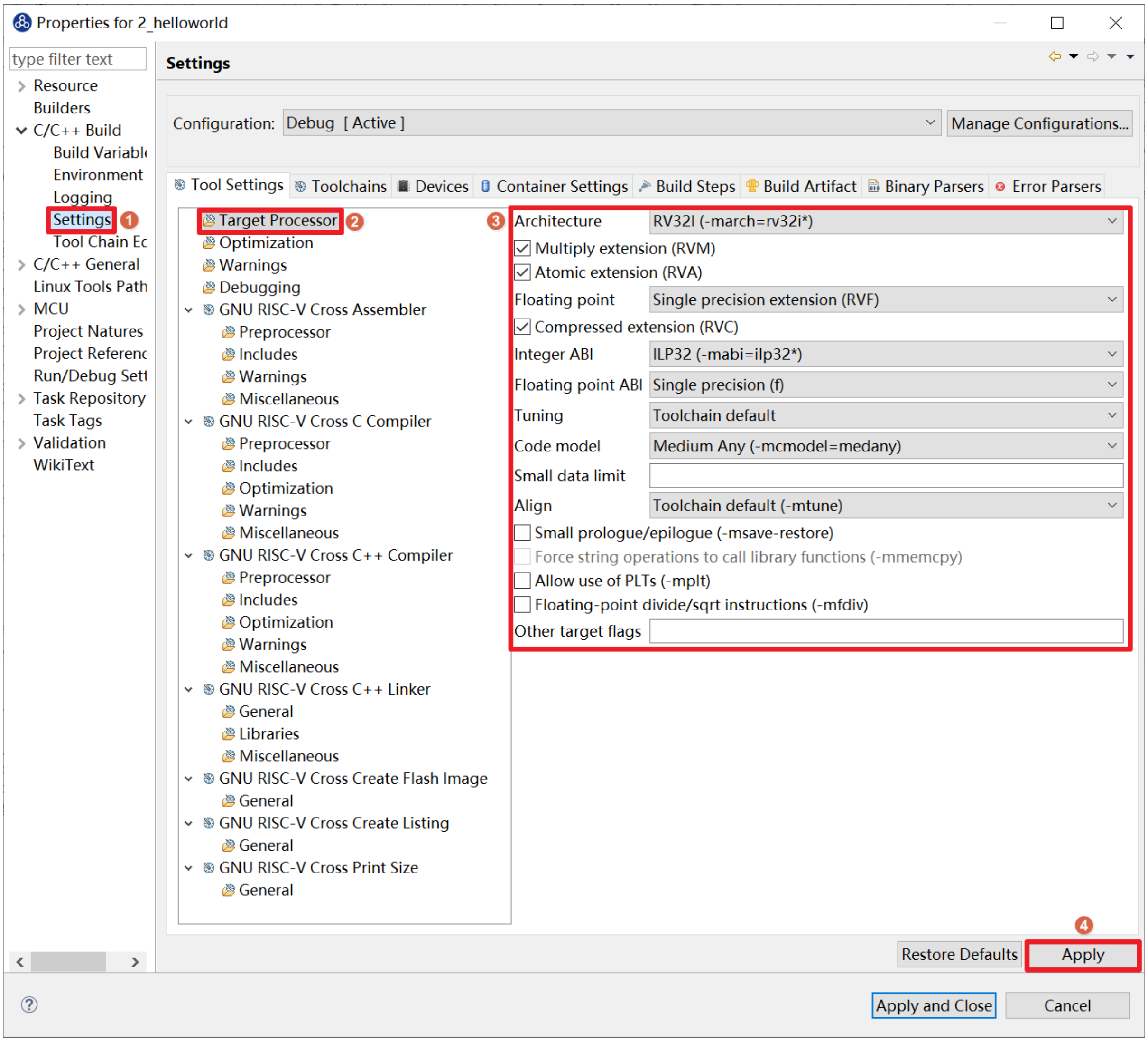
在 Project Explorer 栏中选中 hello_world 项目，单击鼠标右键，选择 `Properties` 。

在弹出的窗口中，展开 C/C++ Build 菜单，单击 `Setting` ，在右侧的 Tool Settings 栏目中进行设置。

选中 Target Processor，我们的内核是 N307，因此需要按照图所示勾选配置选项，分别如下。

- Architecture：选择 `RV32I` 。
- Multiply extension（RVM）：需勾选。
- Atomic extension（RVA）：需勾选。
- Compressed extension（RVC）：需勾选。
- Integer API：选择 `ILP32` 。
- Floting Point ABI：选择 `single precision`

- Code model：选择 `Medium Any` 。
- 单击右下角的 `Apply` 按钮。



选中 `Optimization`，按照图所示勾选配置选项。

- Optimization Level：选择 `Optimization Most (-O2)` 。

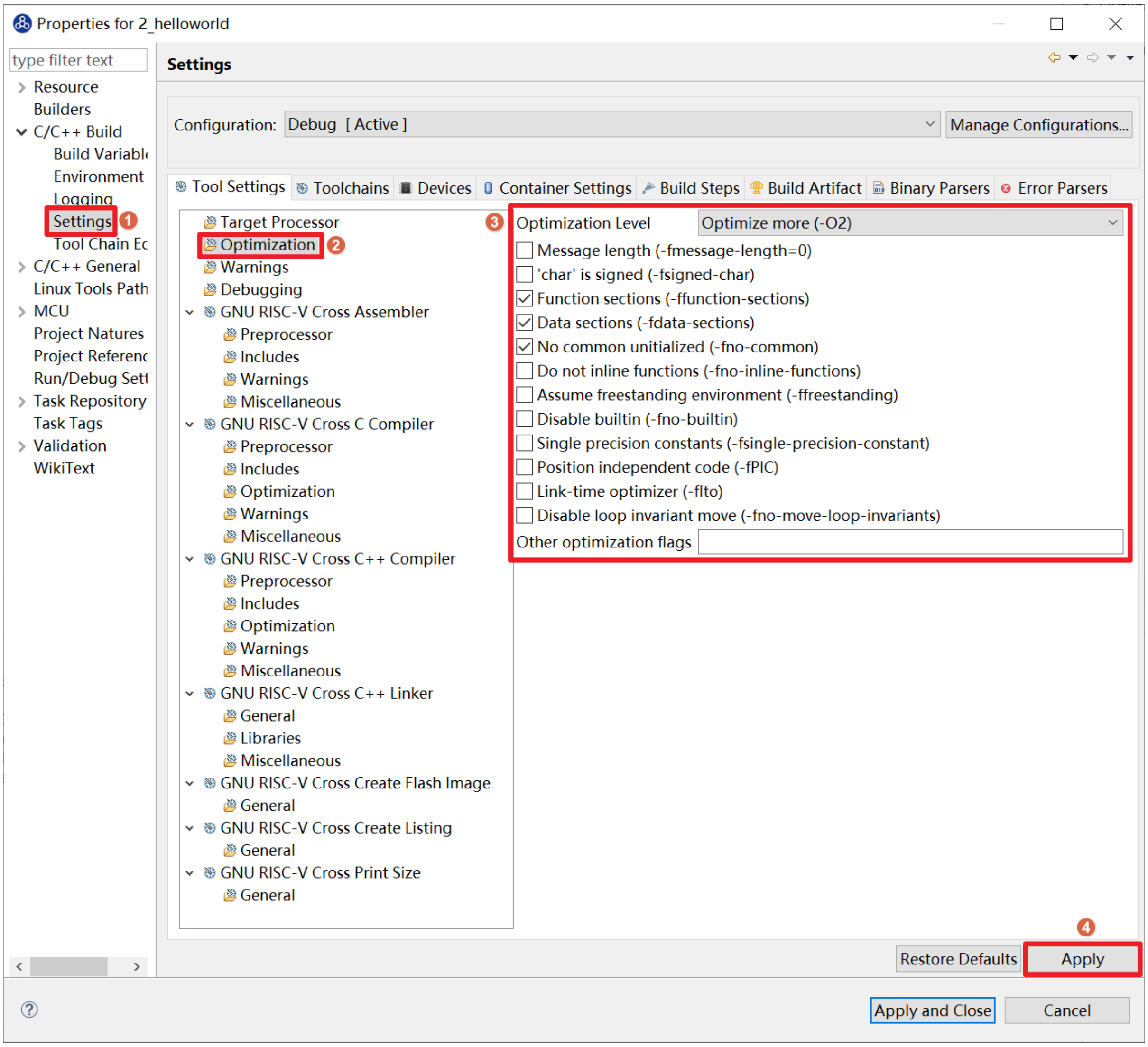
> **Note**
>
> 注意：在 NucleiStudio 2024.06 版本中新增了 `-Oz`，用来优化编译后程序的尺寸。

依次勾选：

- Function Sections (-ffunction-sections)
- Data Sections (-fdata-sections)
- No common unitialized (-fno-common)

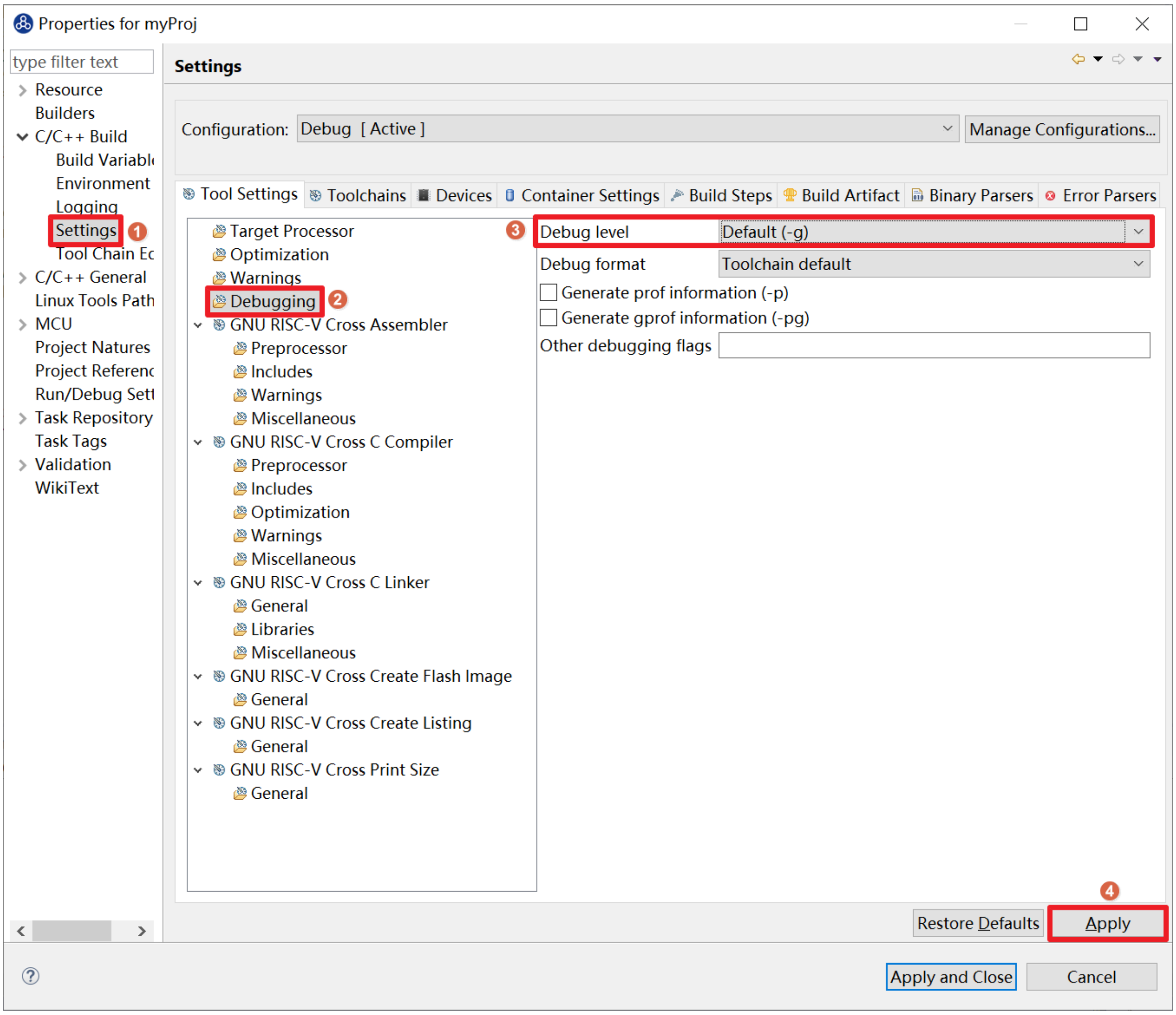
> **Note**
>
> 注意：上述选项均为通用的 GCC 编译优化选项，请用户自行查阅 GCC 手册了解其含义。

单击右下角的 Apply 按钮。



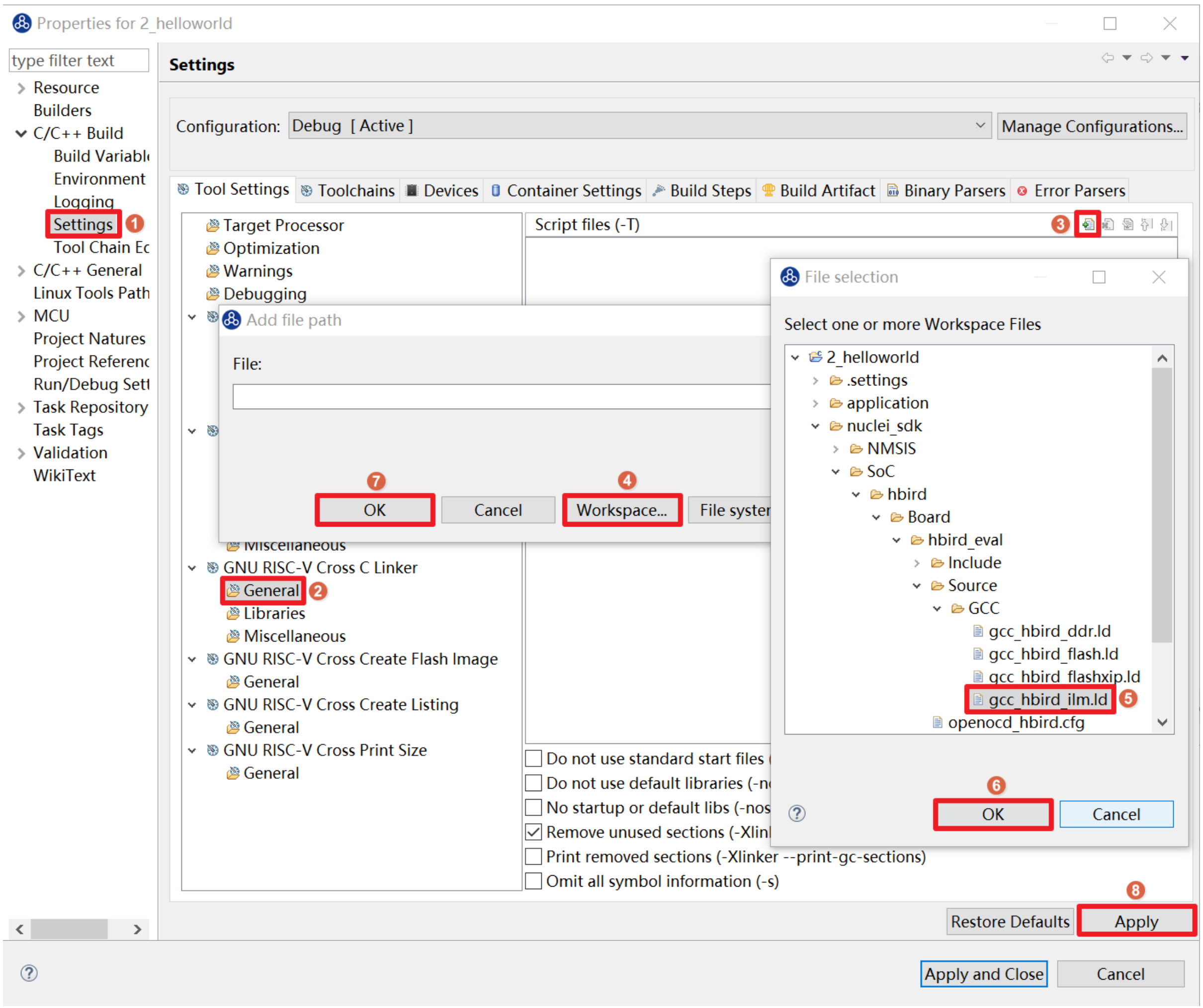
选中 Debugging，按照图中所示勾选配置选项，分别为：

- Debug Level：选择 Default (-g)。
- 单击右下角的 Apply 按钮。



选中 GNU RISC-V Cross C Linker 的 General。按照如下步骤设置链接器的所需的链接脚本。

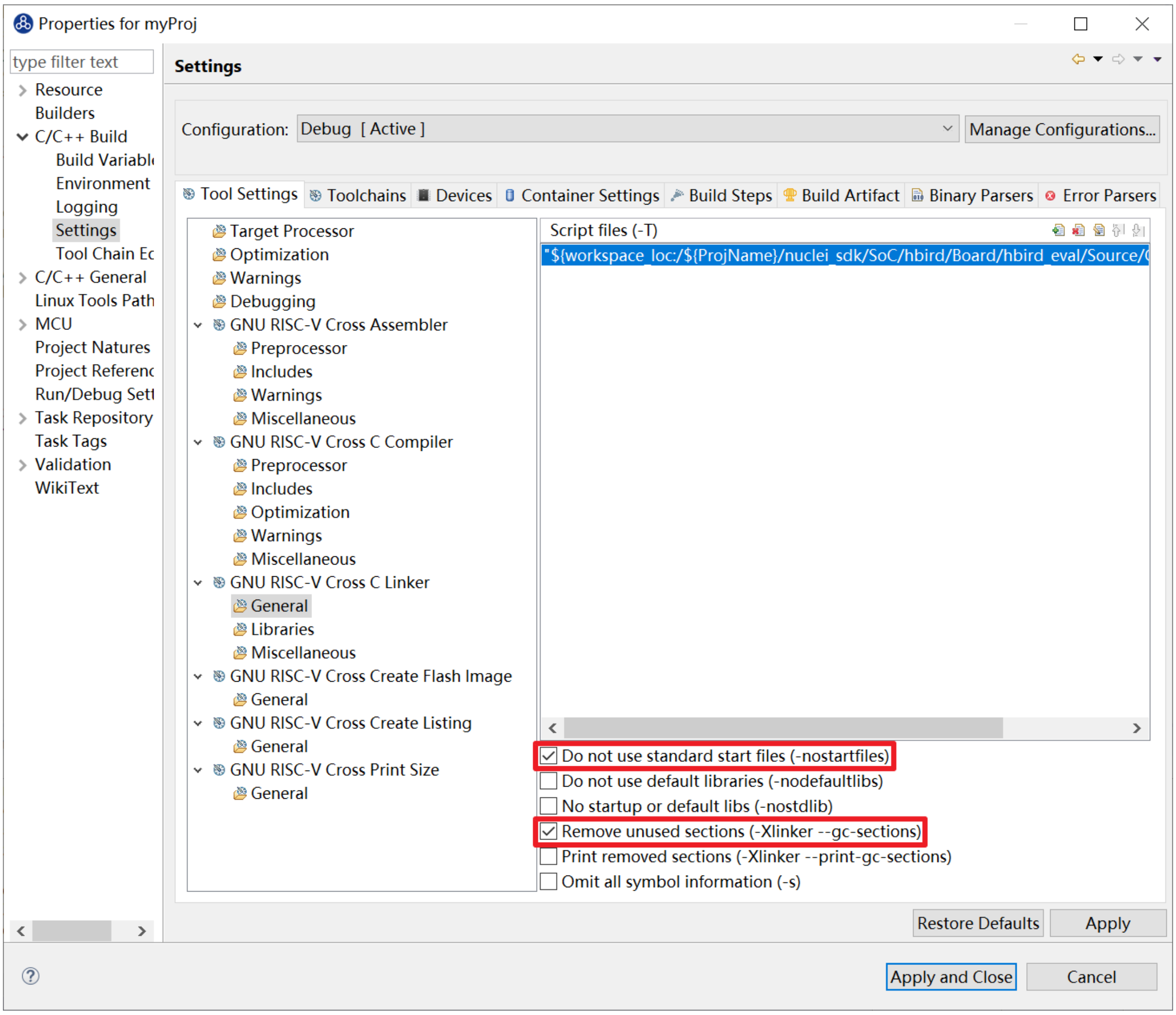
- 选中右上角的加号按键。
- 在弹出的窗口中单击 `Workspace` 按钮。
- 这里我们使用 HummingBird 评估板，所以可以选择 ILM 下载模式对应的 `gcc_hbird_ilm.ld` 文件。在弹出的窗口中选择 Nuclei Studio 文件包中的 `nuclei_sdk/SoC/hbird/Board/hbird_eval/Source/GCC` 文件夹下 `gcc_hbird_ilm.ld` 文件。其他下载模式切换此处文件，各文件详细介绍如下，可根据自己的实际情况选择。
  - `gcc_hbird_ilm.ld` 脚本将程序代码段约束在 ILM 的地址区间，意味着程序将被直接下载在 MCU 的 ILM 中，并从 ILM 开始执行。ILM 由 SRAM 组成，会掉电丢失。
  - `gcc_hbird_flash.ld` 脚本程序代码段的物理地址约束 Flash 区间，将代码段的逻辑地址约束在 ILM 的地址区间，意味着程序将被直接下载在 MCU 的 Flash 中，但是上电后要通过引导程序将代码段搬运到 ILM 中，然后从 ILM 中开始执行。
  - `gcc_hbird_flashxip.ld` 脚本程序代码段约束 Flash 区间，意味着程序将被直接下载在 MCU 的 Flash 中，并直接从 Flash 开始执行。程序被烧写在 Flash 中，不会掉电丢失。
  - 用户可以按照自己的需求选择合适的链接脚本。本节示例选择 `gcc_hbird_ilm.ld` 作为演示。
- 设置完毕请单击右下角的 `Apply` 按钮。



按下图所示勾选配置选项，分别如下。

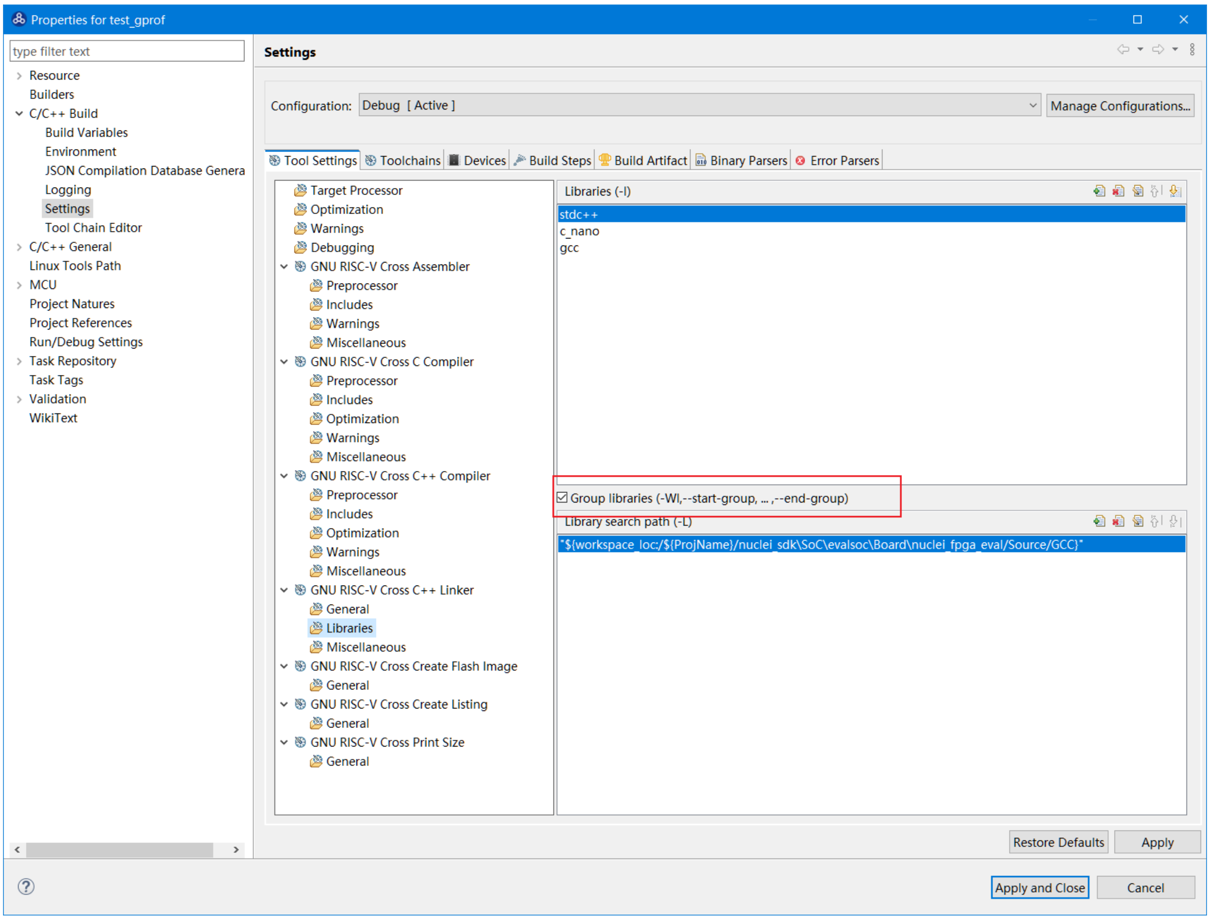
- Do not use standard start files (-nostartfiles) 。
- Remove unused sections (gc-sections)。
- 单击右下角的 `Apply` 按钮。

> **Note**
>
> 注意：上述选项均为通用的 GCC 链接选项，请用户自行查阅 GCC 手册了解其含义。



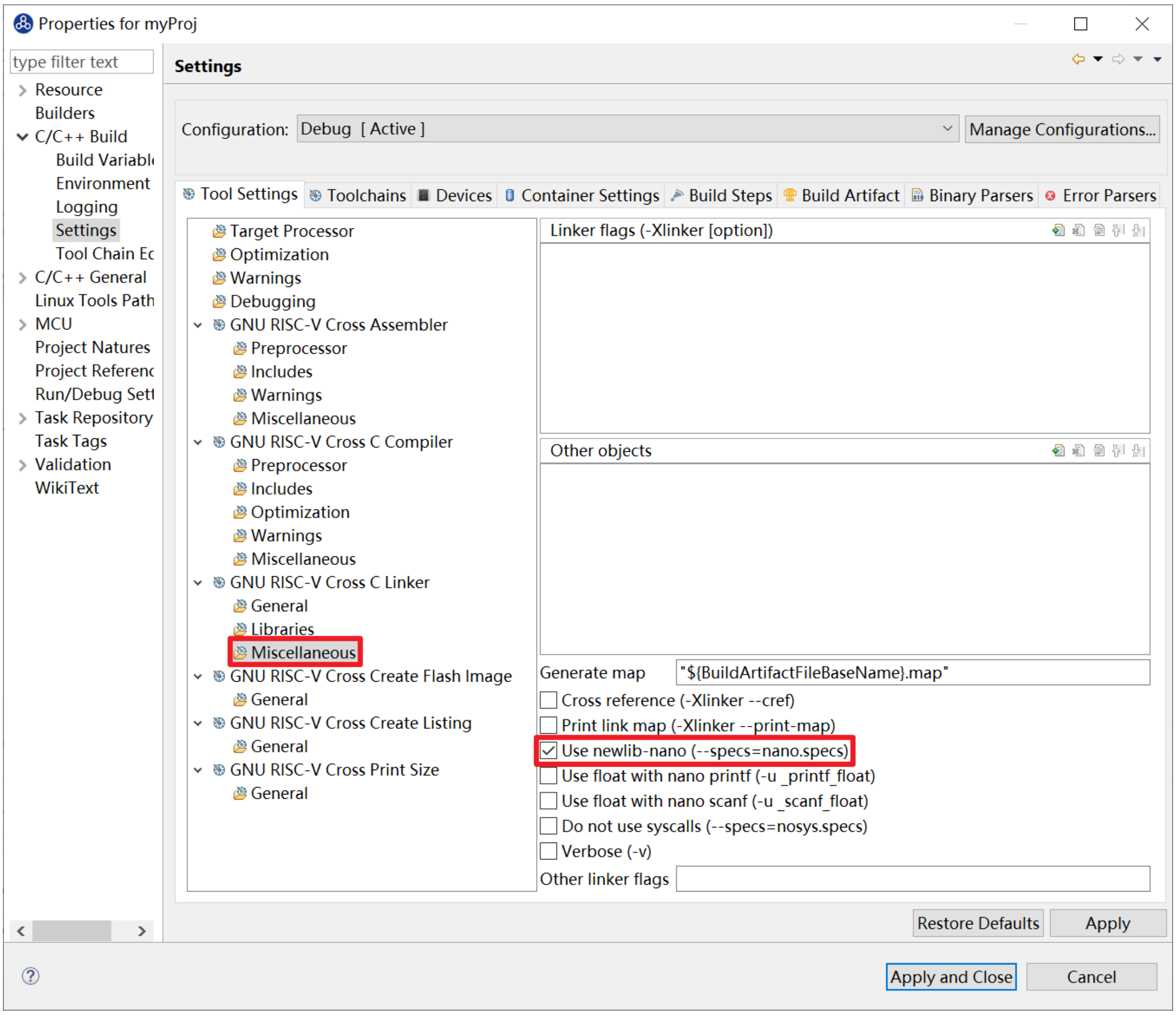
Note

注意：在 NucleiStudio 2024.06 版本中 Libraries 支持 Group 功能，如果勾选了 Group 功能，所有的 Libraries 在编译时会用 `-wl,--start-group,……,--end-group,`，能解决 Libraries 内相互依赖的问题。



选中 GNU RISC-V Cross C Linker 的 Miscellaneous，按照下图所示勾选配置选项。

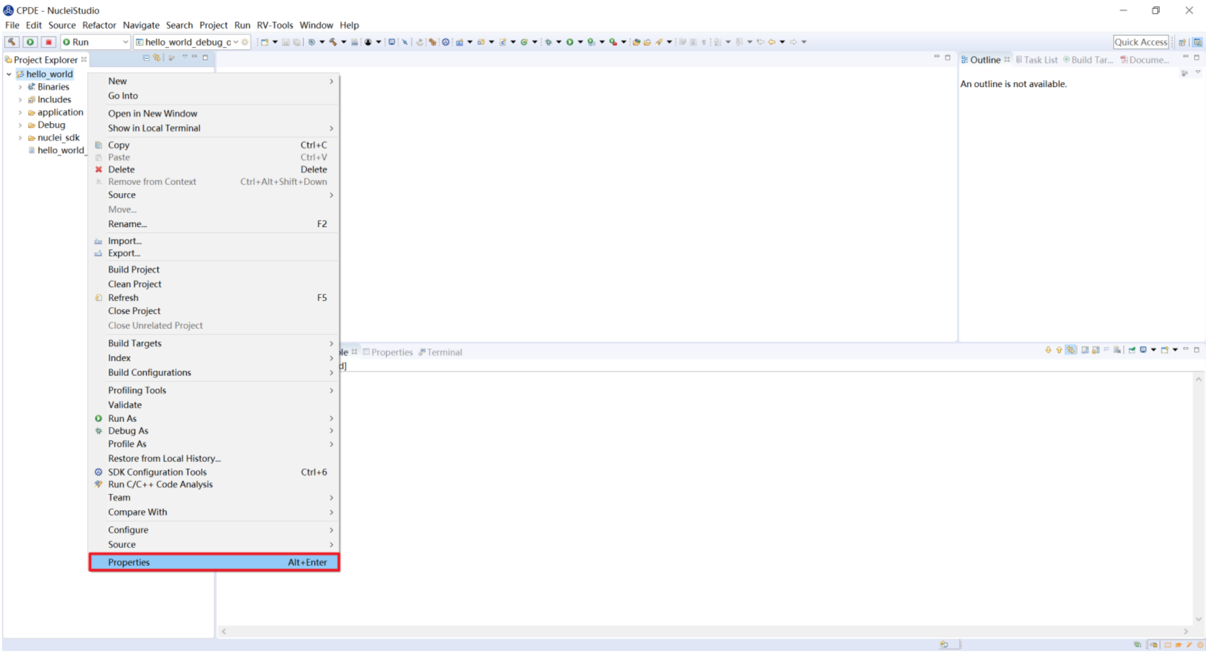
- 勾选 `Use newlib-nano` 。
- 因为 Hello World 程序的 Printf 不需要打印浮点数，所以不要勾选 `Use float with nano printf` 。
- 单击右下角的 `Apply` 按钮。



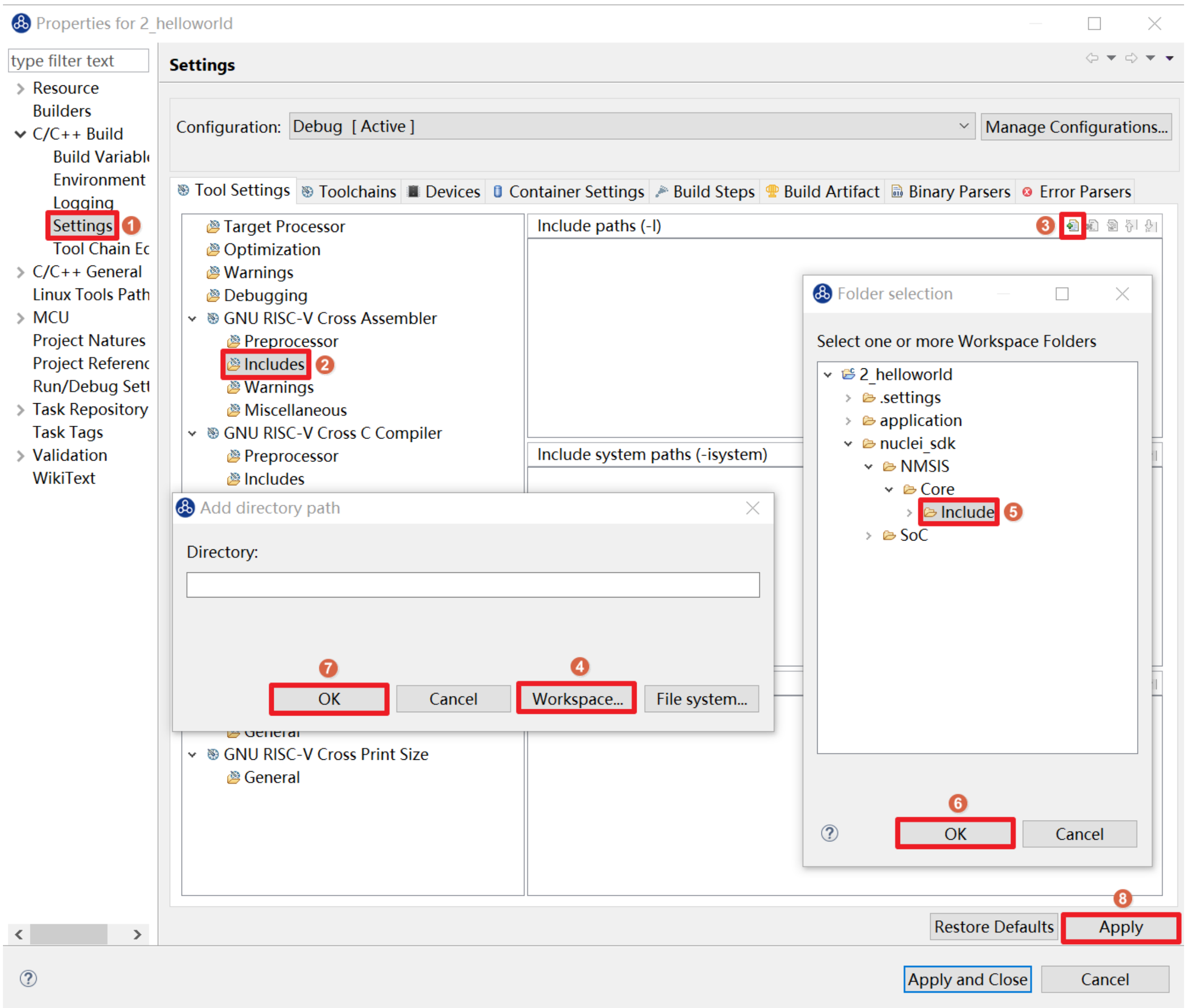
配置项目的包含路径和文件

为了能够正确编译 nuclei_sdk 文件夹中的源文件，需要按照如下步骤配置项目的包含路径和包含文件。

在 Project Explorer 栏中选中 hello_world 项目，点击鼠标右键，选择 `Properties` 。



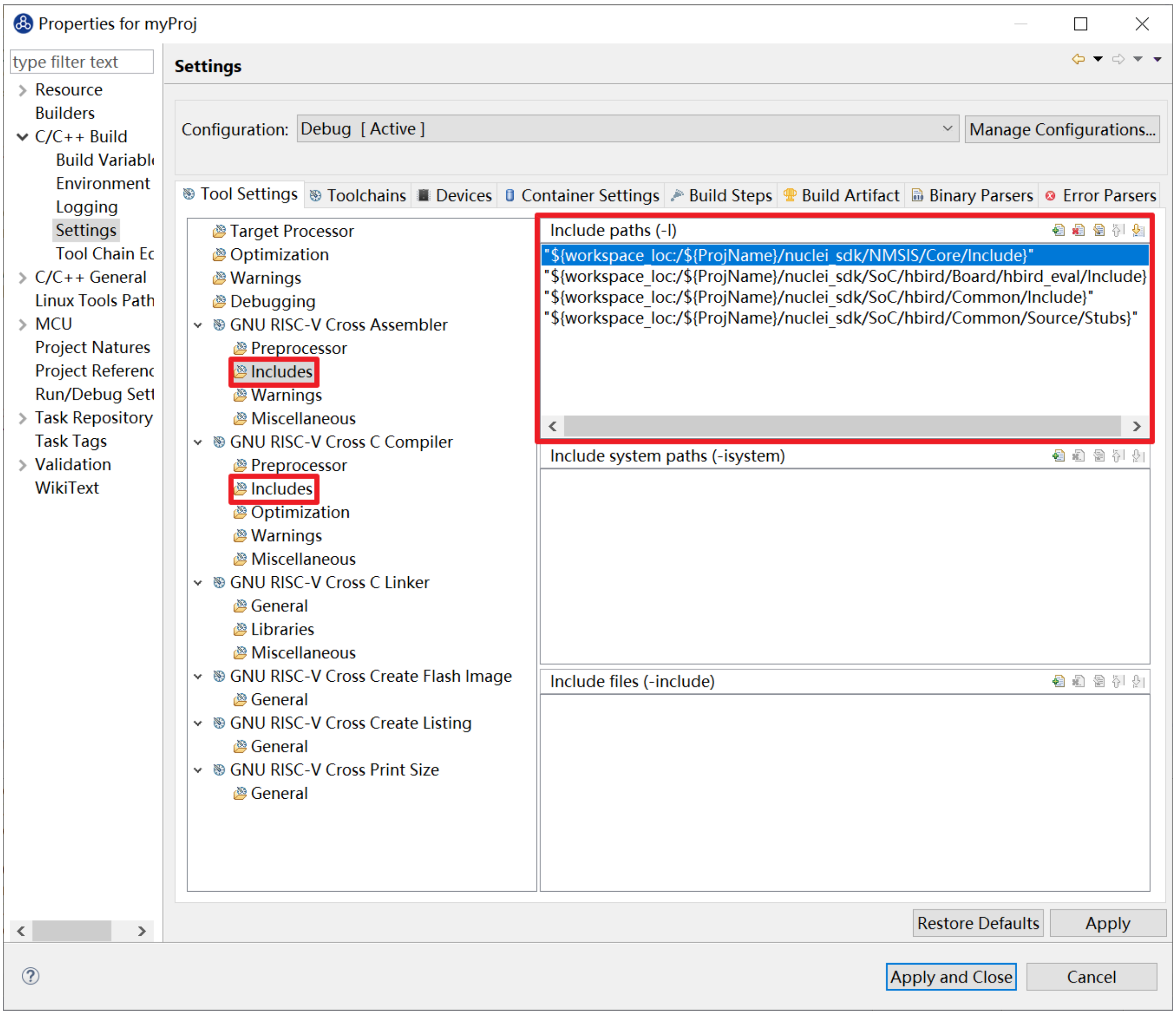
在弹出的窗口中，展开 C/C++ Build 菜单，单击 `Setting`，在右侧的 Tool Settings 栏目中进行设置。选中 GNU RISC-V Cross C Assembler 的 Includes，按照图中所示配置包含文件，步骤如下。



- 在 Include paths 栏目单击加号键。

- 在弹出的窗口中单击 `Workspace`，弹出 Folder selection 窗口。
- 在 Folder selection 窗口中选择项目的 nuclei_sdk 目录下的 NMSIS>Core>Include 文件夹。
- 在右下角单击 `Apply` 完成配置。

采用上述方法，依次添加 nuclei_sdk 目录下的 `SoC>hbird>Board>hbird_eval>Include`，`SoC>hbird>Common>Include` 和 `SoC>hbird>Common>Source>Stubs` 文件夹作为包含路径，并采用同样的方法为 `GNU RISC-V Cross C Compiler` 的 `Includes` 栏目设置包含路径。设置完成后的界面如下图所示。

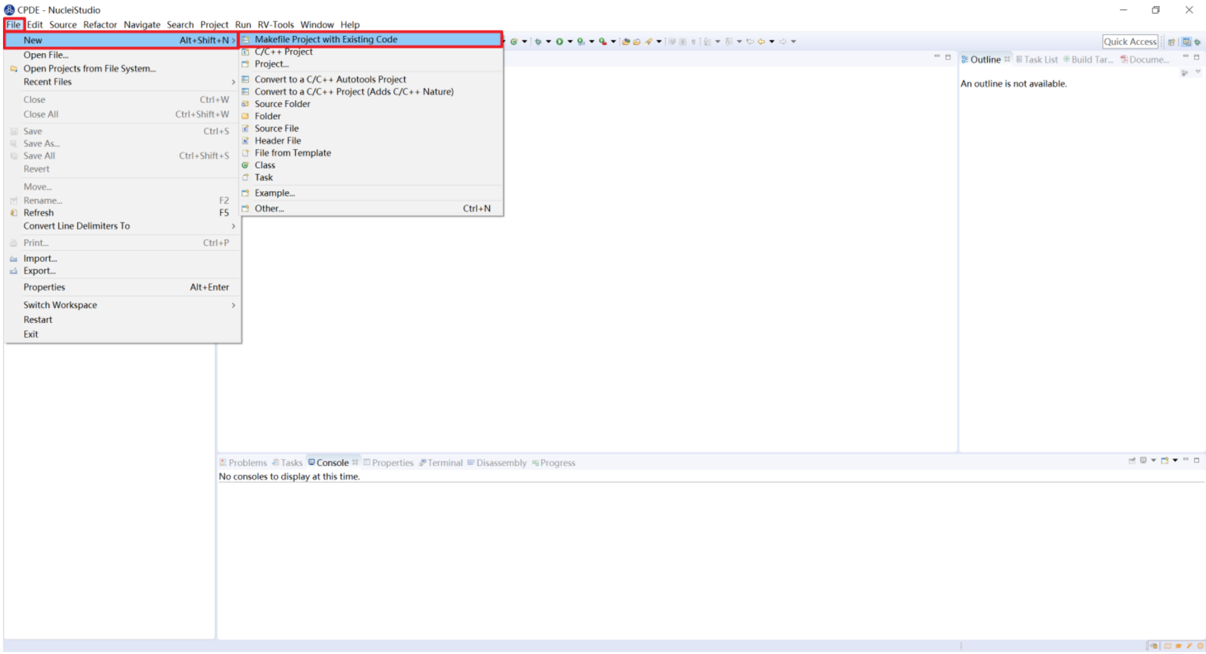


2.7.5 基于已有的 Makefile 创建项目

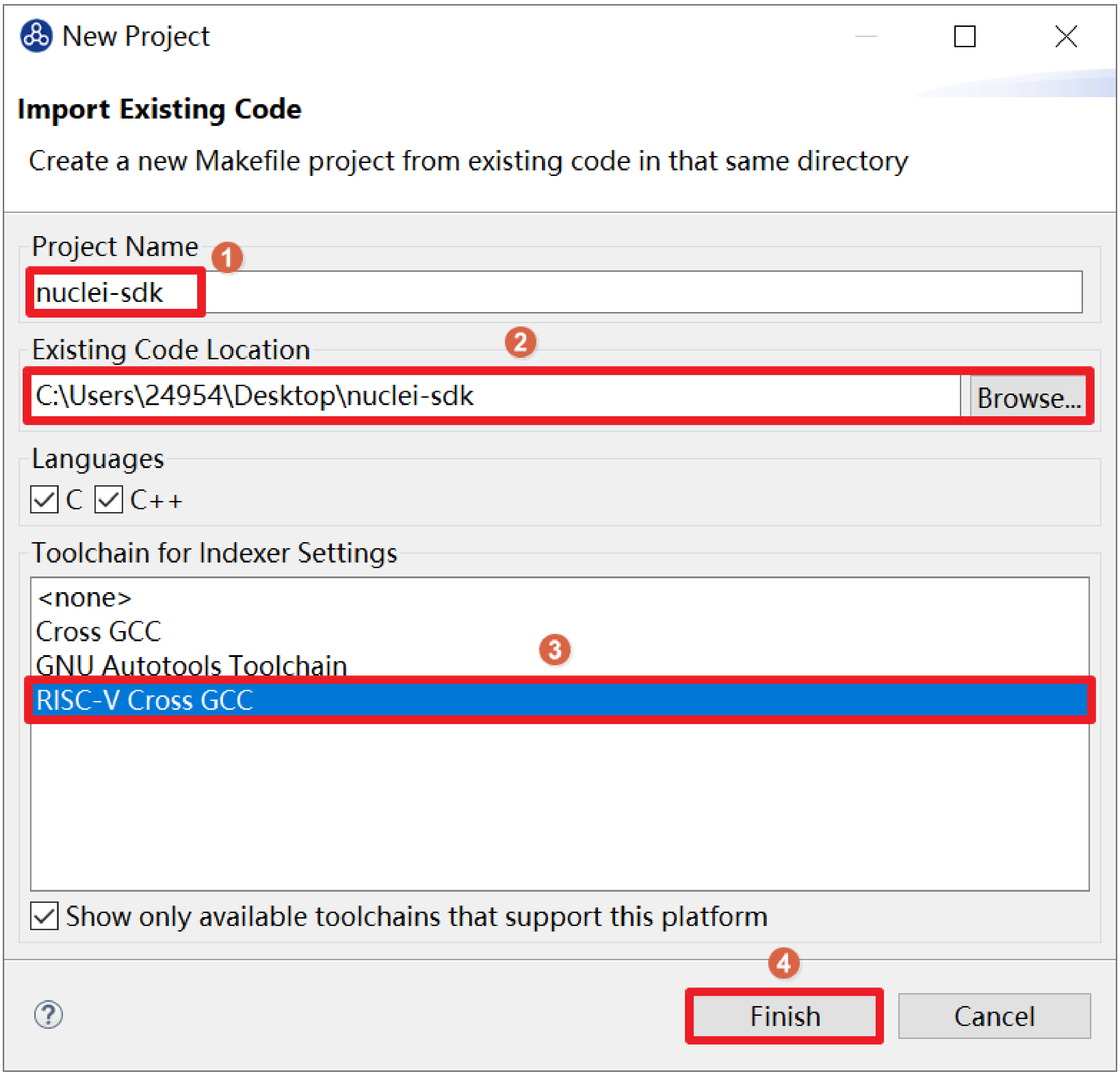
本节将介绍如何使用已有的 Makefile 在 Nuclei Studio IDE 创建一个使用 Makefile 的 Hello World 项目。开发板为 Nuclei FPGA Evaluation Board，内核为 N307。请先下载 Nuclei SDK，Github 链接为：https://github.com/Nuclei-Software/nuclei-sdk。该方法除了创建项目之外，还需要手动设置各种选项和路径，这里以 helloworld 为例，详细步骤如下。

手动新建项目

在菜单栏中选择 `File—> New —> Makefile Project with Existing Code` 。

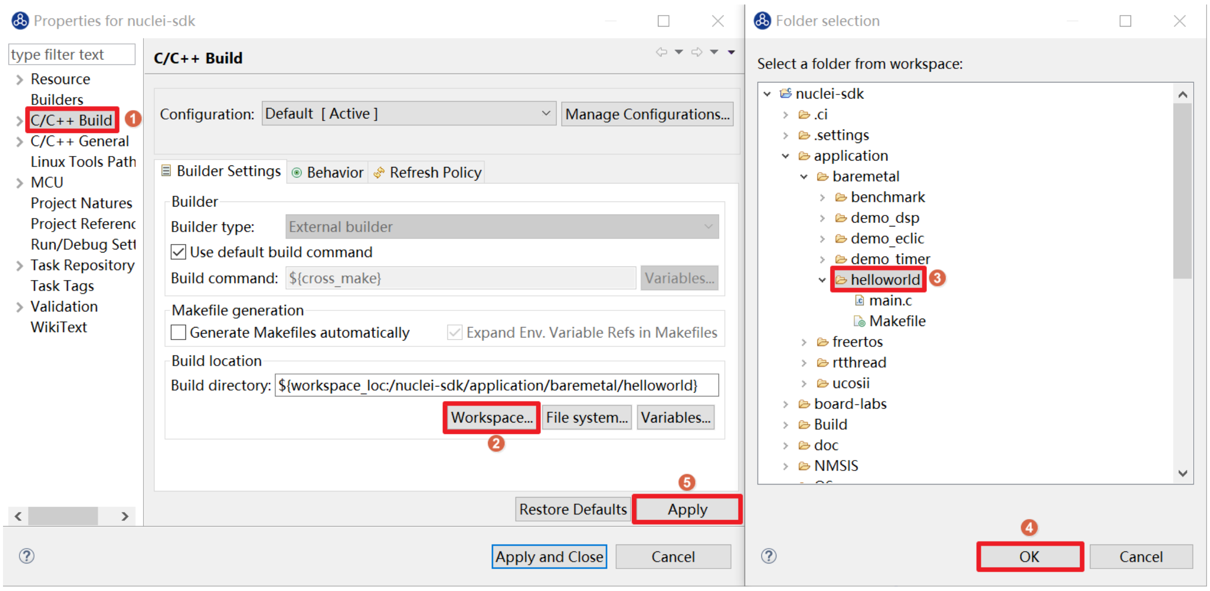


在图标 1 处输入工程名，这里我们命名为 nuclei-sdk。在图标 2 处输入 SDK 的实际路径。在图标 3 处选择 `RISC-V Cross GCC` 。点击图标 4 完成新建项目。

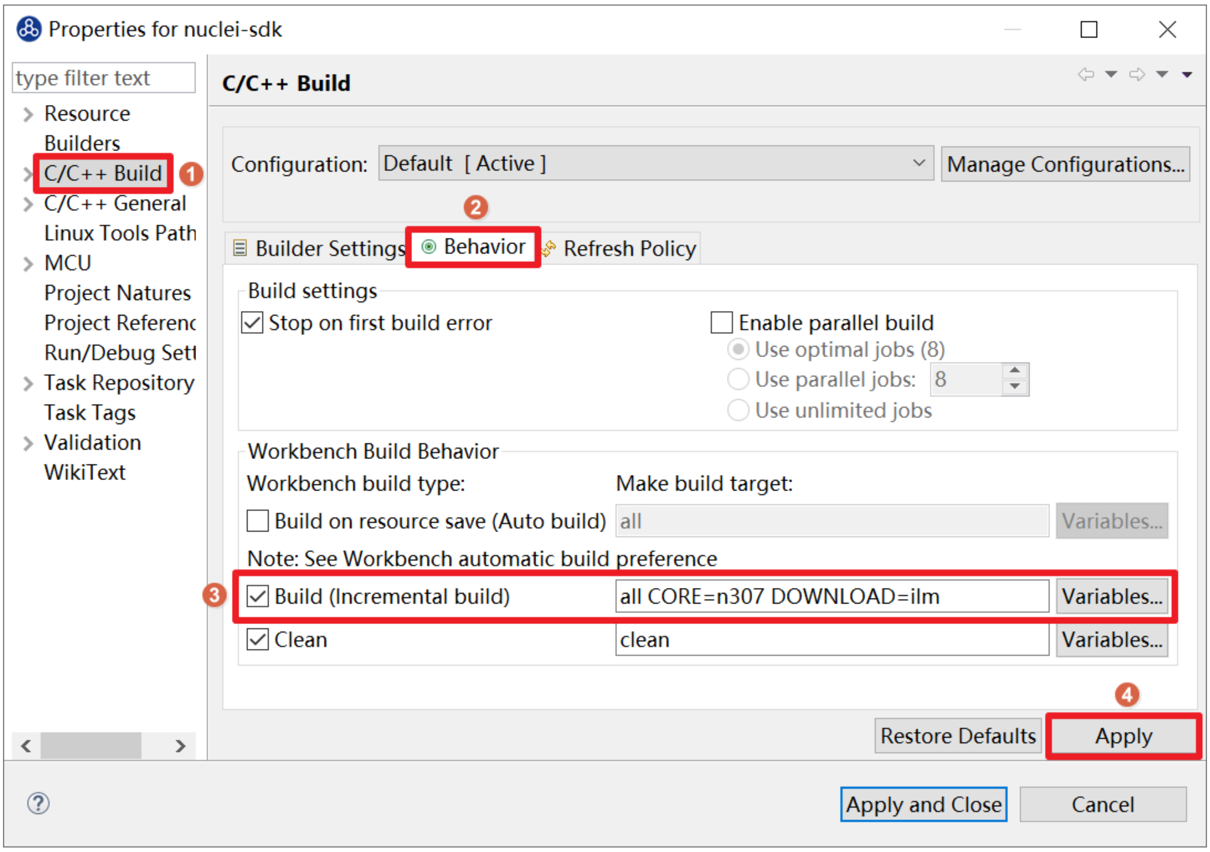


设置 Makefile 路径和 Build 选项

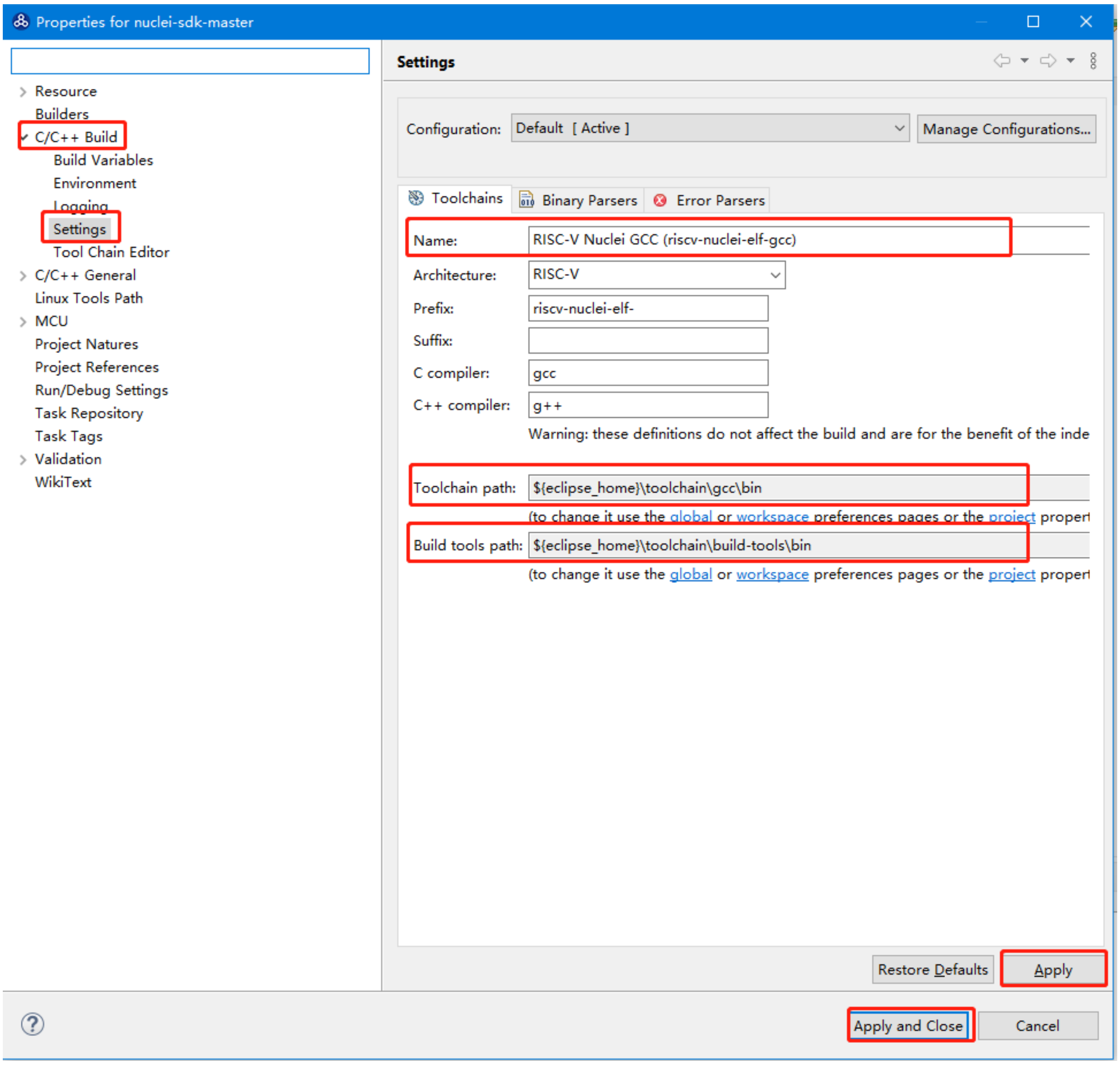
右击新建好的工程，选择 Properties 打开设置页面，选择 C/C++ uild，在 Build Location 中选择 Workspace。在弹出的弹窗中选择 application > baremetal > helloworld 点击 OK 再点击 Apply 保存。



在 C/C++ Build 中选择 Behavior 栏目，确保勾选 Build (Incremental Build) 选项并输入 all CORE=n307 DOWNLOAD=ilm 。其中 CORE 选项根据实际的内核变化，这里以 n307 为例。DOWNLOAD 选项可以修改不同的下载模式，详细请参考 5.1 节，这里以 ilm 模式为例。因为例程使用 HummingBird Evaluation Board，所以 SoC 和 Board 都不必修改，如果使用其他开发板，以 RVSTAR 为例，请在此处设置增加 SOC=gd32vf103 BOARD=gd32vf103v_rvstar ，并且由于 RVSTAR 仅支持 FLASHXIP 模式，需要将 DOWNLOAD 设置为 flashxip ，同时 CORE 修改为 n205 。完成后点击 Apply 保存修改。

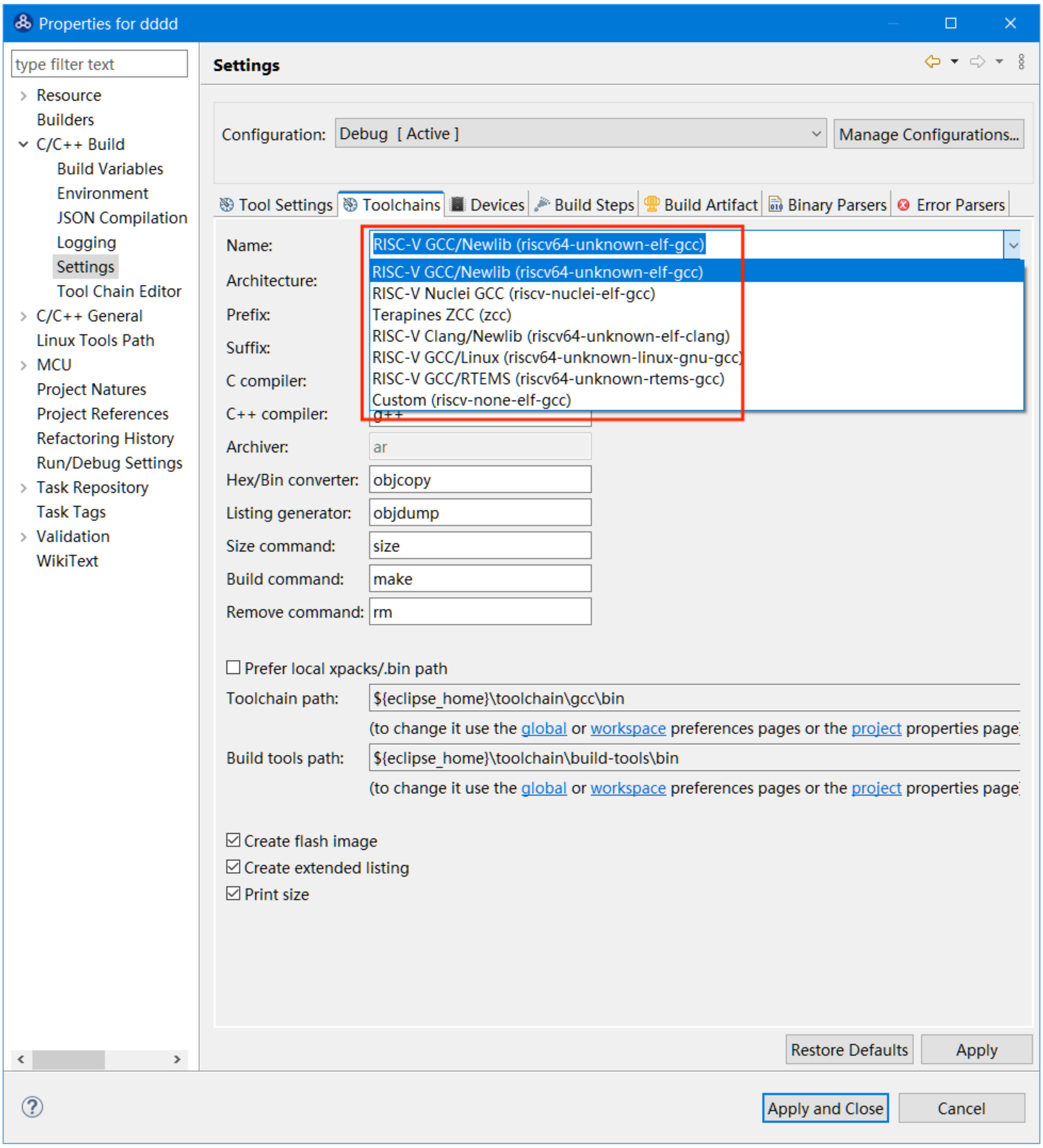


在完成上述操作后，打开工具链配置页，点击 Apply 保存修改。



2.8 Nuclei Studio 编译工程

在 NucleiStudio 中，支持多种 Toolchain，以便用户在创建工程时，可以选择不同的 Toolchain 对工程进行编译。NucleiStudio 中已经集成了 GCC 13、Clang 17、ZCC Lite 三款编译器，用户无需下载即可使用。



- **RISC-V Nuclei GCC (riscv-nuclei-elf-gcc)**

  早期 NucleiStudio 对 GCC 10 编译器的支持，Nuclei Studio 2023.10 及之后版本弃用，如果用户需要对 GCC 10 的支持，需要自行下载 Nuclei RISC-V Toolchain 2022.12(gcc10) 并替换 `<NucleiStudio>/toolchain/gcc10` 目录内容。

- **RISC-V GCC/Newlib (riscv64-unknown-elf-gcc)**

  Nuclei Studio 2023.10 及之后对 GCC 13 编译器的支持，对应的软件包存放在 `<NucleiStudio>/toolchain/gcc` 目录。

- **RISC-V Clang/Newlib (riscv64-unknown-elf-clang)**

  Nuclei Studio 2023.10 及之后对 Clang 17 编译器的支持，对应的软件包存放在 `<NucleiStudio>/toolchain/gcc` 目录。

- **Terapines ZCC (zcc)**

  Nuclei Studio 2024.06 版本对 ZCC 编译器的支持，对应的软件包存放在 `<NucleiStudio>/`

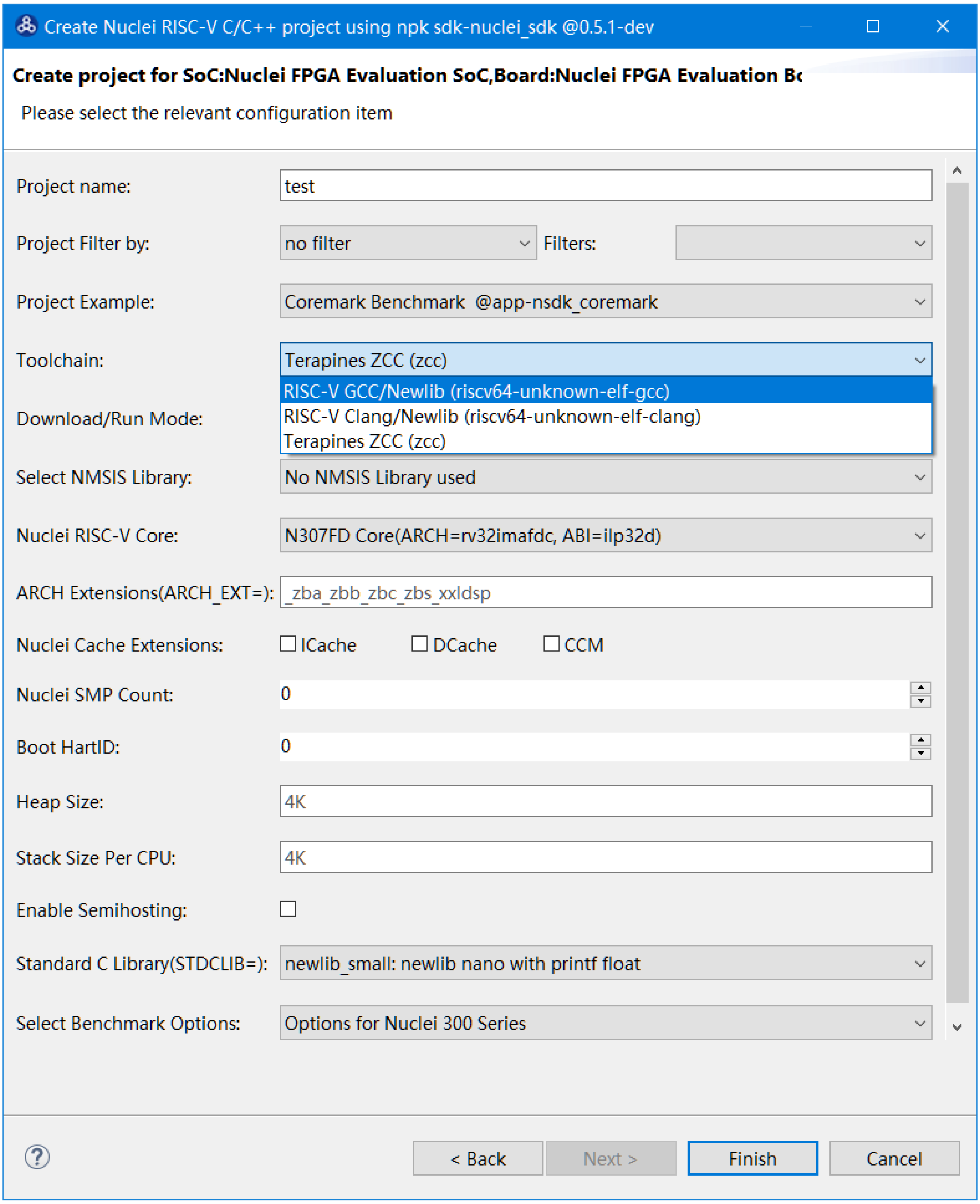
toolchain/zcc 目录。Terapines ZCC 工具链工程的创建需要依赖 Nuclei SDK **> 0.6.0** 之后的版本才可以, 目前可以通过 Nuclei SDK 命令行的方式进行测试。

2.8.1 编译工具

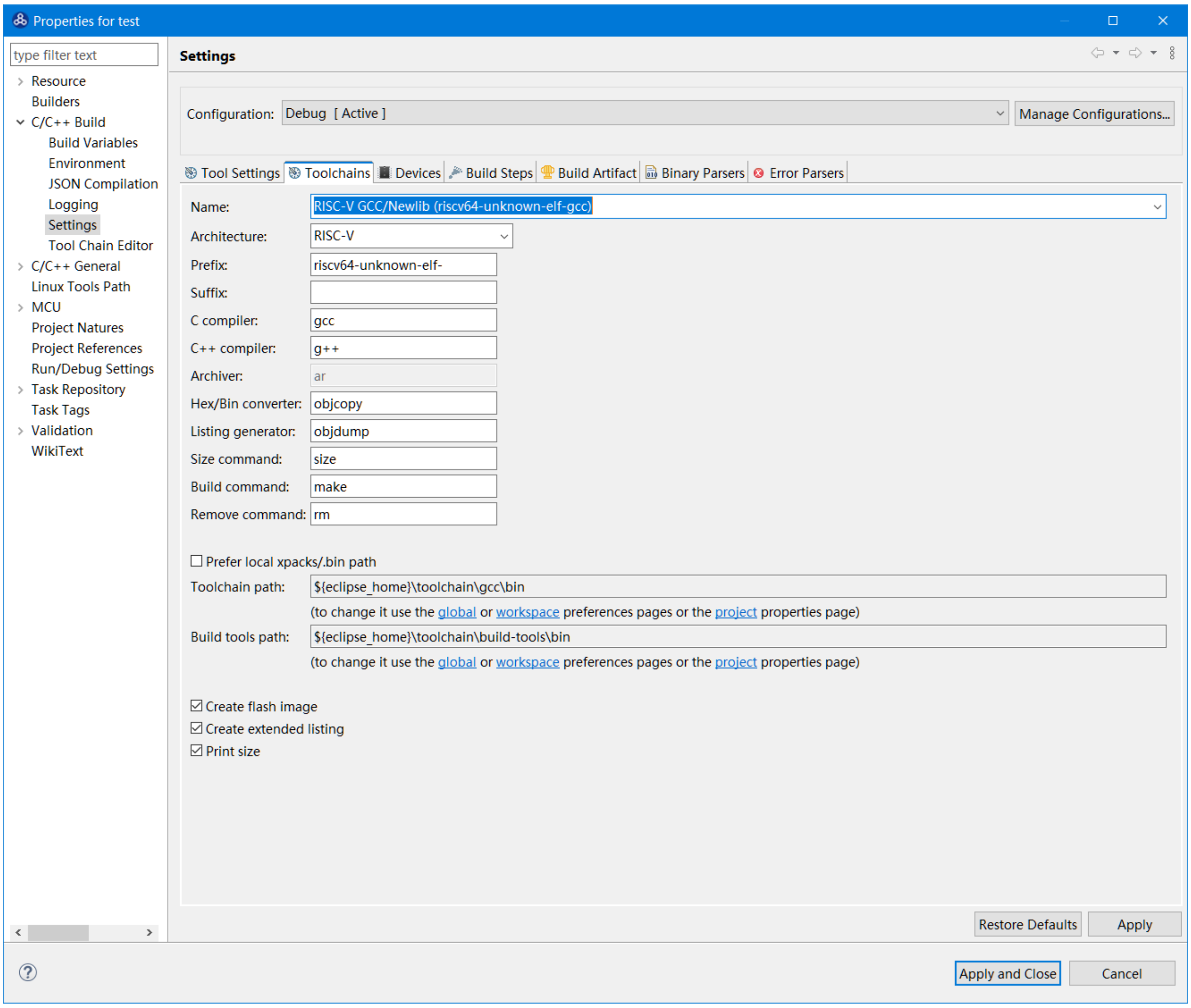
Nuclei GNU Toolchain

GNU Toolchain 是由 GNU 项目提供的一套完整的软件开发工具链，它包括了编译器、调试器、链接器、库文件等一系列用于软件开发和构建的必需工具。GNU Toolchain 以其开源、跨平台、高度可定制和强大的功能特性，成为了全球开发者社区广泛使用的开发工具集。Nuclei Studio 2023.10 之前的版本中集成了 GCC 10 ；Nuclei Studio 2023.10 及之后的版本中，集成了 GCC 13 。

在 nuclei_sdk 0.5.0 之后的版中，在创建工程时，用户可以选择 Toolchain 为 RISC-V GCC/Newlib（riscv64-unknown-elf-gcc）则可以创建一个支持 GCC 13 编译的工程，NucleiStudio 将默认将相对应的编译选项配置好。关于 GCC 10 与 GCC 13 工程的问题，可以参阅通过工具将工程转换成支持 *gcc 13* 的工程 (page 189) 。



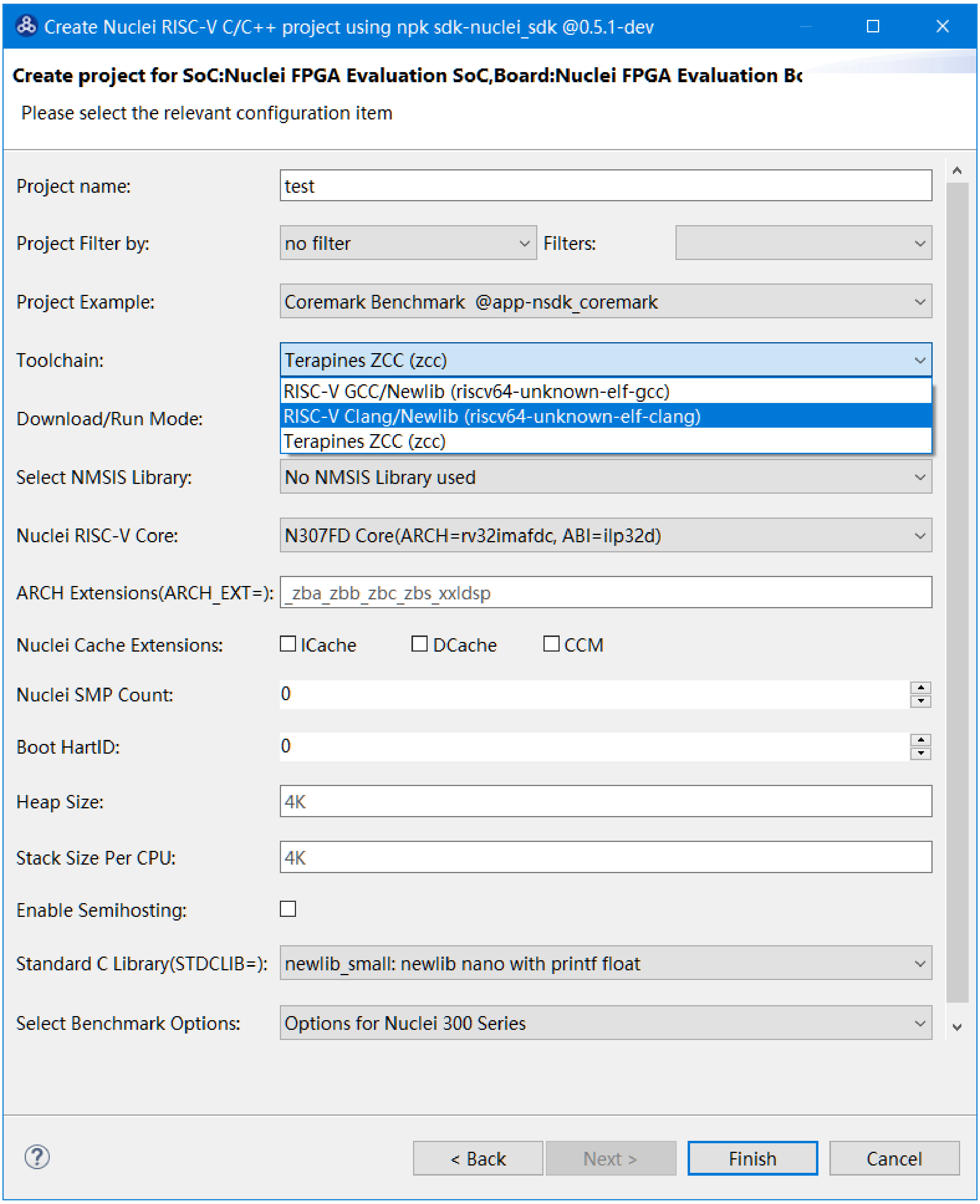
在工程上右键->Properties->C/C++ Build->Setting->Toolchains 中可以看到工程所用到的 Toolchains 有一些工具，以及 Toolchains 所在的路径。



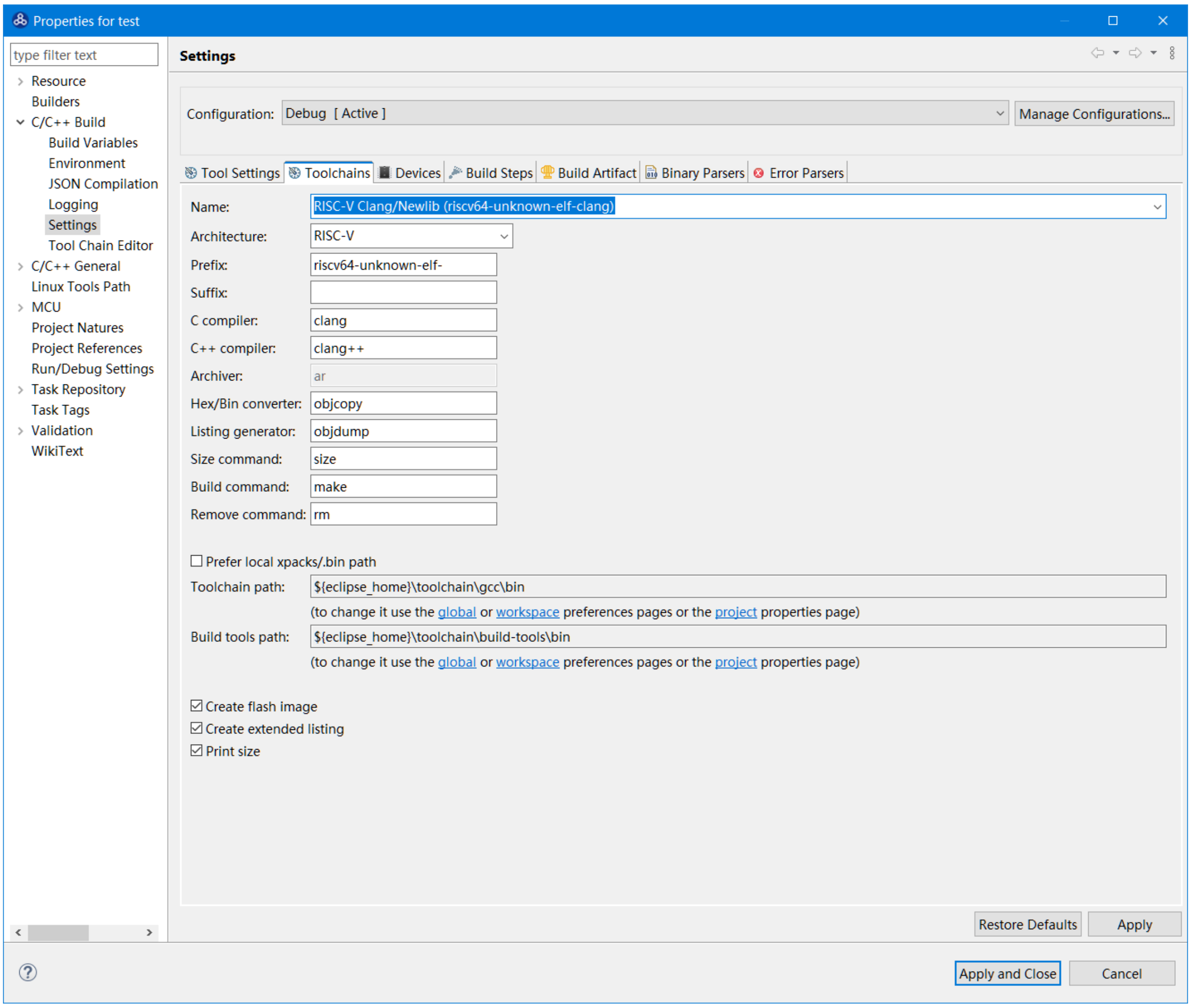
Nuclei LLVM Toolchain

LLVM Toolchain 是一套为 C 系列编程语言设计的完整工具链，旨在提供从源代码到可执行文件的编译和链接过程。LLVM Toolchain 的核心组件包括 Clang 编译器、LLVM 编译器基础设施以及相关的工具和库。LLVM Toolchain 是一套功能强大、灵活可扩展的编译工具链，它采用了先进的编译技术和设计理念，为开发者提供了高效、便捷的编译和构建解决方案。Nuclei Studio 2023.10 及之后的版本中，集成了 LLVM Toolchain。

在 nuclei_sdk 0.5.0 之后的版中，在创建工程时，用户可以选择 Toolchain 为 `RISC-V Clang/Newlib (riscv64-unknown-elf-clang)` 则可以创建一个支持 `Clang 17` 编译的工程，NucleiStudio 将默认将相对应的编译选项配置好。



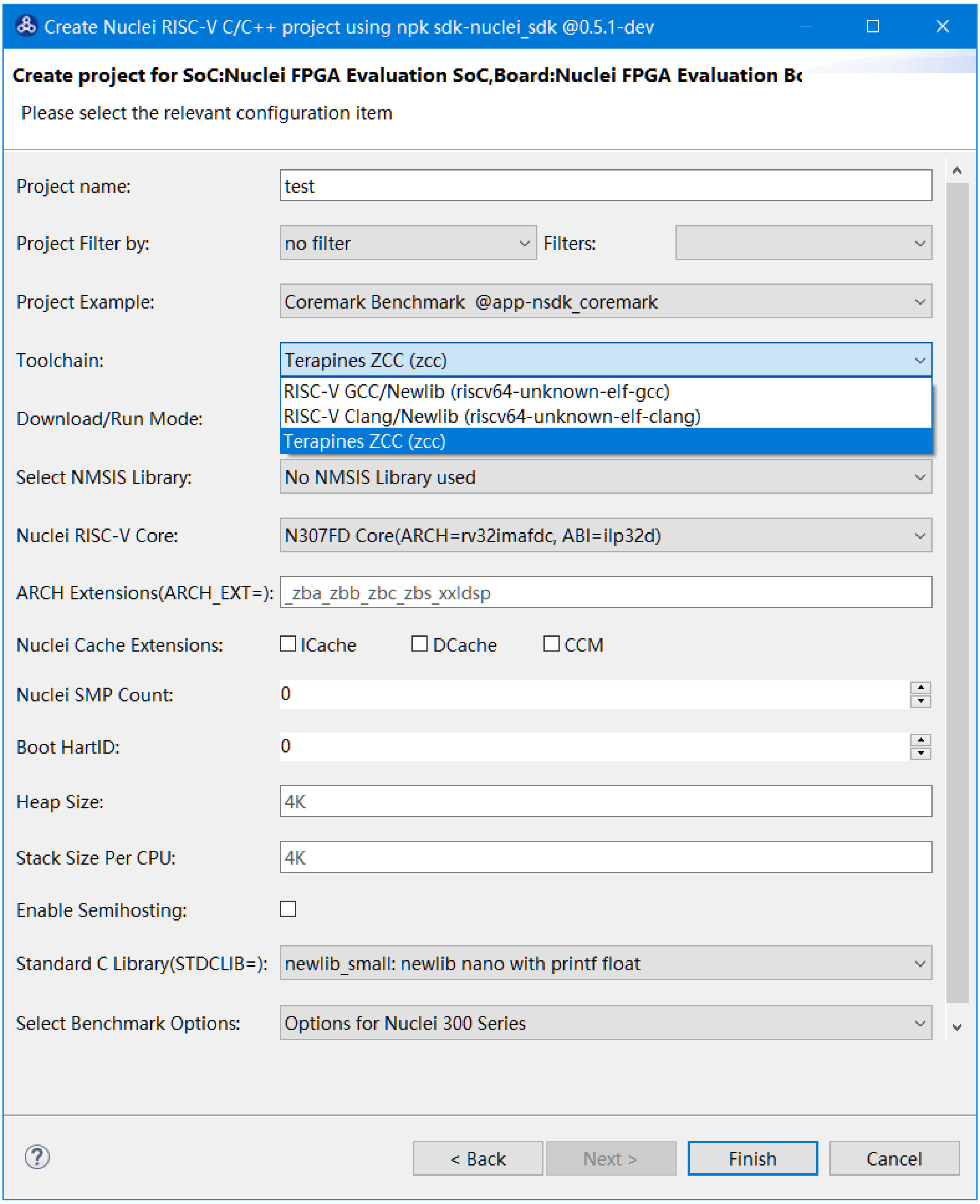
在工程上右键->Properties->C/C++ Build->Setting->Toolchains 中可以看到工程所用到的 Toolchains 有一些工具，以及 Toolchains 所在的路径。



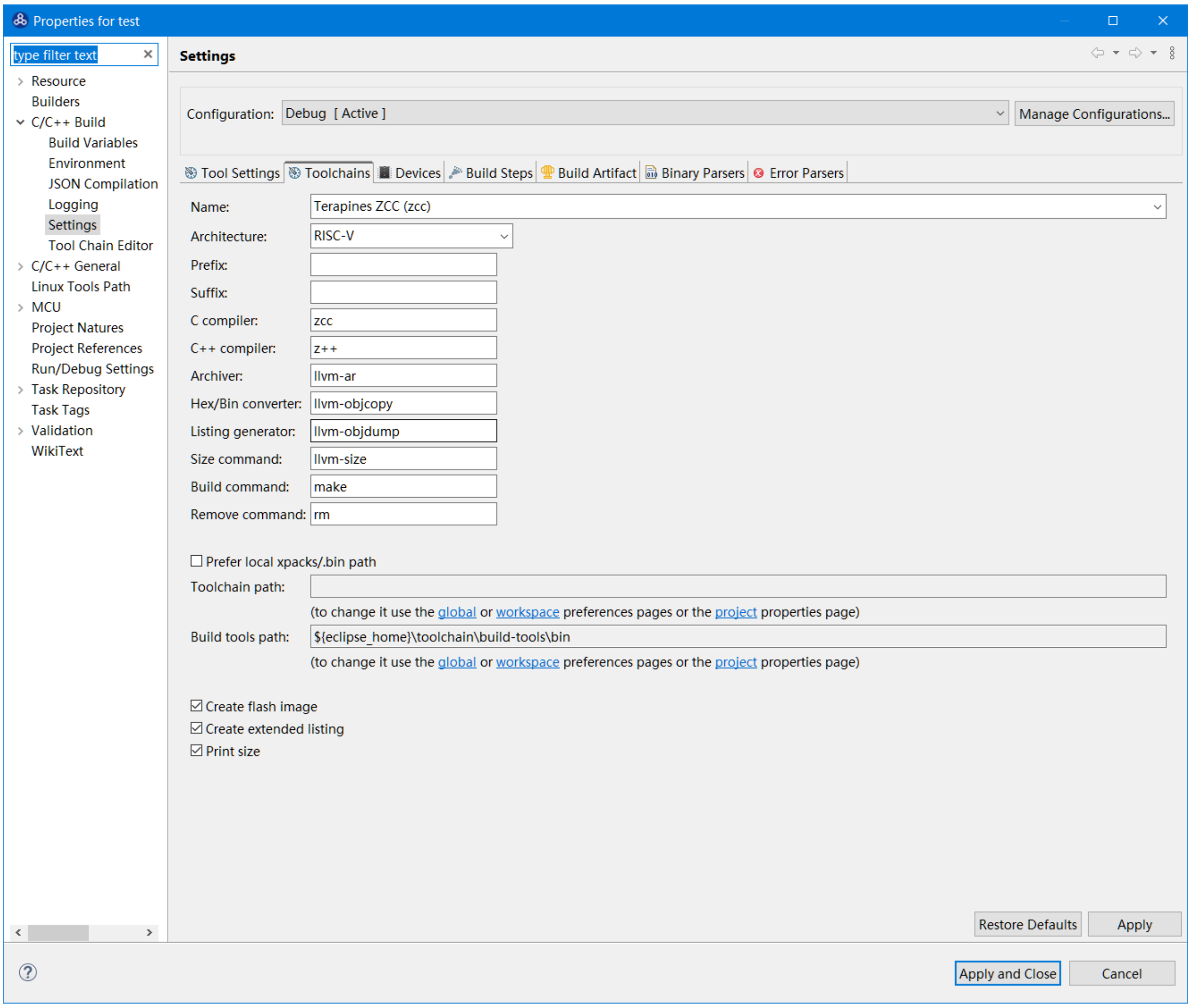
Terapines ZCC

Terapines ZCC 是兆松科技研发的高性能 RISC-V 编译器。Nuclei Studio 2024.06 版中对 Terapines ZCC 进行支持，集成了 **ZCC Lite** 版本的工具链，如果需要更新，可以自行下载好 Terapines ZCC 后，替换 `<NucleiStudio>/toolchain/zcc` 目录下的内容，可以在 NucleiStudio 中直接创建一个支持 Terapines ZCC 的工程，并使用 Terapines ZCC 进行编译。

关于 Terapines ZCC 参见：https://products.terapines.com/downloads



在工程上右键->Properties->C/C++ Build->Setting->Toolchains 中可以看到工程所用到的 Toolchains 有一些工具，以及 Toolchains 所在的路径。在这里可以配置用户所下载的 ZCC 编译器的路径。

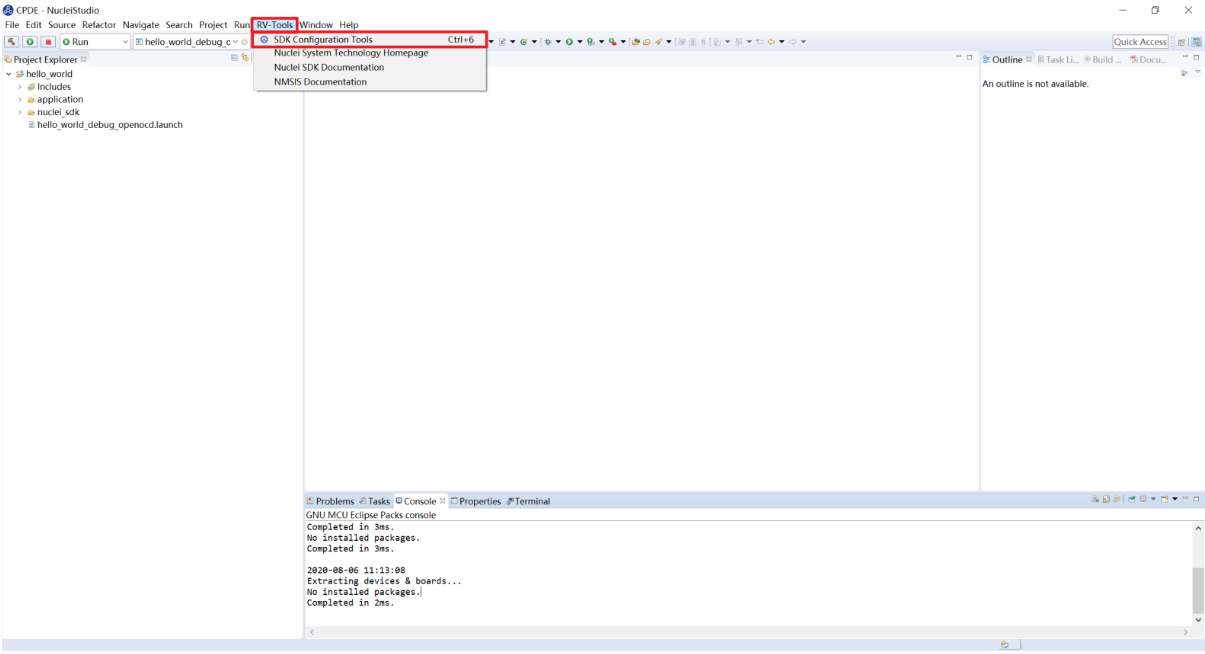


2.8.2 Nuclei SDK 工程设置工具

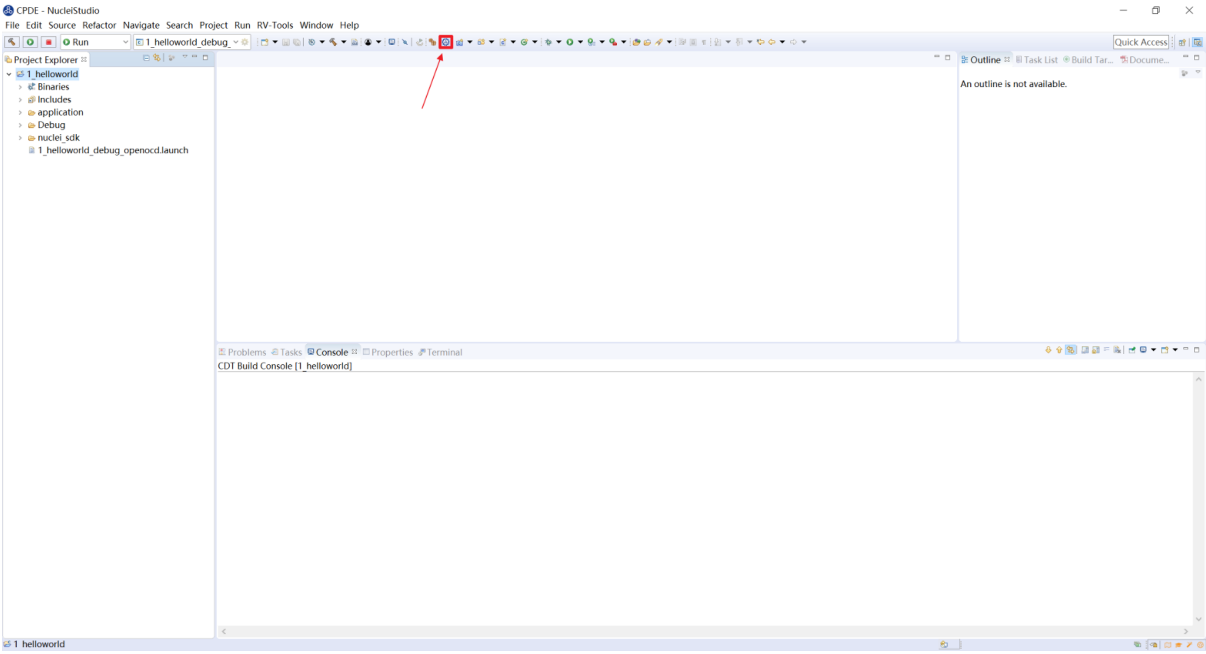
在 Nuclei Studio 中可以通过 Nuclei SDK 工程设置工具一键修改通过 Nuclei SDK Project Wizard 创建的工程的编译链接选项。

> **Note**
>
> 本工具目前仅支持 Nuclei SDK，HBird SDK，Nuclei Subsystem SDK, 不支持无模板手动创建的项目和基于 Makefile 创建的项目, 或者是自行创建维护的项目

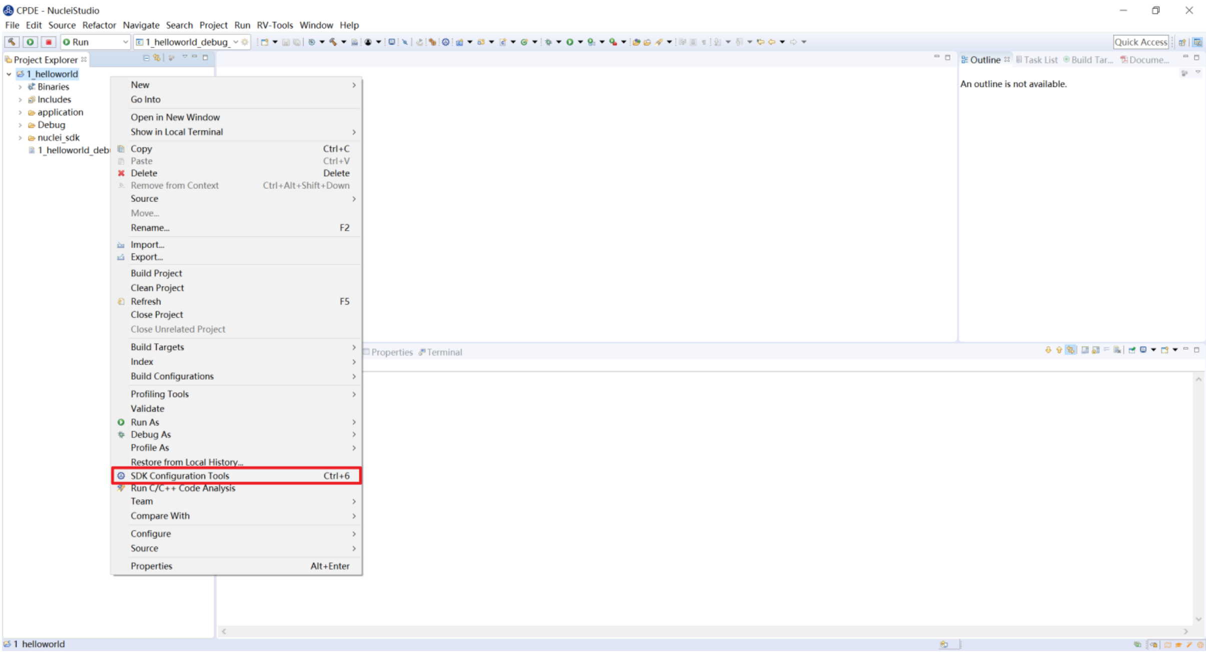
单击选中需要修改的工程，之后如图 6-1 打开 `RV-Tools>SDK Configuration Tools`，可以打开修改编译选项的弹窗，在 **2023.10** 版本以后，将直接打开 **Nuclei Settings** 页面。也可以单击要修改的工程后，点击工具栏的工程设置工具图标。



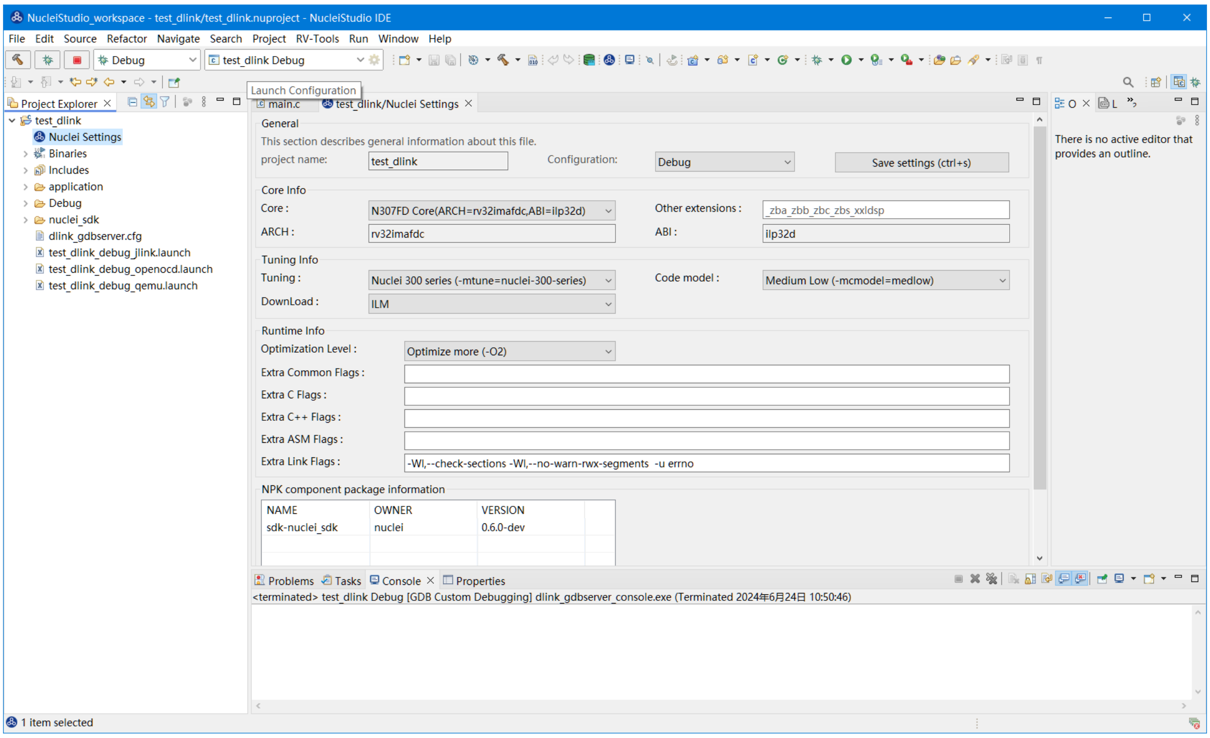
或者单击要修改的工程后，键盘按下 ctrl+6 。



也可以单击要修改的工程后，右击打开右键菜单，选择 SDK Configuration Tools 。



SDK Configuration Tools 各选项详细功能如下：



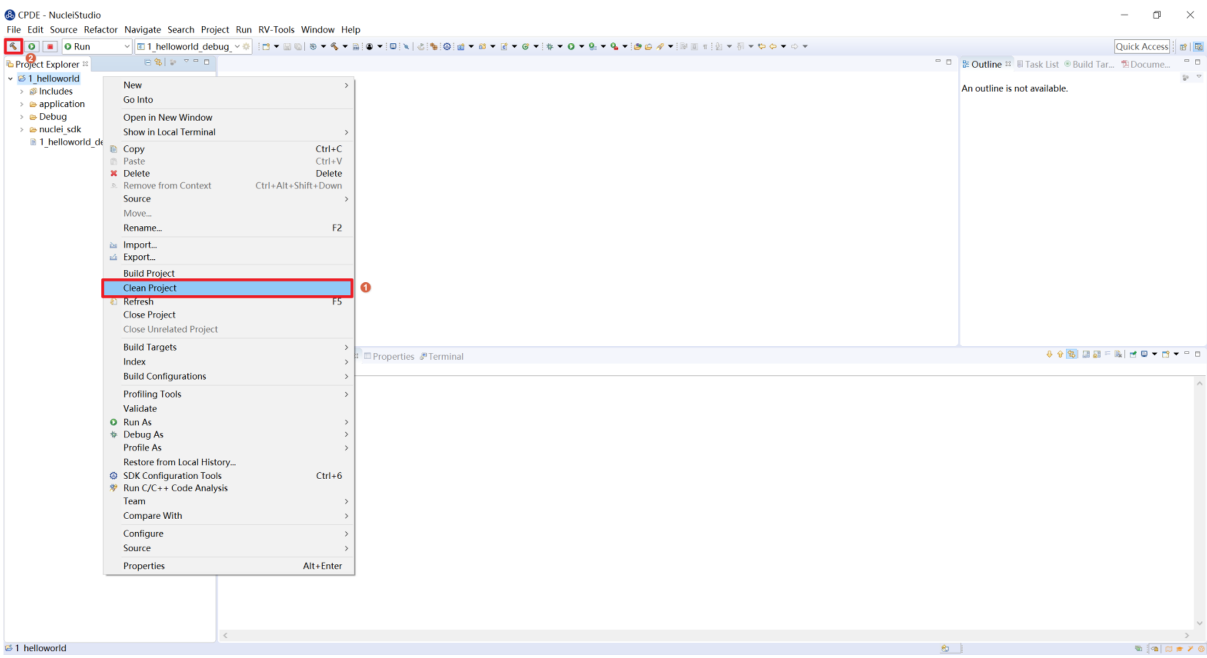
- projectName 为当前选中的工程名。
- Core 为当前工程对应的内核。由于工具根据 ARCH 和 ABI 选项反推出对应的内核，而不同的内核可能有相同的 ARCH 和 ABI 选项，所以显示上可能会有所偏差，只要 ARCH 与 ABI 为正确的选项即可。此选项为方便快速切换内核选项使用。
- 四个勾选项：`Bitmanipulation Extension(RVB)`，`Cryptography Extension(RVK)`，`Packed SIMD/DSP Extension`，`Vector Extension(RVV)` 用于选择对应的扩展指令集 (`B/K/P/V`)。

Note

> 本功能在 2023.10 版本中移除，使用 `Other Extensions` 输入框来制定额外的扩展。

- ARCH 对应的当前工程的 arch 选项, 根据 Core 和勾选项自动组合。
- ABI 对应的当前工程的 abi 选项。
- Tuning 根据不同级别处理器优化的 gcc 选项，选择 Core 会自动选择正确的 Tuning 选项，不建议自己调整。
- Code Model 针对 RV32 处理器，自动选择为 `Medium Low`，而针对 RV64 处理器自动选择为 `Medium High`，选择 Core 以后会自动选择合适的 Code Model，其中 RV64 处理器必须使用 `Medium High`。
- Download 对应当前工程的下载模式，可以切换选择不同的下载模式，目前仅 Nuclei FPGA 评估开发板支持切换下载模式，RVSTAR 仅有 `FLASHXIP` 模式。其中切换到 flash 模式会额外定义 `VECTOR_TABLE_REMAPPED` 宏，其他模式不会定义这个宏
- `Select C Runtime Library` 对应的使用标准 C 库，本功能在 **2023.10** 版本中移除。在工程创建的时候，如果创建的工程采用的是 Newlib，则这里只能进行 newlib 版本的切换，如果创建的工程才用的是 `Nuclei C Runtime Library` (*libncrt* (page 321))，则这里只能进行 libncrt 版本的切换。
- `Optimization Level` 对应编译的优化等级。
- `Extra Common Flags` 对应的是额外的通用编译选项。可以添加额外的通用编译选项。
- `Extra C Flags` 对应的是额外的 C 编译选项。可以添加额外的 C 编译选项。
- `Extra C++ Flags` 对应的是额外的 C++ 编译选项。可以添加额外的 C++ 编译选项。
- `Extra ASM Flags` 对应的是额外的汇编编译选项。可以添加额外的汇编编译选项。
- `Extra Link Flags` 对应的是额外的链接选项。如果此选项已经有默认选项并且需要增加编译选项，可以在编译选项开头或结尾处相隔一个空格字符再增加编译选项。

根据需要修改以上的选项，这里我们修改优化等级为 `-Os` 优化生成可执行文件大小。点击 `save` 一键修改编译选项，`save` 以后一定要先 `clean project`，之后右击修改后的工程打开右键菜单，选择 `Clean Project` 清理一下工程，再点击锤子图标即可完成修改编译选项后重新编译工程。



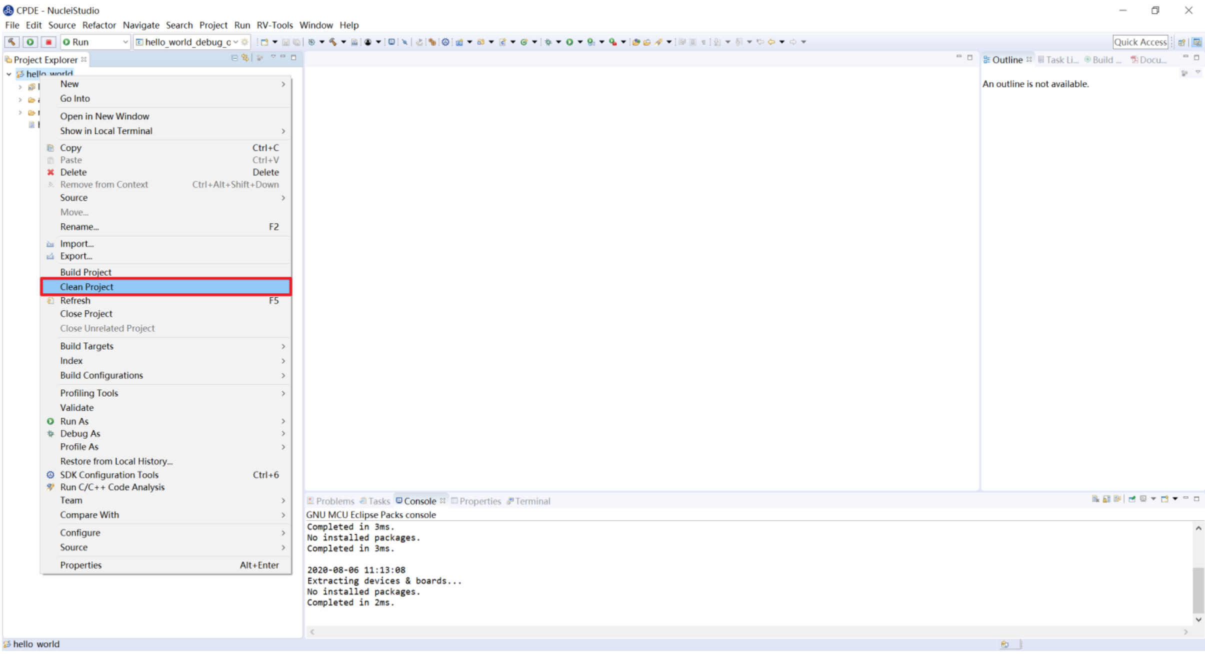
这里的 SDK Configuration Tool 切换不会对 Debug Configuration 选项做任何改动，因此如果切换了 Core 以后，对应的调试配置 (OpenOCD/QEMU/JLink) 也需要手动修改。

需要注意的是如果要切换工程从 32 位变为 64 位，需要打开调试设置页面，修改 `command` 中 `set arch riscv:rv32` 为 `set arch riscv:rv64`，从 64 位切换回 32 位也应当修改这里的参数为对应的数值。

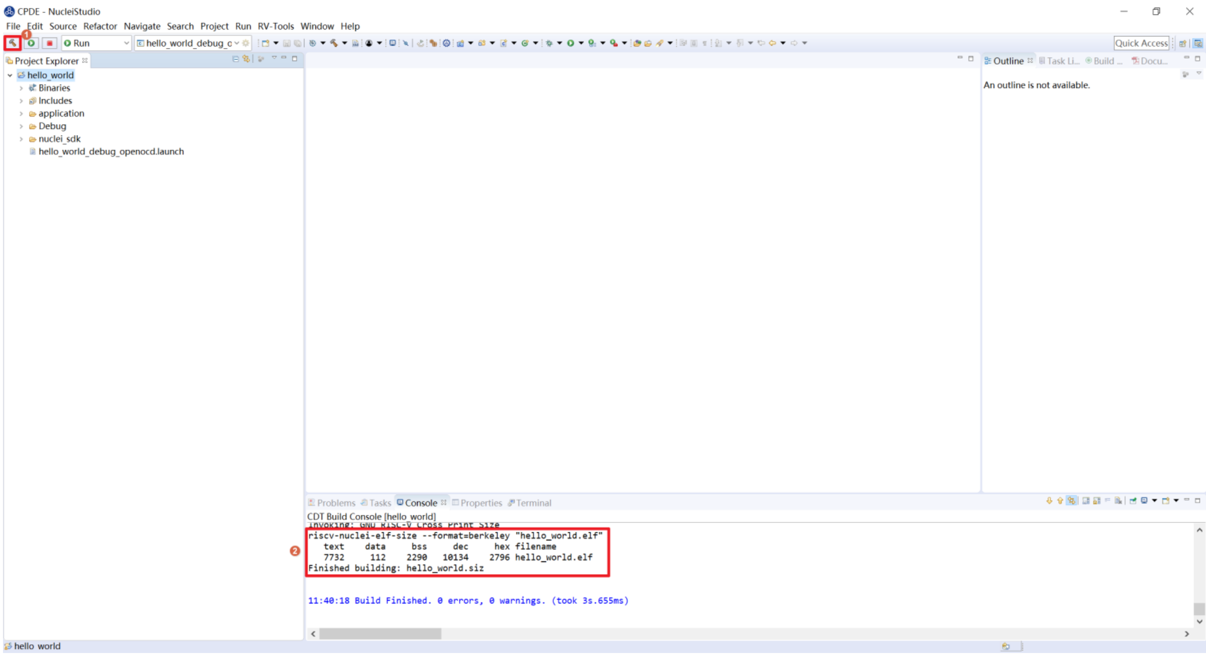
2.8.3 Nuclei Studio 中编译 Hello World 项目

在 Nuclei Studio 中编译 Hello World 项目的步骤如下。

在编译工程前，建议先将项目清理一下。在 `Project Explorer` 栏中选中 `hello_world` 项目，单击鼠标右键，选择 `Clean Project`。

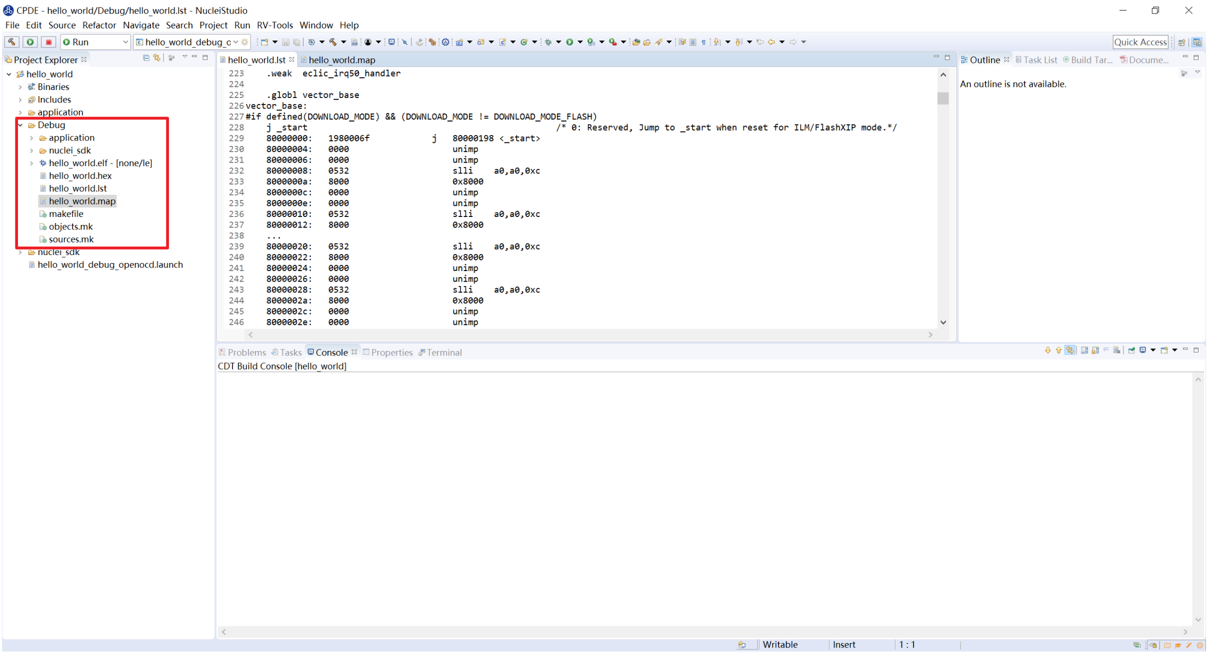


单击 Launch Bar 菜单上的锤子按钮，开始对项目进行编译。如果编译成功，能够看到生成可执行文件的代码体积大小，包括 text 段、data 段和 bss 段，以及总大小的十进制和十六进制数值。使用 Makefile 方式新建的工程需要在右键菜单中选择 `Build Project` 进行编译。



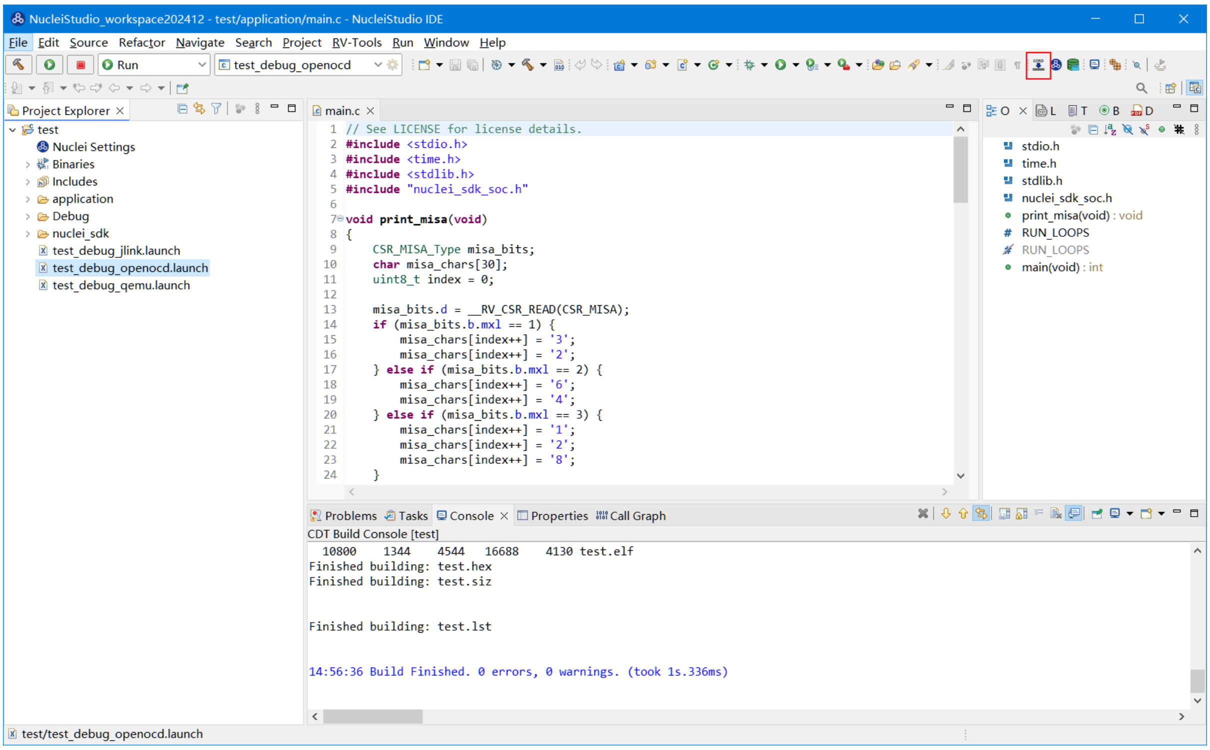
编译成功后可以看到增加了 Debug 文件夹，各文件作用如下：

- hello_world.elf 是生成的可执行文件。
- hello_world.hex 是生成的 Hex 文件。
- hello_world.lst 是生成的 list 文件，可以看到反汇编和简单的代码分部信息。
- hello_world.map 是生成的 map 文件，可以详细的看到生成的代码分布情况。

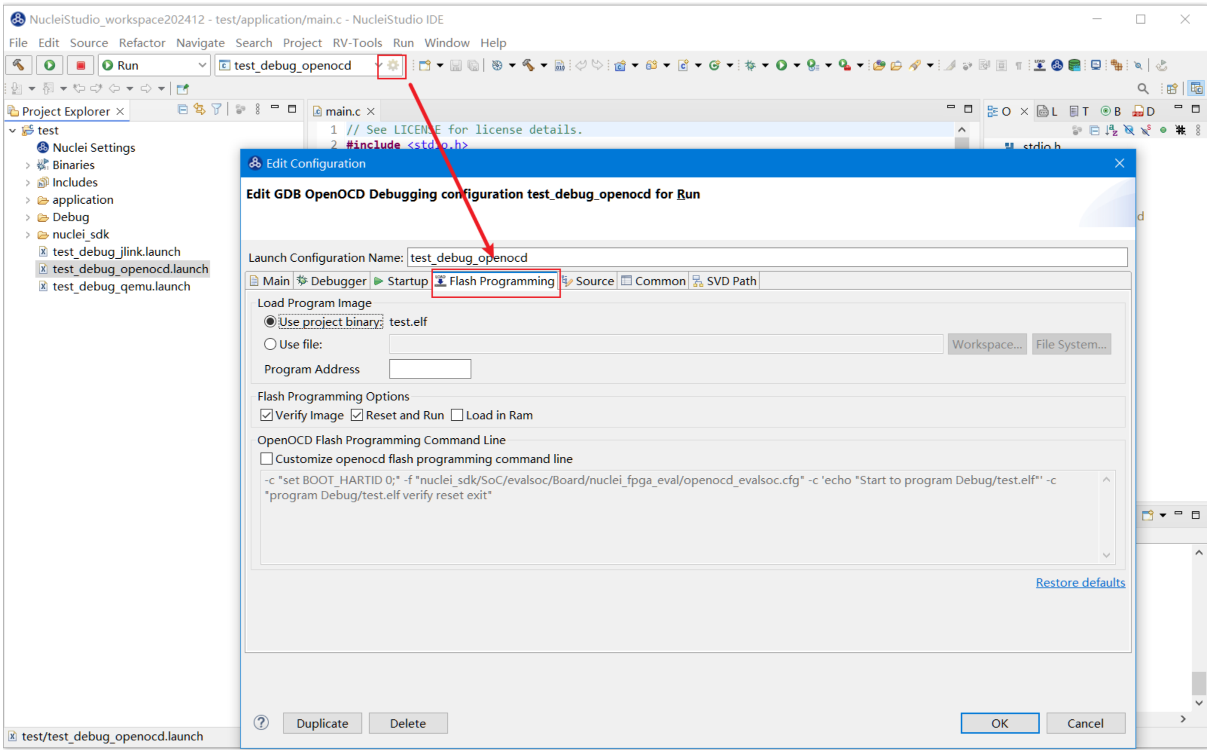


2.8.4 Flash Programming

为了满足用户将编译好的二进制文件直接下载到硬件开发板的需求，Nuclei Studio 新增了 Flash Programming 功能。该功能允许用户快速、便捷地将编译好的二进制文件直接下载到硬件开发板中，极大提升了开发和调试的效率；简化操作流程，用户只需点击一次即可完成二进制文件的下载。工程编译好后，找到 Flash Programming，并点击，即可完成二进制文件的下载。

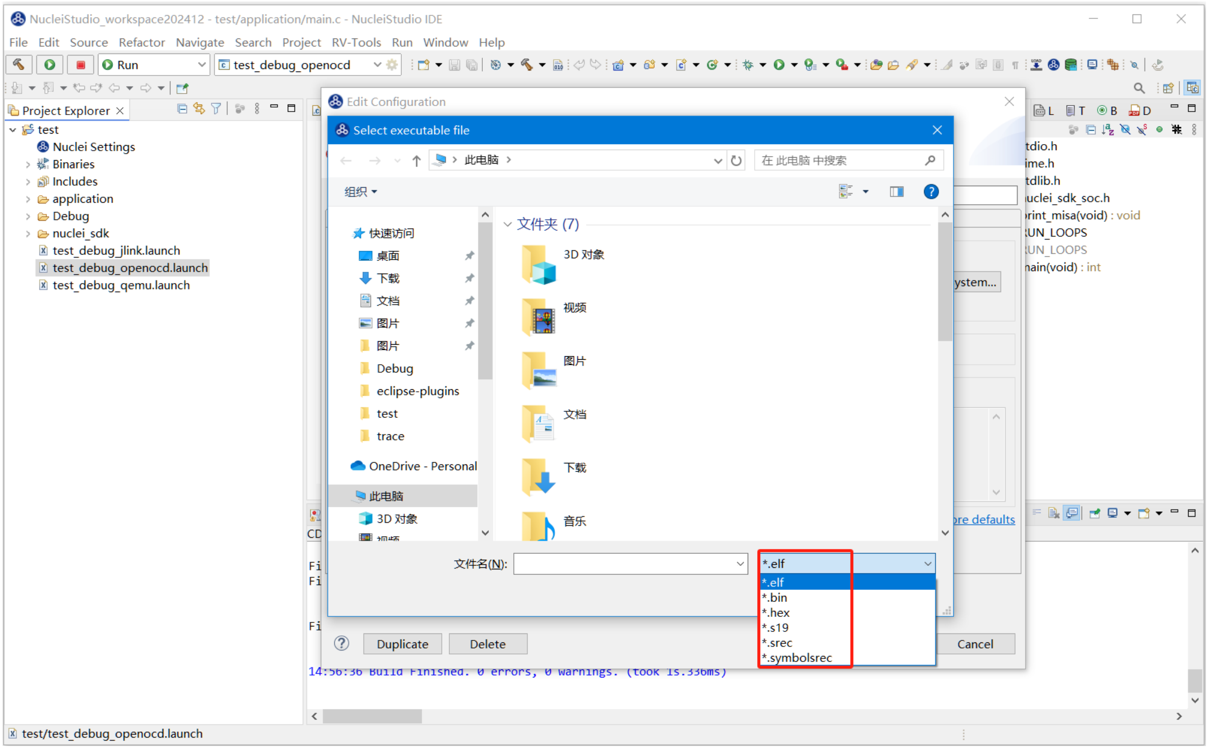


用户也可以修改其相关的配置信息，在 Launch Bar 中点击配置按钮，打开配置页面，然后选中 Flash Programming 选项卡。



Load Program Image

Load 的文件，默认的 elf 格试的文件，也可以支持 `*.bin`、`*.hex`、`*.s19`、`*.srec`、`*.symbolsrec` 等各种格式



Flash Programming Options

Flash Programming 的选项有以下三个

- Verify Image：在烧录完备后，会校验烧录的镜像文件与当前连接的目标设备上下载进去的文件是否一致，确保烧录成功。
- Reset and Run：烧录完成后，让 CPU 复位并并运行，需要注意如果勾选了 Load in RAM，则只会运行不会复位（如程序烧录在 RAM 中，复位将会使程序丢失）。
- Load in Ram：勾选这个表示固件需要下载到内存中，而不是 Flash 等非易失性存储器上，选中后，必须指定 Program Address。

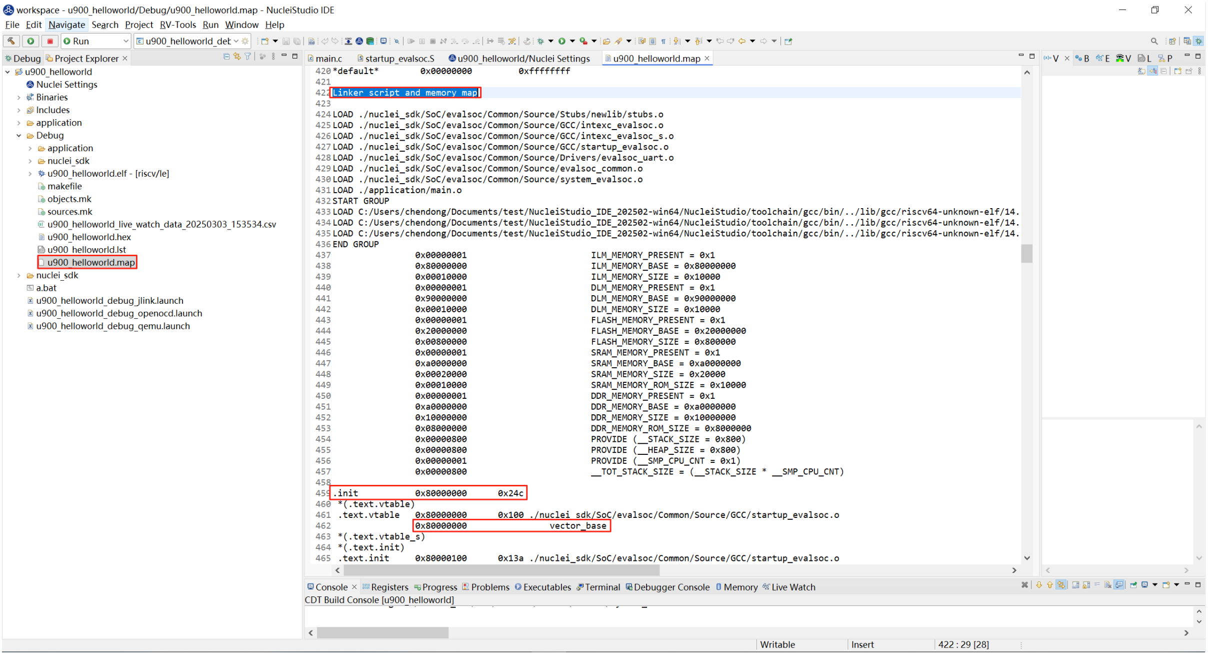
> **Note**
>
> Load in Ram 勾选和不勾选固件下载方式不一样，所使用的 OpenOCD 命令也不同，后续文档中有详细介绍，请仔细阅读并根据实际情况准确选择。

上述参数的使用，与工程的 Download 模式匹配，Nuclei Studio 中默认支持 `DDR/ILM/SRAM/FLASH/FLASHXIP`。

当工程 Download 模式是 `DDR/ILM/SRAM` 时，`Load in Ram` 必须选中，同时 `Program Address` 地址也不能为空，这里的 `Program Address` 地址是程序烧写入 Ram 时的第一个地址。

一般 `Program Address` 可以在 `*.map` 文件中找到，或者执行 riscv64-unknown-elf-readelf -h /path/to/your.elf 后查看 Entry point address，本文档仅介绍从 `*.map` 文件中查找 `Program Address`。

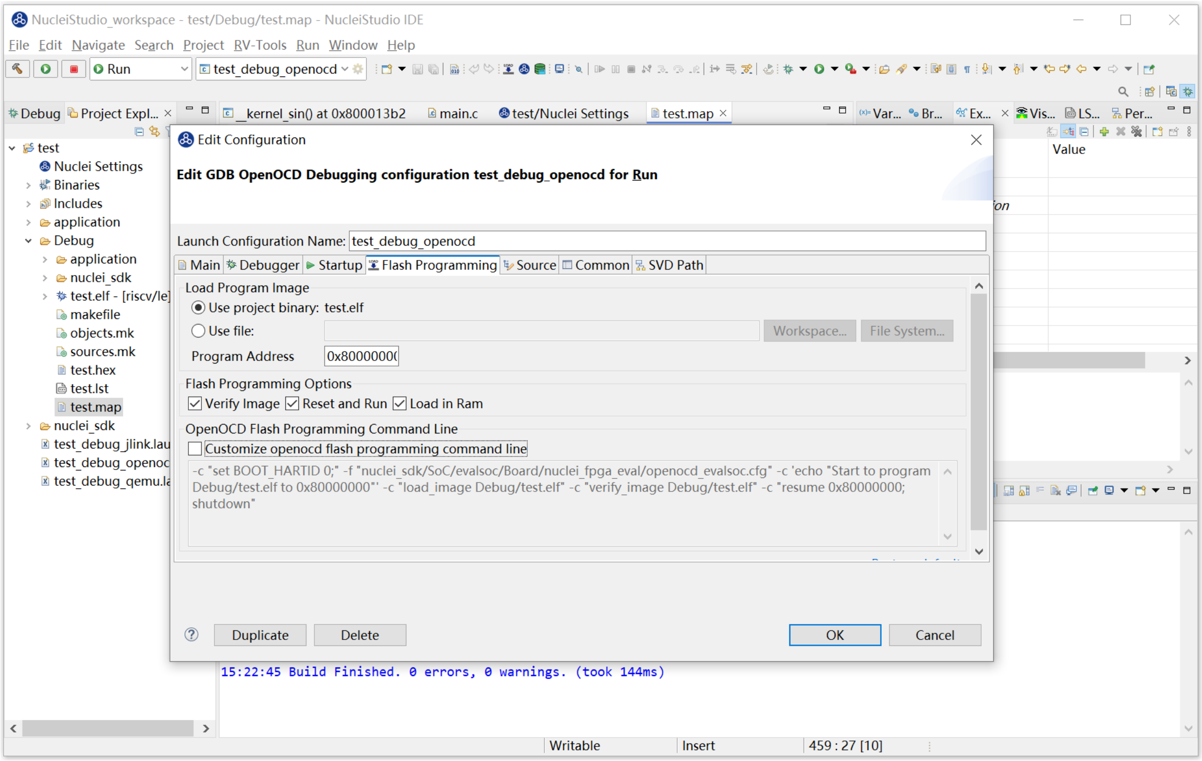
打开 `*.map` 文件，搜索 `Linker script and memory map`，然后找到 `.init` 后面的这个地址就是 `Program Address`。



当选中 Load in Ram 时，同时选中 Verify Image，命令行中将多一行命令 -c "verify_image Debug/test.elf"，此时是通过 verify_image 命令来实现镜像文件的检查。

当选中 Load in Ram 时，同时选中 Reset and Run 、命令行中将多一行命令 -c "resume 0x80000000; shutdown"，此时是通过 resume 命令来实现 load 后可能强制系统复位。

```
-c "set BOOT_HARTID 0;"
-f "nuclei_sdk/SoC/evalsoc/Board/nuclei_fpga_eval/openocd_evalsoc.cfg"
-c 'echo "Start to program Debug/test.elf to 0x80000000"'
-c "load_image Debug/test.elf"
-c "verify_image Debug/test.elf"
-c "resume 0x80000000; shutdown"
```



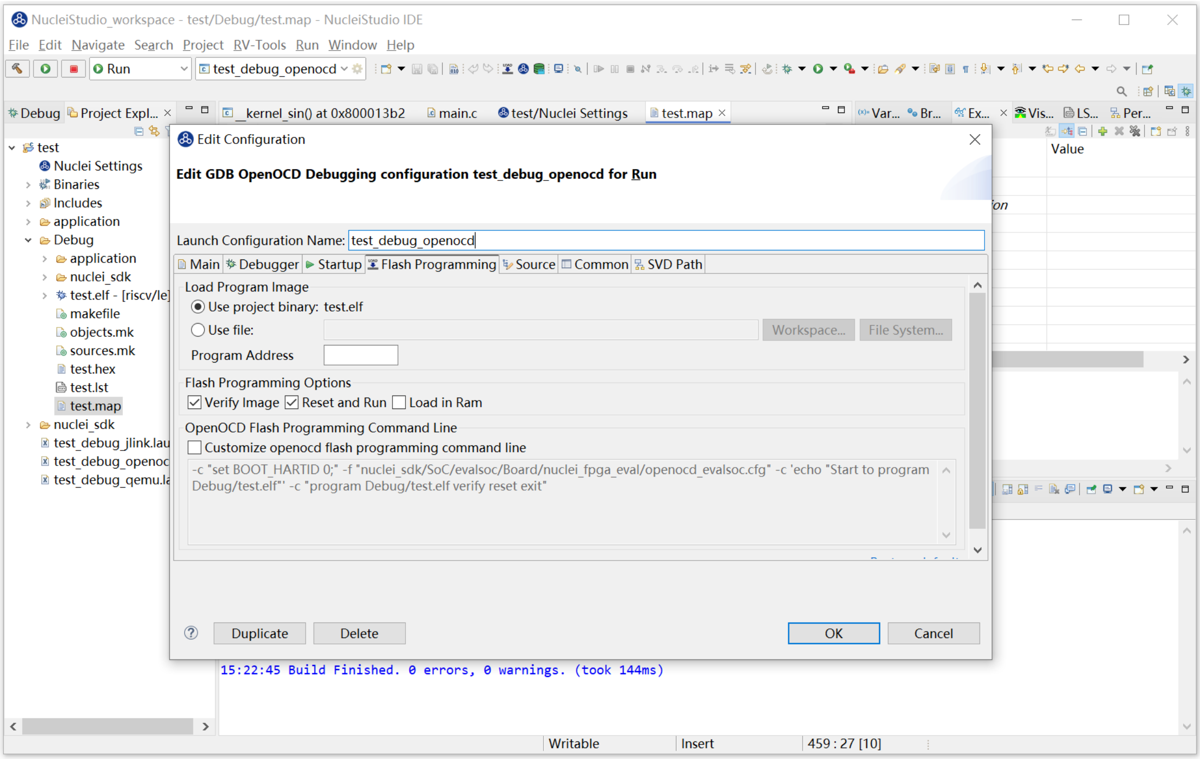
当工程 Download 模式是 FLASH/FLASHXIP 时，则不勾选 Load in Ram，并且 Program Address

为必须为空。

当选中 Verify Image，命令行中将多一行命令 verify，此时是通过 verify 命令来实现镜像文件的检查。

当选中 Reset and Run，命令行中将多一行命令 reset，此时是通过 reset 命令来实现 load 后可能强制系统复位。

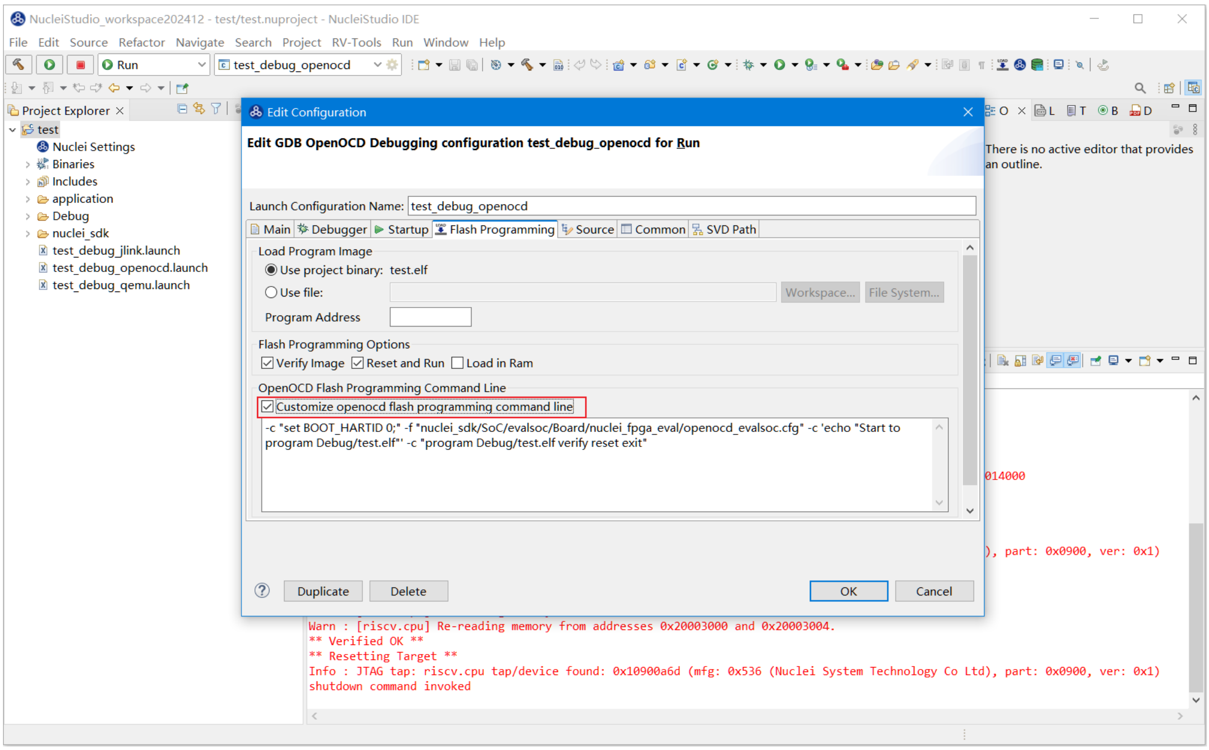
```
-c "set BOOT_HARTID 0;"
-f "nuclei_sdk/SoC/evalsoc/Board/nuclei_fpga_eval/openocd_evalsoc.cfg"
-c 'echo "Start to program Debug/test.elf"'
-c "program Debug/test.elf verify reset exit"
```



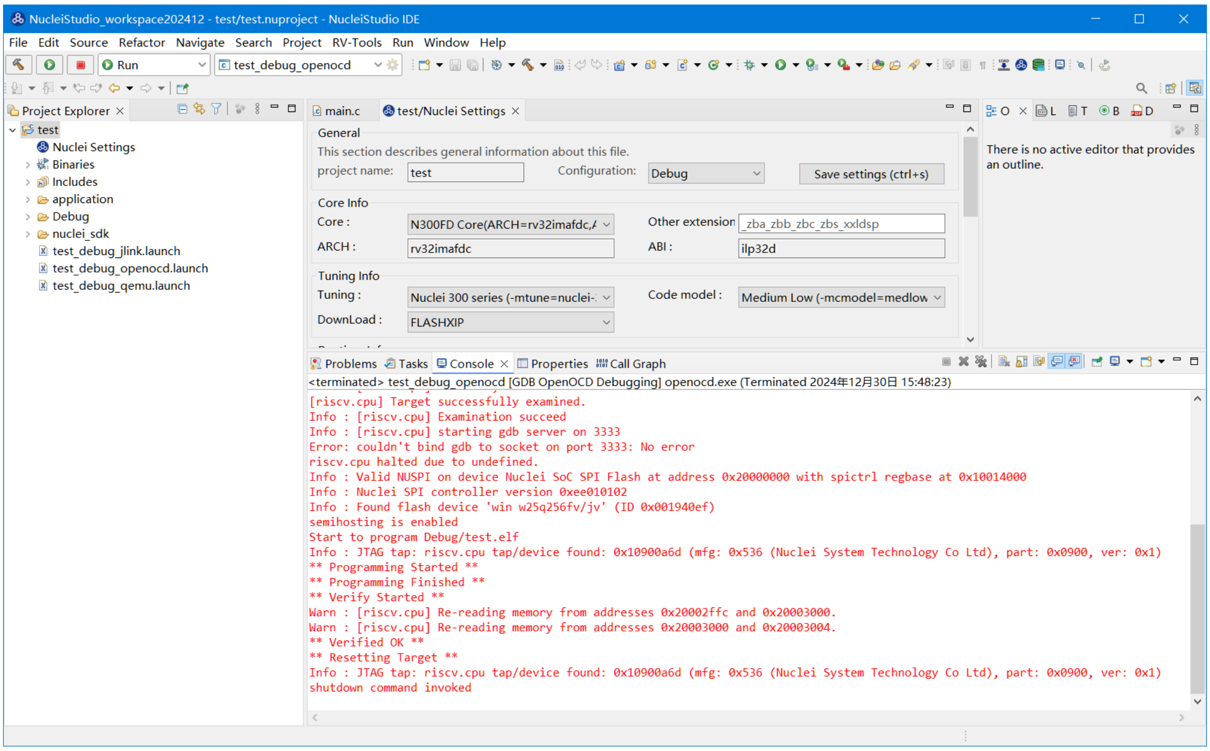
OpenOCD Flash Programming Command line

Flash Programming 中的参数最终将通过 OpenOCD 执行。默认情况下，Customize openocd flash programming command line 选项是未选中的。当您勾选 Customize openocd flash programming command line 时，所有其他相关选项将失效。此时，您可以直接在下方的输入框中输入自定义命令。

如果您对 OpenOCD 命令有足够的了解，可以尝试自定义命令以满足特定需求。然而，如果您不熟悉 OpenOCD 命令行操作，建议不要勾选此选项，以免导致配置错误。



根据需求配置好参数后，点击 Flash Programming 就可以下载二进制代码到硬件中，下载成功的结果如下图。

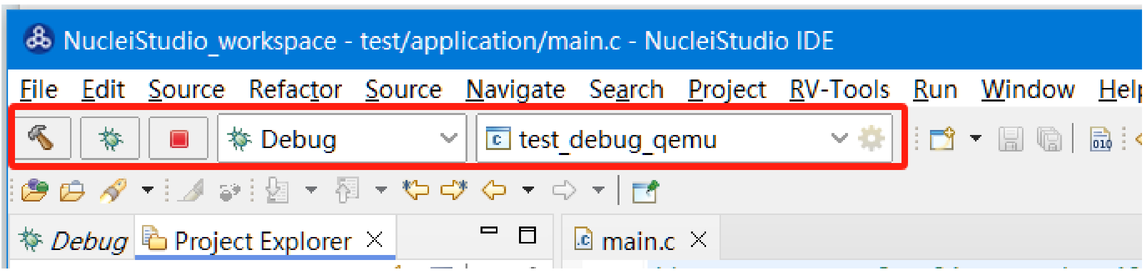


2.9 Nuclei Studio 调试运行工程

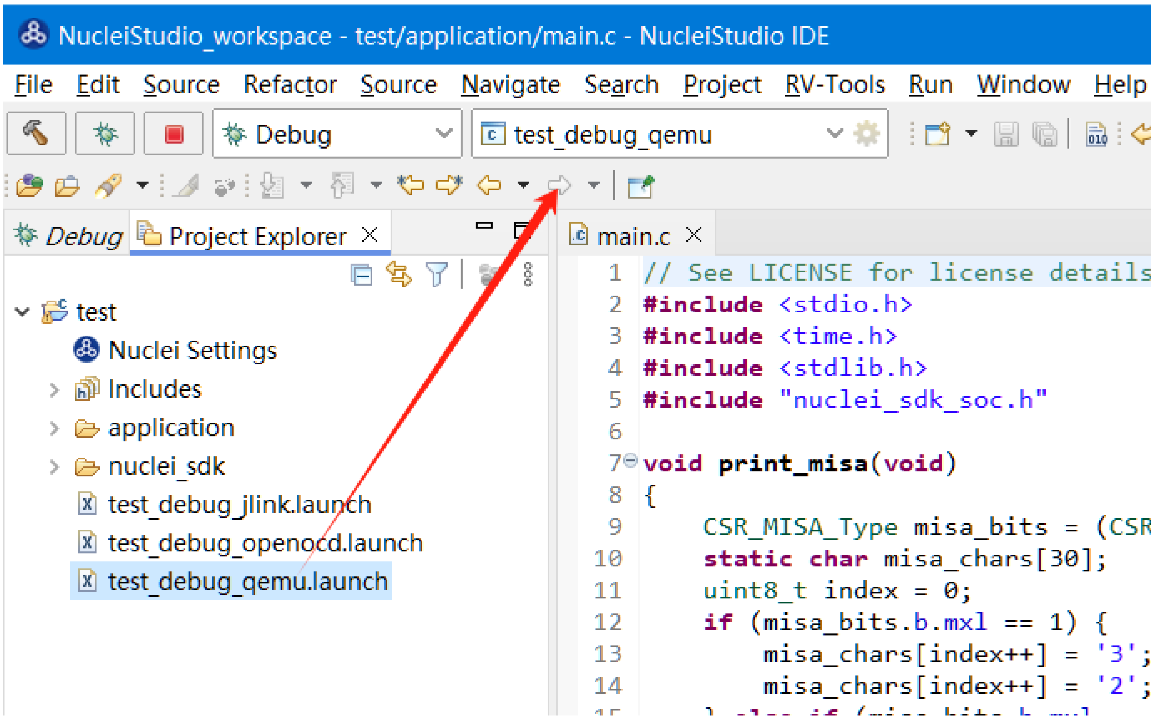
在进行工程调试前，可以先检验一下当前的硬件环境是否正确无误，可以参考*Nuclei Studio* 调试工程前的硬件检测 (page 291) 。

2.9.1 调试模式管理

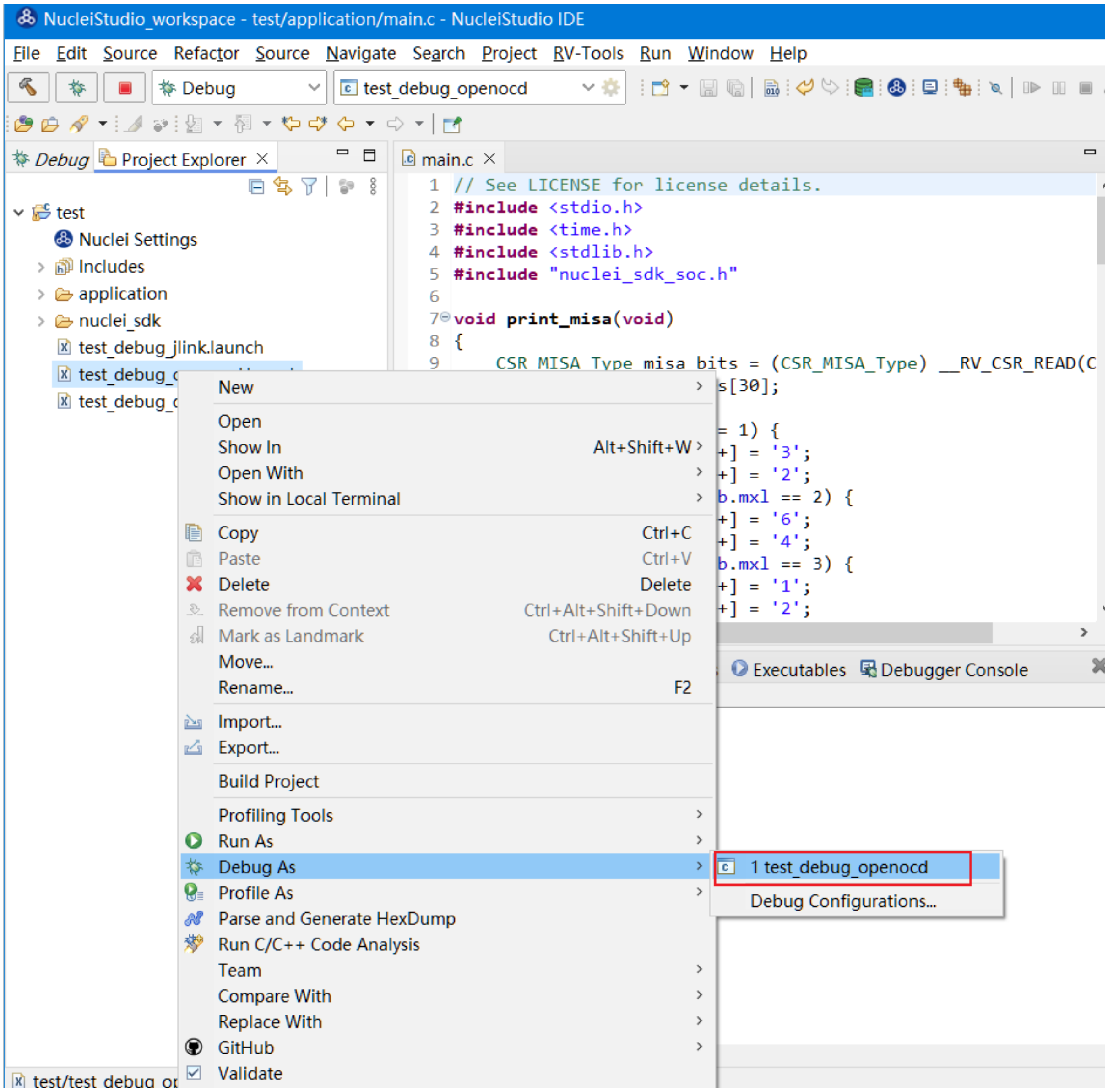
在 NucleiStudio 中，使用 Launch Bar 管理不同的调试器，默认情况下，NucleiStudio 会为 OpenOCD、Jlink、Qemu 生成对应的 `*.launch` 调试文件，NucleiStudio 识别到 `*.launch` 文件后，会将其加入到 Launch Bar 中进行管理，用户可以通过以下三种方式来使用指定的调试模式。



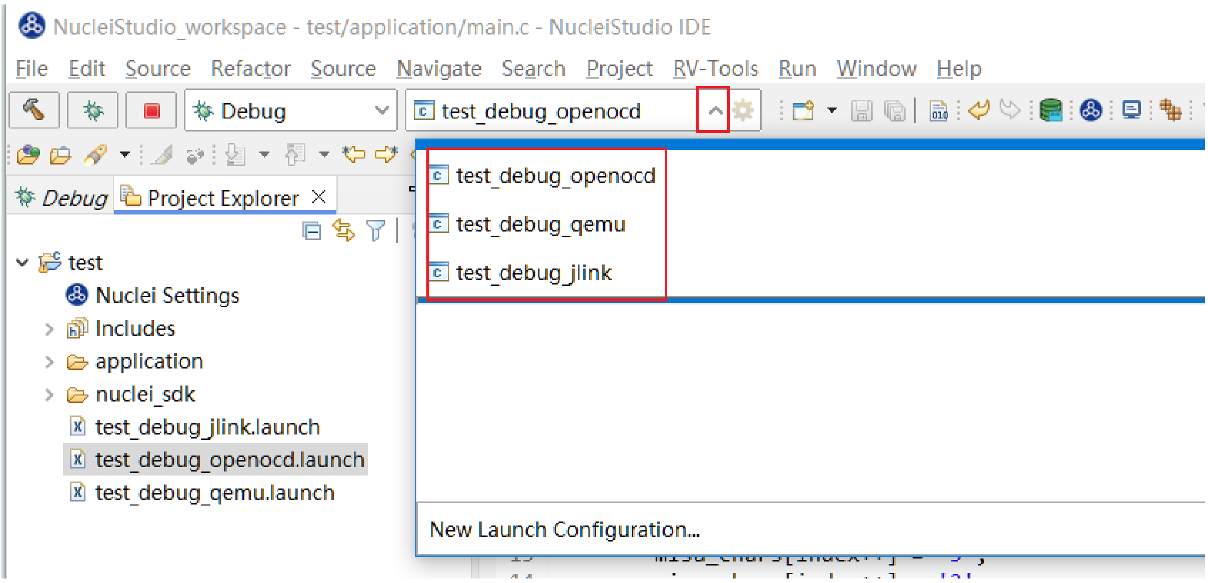
通过点击工程中的 `*.launch` 文件切换不同的调试模式, 当用户单击工程中的某一个 `*.launch` 文件时，NucleiStudio 会将该文件设置为 Launch Bar 中的选中文件，然后就可以通过 Launch Bar 执行 Run/Debug 操作。



在工程展开文件，找到 `*.launch` 文件并在 `*.launch` 文件上点击鼠标右键，在弹出菜单中选中 `Debug As/Run As`，在下一级菜单中点对应的模式开始 `Run/Debug` 操作。



用户可以通过 Launch Bar 中的下接框，来切换成不同的调试模式，在 Launch Bar 中点击展开按钮，然后选中对应的调试模式，并执行 Run/Debug 操作。



2.9.2 使用蜂鸟调试器结合 OpenOCD 调试运行项目

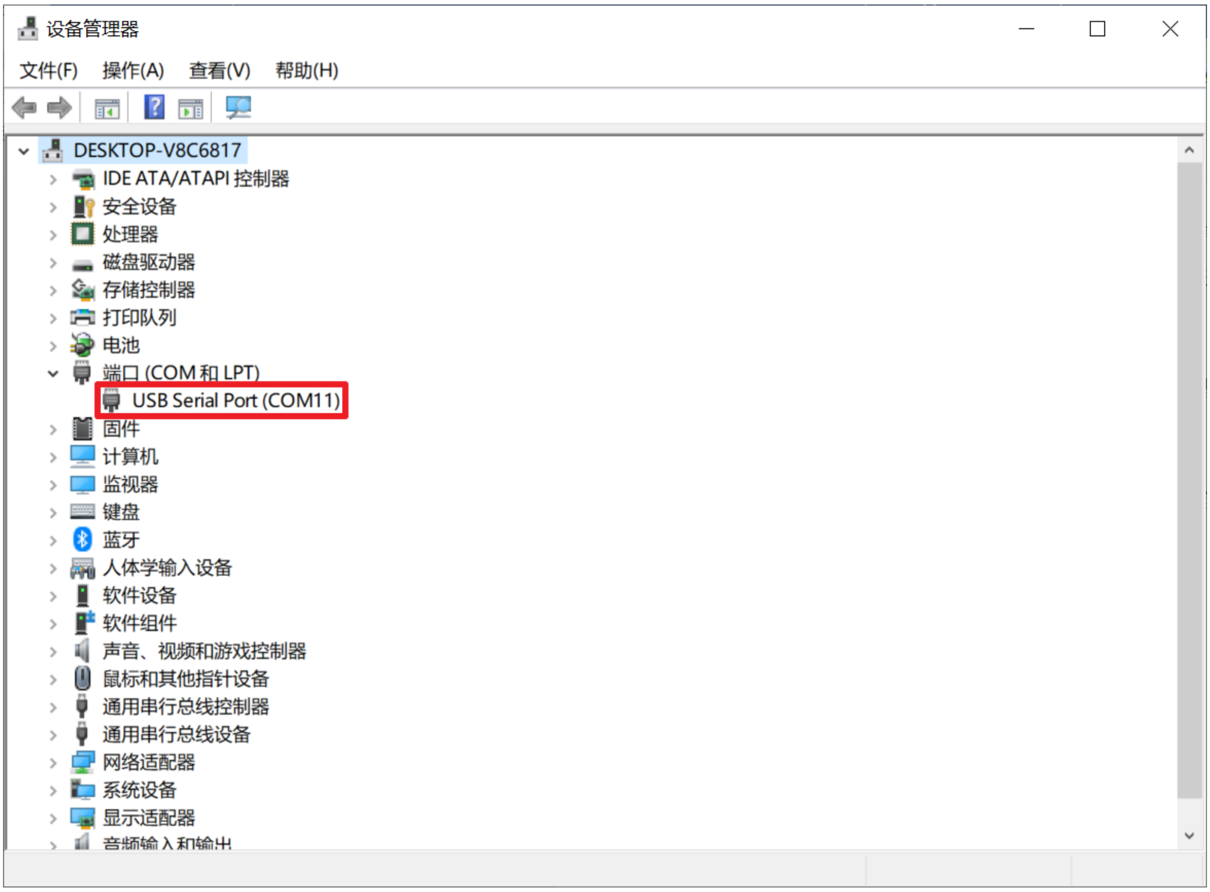
安装蜂鸟调试器驱动

> **Note**
>
> Nuclei Studio 自 2021.02 版本起 openocd 实现 windows 免驱功能，使用此版本及以上 windows 环境的用户可跳过此节，Linux 环境仍需配置驱动。低于此版本的用户需按照以下方法安装驱动。

在 Windows 系统中安装驱动

程序编译成功后，便可以将程序下载到 FPGA 原型开发板运行。首先将原型开发板与主机 PC 进行连接，步骤如下。

将蜂鸟调试器的一端插入主机 PC 的 USB 接口，另一端与原型开发板连接。由于蜂鸟调试器还包含了将原型开发板输出的 `UART` 转换成 `USB` 的功能，因此如果蜂鸟调试器被主机 PC 识别成功（且驱动安装成功），那么将能够被主机识别成为一个 COM 串口。在主机 PC 的设备管理器中的端口（`COM` 和 `LPT`）栏目中可以查询到该 COM 的串口号（譬如 COM11）。此串口在后续的程序运行过程中将充当原型开发板运行程序的 printf 输出显示接口。



> **Note**
>
> 注意：如果使用蜂鸟调试器，主机 PC 的 Windows 系统不能够识别蜂鸟调试器的 USB，需要安装驱动，可以在 https://www.nucleisys.com/developboard.php#debuggerkit 下载驱动程序并安装。

Note

注意：在通过蜂鸟调试器下载之前，需要注意蜂鸟调试器被 Windows 正确识别，检验的标准即为本节中所述正确地安装了 HBird-Driver.exe 的驱动，且能够在设备管理器中查询到 COM 的串口号。

在 Linux 系统中安装驱动

在 Linux 环境下安装驱动步骤如下：

- 1：连接开发板到 Linux 中，确保 USB 被 Linux 识别出来。如果使用虚拟机，确保开发板连到了虚拟机当中。

- 2：在控制台中使用 lsusb 指令查看信息，参考的打印信息如下：

```
Bus 001 Device 010: ID 0403:6010 Future Technology Devices International, Ltd␣
↪FT2232xxxx
```

- 3：控制台中输入 sudo vi /etc/udev/rules.d/99-openocd.rules 指令打开 99-openocd.rules 文件，输入如下内容，保存退出。

```
SUBSYSTEM=="usb", ATTR{idVendor}=="0403",

ATTR{idProduct}=="6010", MODE="664", GROUP="plugdev"

SUBSYSTEM=="tty", ATTRS{idVendor}=="0403",

ATTRS{idProduct}=="6010", MODE="664", GROUP="plugdev"
```

- 4：断开调试器再重新连接到 Linux 系统中。
- 5：使用 `ls /dev/ttyUSB*` 命令查看 `ttyUSB` 信息，参考输出如下：

```
/dev/ttyUSB0

/dev/ttyUSB1
```

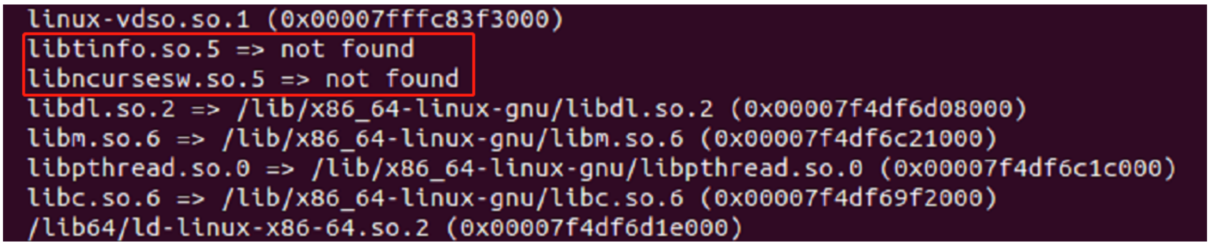
- 6：使用 ls -l /dev/ttyUSB1 命令查看分组信息，参考输出如下:

```
crw-rw-r-- 1 root plugdev 188, 1 Nov 28 12:53 /dev/ttyUSB1
```

可以看到 ttyUSB1 已经加入 `plugdev` 组，接下来我们要将自己添加到 plugdev 组（不同环境可能名字不同，请根据实际情况修改）。使用 whoami 命令查看当前用户名，我们将其记录为 `< your_user_name >`。

- 7：使用 `sudo usermod -a -G plugdev <your_user_name>` 命令将自己添加进 plugdev 组。加入以后一定要重启或者注销操作系统。
- 8：再次确认当前用户名已属于 `plugdev` 组，使用 `groups` 命令，可以看到打印信息中有 `plugdev` 即成功将当前用户添加至 plugdev 组。如果没有可以尝试重启。
- 9：查看 gcc 的依赖是否完整，如果有依赖需要安装，可以执行 `sudo apt install libncurses-esw5libtinfo5` 进行安装

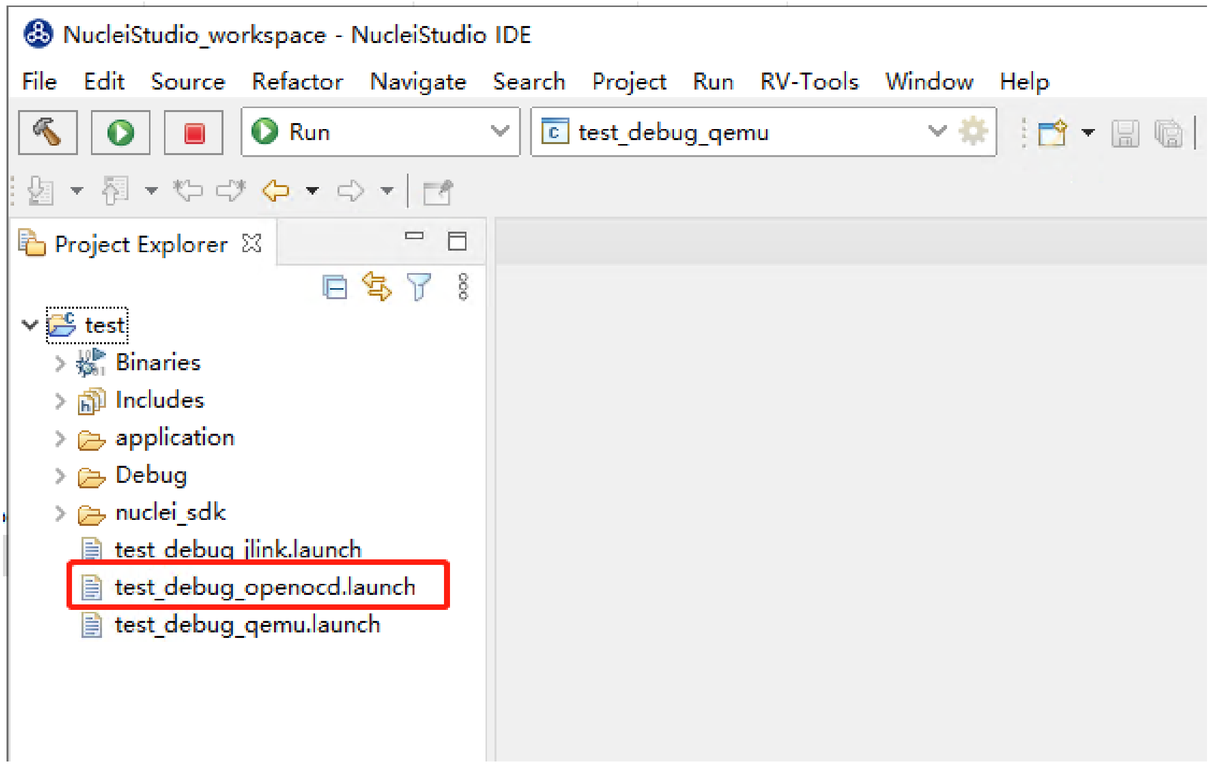
```
cd Nuclei Studio/toolchain/gcc/bin/

ldd ./riscv-nuclei-elf-gdb
```



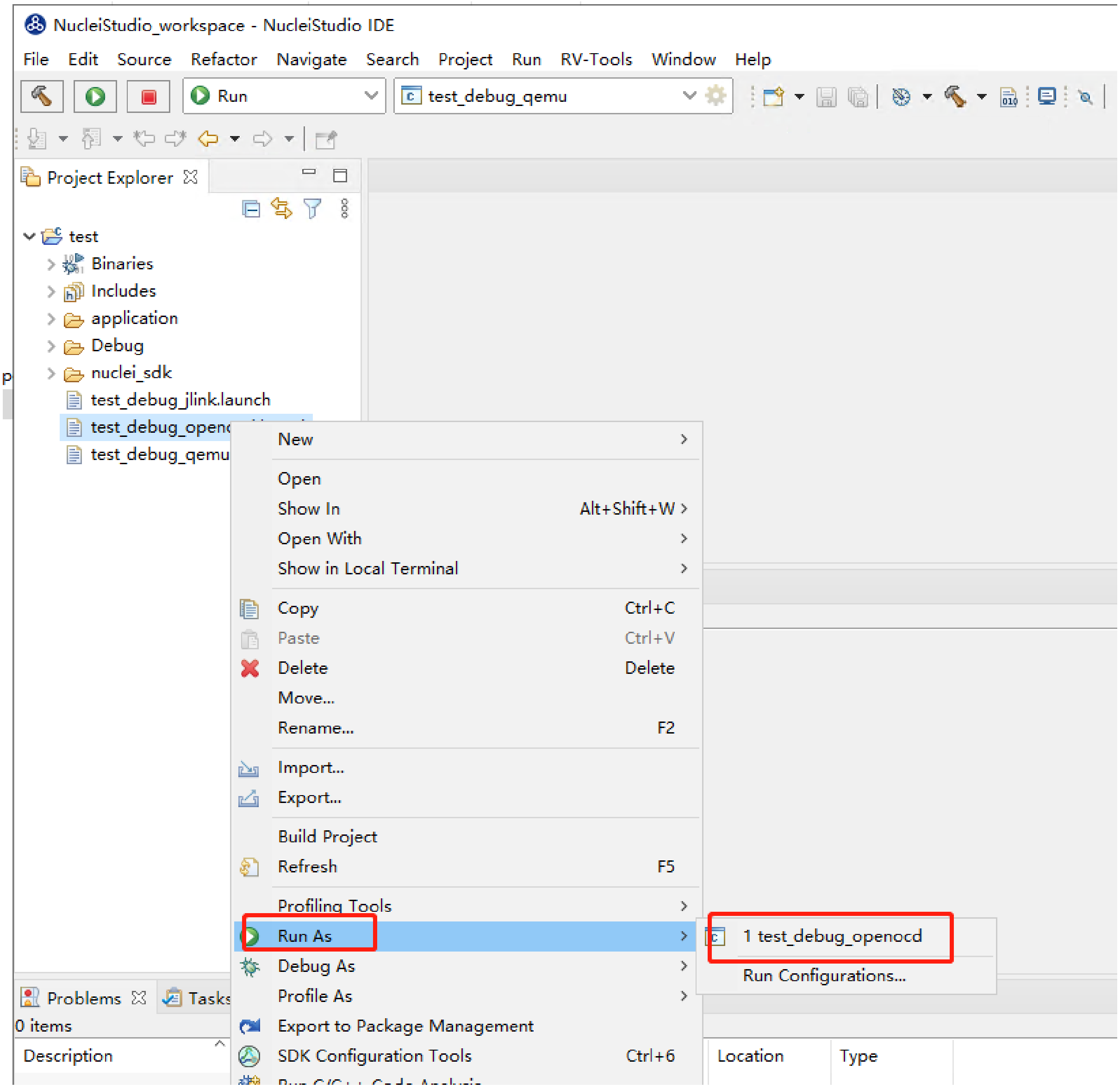
Debug Configuration

使用 Nuclei Studio 生成的 Debug Configuration

为了方便用户调试，Nuclei Studio 在创建工程时，会根据 NPK 的配置，默认的生成 Debug Configurations 的 Launch 文件。

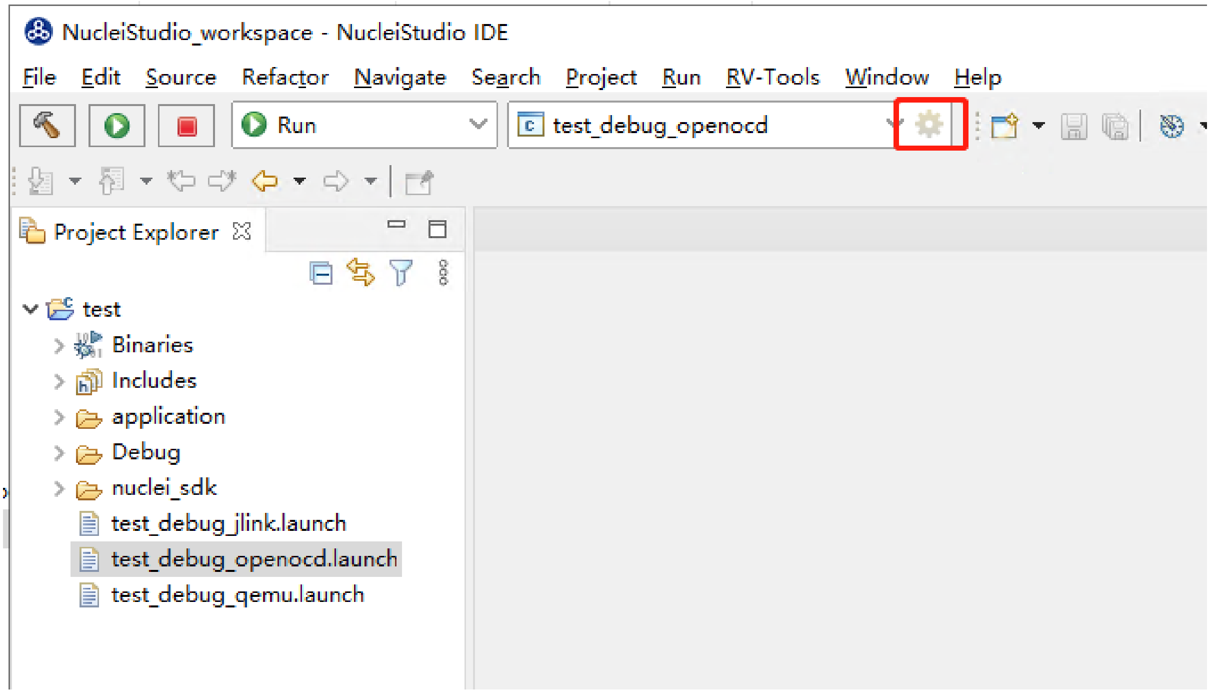


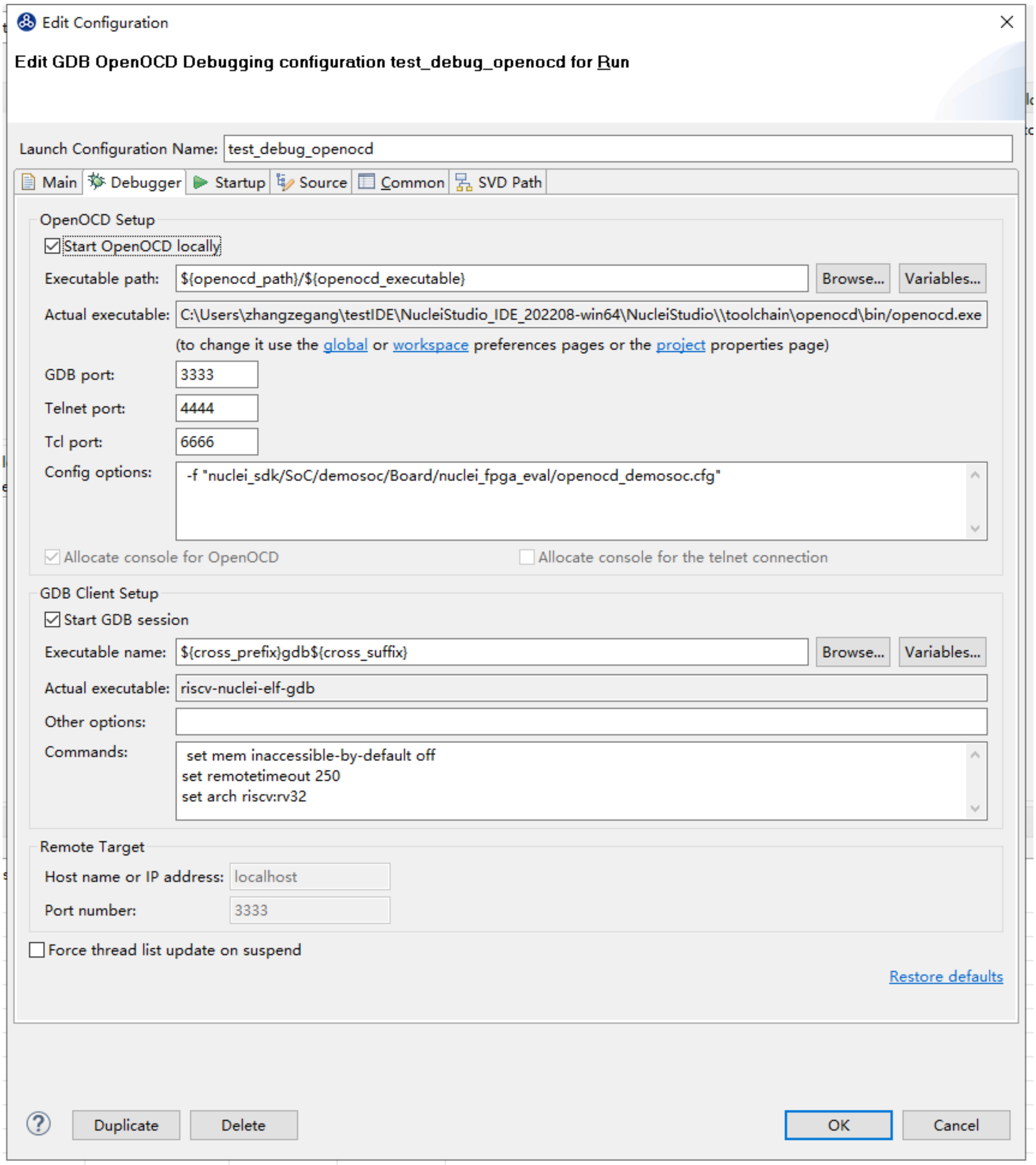
用户可以展开工程，选中对应的 `test_debug_openocd.launch` 文件，在右键菜单中，可以 `Run as/Debug as->test_debug_openocd`，就可以按照对应的 Debug Configurations 操作工程程了。



> **Note**
>
> 注意：配图可能没有及时更新，导致图文不一致，以文字为准，结合对应版本进行使用。

具体的 Debug Configurations 的内容可以在 Launch Bar 中进行详情查看。

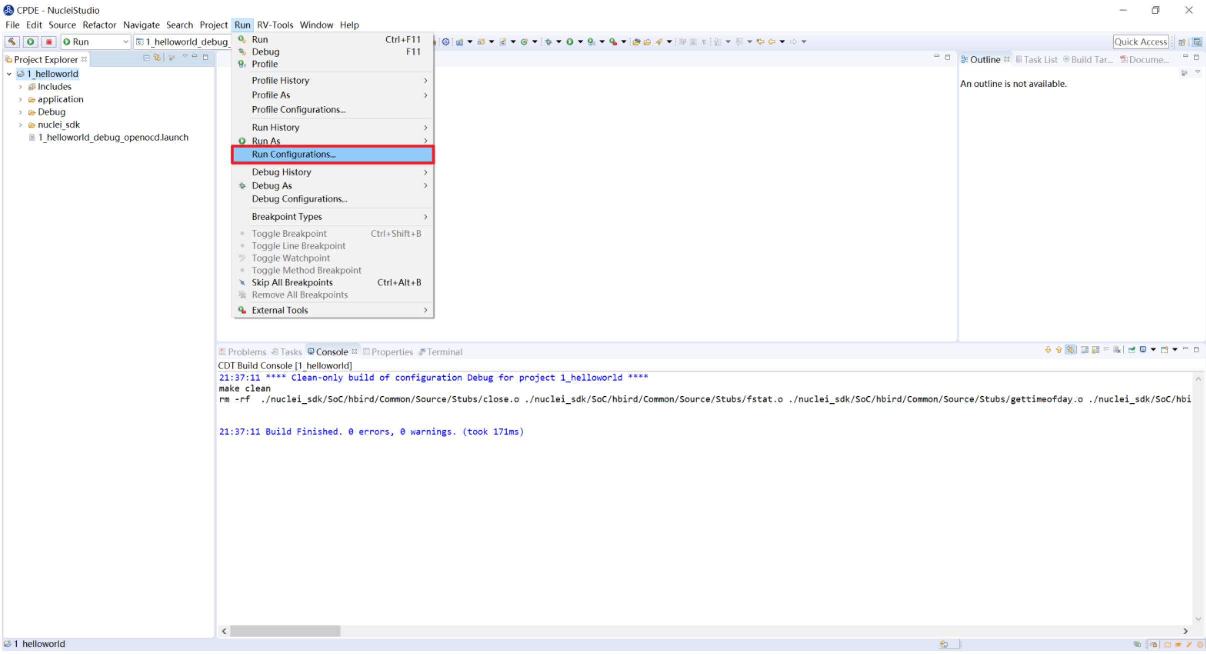

NucleiStudio_workspace - NucleiStudio IDE
File Edit Source Refactor Navigate Search Project Run RV-Tools Window Help
Run
test_debug_openocd
Project Explorer
test
Binaries
Includes
application
Debug
nuclei_sdk
test_debug_jlink.launch
test_debug_openocd.launch
test_debug_qemu.launch



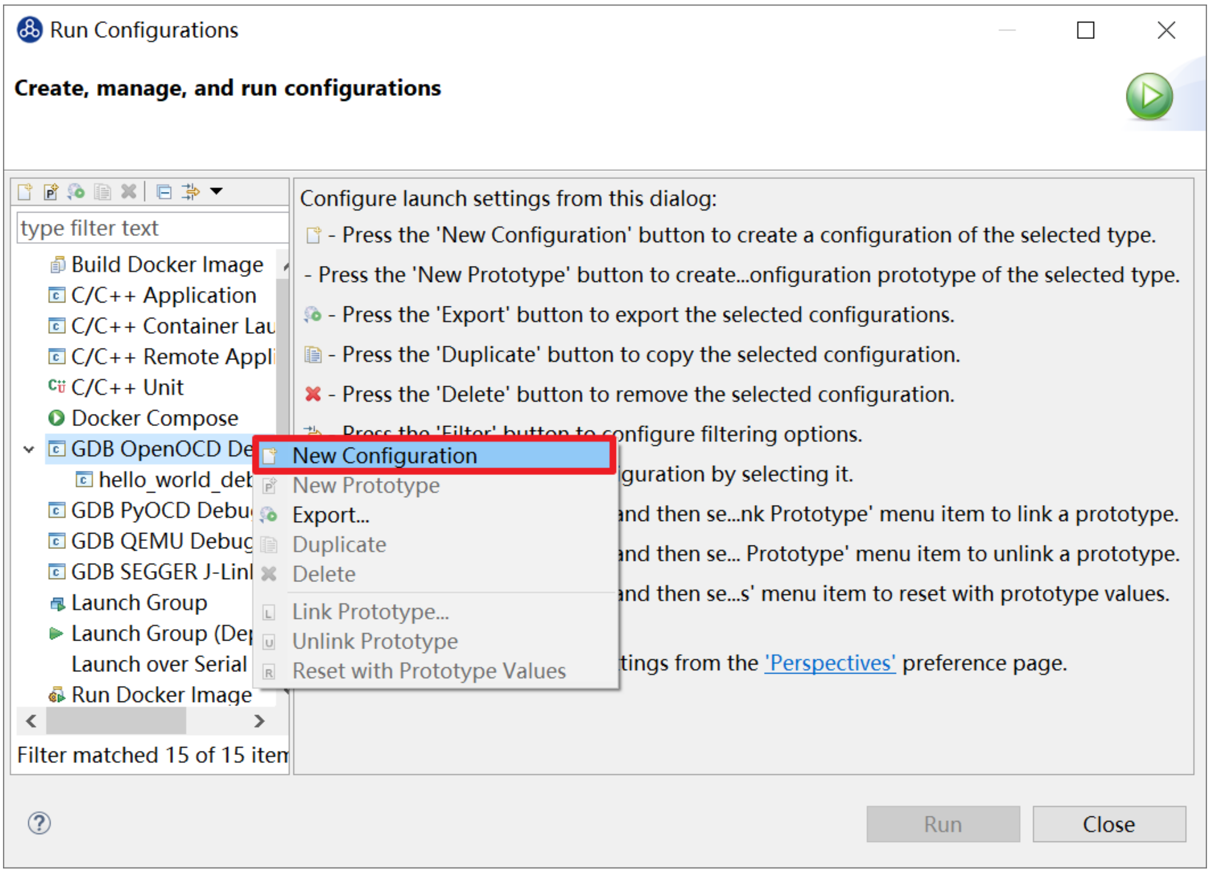
新建并配置 Debug Configuration

通过 Nuclei Studio 新建并配置 Debug Configuration 内容的步骤如下。

在 Nuclei Studio 的主菜单栏中选择 `Run—>Debug Configurations`。



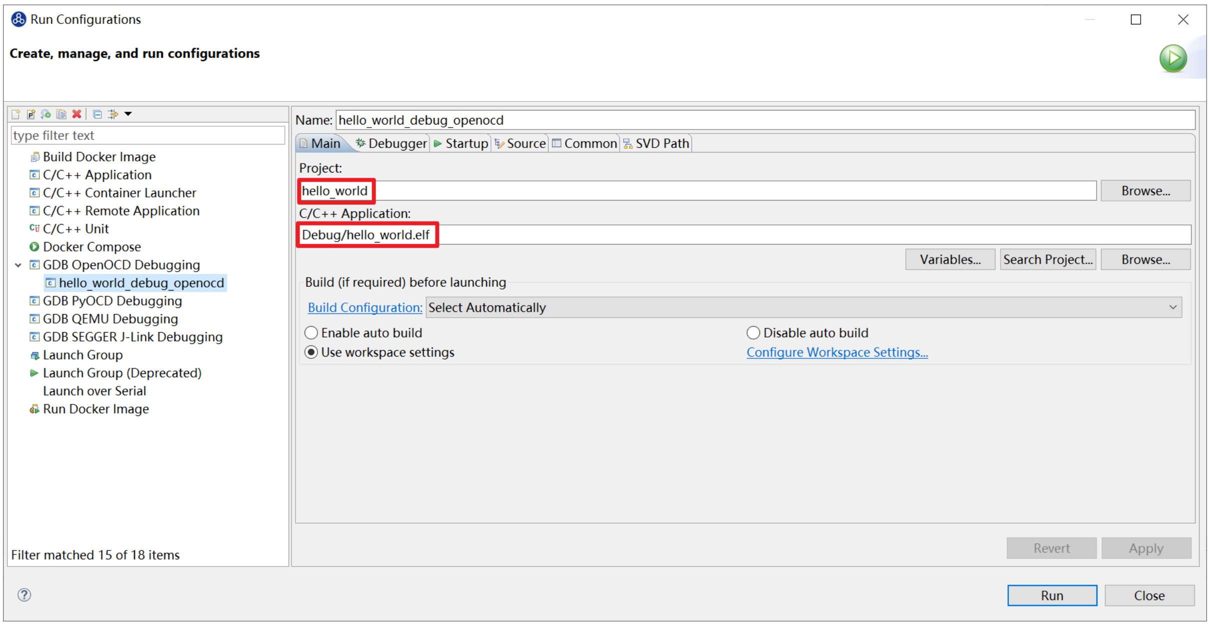
在弹出的窗口中，如果没有当前工程的调试设置内容，右键单击 `GDB OpenOCD Debugging`，选择 `New`，将会为本项目新建出一个调试项目 `hello_world_demo Debug`。确保 `Project` 是当前需要调试的工程，`C/C++ Application` 中选择了正确的需要调试的 ELF 文件。



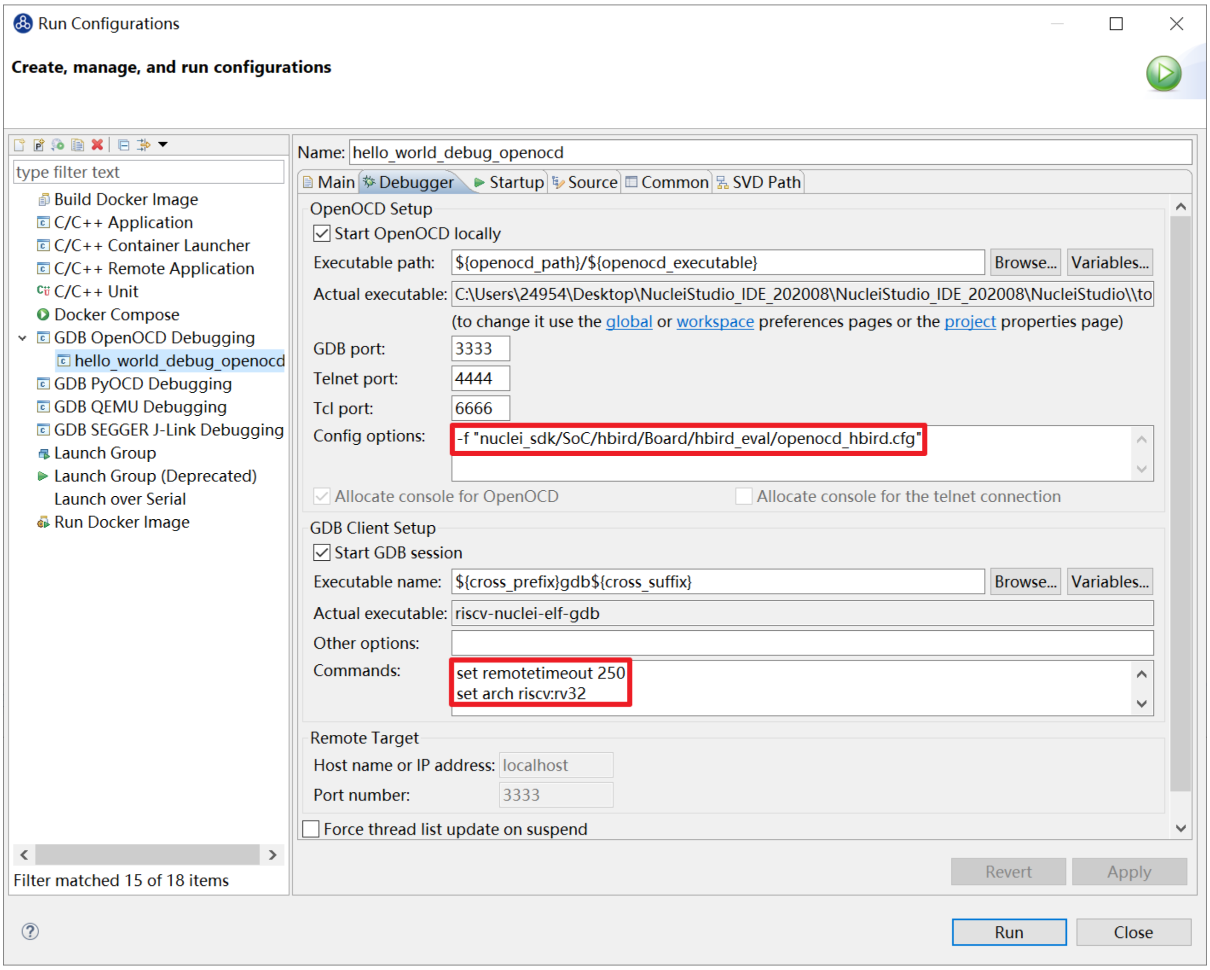
选择调试项目 `hello_world_demo Debug` 的 Debugger 菜单，在 Config options 栏目中填入 `-f "nuclei_sdk/SoC/evalsoc/Board/nuclei_fpga_eval/openocd_evalsoc.cfg"`，以确保 OpenOCD 使用正确的配置文件。这里的配置文件 (*nuclei_sdk/SoC/evalsoc/Board/nuclei_fpga_eval/openocd_evalsoc.cfg*) 根据实际工程中 openocd 的配置文件路径而定。例如：如果使用 makefile 方式导入工程，修改此处的内容为 `-f "SoC/evalsoc/Board/nuclei_fpga_eval/openocd_evalsoc.cfg"`。

如果当前内核是 RISC-V 32 位内核，请确保 Commands 内容包含 `set arch riscv:rv32`

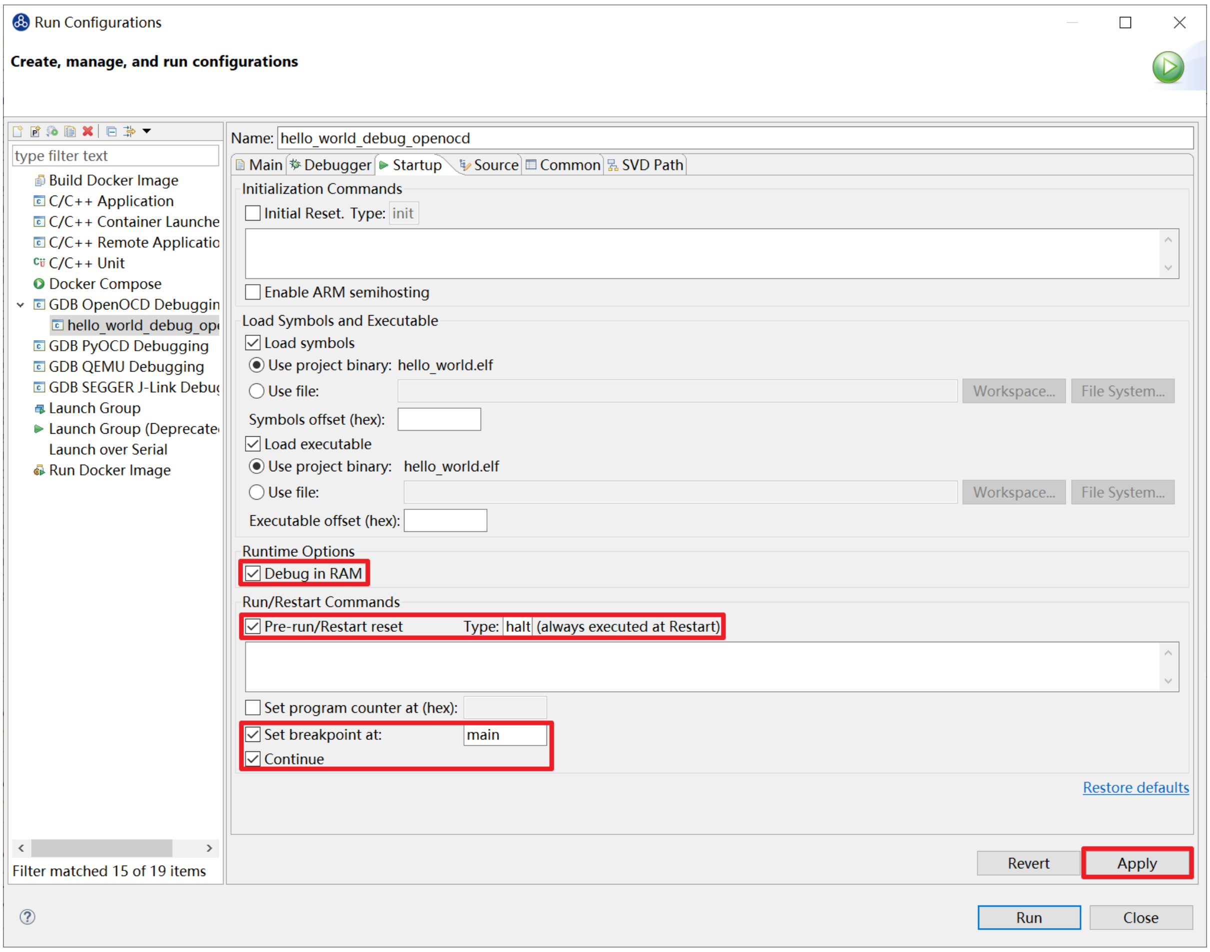
如果当前内核为 64 位，应确保替换为 `set arch riscv:rv64`



选择调试项目 `hello_world_demo Debug` 的 Startup 菜单，确保 `Debug in RAM`，`Pre-run/Restart reset`，`Set Breakpoint at Main` 和 `Continue` 被勾选。



完成配置后点击右下方 Apply 保存设置。



关于 Debug Configuration 中 Startup 各项设置的具体含义，详细说明如下。这里的设定内容最终将以 GDB 命令的方式实现，所以在执行 GDB 命令时也是按此顺序来执行。

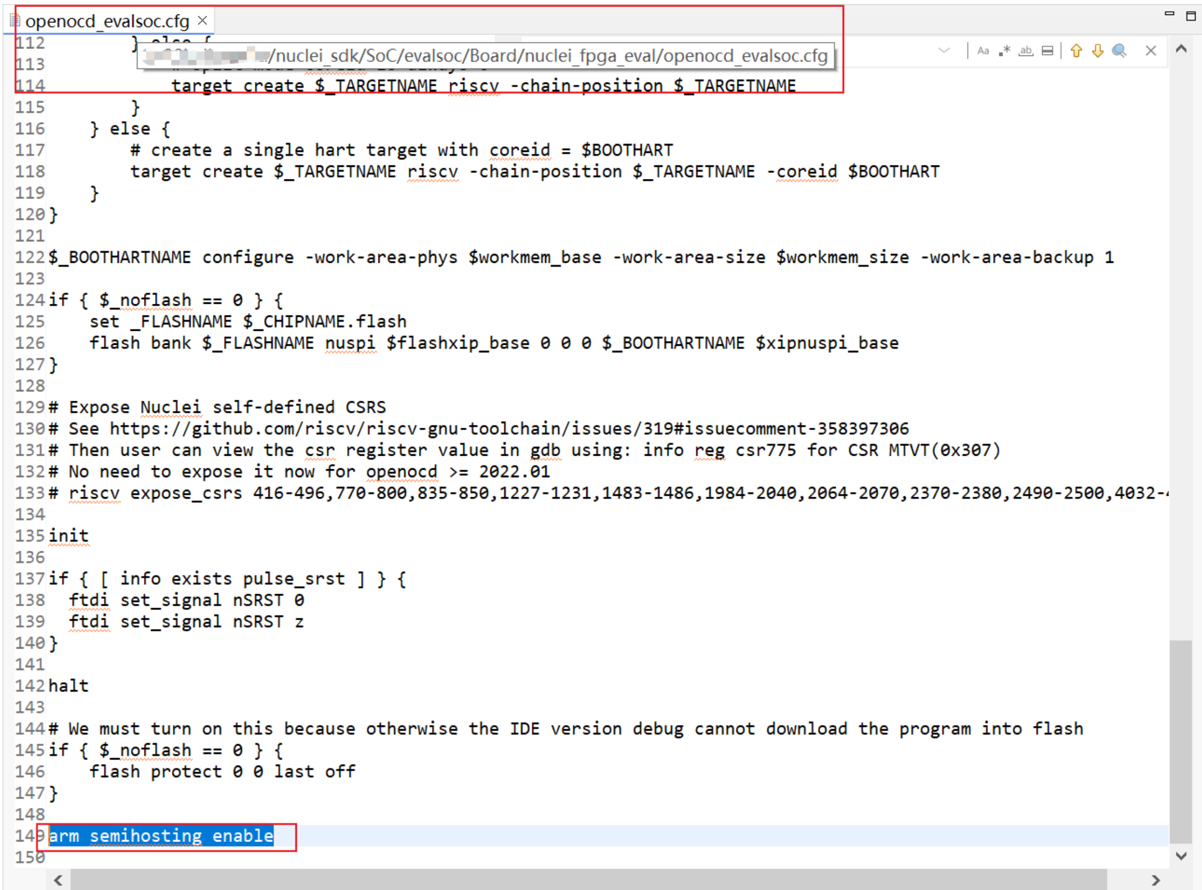
Eclipse embedded CDT 官方文档中对 Startup 中的参数也做了一些基本介绍，具体参见：https://eclipse-embed-cdt.github.io/debug/openocd/riscv/

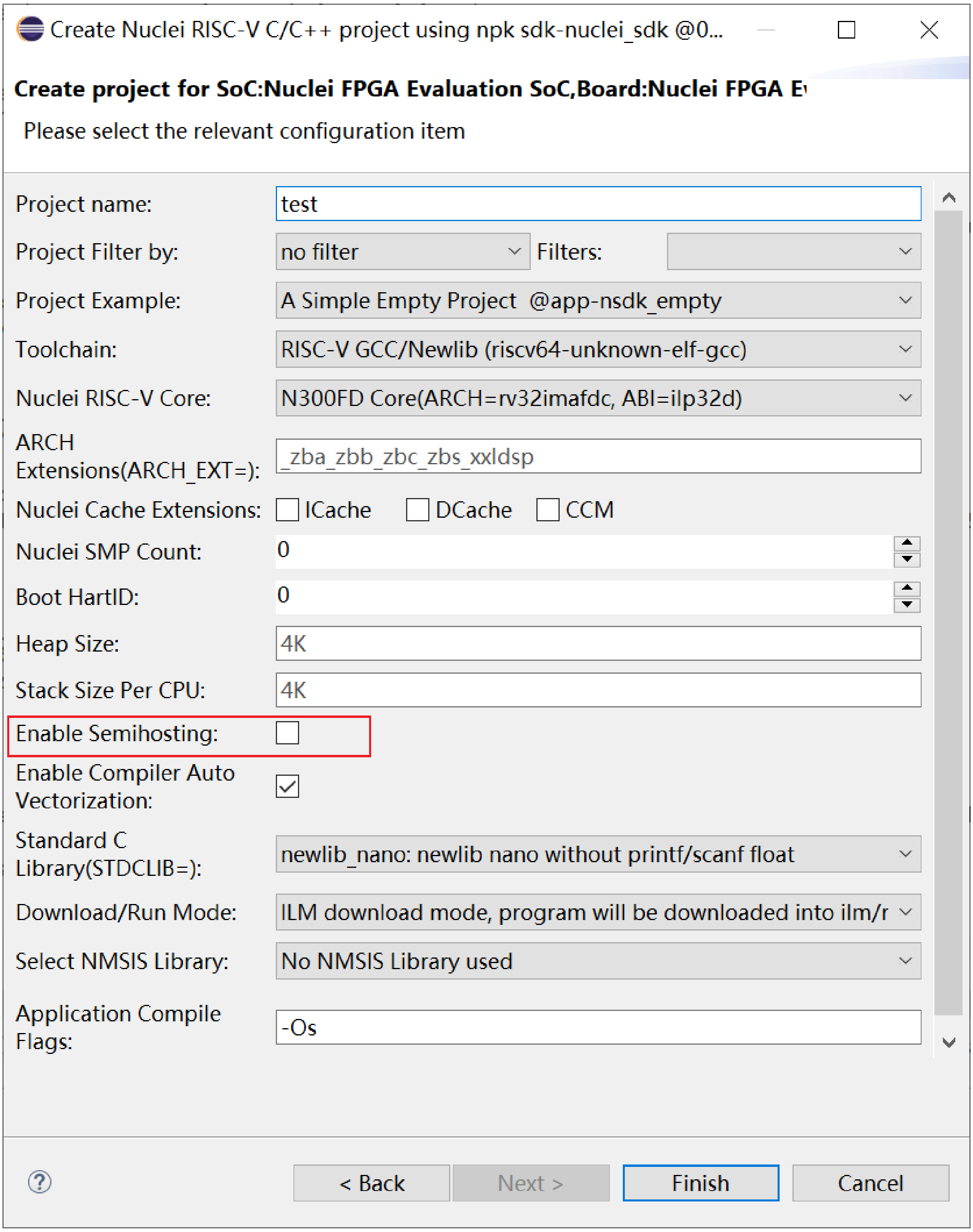
Initial Reset

可以设定一些让 GDB 做 init 的命令，具体可以参考 GDB init command 的一些用法，一般不需勾选。

Enable semithost

如果勾选此项，工程必须支持 semihosting 功能。在使用 RISC-V OpenOCD 时，需要在配置文件中添加 ``arm semihosting enable`` 来启用该功能，并且在创建 Nuclei SDK 工程时也需要开启 semihost 支持。J-Link 和 GDB J-Link 调试方式也可以实现 semihost 功能，具体请参考使用 *J-Link* 调试运行项目 (page 163)。默认情况下，无需勾选此选项。


openocd_evalsoc.cfg
/nuclei_sdk/SoC/evalsoc/Board/nuclei_fpga_eval/openocd_evalsoc.cfg
target create $_TARGETNAME riscv -chain-position $_TARGETNAME
}
} else {
# create a single hart target with coreid = $BOOTHART
target create $_TARGETNAME riscv -chain-position $_TARGETNAME -coreid $BOOTHART
}
}
$_BOOTHARTNAME configure -work-area-phys $workmem_base -work-area-size $workmem_size -work-area-backup 1
if { $_noflash == 0 } {
set _FLASHNAME $_CHIPNAME.flash
flash bank $_FLASHNAME nuspi $flashxip_base 0 0 0 $_BOOTHARTNAME $xipnuspi_base
}
# Expose Nuclei self-defined CSRS
# See https://github.com/riscv/riscv-gnu-toolchain/issues/319#issuecomment-358397306
# Then user can view the csr register value in gdb using: info reg csr775 for CSR MTVT(0x307)
# No need to expose it now for openocd >= 2022.01
# riscv expose_csrs 416-496,770-800,835-850,1227-1231,1483-1486,1984-2040,2064-2070,2370-2380,2490-2500,4032-
init
if { [info exists pulse_srst] } {
ftdi set_signal nSRST 0
ftdi set_signal nSRST z
}
halt
# We must turn on this because otherwise the IDE version debug cannot download the program into flash
if { $_noflash == 0 } {
flash protect 0 0 last off
}
arm semihosting enable



Load symbols

使用 GDB 的 `file` 命令，通过 GDB 读取 elf 的 debug information 信息，使 GDB 能方便且正确的进行 Debug 操作，默认需勾选。

Load executable

使用 GDB 的 `load` 命令，让 GDB 下载程序到 target 端（GDB 执行 load 命令，下载 elf 内容到 target 端，并将 target 端 CPU 的 PC 改成当前 elf 的 entry 位置），如果调试 RAM（比如 LM）中的程序，默认勾选；如果调试 Flash 中的程序，需确认 openocd 是否支持 Flash 的烧写以及当前调试是否要重新做 Flash 烧写，如果支持且需要重新烧写，则此项需勾选，否则不勾选；如果是 ROM 的代码，则不需要勾选。

Debug in Ram

如果 GDB 在 `load excutable` 后有 reset 等动作，避免 CPU 的启动地址和 elf 文件的启动地址不一样；或者 RAM 中的程序因为 reset 后被清掉。故每次在 GDB reset 后需重新 load elf 文件。如果是调试 RAM 里的程序，必需勾选此项。

Pre-run/Reset

使用 GDB 的 `monitor reset` 命令，它给 openocd 发 `reset` 命令，openocd 会按 RISC-V Debug Spec 去驱动 `nReset` 信号，这个信号会让 `core` 和 `peripherals` 都 `reset` （这里也要看具体实现，不过 RISC-V Dedbug Spec 推荐如此，且如果是 Nuclei 做的 example SoC 或者 FPGA ，都是 follow 这个 Spec ），执行 `reset` 后，CPU 的 PC 就是 `reset_vector` 地址，因为外设都 reset，所以 RAM 的内容也都会被清掉。如果是在 RAM 里 debug，且勾选了此项，那么必须要勾选 `Debug in RAM` 项。

下图是 RISC-V Debug Spec 中关于 `nRESET` 的说明。

Table 6.7: JTAG Connector Pinout

1	VREF DEBUG	Reference voltage for logic high.
2	TMS	JTAG TMS signal, driven by the debug adapter.
4	TCK	JTAG TCK signal, driven by the debug adapter.
6	TDO	JTAG TDO signal, driven by the target.
7	GND or KEY	This pin may be cut on the male and plugged on the female header to ensure the header is always plugged in correctly. It is, however, recommended to use this pin as an additional ground, to allow for fastest TCK speeds. A shrouded connector should be used to prevent the cable from being plugged in incorrectly.
8	TDI	JTAG TDI signal, driven by the debug adapter.
10	nRESET	Active-low reset signal, driven by the debug adapter. Asserting reset should reset any RISC-V cores as well as any other peripherals on the PCB. It should not reset the debug logic. This pin is optional but strongly encouraged. If necessary, this pin could be used as nTRST instead. nRESET should never be connected to the TAP reset, otherwise the debugger might not be able to debug through a reset to discover the cause of a crash or to maintain execution control after the reset.
12	RTCK	Return test clock, driven by the target. A target may relay the TCK signal here once it has processed it, allowing a debugger to adjust its TCK frequency in response.
14	nTRST_PD	Test reset pull-down (optional), driven by the debug adapter. Same function as nTRST, but with pull-down resistor on target.
16	nTRST	Test reset (optional), driven by the debug adapter. Used to reset the JTAG TAP Controller.
18	TRIGIN	Not used, driven low by the debug adapter.
20	TRIGOUT	Not used, driven by the target.

Halt

使用 GDB 的 `monitor halt` 命令，它实现就是通过 OpenOCD 发 `reset` 命令后发 `halt` 命令，让 CPU reset 完后能马上 halt 住。

Set Program counter at

使用 GDB 的 `set $pc` 命令，可以再次修改 target 端 CPU 的 PC 地址。

Set breakpoint at

使用 GDB 的 `break` 命令，默认是 main 方法，如果是初次调试或者调试启动代码，建议修改为其他，如 _start, 或者某一个绝对的地址。

Continue

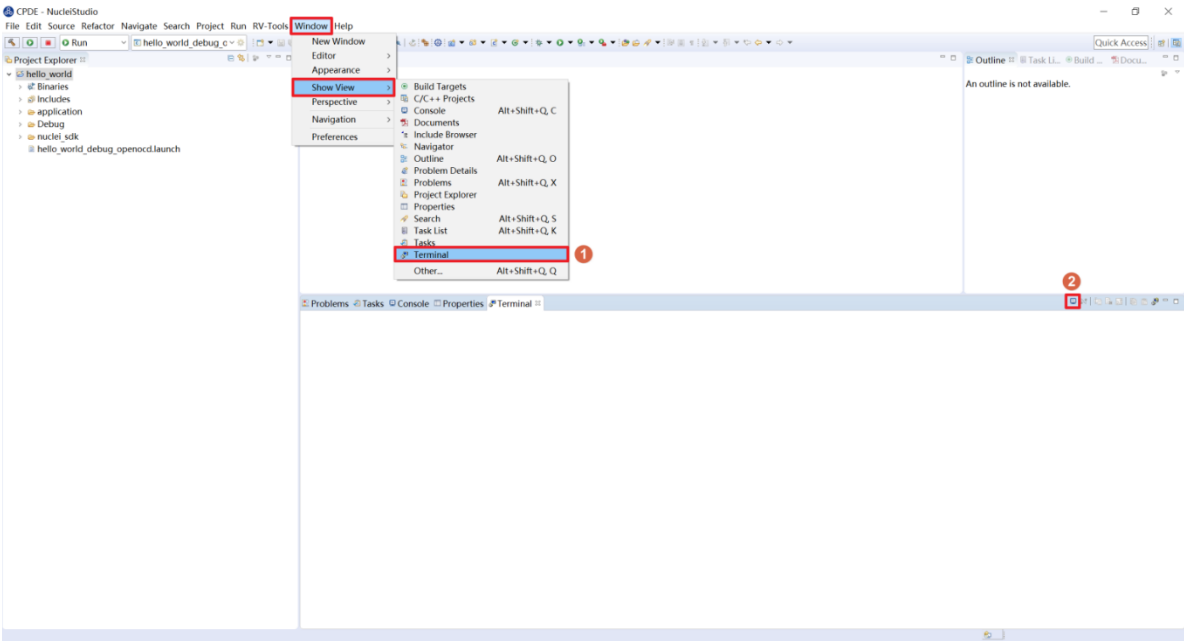
使用 GDB 的 `continue` 命令。

在原型开发板上调试程序

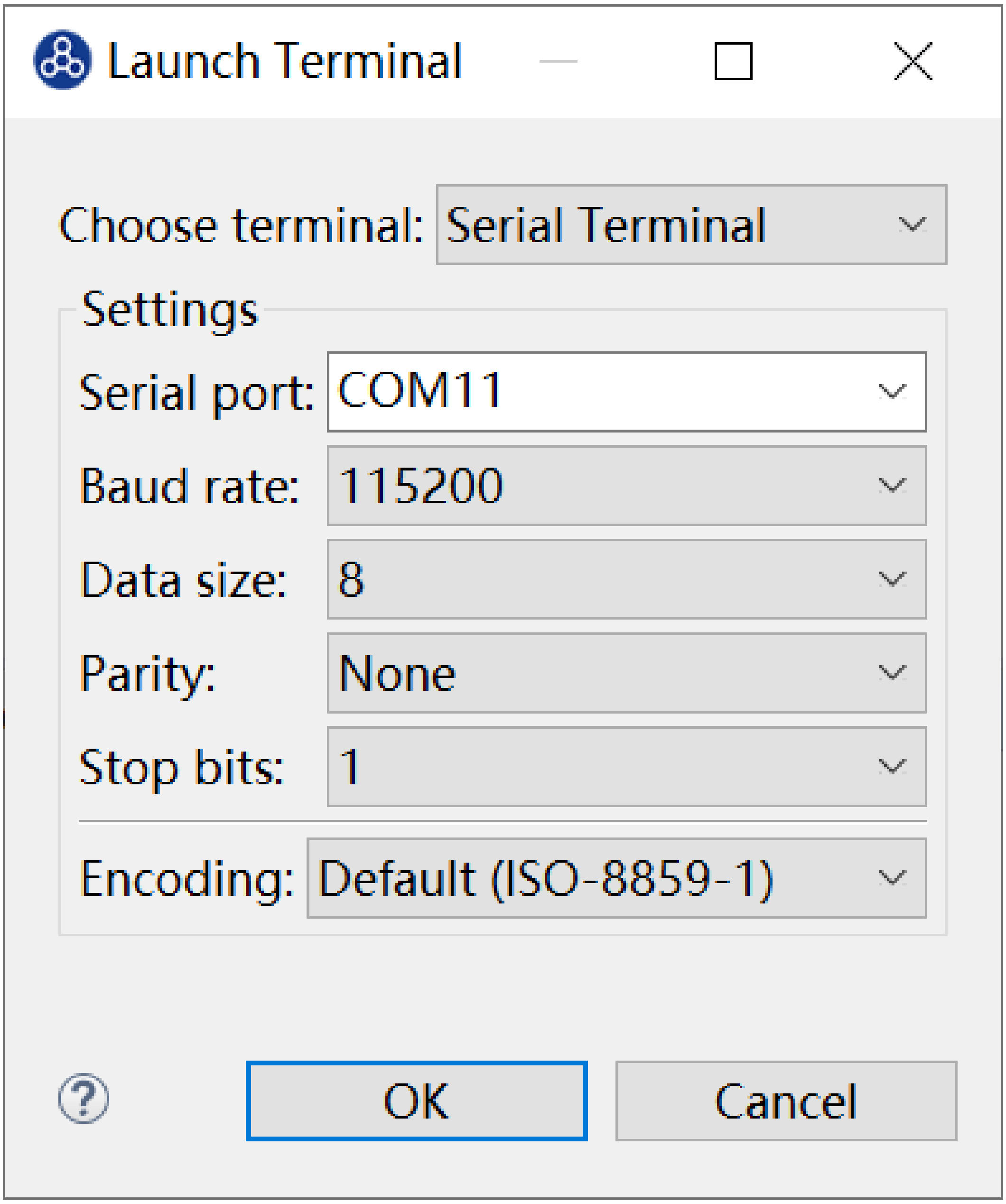
在开发板上调试之前，需要打开串口以便观察 Printf 函数打印信息。

使用 Windows 系统打开串口的方法如下：

打开 Nuclei Studio 自带的串口打印通道，选择 `Window>Show View>Terminal`，如图 7-13 所示，点击显示器图标打开串口设置选项。

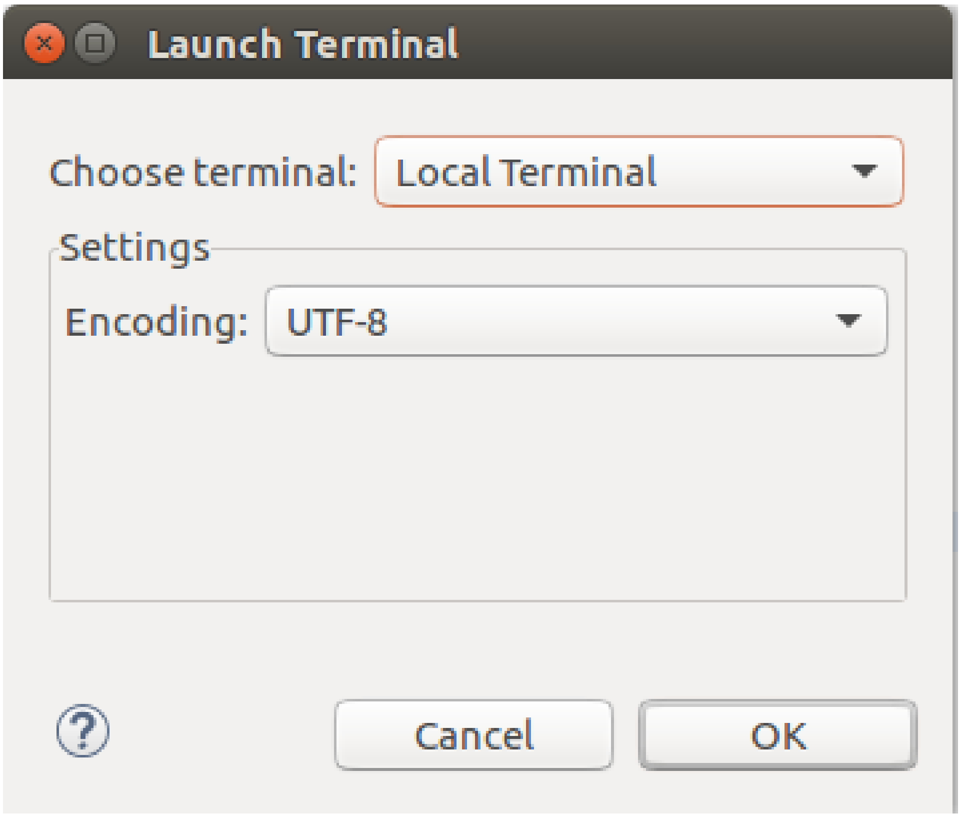


在其窗口中设置 Choose terminal（选择串口，即 Serial Terminal）、串口号（这里以 COM11 为例）、波特率（设置为 `115200`）等参数后，单击 `OK` 按钮。

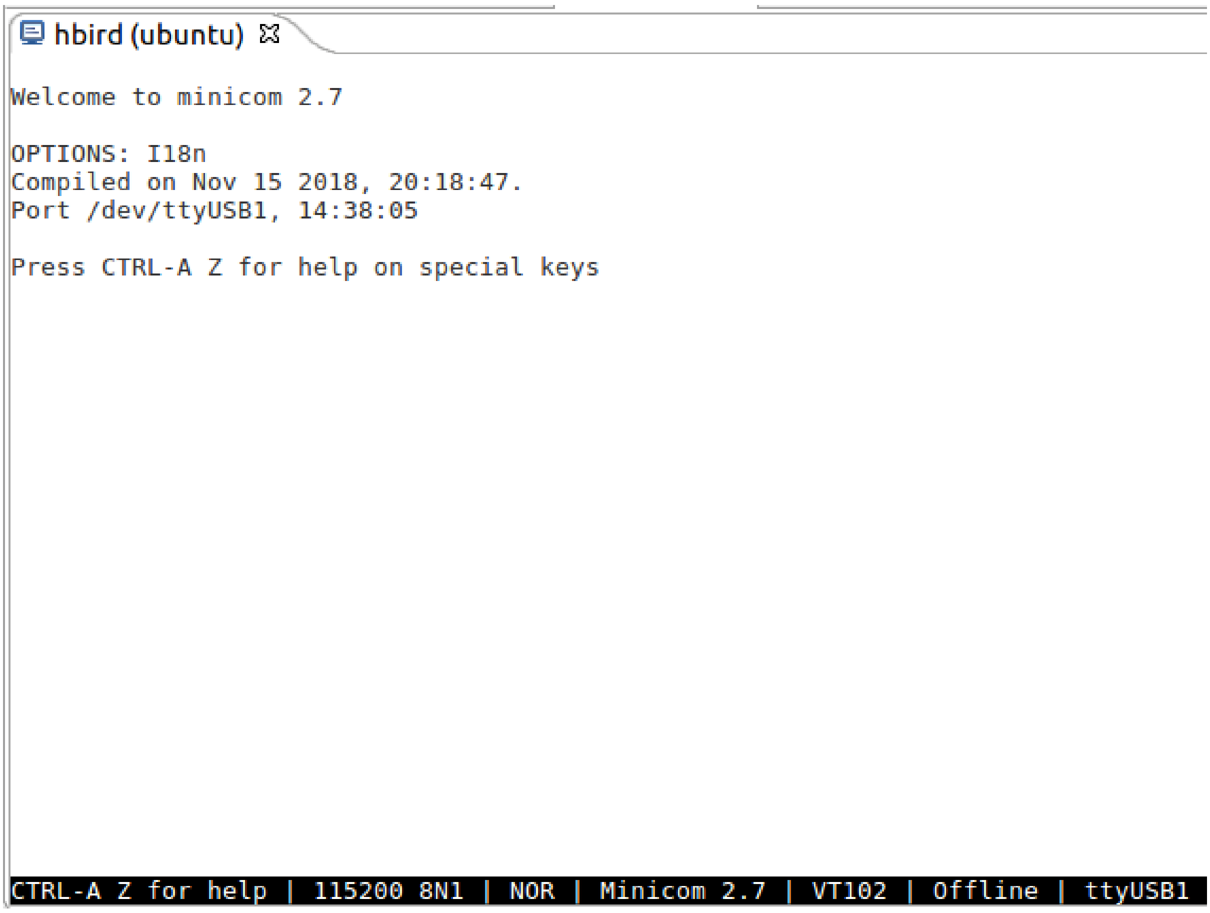


使用 Linux 系统打开串口方法如下：

打开 Nuclei Studio 自带的 Terminal 终端，选择 `Window>Show View>Terminal`，点击显示器图标打开串口设置选项。choose terminal 选择 `Local Terminal`，点击 OK 打开 Terminal 终端。



在框口中输入 `minicom /dev/ttyUSB1 115200` 打开串口，即可在 Nuclei Studio 中查看串口打印信息。



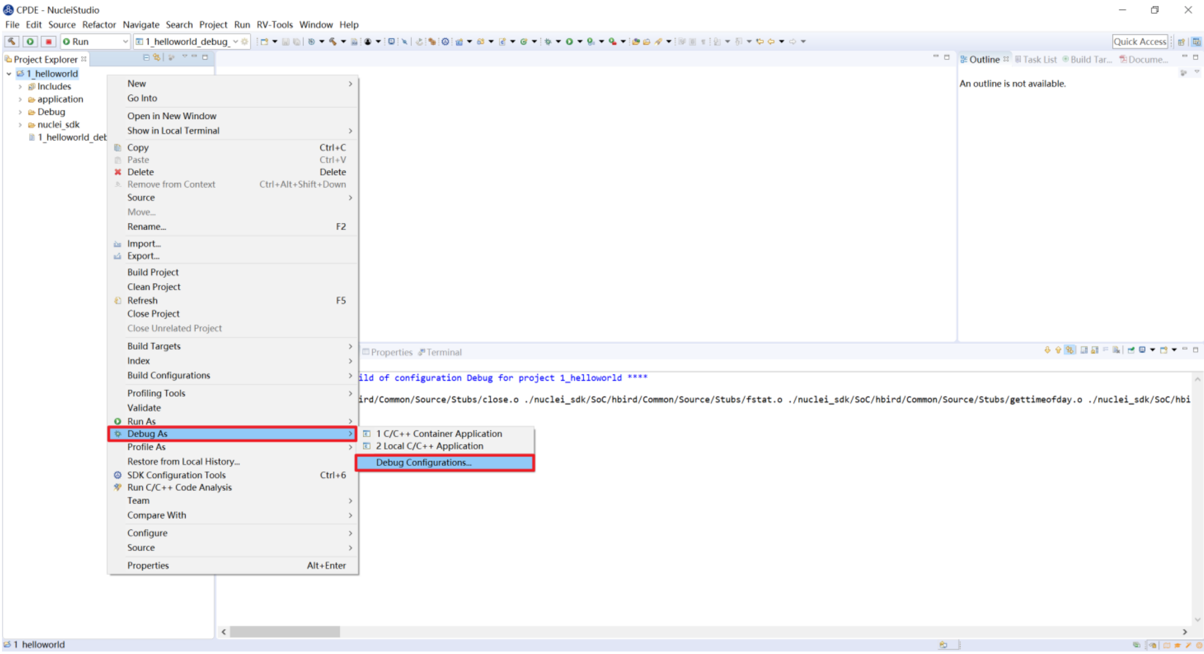
如果程序员希望能够调试运行于原型开发板中程序，可以使用 Nuclei Studio IDE 进行调试。由于 IDE

运行于主机 PC 端，而程序运行于原型开发板上，因此这种调试也称为 在线调试或者 远程调试。

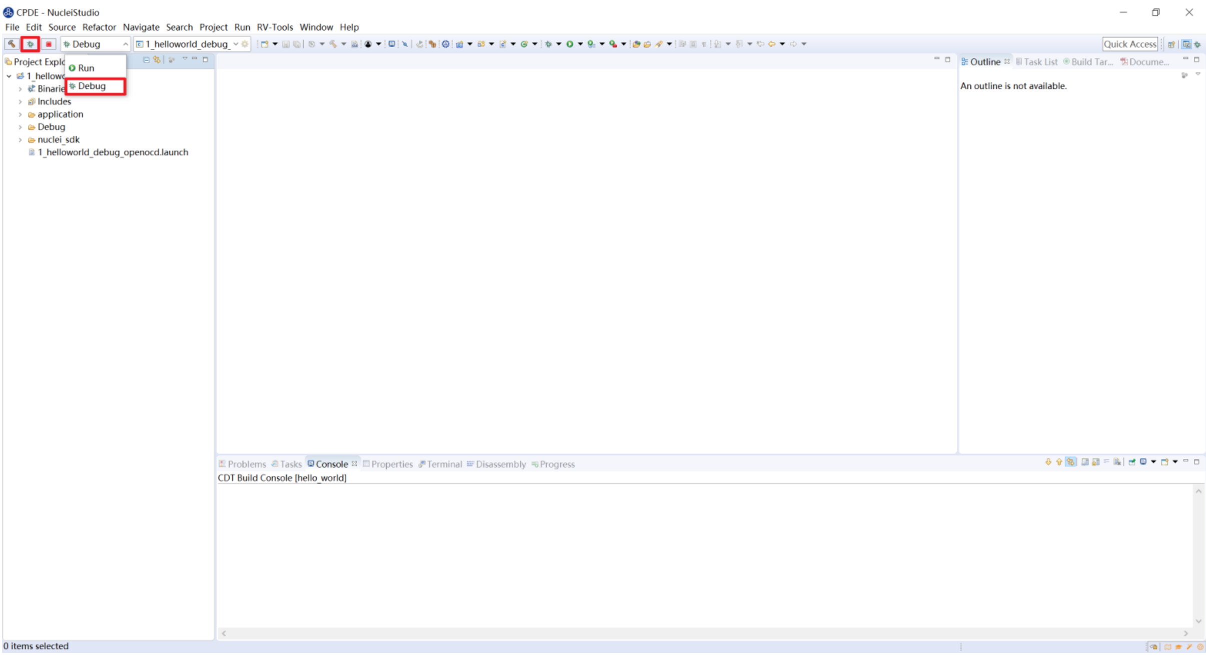
这里以 1_helloworld 为例，使用 Nuclei Studio IDE 对 evalsoc 原型开发板进行在线调试的步骤如下：

> **Note**
>
> 注意 demosoc 在 Nuclei SDK 5.0 中被移除，请使用 evalsoc 作为替代。

确保 Debug 设置内容正确，可以打开 Debug 设置选项确认。在 1_helloworld 工程处右击，选择 `Debug As >Debug Configuration` 打开 Debug 设置页面选择之前新建的设置进行检查。



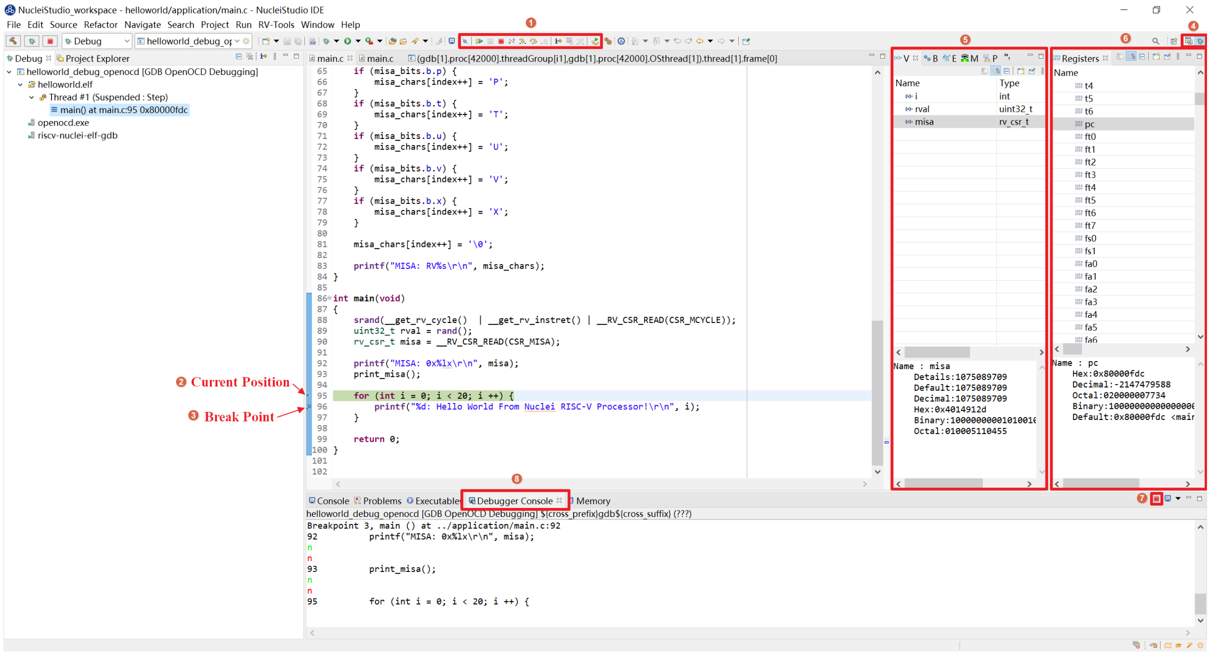
确定设置无误后，在下拉框选中 Debug，之后左侧图标会变为甲虫图标，单击即可进入调试模式并下载程序进入开发板中。



切换至 Debug 模式，如果下载成功，则会启动调试界面。

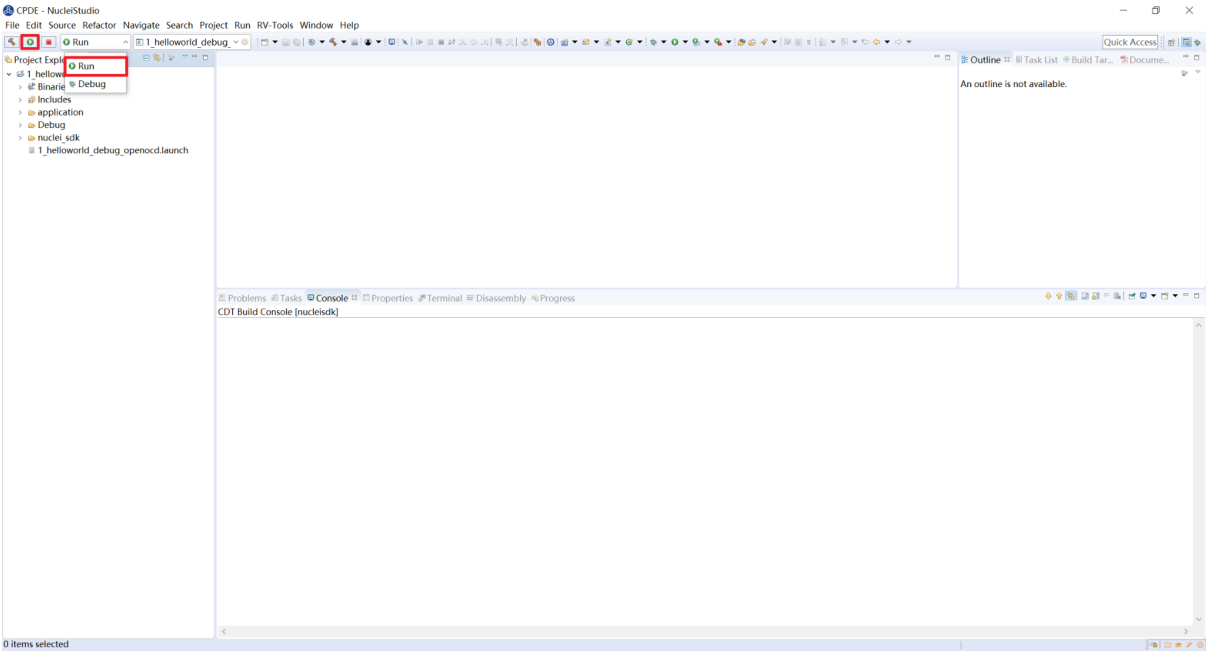
- 如图 1 号标注位置，这里功能包括单步，运行，汇编级调试等。

- 如图 2 号标注位置，这个箭头表示当前程序运行位置。
- 如图 3 号标注位置，在代码的左侧双击即可在该行设置断点，再次双击可以取消断点。
- 如图 4 号标注位置，这里可以切换编辑模式和调试模式。
- 如图 5 号标注位置，这里是函数内变量显示的位置。
- 如图 6 号标注位置，这里是查看寄存器数值的位置。图中显示的是 PC 寄存器当前的数值。
- 如图 7 号标注位置，点击这里红色按钮可以退出调试模式。
- 如图 8 号标注位置的下方，这里可以使用 GDB 控制台指令进行调试。

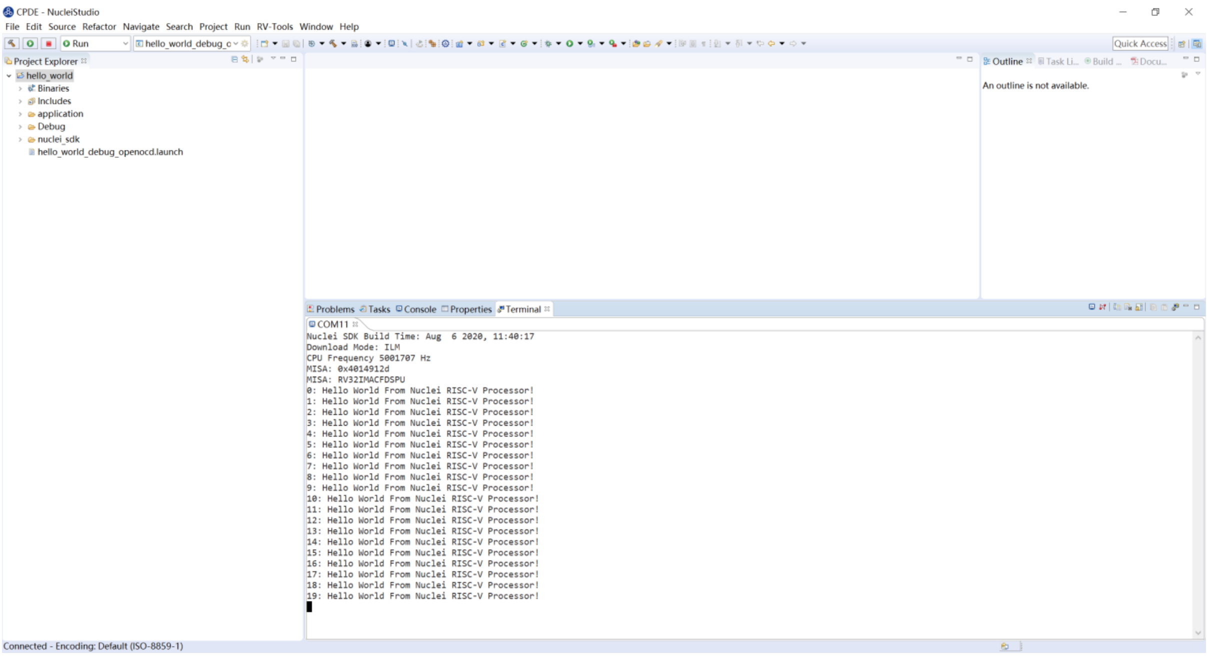


下载运行程序

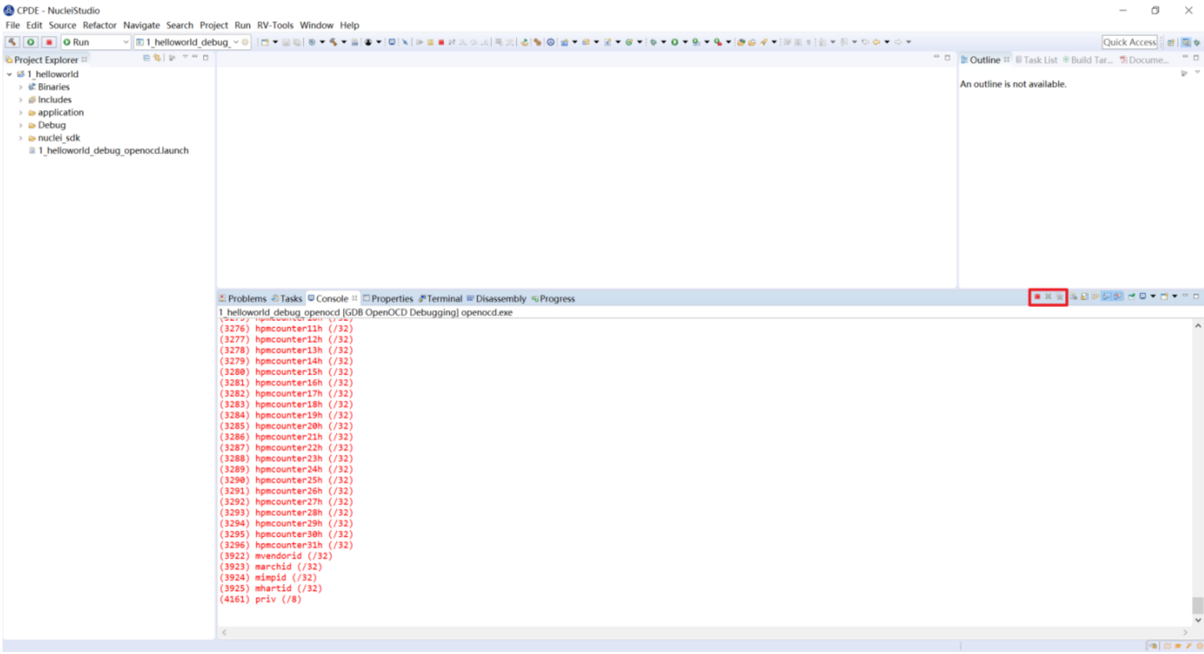
调试程序没有出现问题后，可以将程序下载进开发板。点击下拉框切换至运行模式，此时左侧图标会切换为绿色运行按键，单击即可将程序下载至开发板并运行。由于调试和下载运行使用相同的设置文件，所以不需要再次设置。



程序正常运行后，可以看到串口正确打印出 helloworld 等信息。



如果想要结束程序运行并需要断开连接，在 console 栏目下点击红色按钮断开连接。



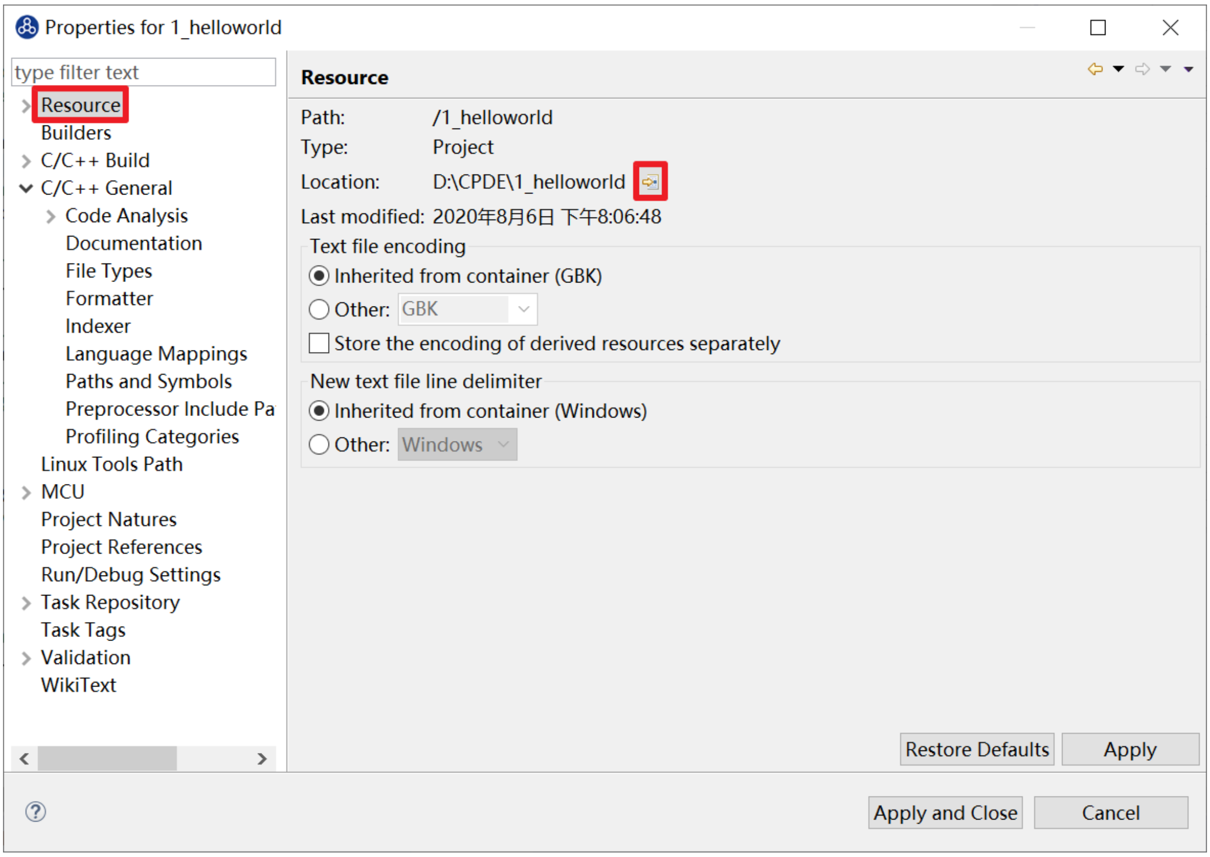
2.9.3 使用 J-Link 调试运行项目

安装 J-Link 驱动并导入 RTT 文件

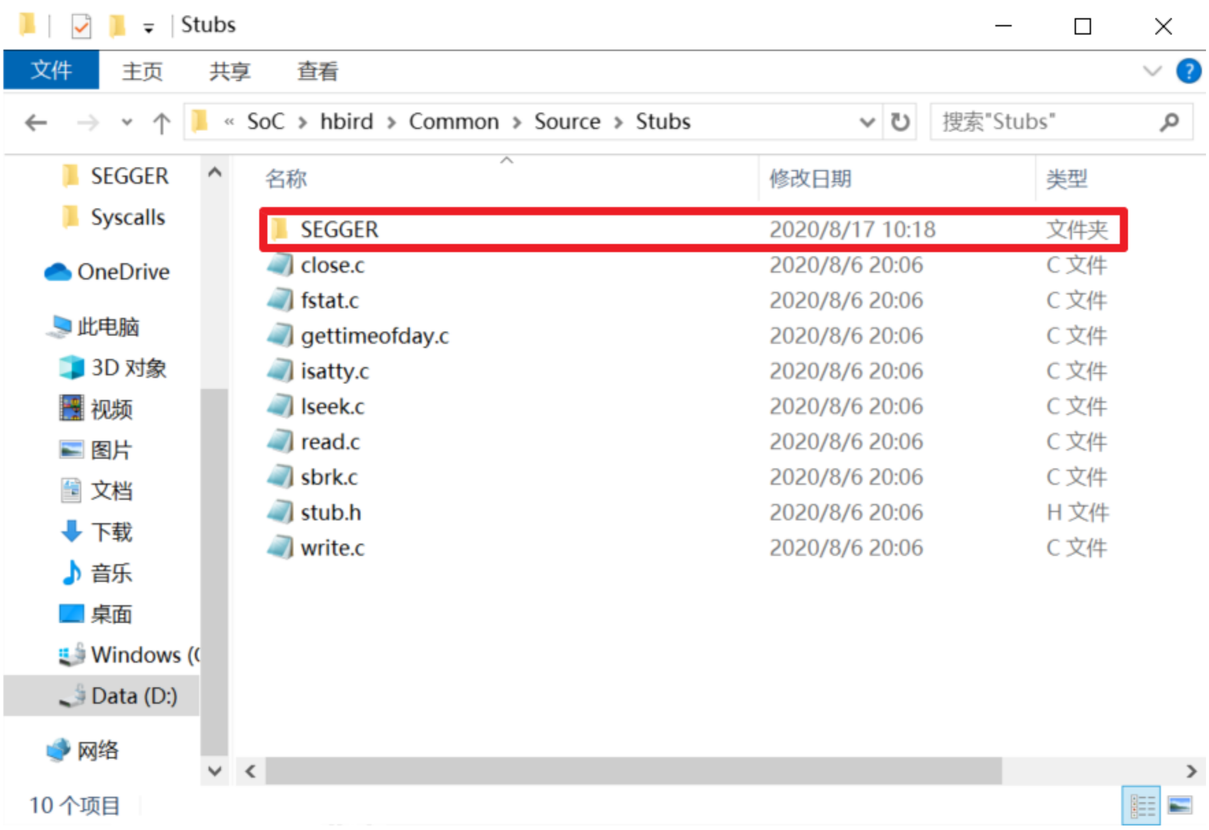
HummingBird Evaluation Board 也支持使用 J-Link 调试。前往 SEGGER 官网 J-Link 页面 (https://www.segger.com/downloads/jlink/#J-LinkSoftwareAndDocumentationPack)，根据自己的操作系统下载最新的 J-Link 驱动并安装。注意，J-Link 的版本必须高于 v6.62 版本。

如果使用串口进行打印输出，则可以略过本节后续内容。如果想使用 J-Link 的 RTT 打印输出，请按照以下步骤配置。

打开当前工程的设置页面，在 Resource 选项点击红框标注的图标快速打开工程所在的目录。

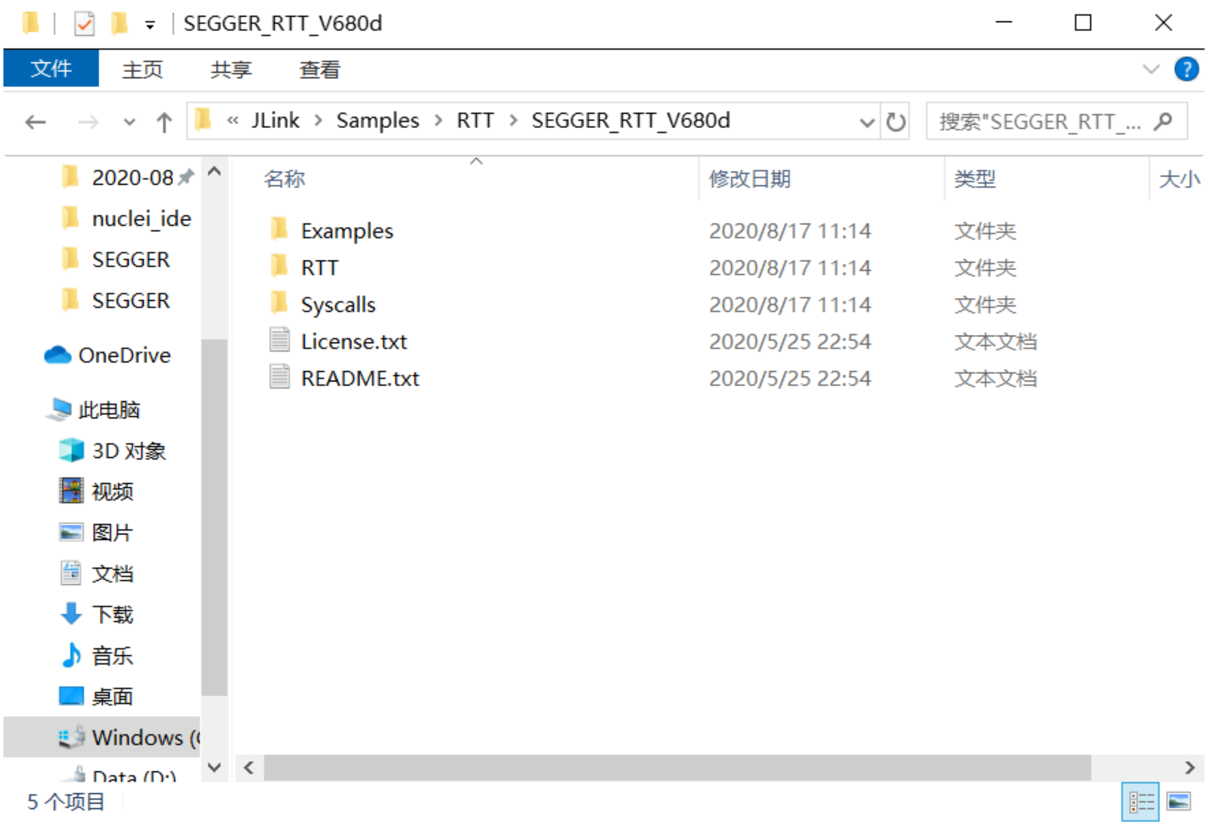


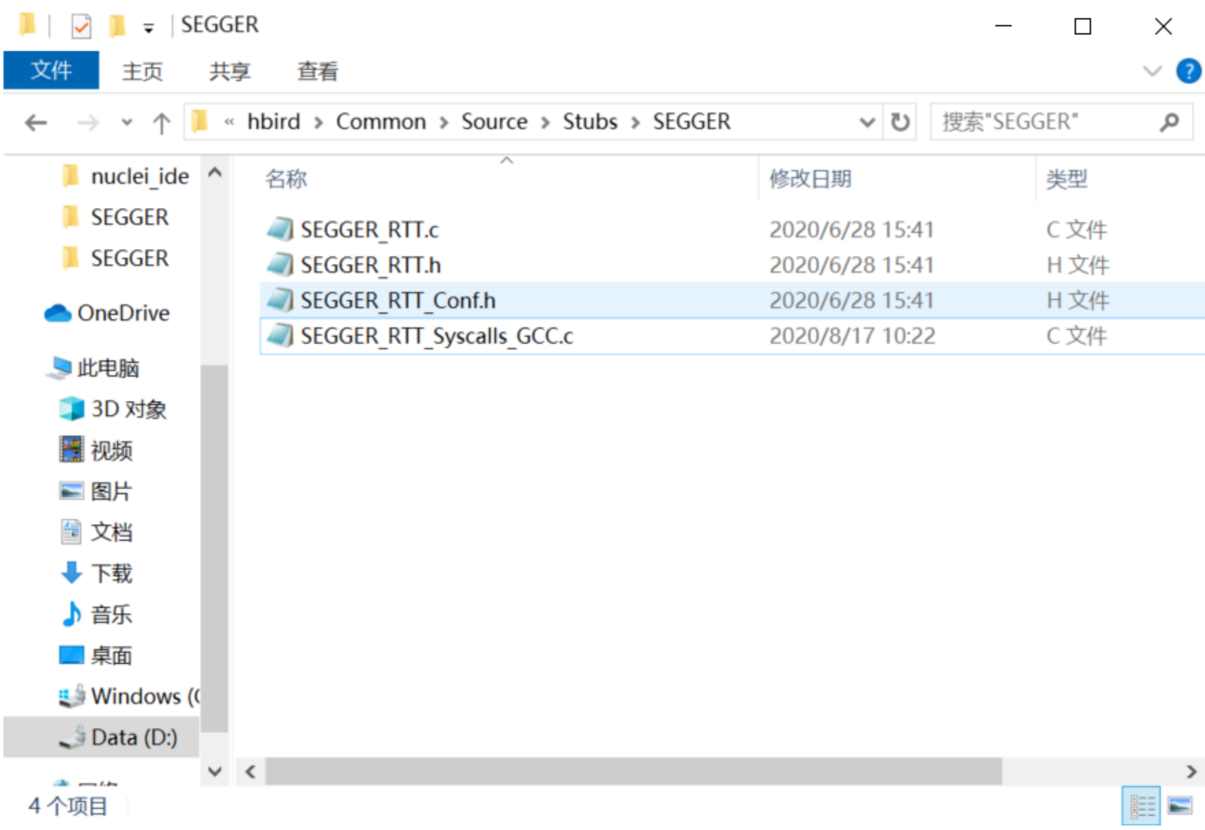
在 nuclei_sdk/SoC/evalsocsoc/Common/Source/Stubs 路径下新建一个 SEGGER 文件夹，此文件夹用来存放 RTT 相关文件。



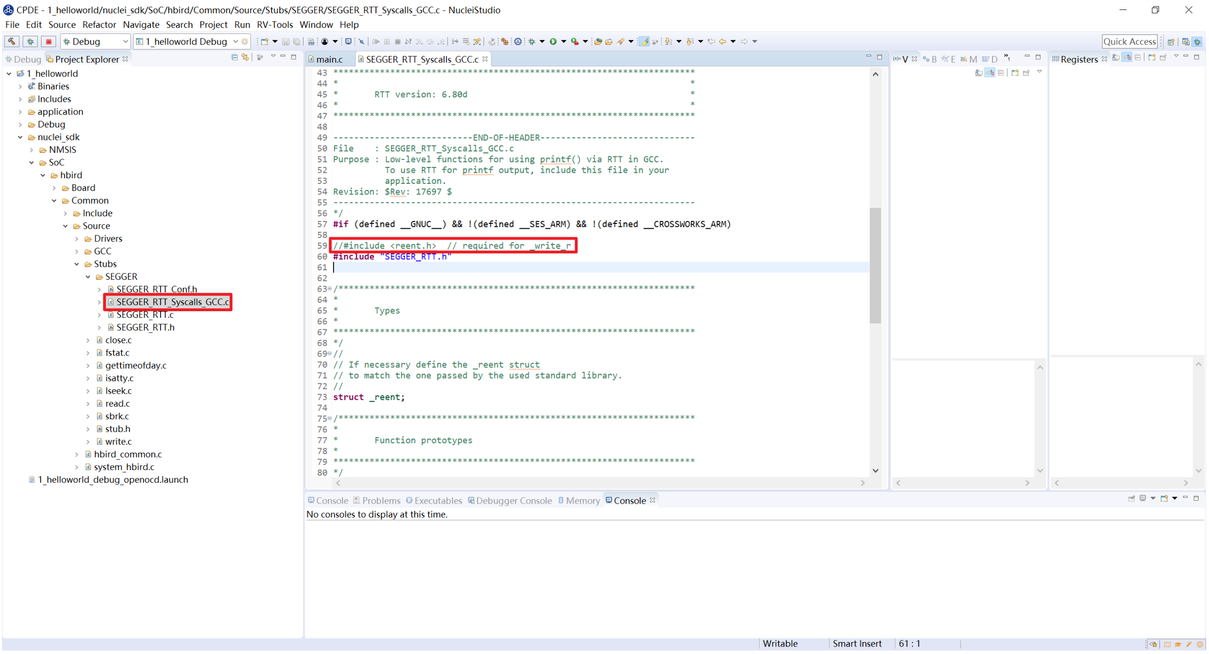
安装完成后打开 J-Link 驱动的根目录，将 Samples -> RTT 路径下的 SEGGER_RTT_V680d.zip 解压缩（具体压缩包名可能因版本不同而变化）。解压缩后文件内容，将 RTT 文件夹下的

SEGGER_RTT.c，SEGGER_RTT.h 和 SEGGER_RTT_Conf.h 三个文件以及 Syscalls 文件夹下的 SEGGER_RTT_Syscalls_GCC.c 这些文件复制到之前新建的 SEGGER 文件夹中。

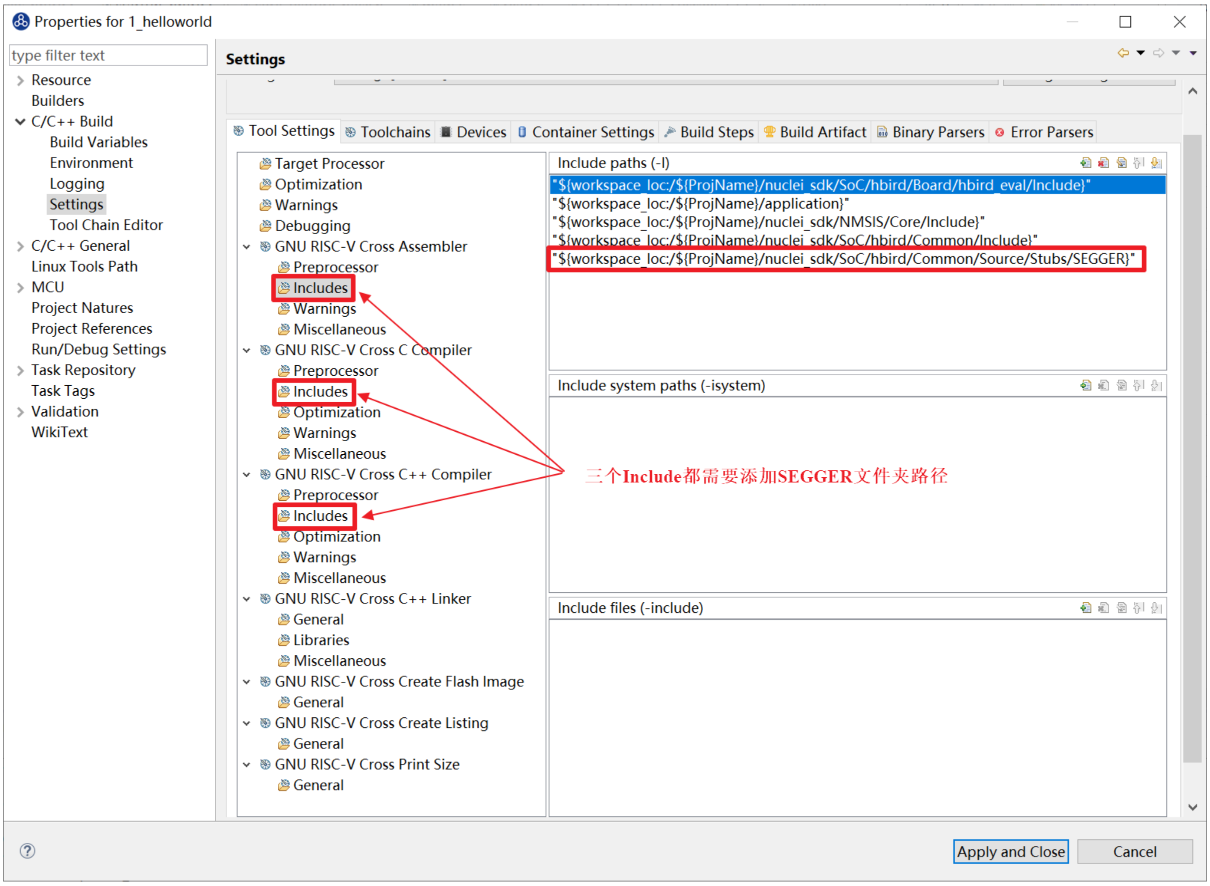




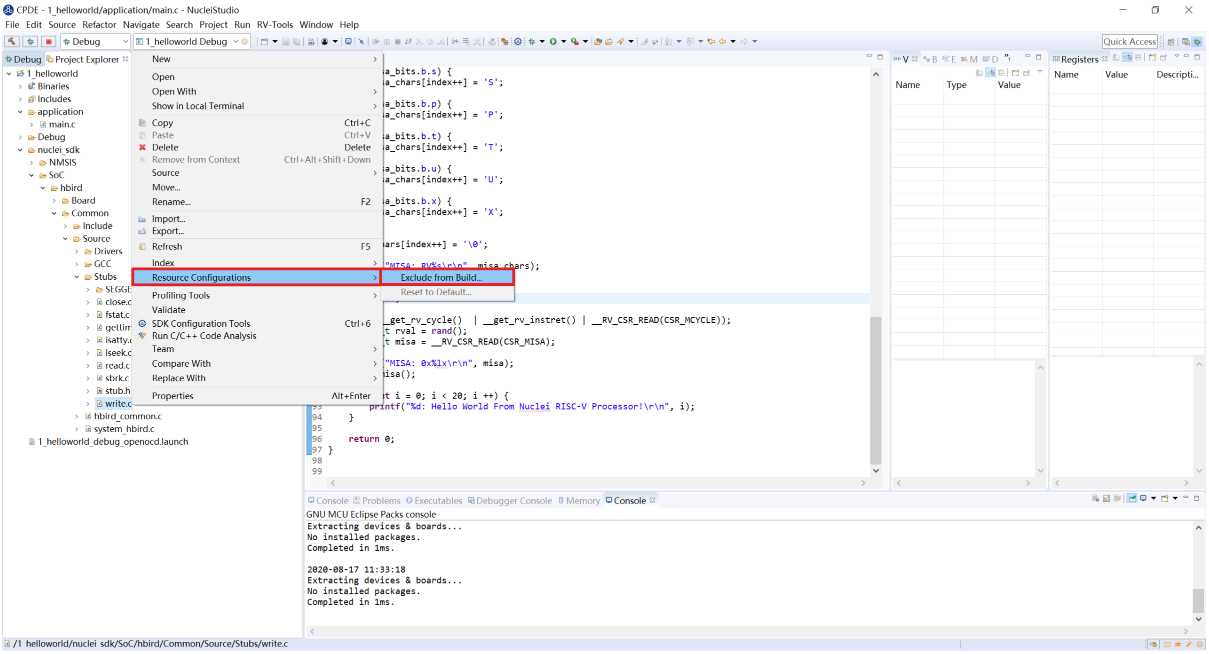
最后在 IDE 中打开 SEGGER_RTT_Syscalls_GCC.c，注释 #include <reent.h> 所在的这一行。



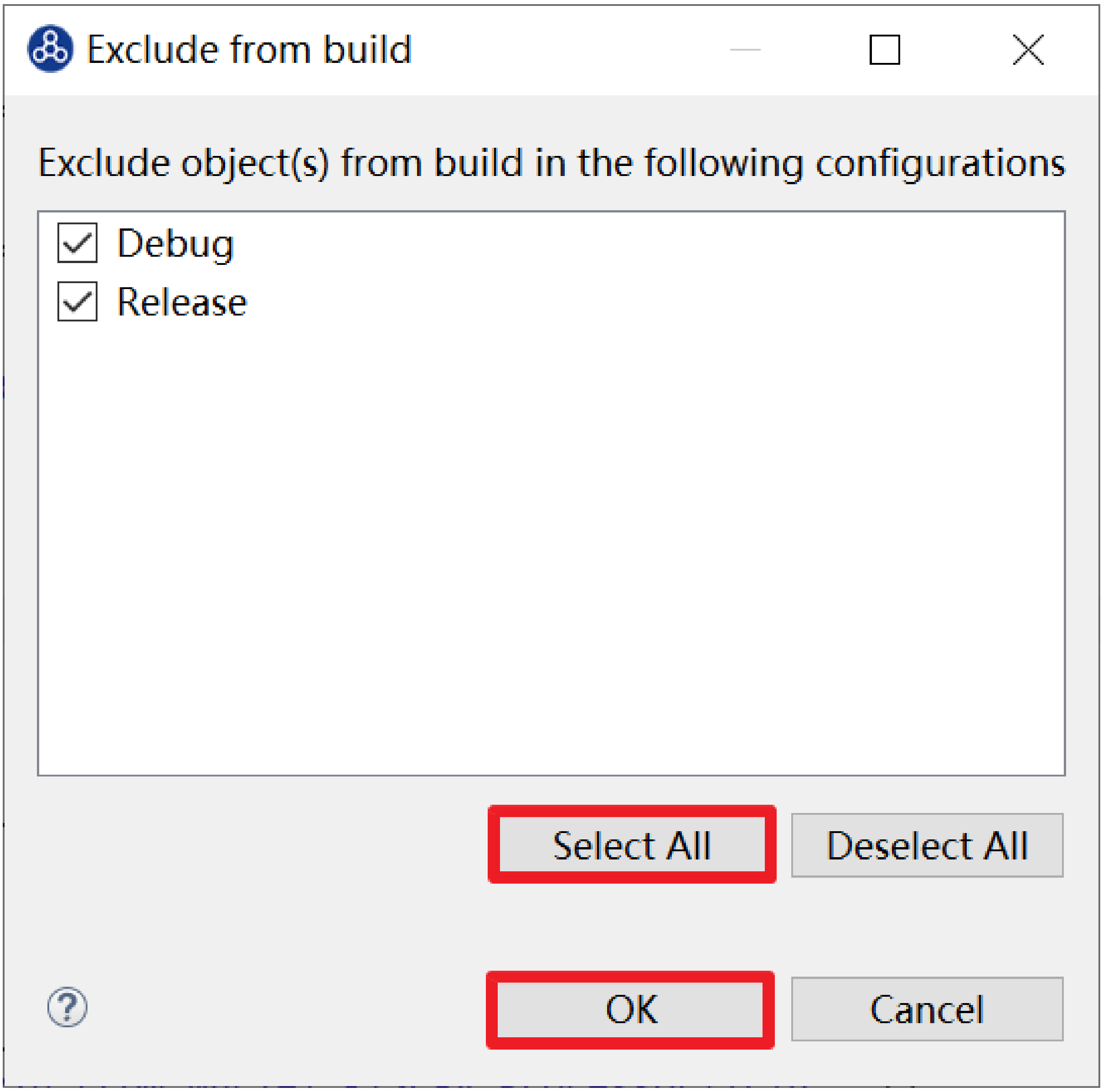
文件添加完成后添加 SEGGER 文件夹路径至 include，打开当前工程的设置页面，添加 SEGGER 文件夹路径至 include 中。



接下来移除原有的 write 函数。在 `nuclei_sdk/SoC/evalsoc/Common/Source/Stubs` 下的 `write.c` 文件处右击，选择 `Resource Configurations > Exclude from Build`。如图 7-30，选择 `Select All`，点击 `OK`。



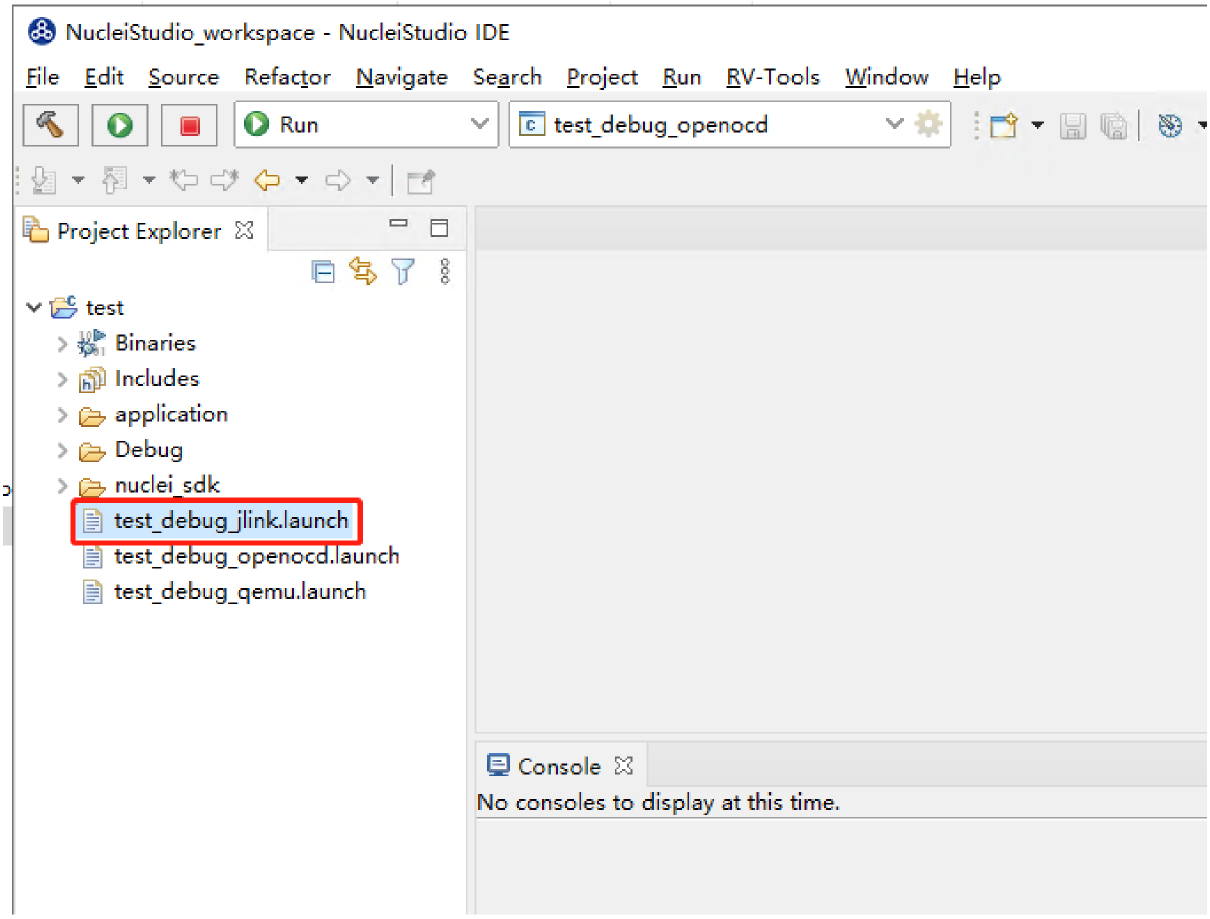
以后如果想切换回使用串口打印，可以使用相同的方式移除 SEGGER 文件夹并把 `write.c` 文件添加回工程。



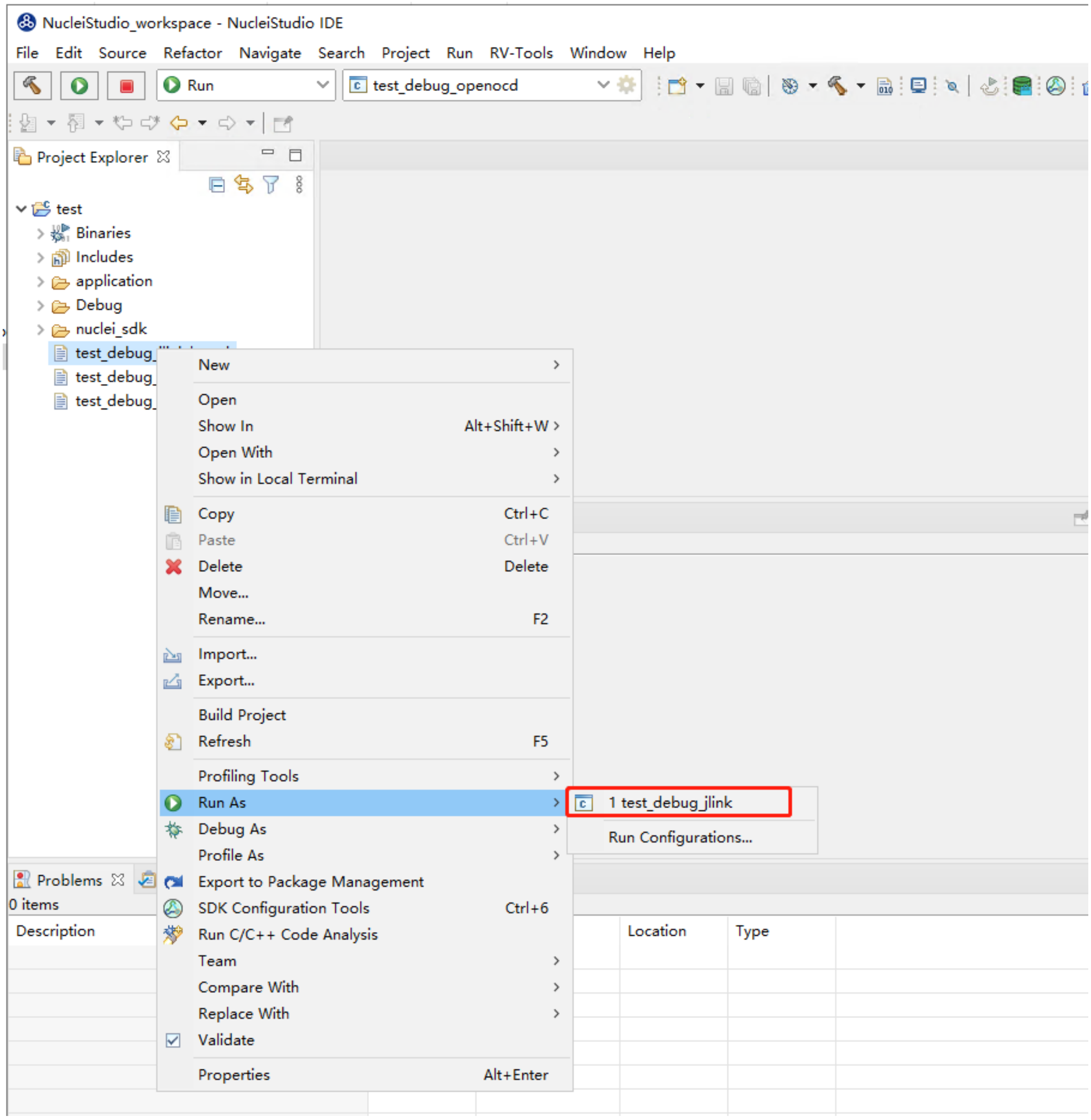
Debug Configuration

使用 Nuclei Studio 生成的 Debug Configuration

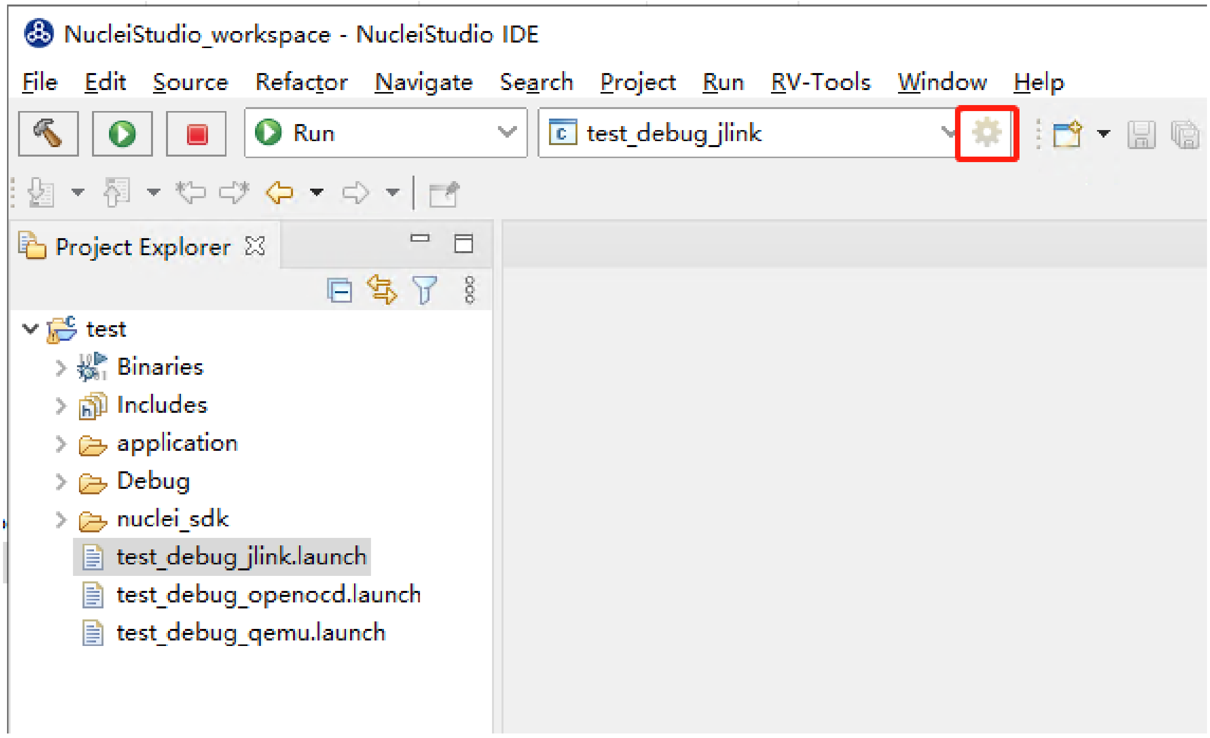
为了方便用户调试，Nuclei Studio 在创建工程时，会根据 NPK 的配置，默认的生成 Debug Configurations 的 Launch 文件。

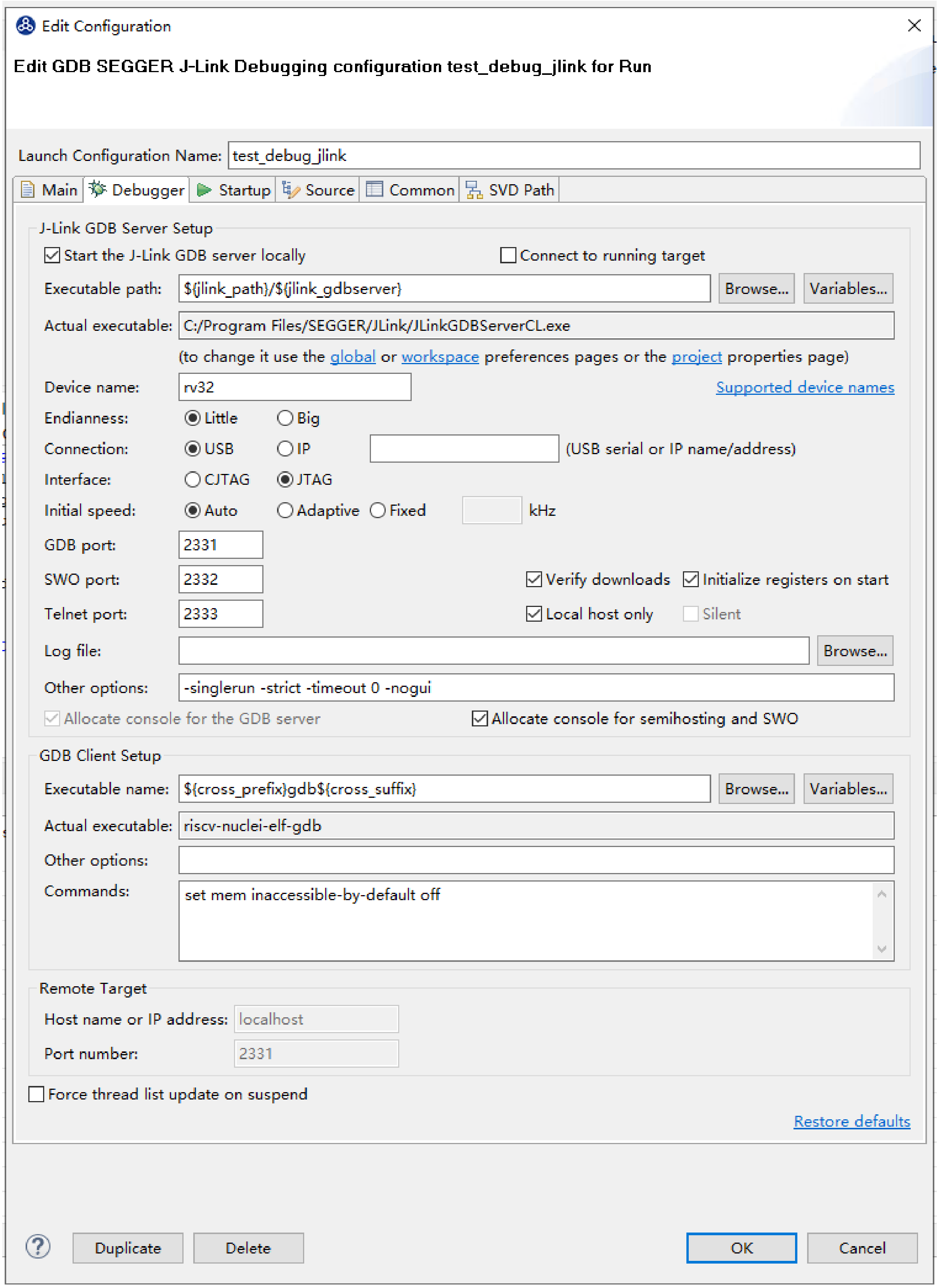


用户可以展开工程，选中对应的 `test_debug_jlink.launch` 文件，在右键菜单中，可以 `Run as/Debug as->test_debug_jlink`，就可以按照对应的 Debug Configurations 操作工程程了，



具体的 Debug Configurations 的内容可以在 Launch Bar 中进行详情查看。

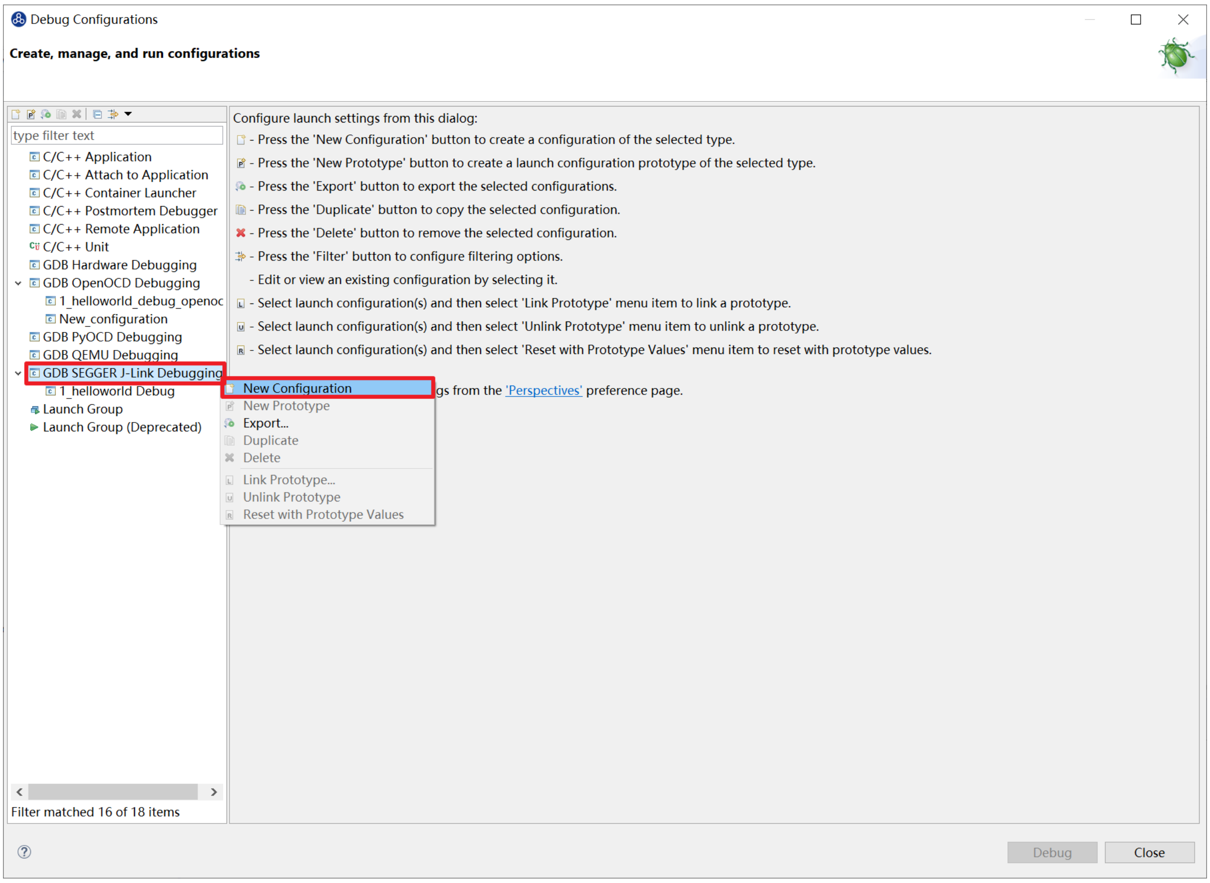

NucleiStudio_workspace - NucleiStudio IDE
File Edit Source Refactor Navigate Search Project Run RV-Tools Window Help
Run
test_debug_jlink
Project Explorer
test
Binaries
Includes
application
Debug
nuclei_sdk
test_debug_jlink.launch
test_debug_openocd.launch
test_debug_qemu.launch


Edit Configuration
Edit GDB SEGGER J-Link Debugging configuration test_debug_jlink for Run
Launch Configuration Name: test_debug_jlink
Main
Debugger
Startup
Source
Common
SVD Path
J-Link GDB Server Setup
Start the J-Link GDB server locally
Connect to running target
Executable path: ${jlink_path}/${jlink_gdbserver}
Browse...
Variables...
Actual executable: C:/Program Files/SEGGER/JLink/JLinkGDBServerCL.exe
(to change it use the global or workspace preferences pages or the project properties page)
Device name: rv32
Supported device names
Endianness: Little Big
Connection: USB IP (USB serial or IP name/address)
Interface: CJTAG JTAG
Initial speed: Auto Adaptive Fixed kHz
GDB port: 2331
SWO port: 2332
Verify downloads
Initialize registers on start
Telnet port: 2333
Local host only
Silent
Log file:
Browse...
Other options: -singlerun -strict -timeout 0 -nogui
Allocate console for the GDB server
Allocate console for semihosting and SWO
GDB Client Setup
Executable name: ${cross_prefix}gdb${cross_suffix}
Browse...
Variables...
Actual executable: riscv-nuclei-elf-gdb
Other options:
Commands: set mem inaccessible-by-default off
Remote Target
Host name or IP address: localhost
Port number: 2331
Force thread list update on suspend
Restore defaults
Duplicate
Delete
OK
Cancel

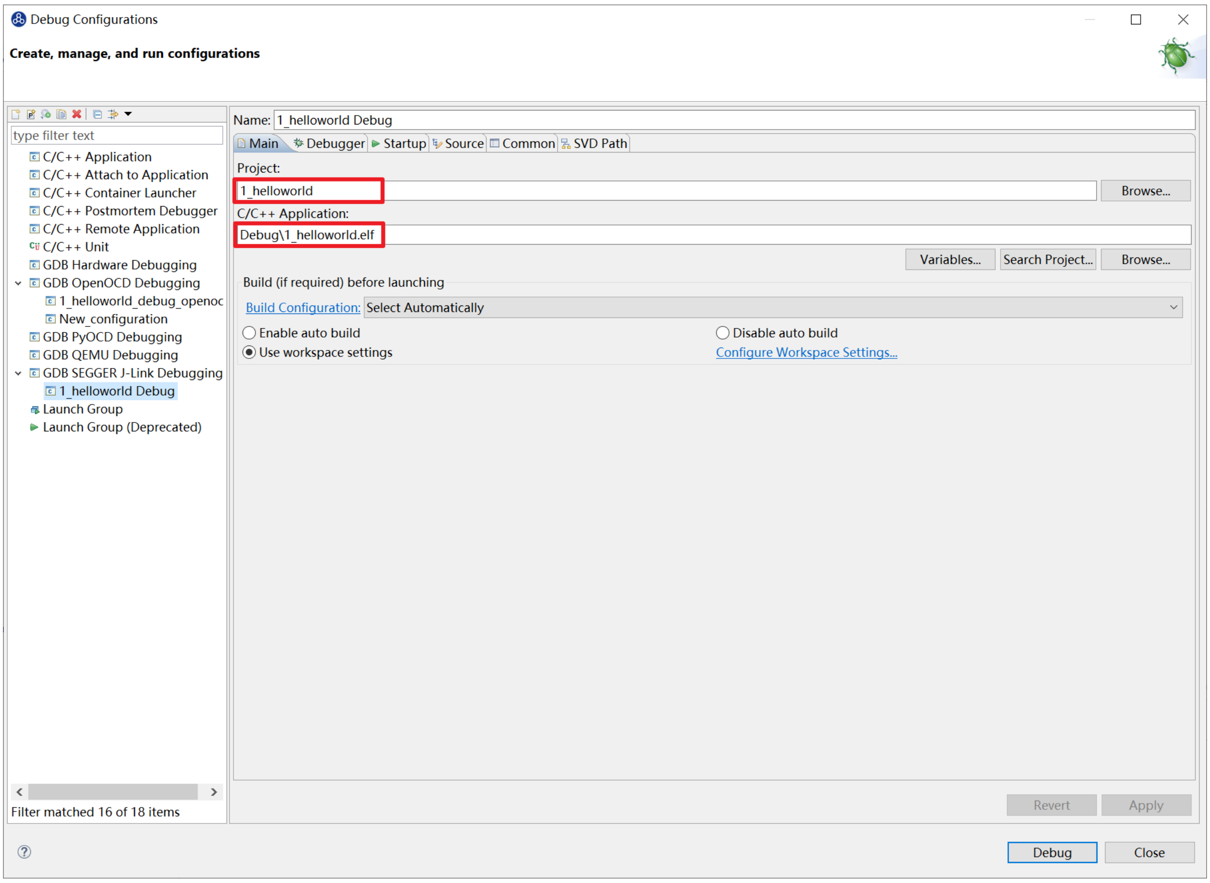
新建并配置 Debug Configuration

新建并配置 J-Link 调试下载的 Debug Configuration 步骤如下：

在菜单栏中选择 `Run—>Debug Configurations`。在弹出的窗口中，如果没有当前工程的调试设置内容，右键单击 `GDB SEGGER J-Link Debugging`，选择 `New Configuration`，将会为本项目新建出一个调试项目 `1_helloworld Debug`。



确保 `Project` 是当前需要调试的工程，`C/C++ Application` 中选择了正确的需要调试的 ELF 文件。



打开 Debugger 栏目，确保 1 号位置 `Start the J-Link GDB server locally` 被选中。

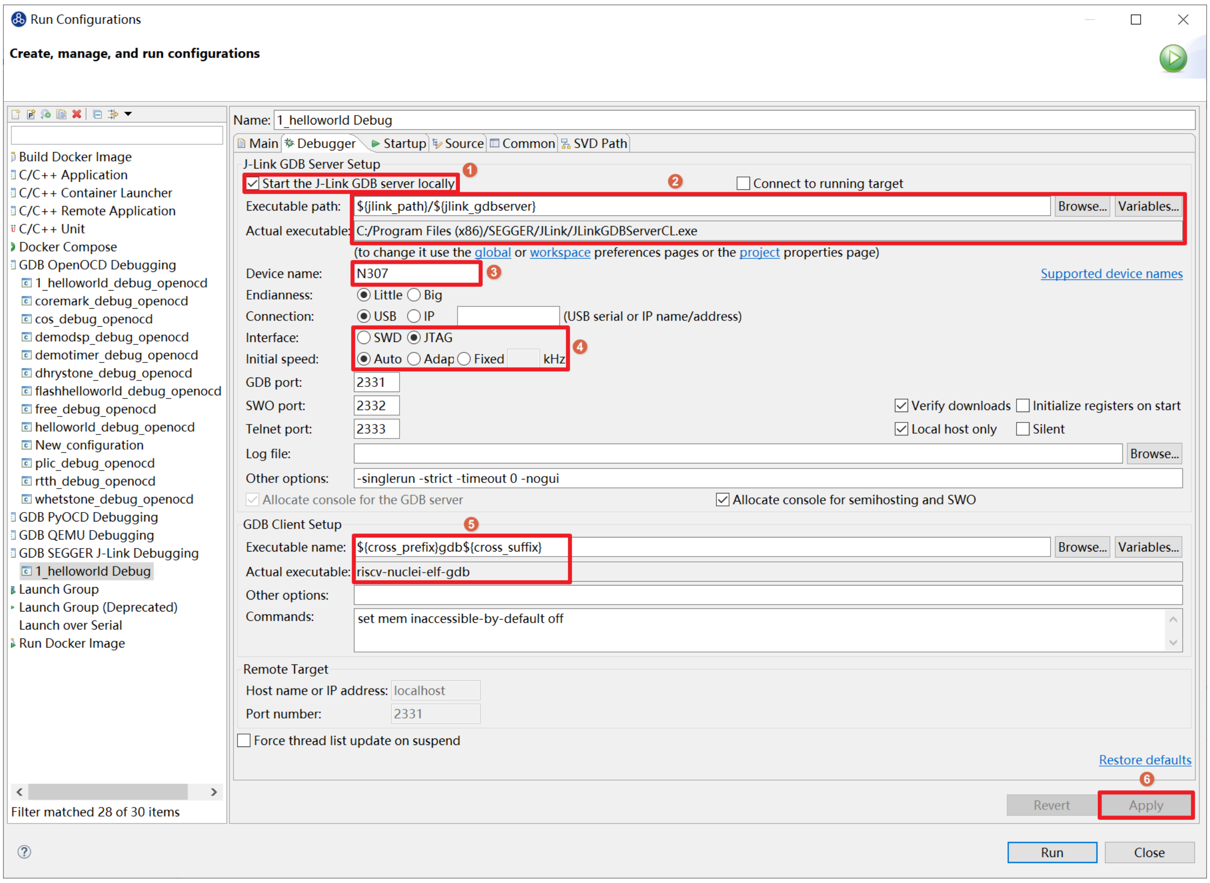
2 号位置正确指向 JLinkGDBServerCL.exe 的路径。

3 号内是当前使用的内核，这里以 N307 为例，输入 N307 即可。如果使用 RV-STAR 开发板，这里输入 GD32VF103VBT6。如果使用其他开发板请参考 J-Link Support Device 网页，链接如下：https://www.segger.com/downloads/supported-devices.php

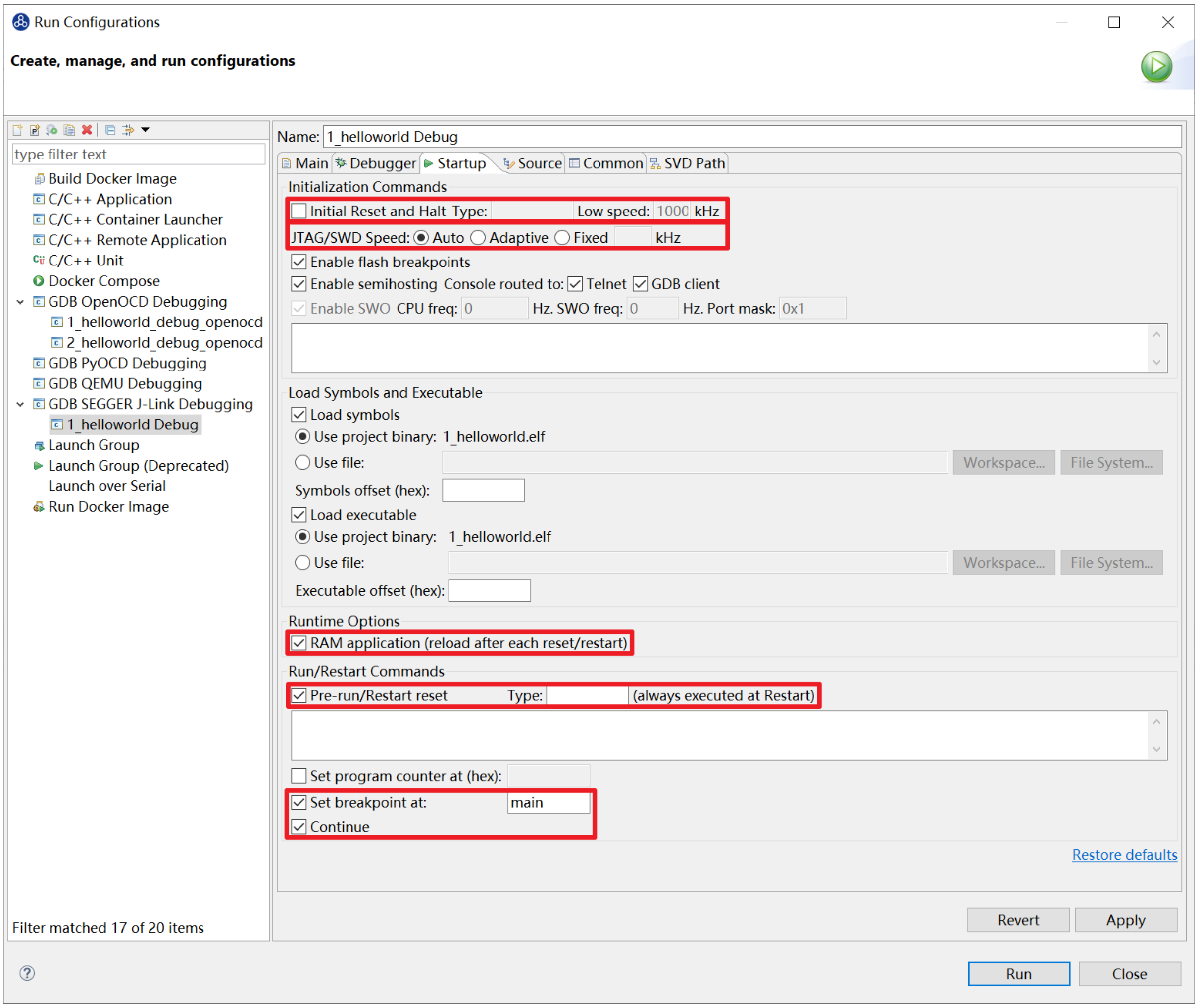
4 号选择 `Interface` 为 JTAG，`initial speed` 为 Auto。

5 号确认与使用的 GDB 设置一致。

如果有修改的内容，点击 6 号位置 `Apply` 保存。



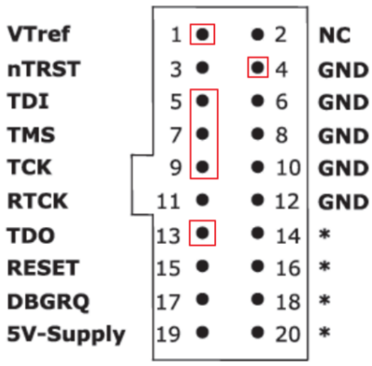
打开 Startup 栏目，确保 JTAG/SWD Speed 为 Auto，set Breakpoint at main，Continue，Pre-run/Restart reset 和 RAM application 选项被勾选，并且取消勾选 Initial Reset and Halt 选项。



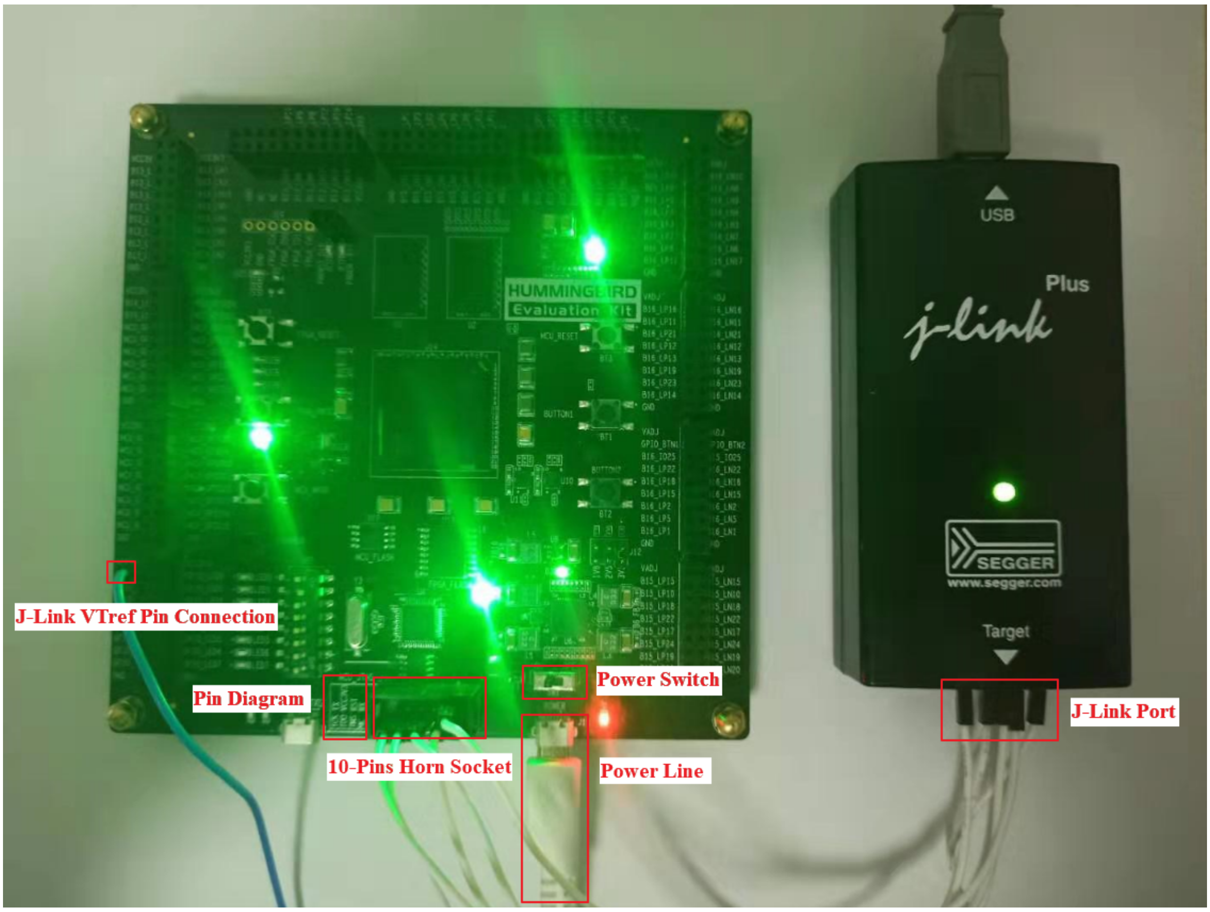
以上设置内容完成后，如果有变动需要点击右下角 Apply 保存设置，如果没有变动点击 close 即可。

在原型开发板上调试程序

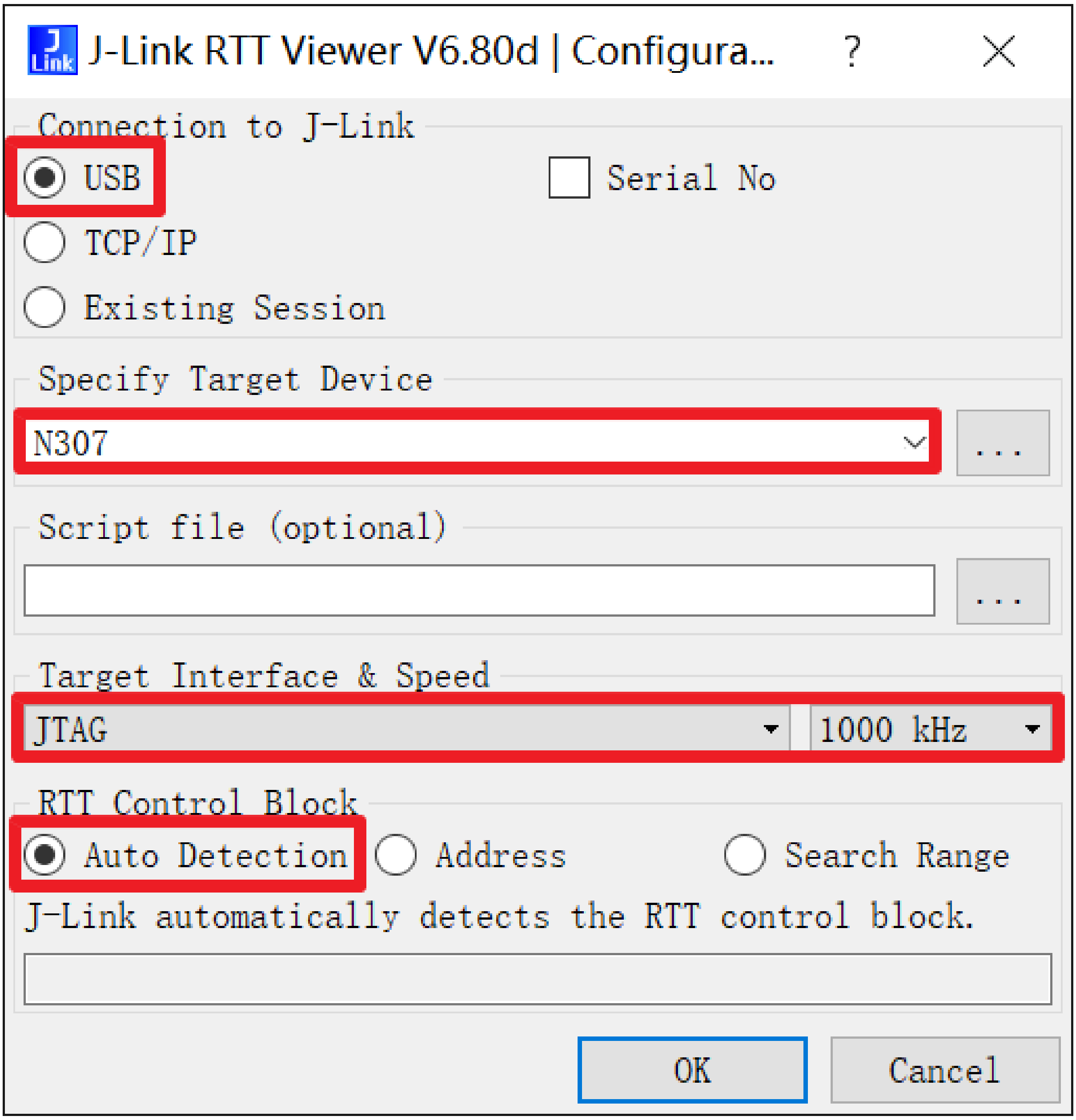
使用 J-Link 在 HummingBird Evaluation Board 调试需要连接跳线。红框标注的部分是 J-Link 需要连接到板子的部分。



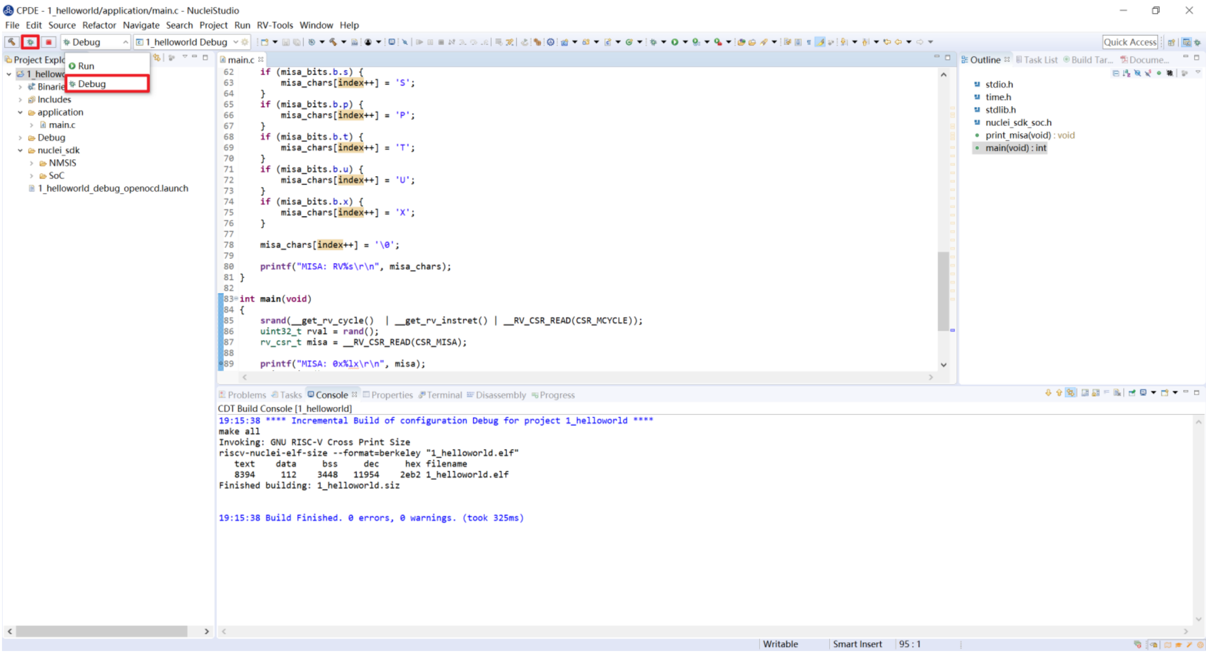
其中 VTref 连接到板子上 V3.3 的接口，其他部分连接到 JTAG 接口，各引脚的丝印就在旁边，一一对应连接即可，最后实物连接如下图。



在开发板上调试之前，如果使用串口打印，需要连接 JTAG 上的串口引脚到自己的主机上，再打开串口以便观察 Printf 函数打印信息。如果使用 RTT 打印，需要打开 `J-Link RTT Viewer` 查看 printf 打印信息。按照图中内容设置，选择 USB 方式连接。Specify Target Device 根据使用的内核来修改，这里以 N307 为例。Target Interface & Speed 设置为 `1000kHz`，可根据实际使用情况来修改。RTT Control Block 选择 `Auto Detection`。

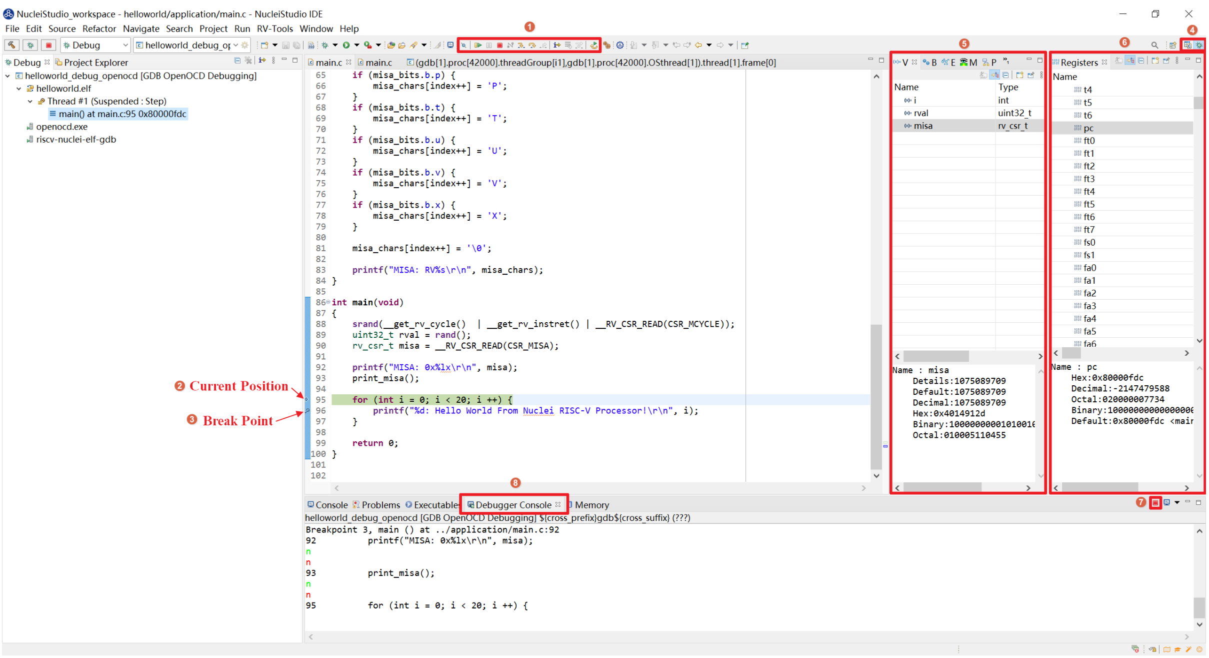


在 Launch Bar 的下拉框选中 Debug，之后左侧图标会变为甲虫图标，单击即可进入调试模式并下载程序进入开发板中。



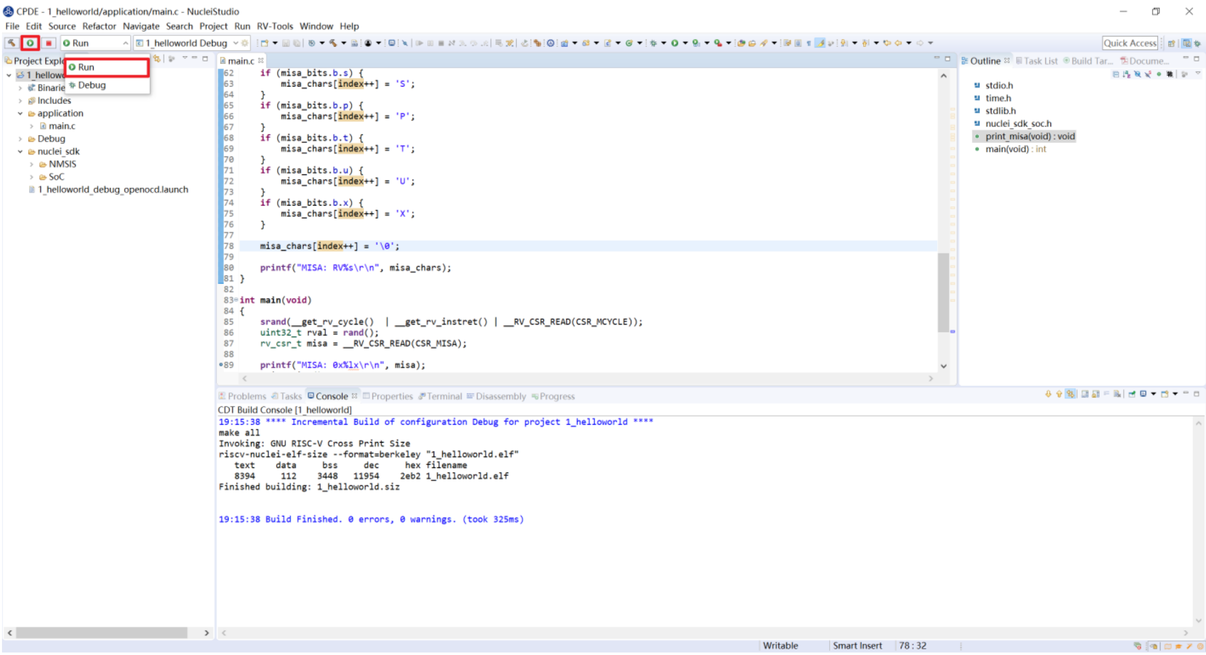
如果程序下载成功，并且 Nuclei Studio 会启动调试界面。

- 如图 1 号标注位置，这里功能包括单步，运行，汇编级调试等。
- 如图 2 号标注位置，这个箭头表示当前程序运行位置。
- 如图 3 号标注位置，在代码的左侧双击即可在该行设置断点，再次双击可以取消断点。
- 如图 4 号标注位置，这里可以切换编辑模式和调试模式。
- 如图 5 号标注位置，这里是函数内变量显示的位置。
- 如图 6 号标注位置，这里是查看寄存器数值的位置。
- 如图 7 号标注位置，点击这里红色按钮可以退出调试模式。
- 如图 8 号标注位置下方，这里可以使用 GDB 控制台指令进行调试。

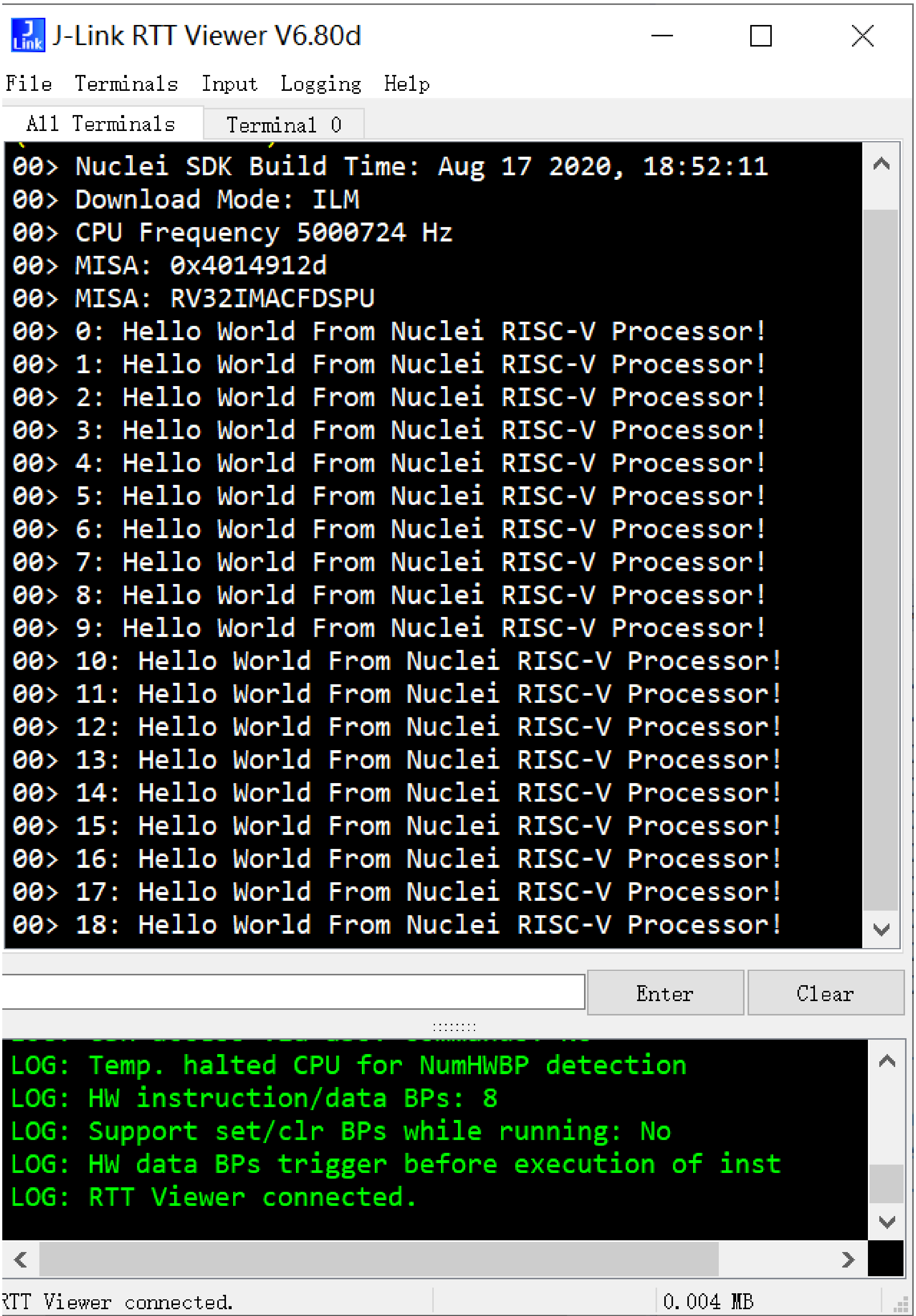


下载运行程序

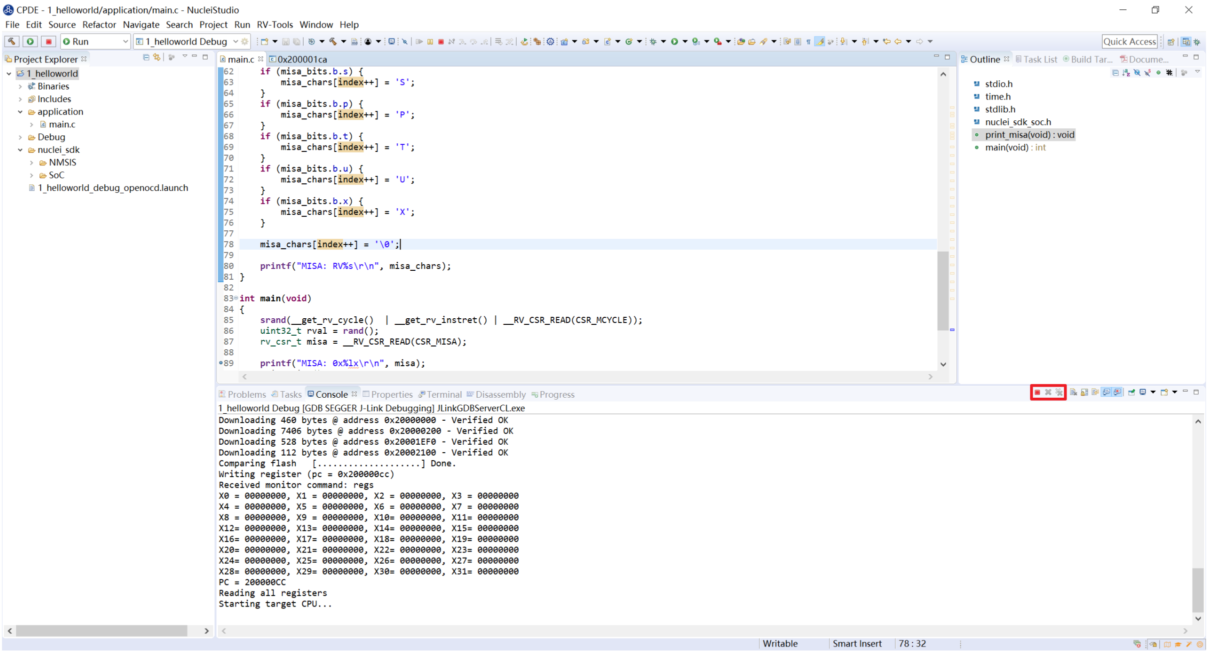
调试程序没有出现问题后，可以将程序下载进开发板，点击下拉框切换至运行模式，此时左侧图标会切换为绿色运行按键，单击即可将程序下载至开发板并运行。



由于调试和下载使用相同的设置文件，所以不需要再次设置。可以看到 RTT Viewer 正确打印出 helloworld 等信息。



如果需要断开连接，在 console 栏目下点击红色按钮即可断开连接。

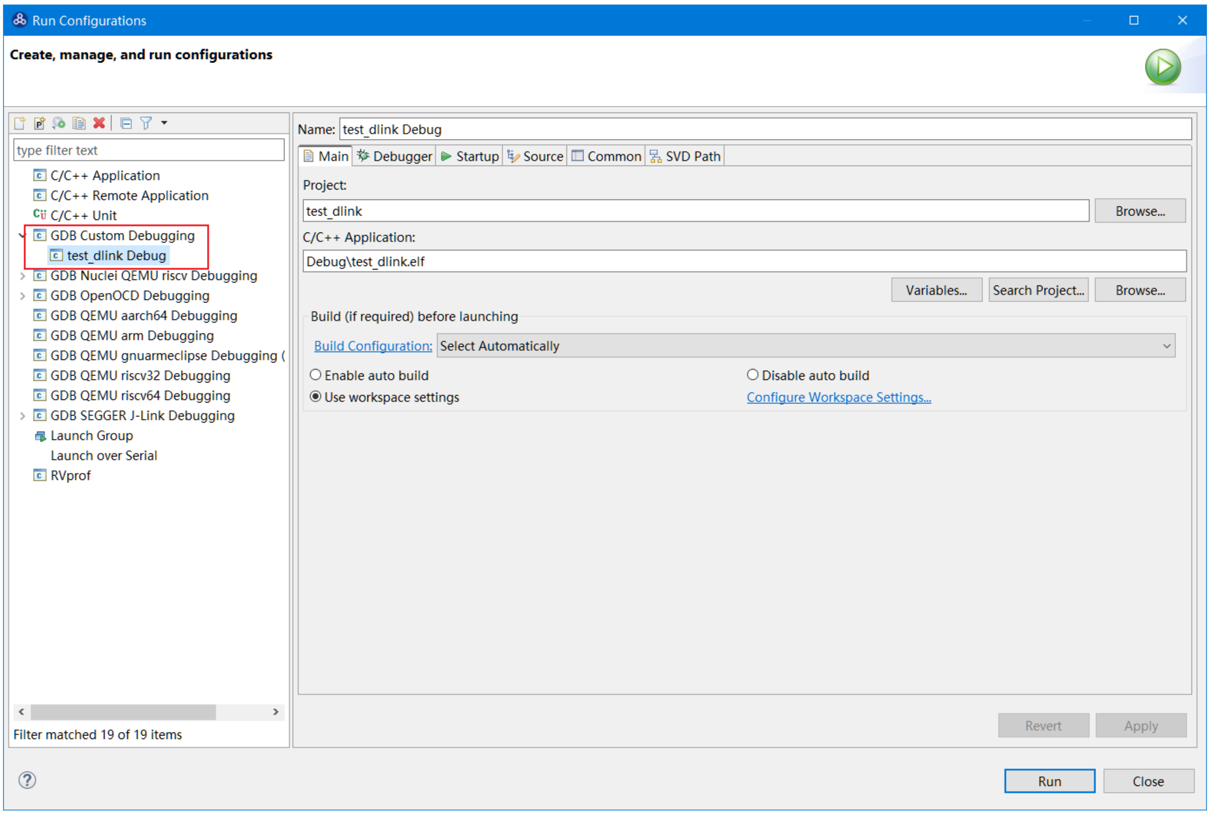


2.9.4 使用 DLink 调试运行项目

DLink 是芯来科技基于 RV Link，并在其基础上做功能迭代升级后，所研发的 RISC-V 调试器，使之更适应于 Nuclei Studio 的应用场景。目前 Dlink 仅针对单核 RISC-V 工程实现调试运行，且已实现量产，具体实物如下, 具体关于 Dlink 固件下载参见 https://github.com/Nuclei-Software/nuclei-dlink/wiki/upload-dlink-firmware 。

> **Note**
>
> 在 芯来科技视频号中有 如何在 **Nuclei Studio** 中通过 **Dlink** 调试程的视频，您可以在微信中搜索芯来科技视频号点击查看相关内容。

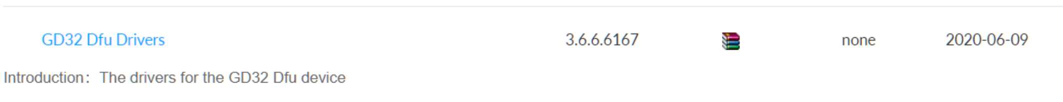
如需使用 Dlink 进行调试，可以在 NucleiStudio 菜单中 `Run -> Run Configurations` 打开 Run Configurations 的配置页面，并配置一个 Dlink Debug Configuration，双击 GDB Custom Debugging 新建一个配置项，并在 Main 选项卡中配置内容如下：



在 **windows** 环境下安装驱动步骤如下：

打开 GB 网站并搜索 GD32VF1，在列表中打到 GD 32 Dfu Drivers，下载并安装。

地址：https://www.gd32mcu.com/en/download/7?kw=GD32VF1



在 **Linux** 环境下安装驱动步骤如下：

- 1：连接开发板到 Linux 中，确保 USB 被 Linux 识别出来。
- 2：在控制台中使用 lsusb 指令查看信息，参考的打印信息如下：

```
Bus 003 Device 057: ID 28e9:018a GDMicroelectronics Dlink Low Cost Scheme
```

- 3：控制台中输入 `sudo vi /etc/udev/rules.d/50-dlink.rules` 指令打开 50-dlink.rules 文件，输入如下内容，保存退出，并执行 `sudo udevadm control --reload`。

```
SUBSYSTEM=="usb", ATTR{idVendor}=="28e9",

ATTR{idProduct}=="018a", MODE="664", GROUP="plugdev"

SUBSYSTEM=="tty", ATTRS{idVendor}=="28e9",

ATTRS{idProduct}=="018a", MODE="664", GROUP="plugdev"
```

- 4：断开调试器再重新连接到 Linux 系统中。
- 5：使用 `ls /dev/ttyACM*` 命令查看 ttyACM 信息，参考输出如下：

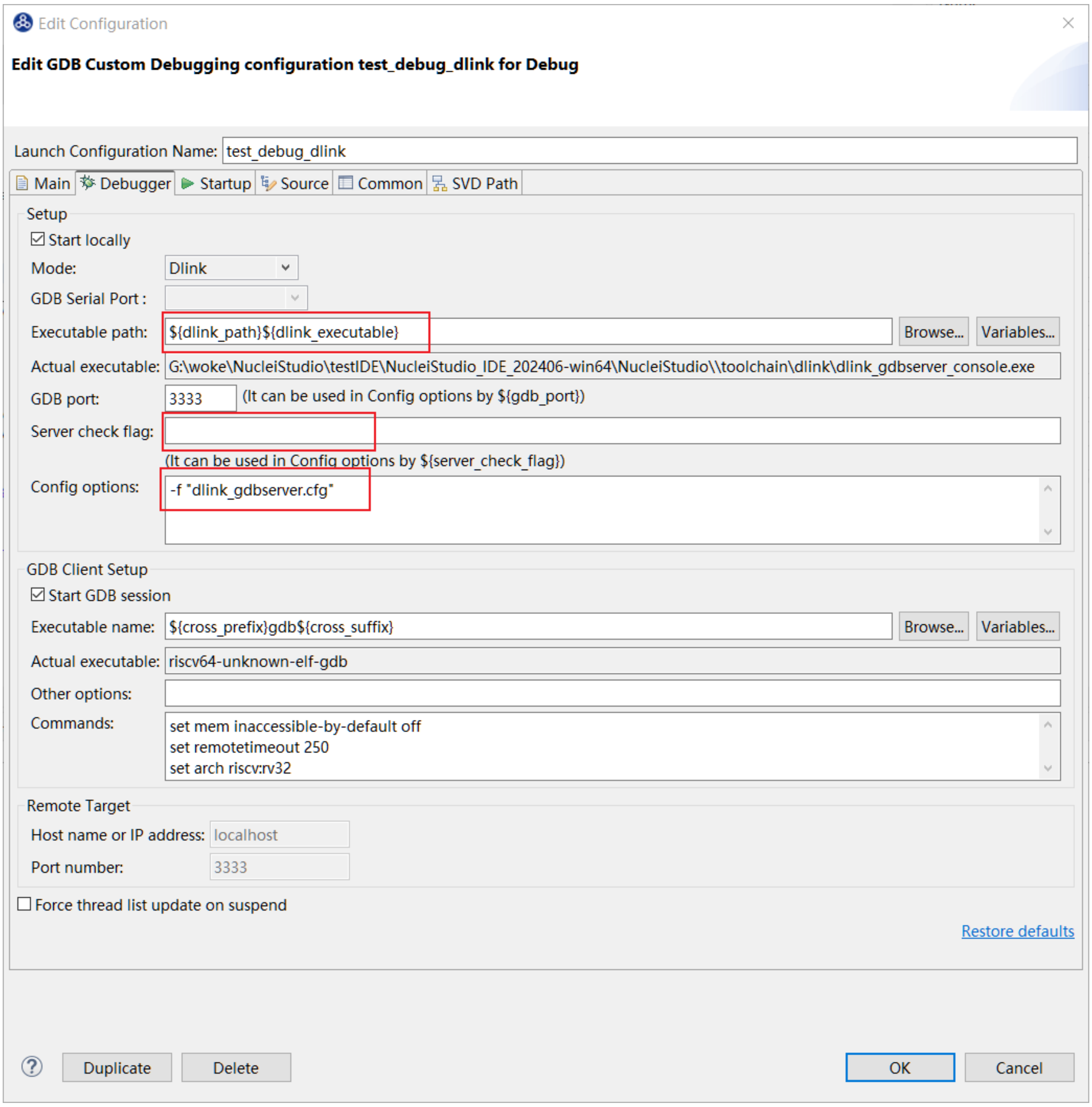
```
/dev/ttyACM0 /dev/ttyACM1
```

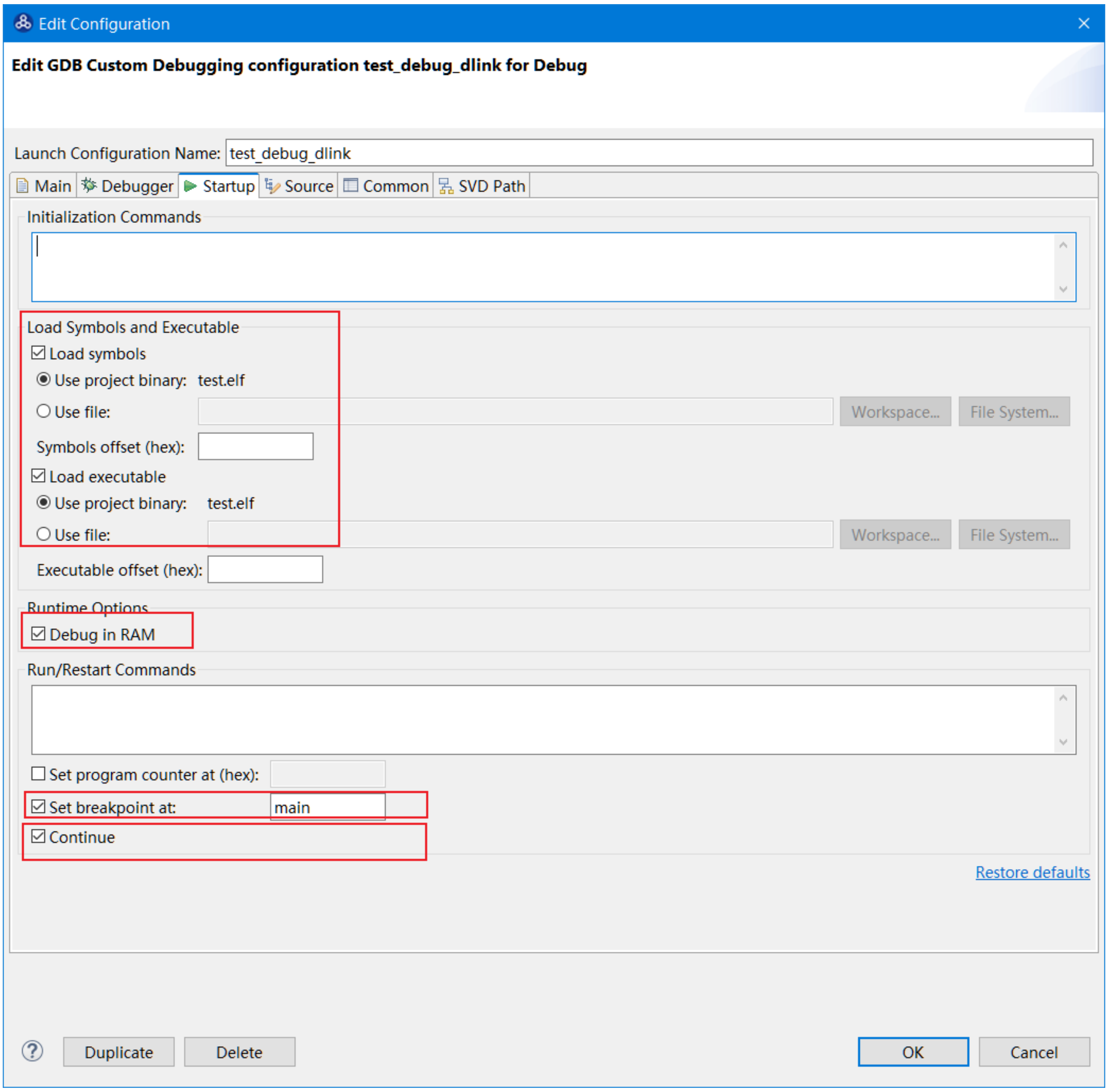
- 6：使用 `ls -l /dev/ttyACM0` 命令查看分组信息，参考输出如下, 可以看到 ttyACM0 已经被加入到 dialout 组，接下来我们要将自己添加到 `dialout` 组（不同环境可能名字不同，请根据实际情况修改）。使用 whoami 命令查看当前用户名，我们将其记录为 < `your_user_name` > 。

```
shell crw-rw-r-- 1 root dialout 166, 0 6月 28 15:25 /dev/ttyACM0
```

- 7：使用 `sudo usermod -a -G dialout <your_user_name>` 命令将自己添加进 dialout 组。加入以后一定要重启或者注销操作系统。
- 8：再次确认当前用户名已属于 dialout 组，使用 groups 命令，可以看到打印信息中有 dialout 即成功将当前用户添加至 dialout 组。如果没有可以尝试重启。

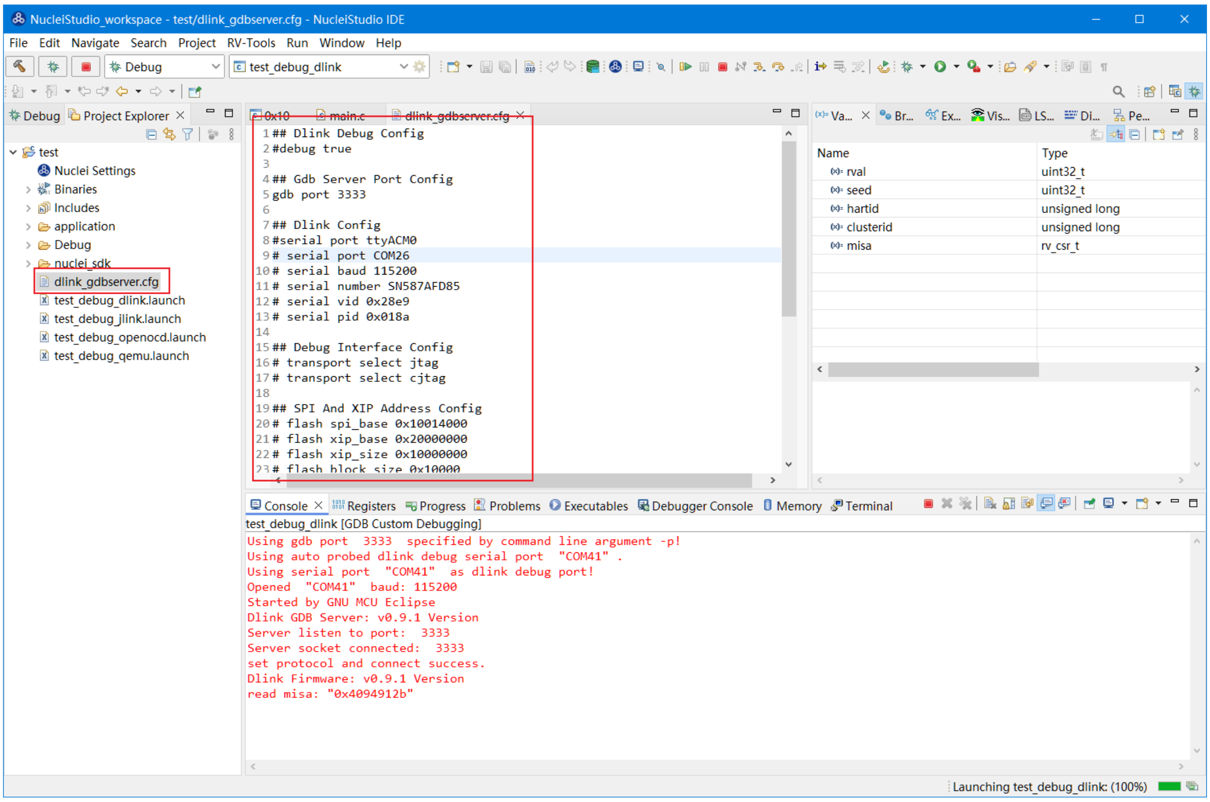
然后在 Debugger 选项卡内配置内容如下，因为在 Custom Debugging 中支持多种 Mode，我们现在需要使用 Dlink，所以选中 Dlink；`Server check flag` 是在 NucleiStudio 中用以确认服务是否正常启动，在 Custom GDB Server 中如果服务正常启动，会输出一段字符串，NucleiStudio 通过判断该字符串以确认 Custom GDB Server 正常启动，在使用 Dlink 时这里可以为空；在 Config options 中需要配置对应的链接文件 `dlink_gdbserver.cfg`，参考配置文件可以在 `<NucleiStudio>/toolchain/dlink` 目录下找到。



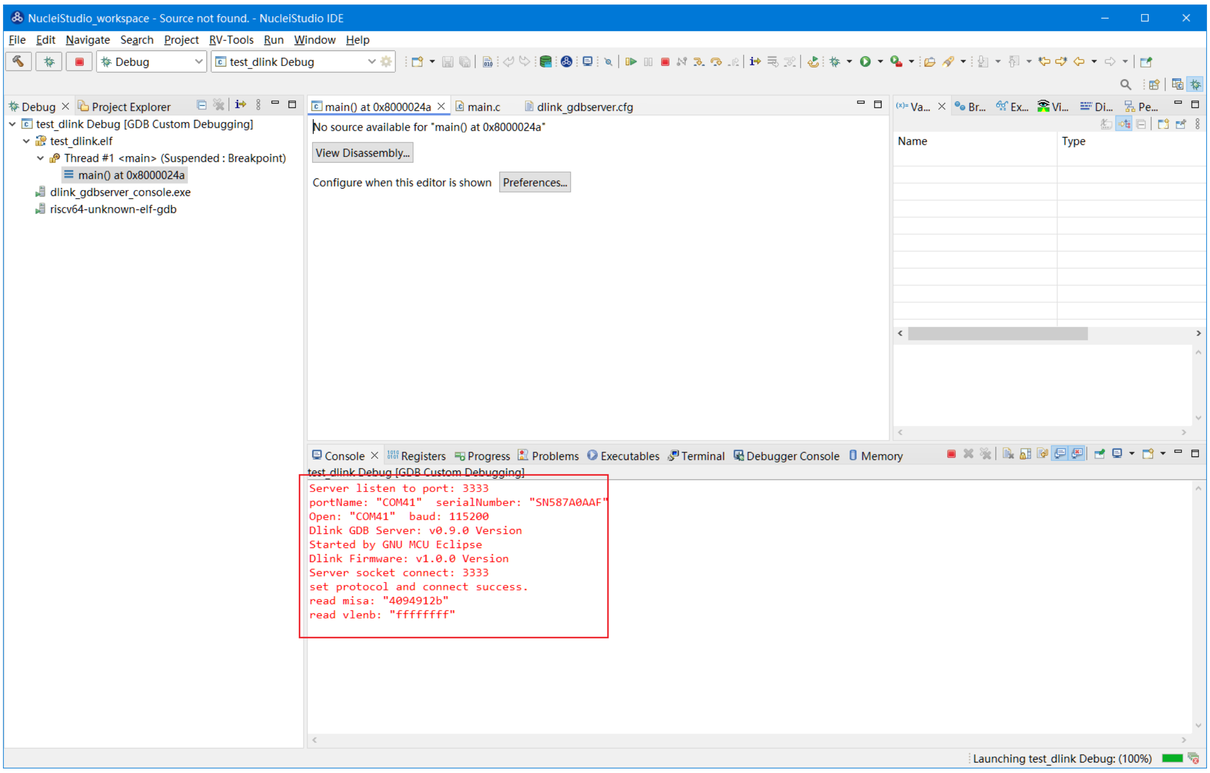


在 windows 下，Dlink 连接上 PC 和开发板后，亮一个绿色灯和一个蓝色灯，说明 Dlink 处理正常工作状态，否则不正常，可以安 NRST 键尝试复位。

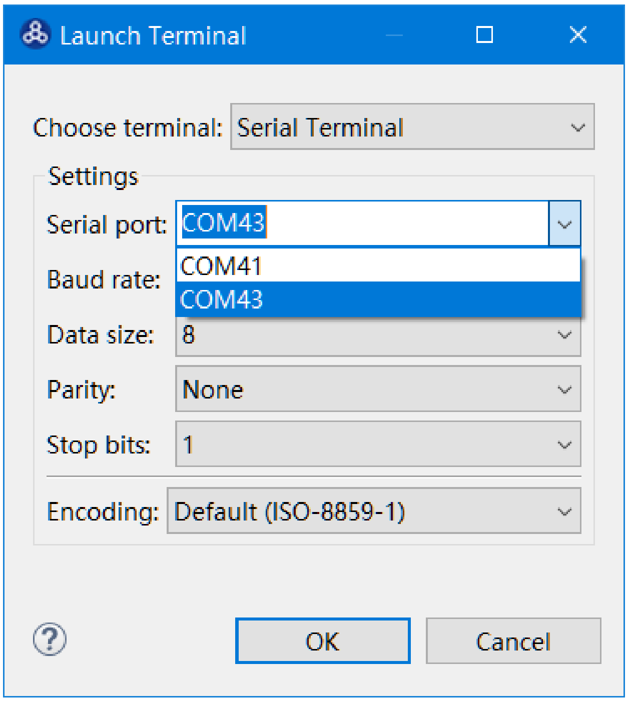
Dlink 连接后，在串口工具下，可以看到两个 COM 口，一个 COM 串用于串口输出，一般情况下数字低的 COM 口是调试的，另一个用于串口数据交换，用户需要在 dlink_gdbserver.cfg 指明用于数据交互的 COM 口，并配置 serial port 和 serial baud，例于在 Windows 下 `serial port COM1`、`serial baud 115200`；在 Linux 下 `serial port ttyACM0`、`serial baud 115200`，如果用户不配置，Dlink 会使用 COM 口中编号较小的那个为 serial port 默认值，并且以其对应的设备号为 serial baud 默认值，以 Windows 下为例，可以参考配置如下：



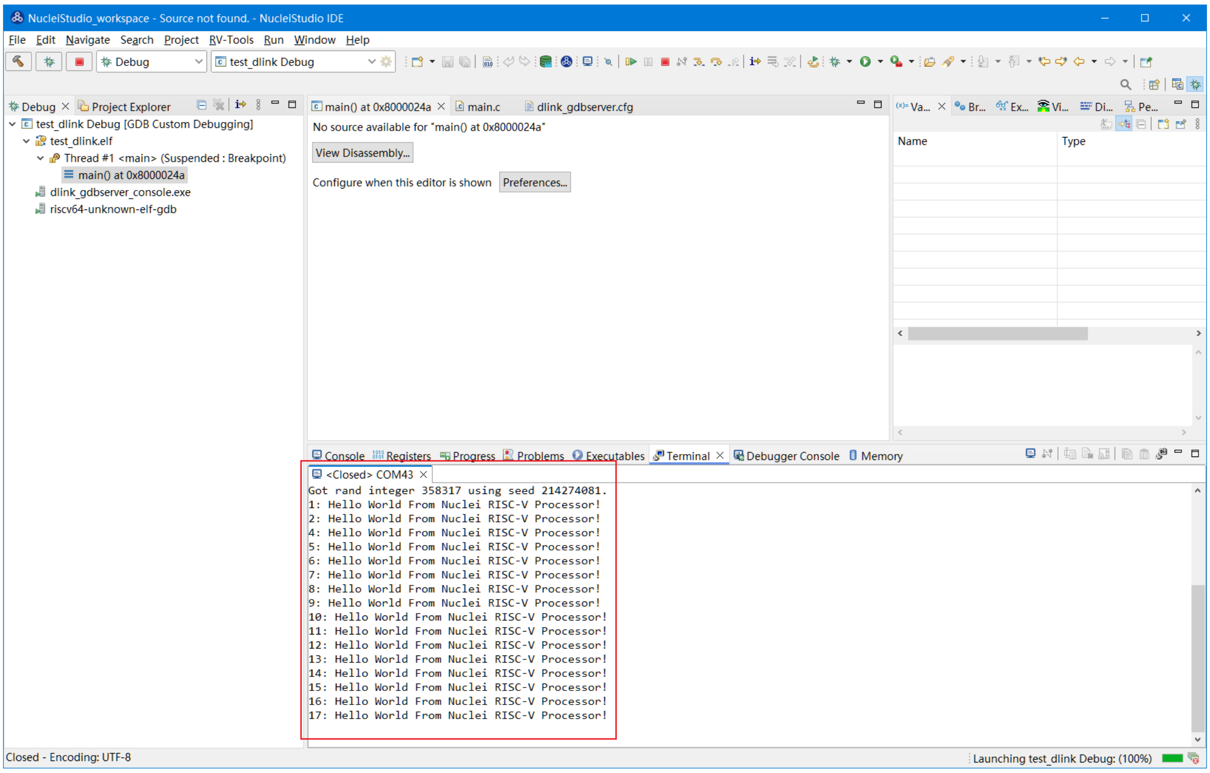
开始 Debug，如果配置正确，则在 Console 中有输出如下，并且 Dlink 亮绿灯。



通过串口工具，联接上另一个串口。



并可以查看到串口中有正确的内容输出，与预期一致，则 Dlink 可以正常调试，其他操作步骤与 OpenOCD 大体一致。



2.10 Nuclei Studio 高级功能

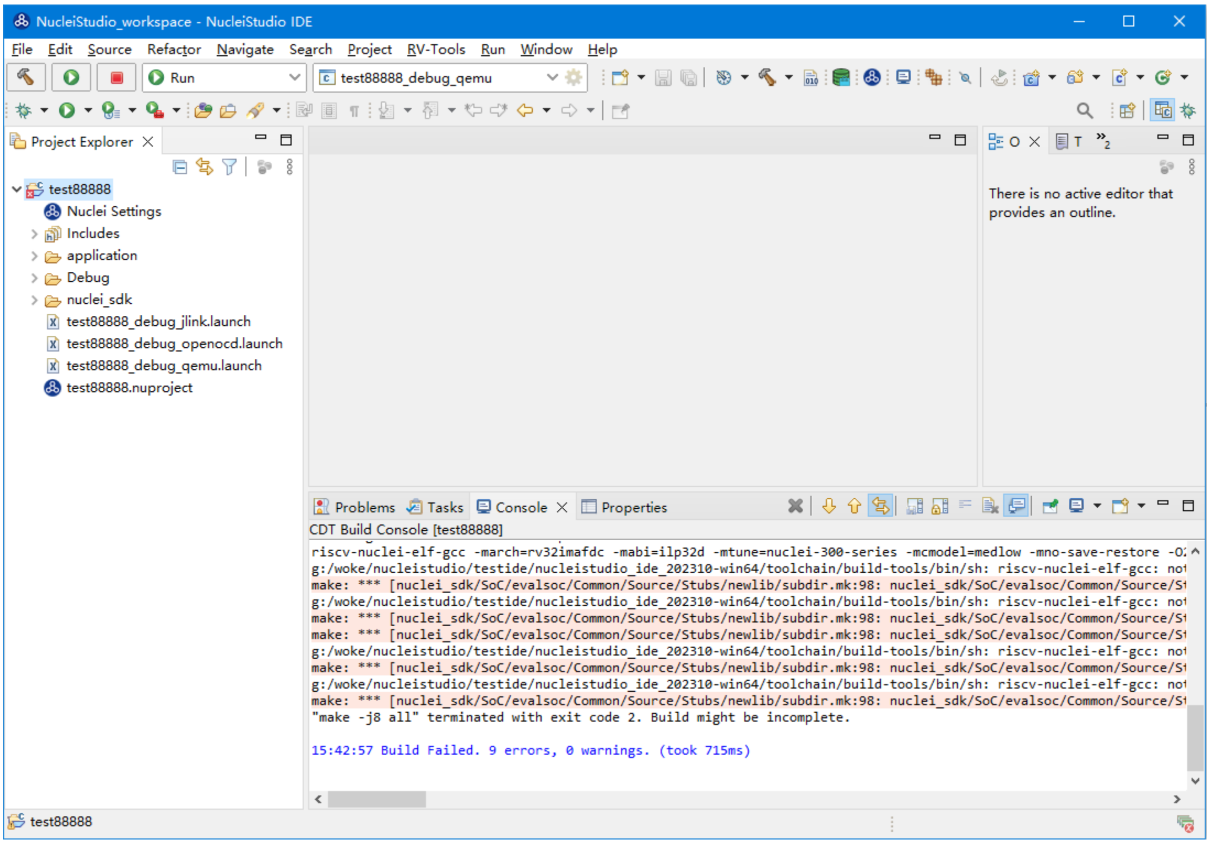
我们在 https://doc.nucleisys.com/nuclei_studio_supply 这里提供了很多且不断完善的 Nuclei Studio 和 Nuclei Tools 使用问题和案例，帮助你更好的使用 Nuclei Studio 以及 Tools，可以点进去了解下。

2.10.1 导入旧版本创建的工程

Nuclei Studio 2023.10 版导导入旧工程

在 Nuclei Studio 2023.10 版本中，因为工具链、sdk 等增均做了较大的修改，如果用户在新的 Nuclei Studio 中想要使用旧版的 Nuclei Studio 创建的工程，或者使用旧的 sdk，需要参考本章节内容进行操作。

将旧的 Nuclei Studio 中的工程导入到 Nuclei Studio 2023.10 中时（具体导入工程的方法，可以阅读*Nuclei Studio 2022.12 之后版本导入旧工程* (page 192)），或者使用旧的 sdk（旧的 sdk 指的是在 Nuclei Studio 2023.10 发布之前所发布的 sdk）所创建的工程，因为工程配置使用使用的是 gcc 10，当找不到对应的工具链，会出现编译报错等问题，导致工程无法正常使用。

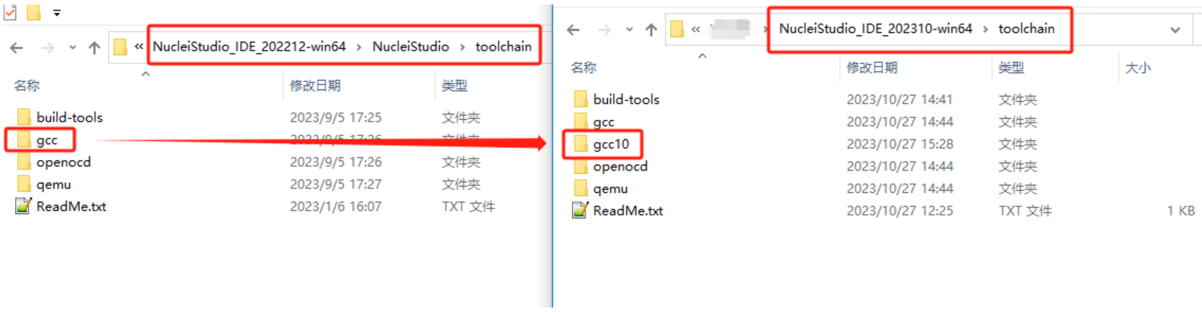


我们提供了两种解决方案，第一种方案是手动导入 gcc 10 工具链，第二种方案是通过 Nuclei Studio 2023.10 所带的转换工具进行转换。在此推荐使用第二种方案，能比较好的将工程转换成 Nuclei Studio 2023.10 所支持的工程，并能使用其最新特性。

手动导入 GCC 10 工具链

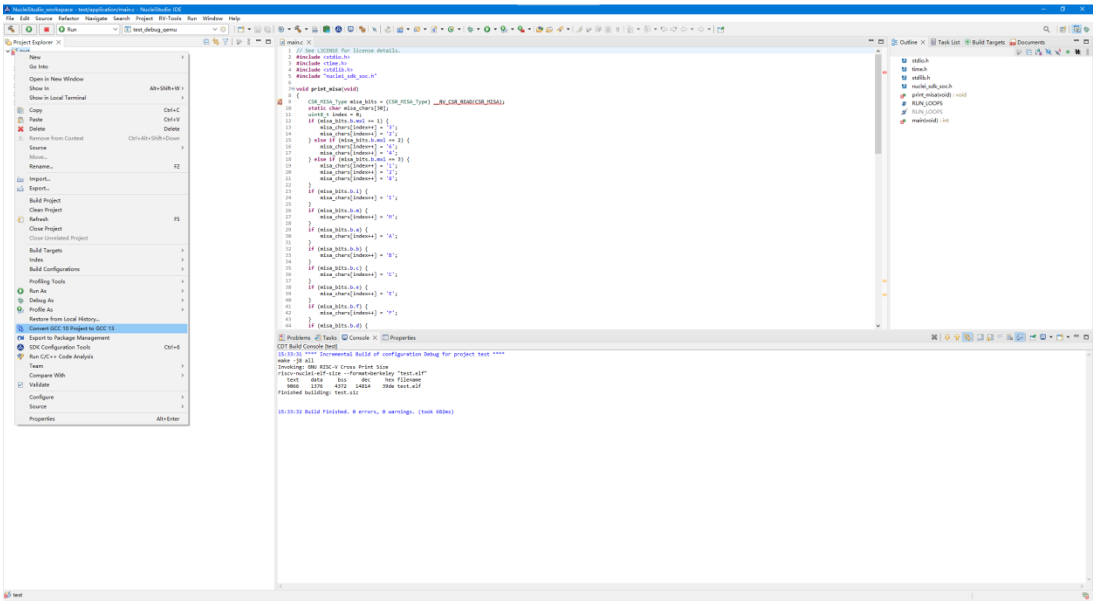
为了方便用户使用，在 Nuclei Studio 2023.10 中保留了 GCC 10 相关的配置，但没有将 GCC 10 打包到 IDE 中，如果用户想要继续在 GCC 10 下进行开发，可以手动将 GCC 导入进来。

首先，在旧版 Nuclei Studio 的安装目录中的 `toolchain\gcc\` 目录找到 GCC 10，并将其中内容复制到 Nuclei Studio 2023.10 的安装目录下的 `toolchain\gcc10\` 内。然后重新编译工程，工程可以正常编译，但这种方法，Nuclei QEMU 是无法正常使用，如果有 Nuclei QEMU 需求的项目，需要收到修改下 QEMU 对应的调试配置。

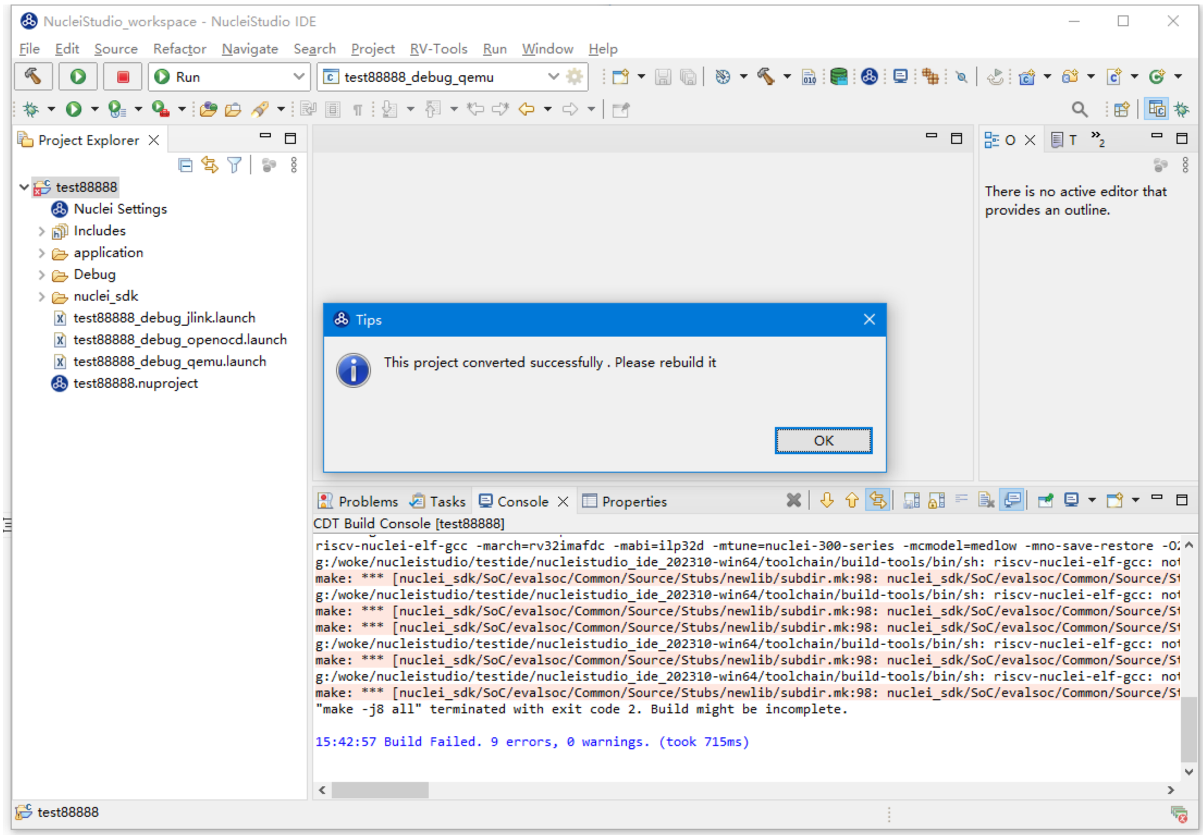


通过工具将工程转换成支持 GCC 13 的工程

为了方便用户导入旧的工程，并能正常使用 Nuclei Studio 2023.10 特性，我们提供了快速转换工具 `Convert GCC 10 Project to GCC 13`，选中工程点击鼠标右键，在弹出的菜单中找到 `Convert GCC 10 Project to GCC 13` 并点击。

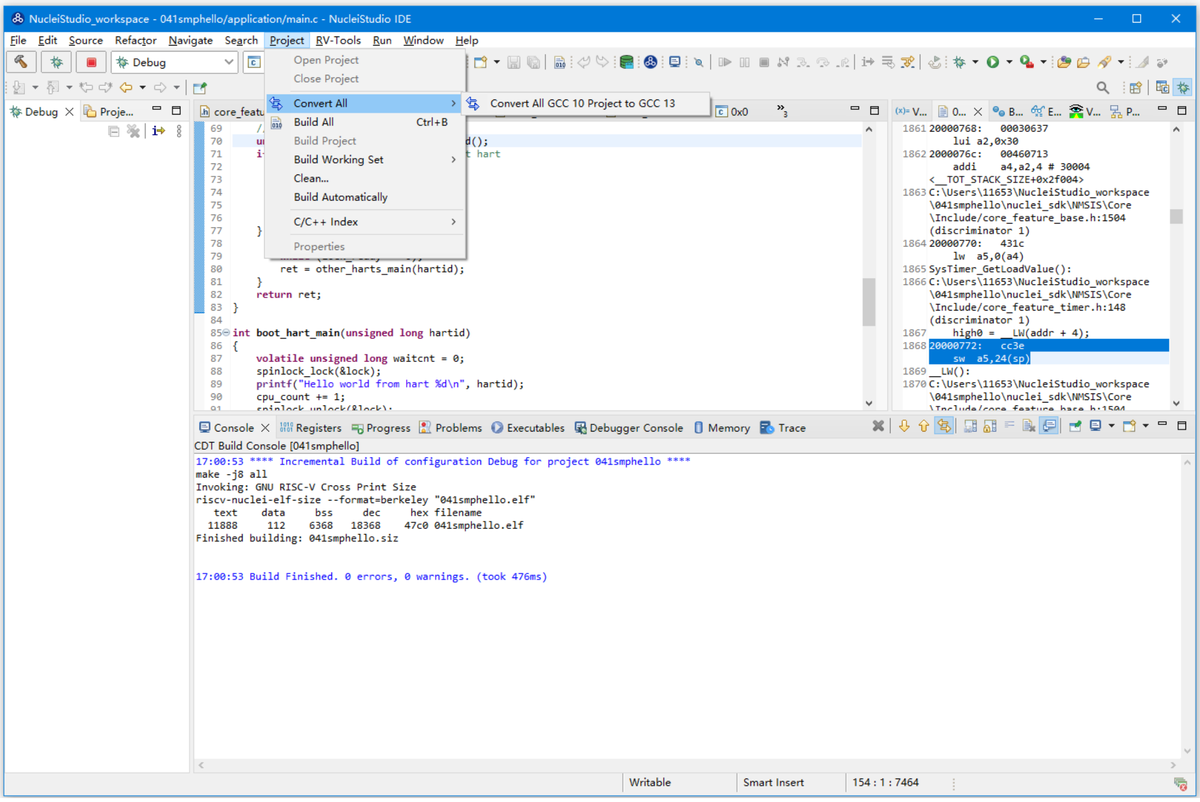


工程转换成功后，重新编译工程，此时 Nuclei Studio 将调用 GCC 13 编译工程，并且 QEMU 等功能也能正常使用。

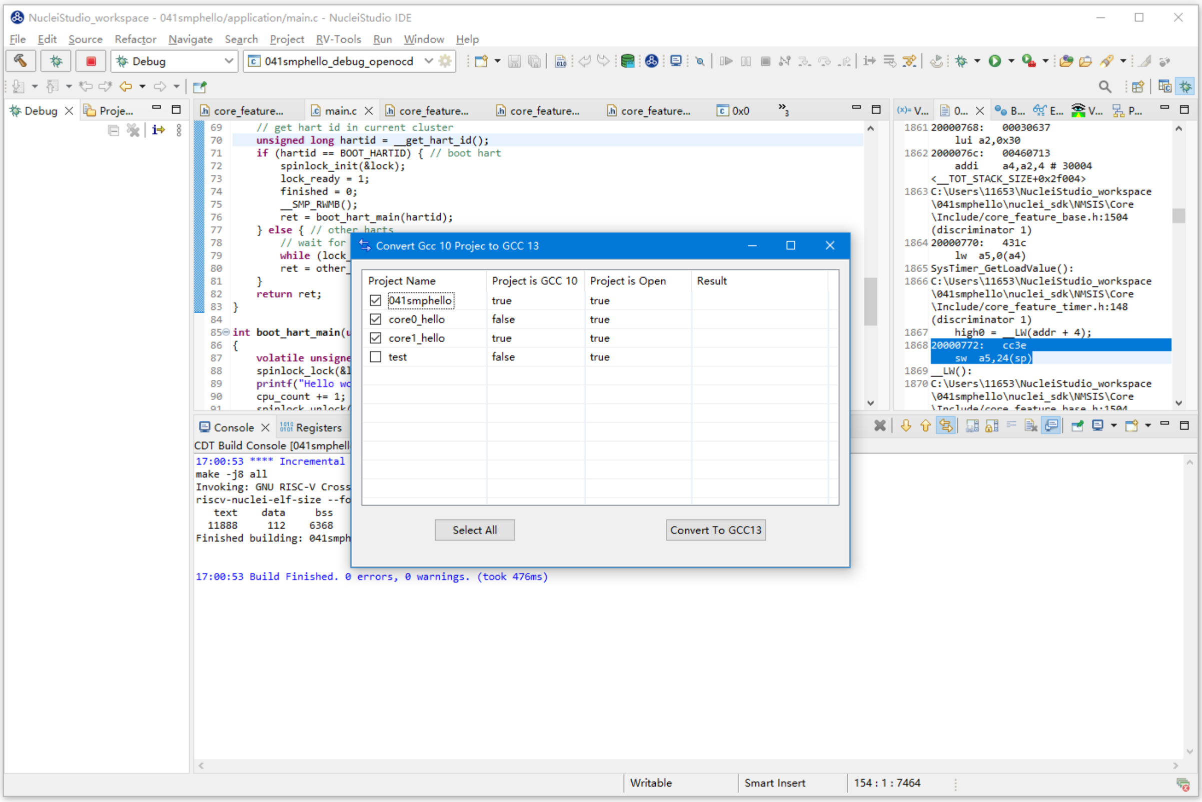


批量将工程转换成支持 GCC 13 的工程

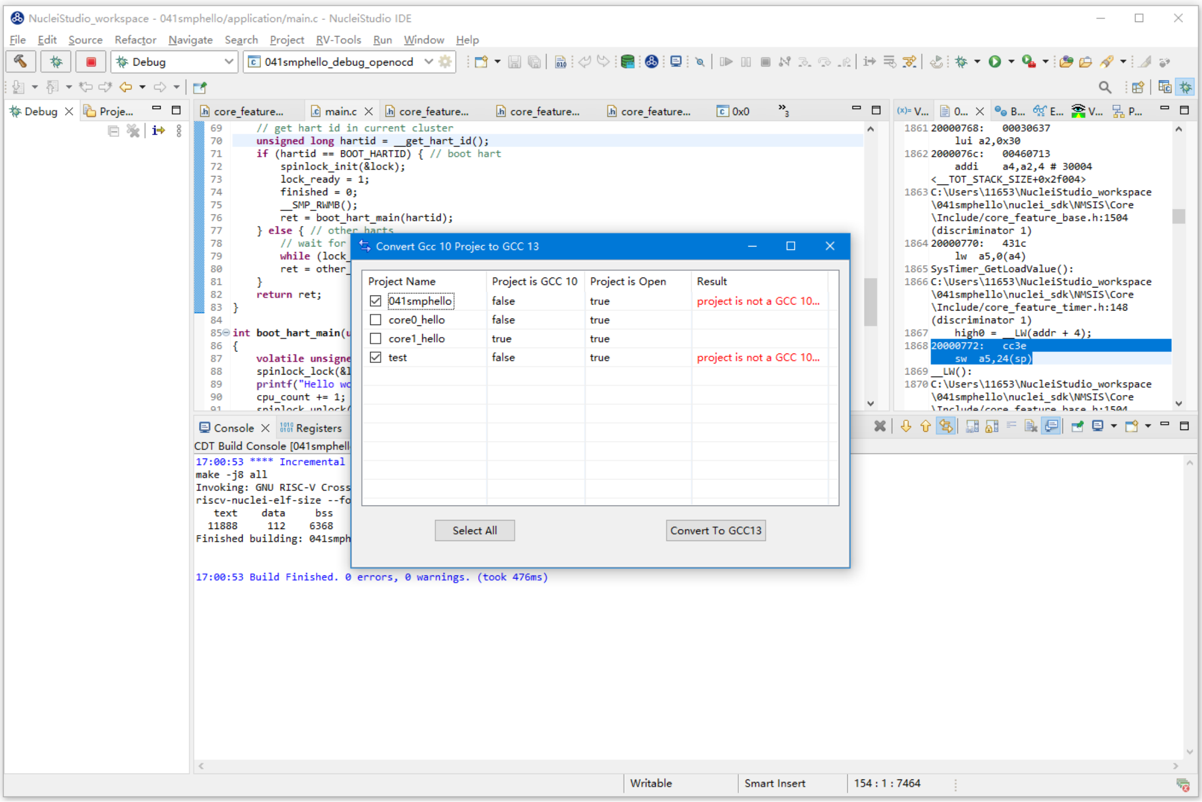
为了方便用户导入旧的工程，并能正常使用 Nuclei Studio 2023.10 以上版本的特性，批量将工程转换成支持 GCC 13 的工程。



打开转换工具，然后选择需要转换的工程，开始转换。



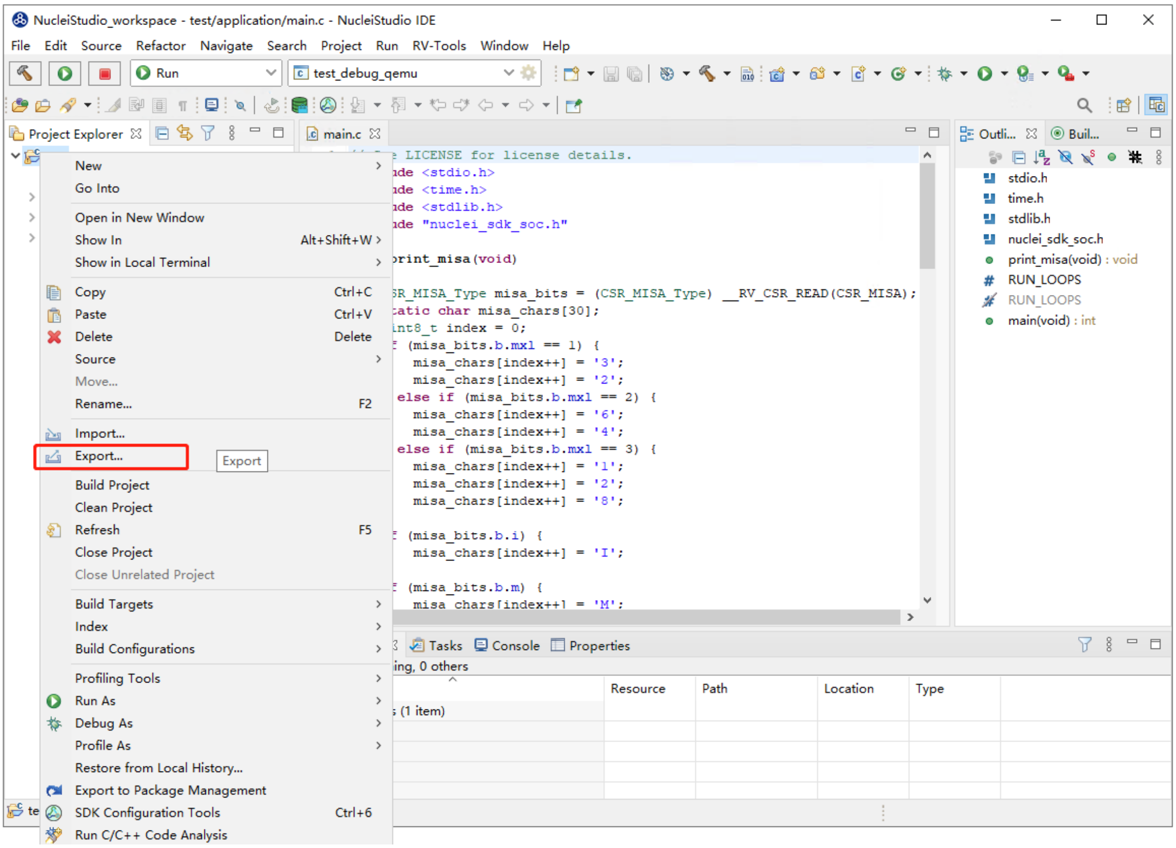
工程转换完成后，页面会显示工程转换成 GCC 13 后的结果。



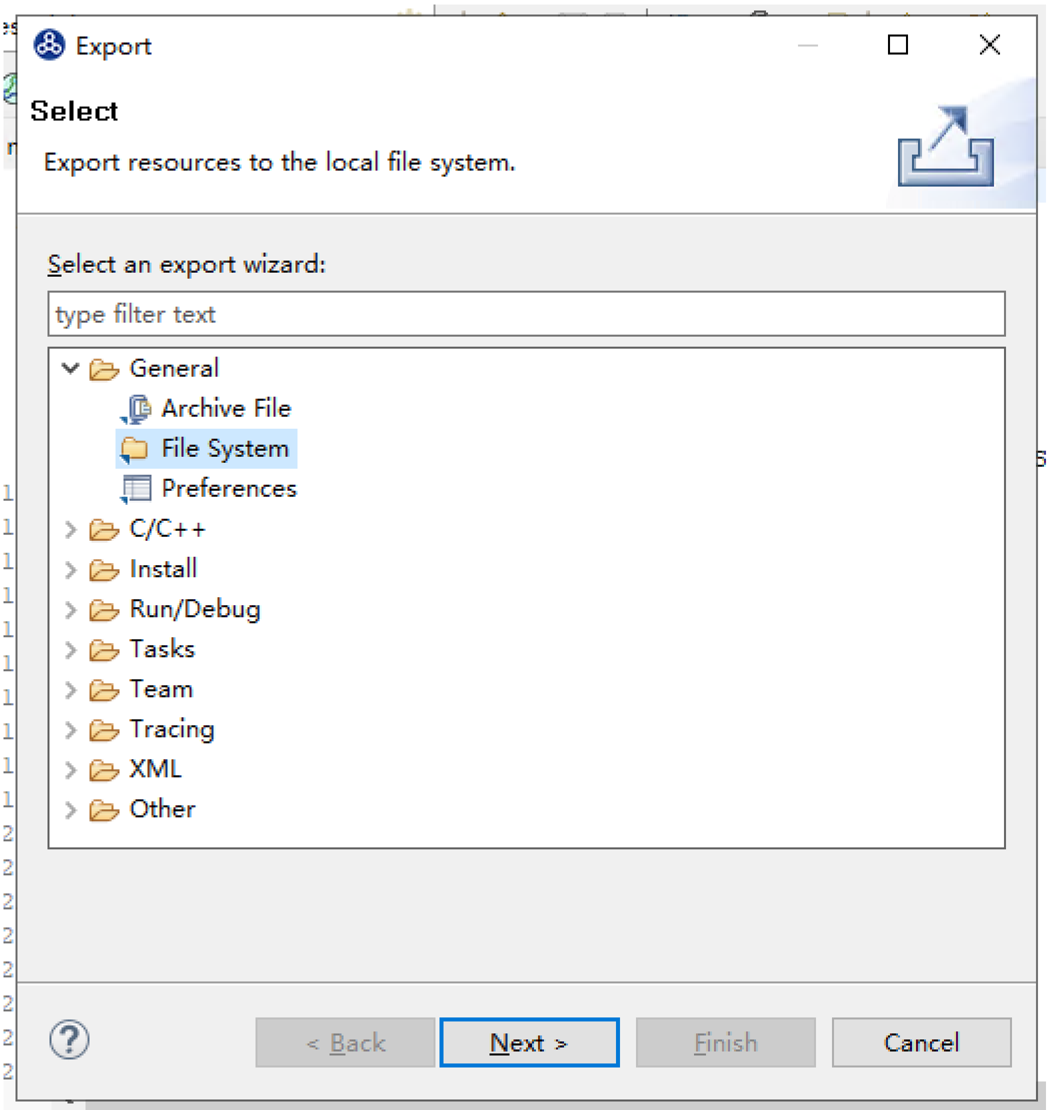
Nuclei Studio 2022.12 之后版本导入旧工程

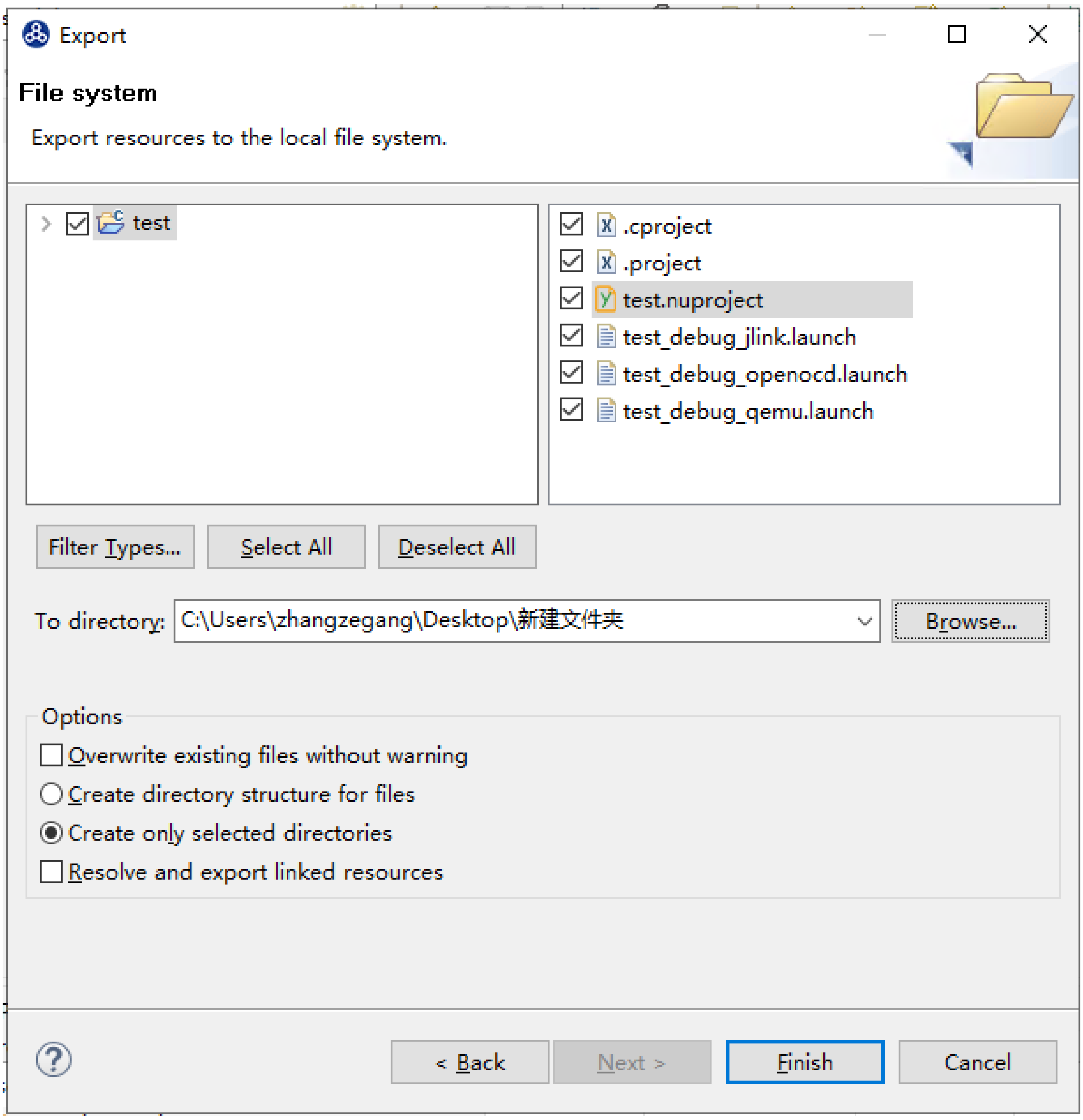
为了方便工程共享，Nuclei Studio 2022.12 版对工程的导入做了优化，在 IDE 中将一个工程导出后，可以双击打开 .nuproject 的方式，快速将工程在 Nuclei Studio 中导入并打开。

首先，在 Nuclei Studio 2022.12 版中导出一个工程，在需要导出的工程上右键，选中 Export。

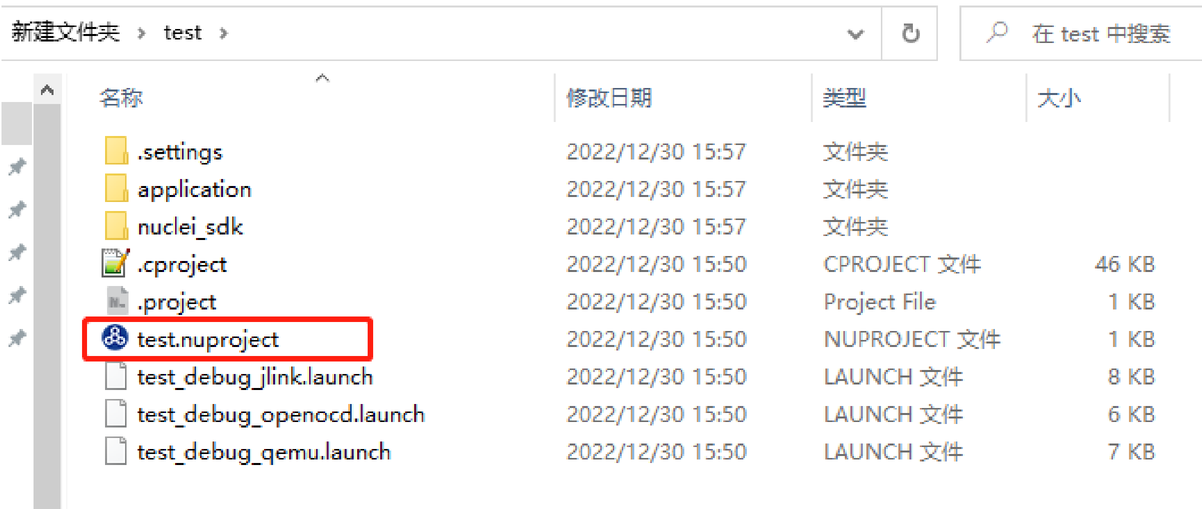


在弹出的 Export 对话框中 General->File System，按向导依次操作，可以把工程导出到指定文件夹。


Export
Select
Export resources to the local file system.
Select an export wizard:
type filter text
General
Archive File
File System
Preferences
C/C++
Install
Run/Debug
Tasks
Team
Tracing
XML
Other
< Back
Next >
Finish
Cancel



其次，导入工程。查看导出的工程，可以看到工程中有一个 test.nuproject 文件，点击这个文件，就可以将工程导入到 Nuclei Studio 中去了，具体的可以参考通过应用关联文件导入工程 (page 97)。

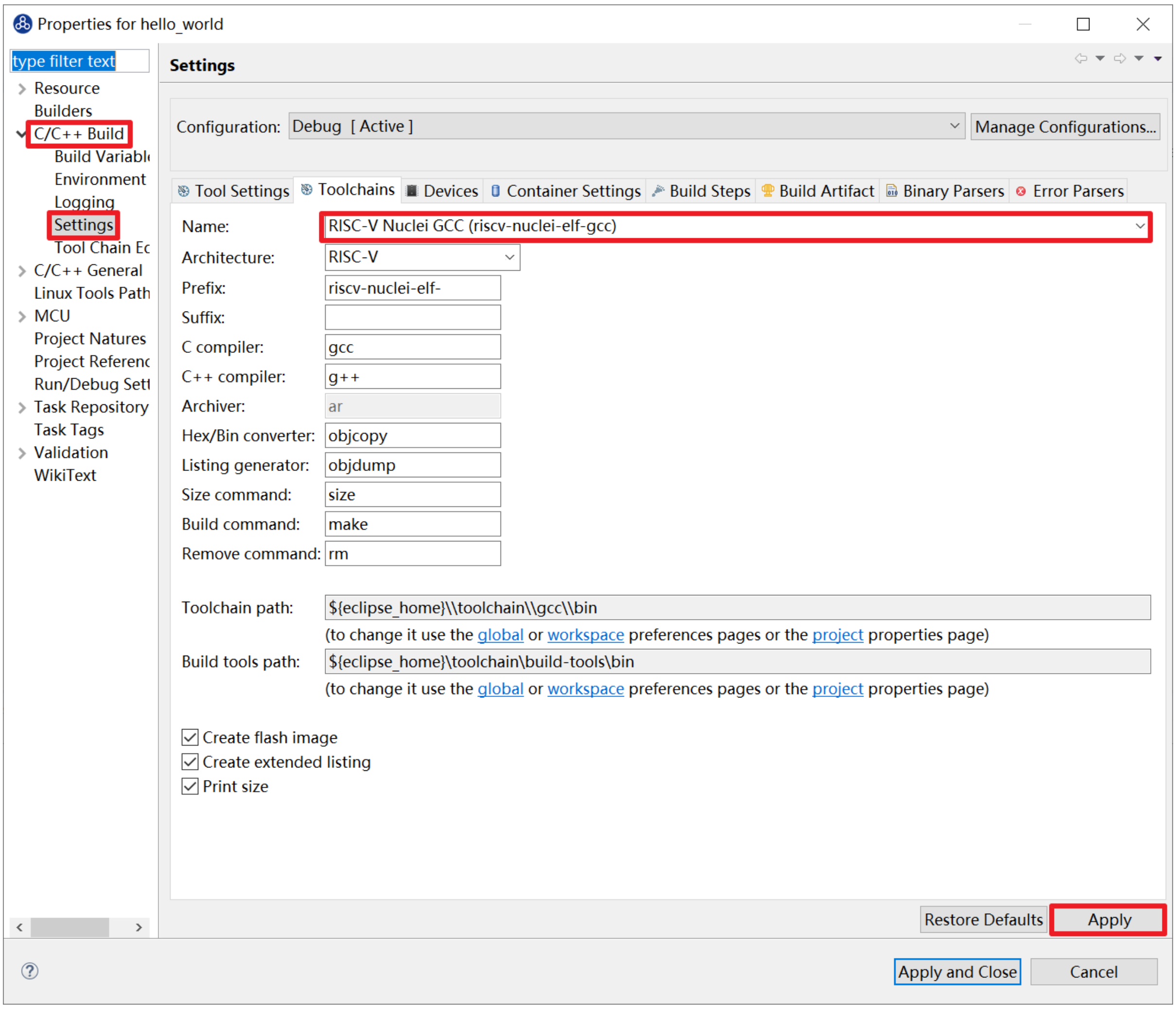


Nuclei Studio 2022.12 之前版本导入旧工程

Nuclei Studio 2023.10 版本工具链名称从 riscv-nuclei-elf-gcc 更名为 riscv64-unknown-elf-gcc，且 gcc 版本从 gcc10 升级到 gcc13。

Nuclei Studio 从 2020.08 版本开始，官方工具链从 RISC-V Nuclei GCC (riscv-nuclei-elf-gcc) 升级到 GNU MCU RISC-V GCC (riscv-none-embed-gcc)，因为编译前缀发生变化，所以使用 201909 及其之前版本的 IDE 生成的工程，经过调整设置后才可以在新版本 IDE 中使用。这里以 201909 版本的 Nuclei Studio 生成的 helloworld 工程为例，其导入及修改设置的详细步骤如下：

- 导入 201909 版本生成的 helloworld 工程，详细的导入方式请参考 5.2 节，这里不做赘述。
- 导入工程后右击选择 Properties 打开设置页面，选择 C/C++ Build ?? Settings，打开 Toolchains 栏目然后修改 Name 下拉选项为 RISC-V Nuclei GCC (riscv-nuclei-elf-gcc)。修改后点击 Apply 保存修改。

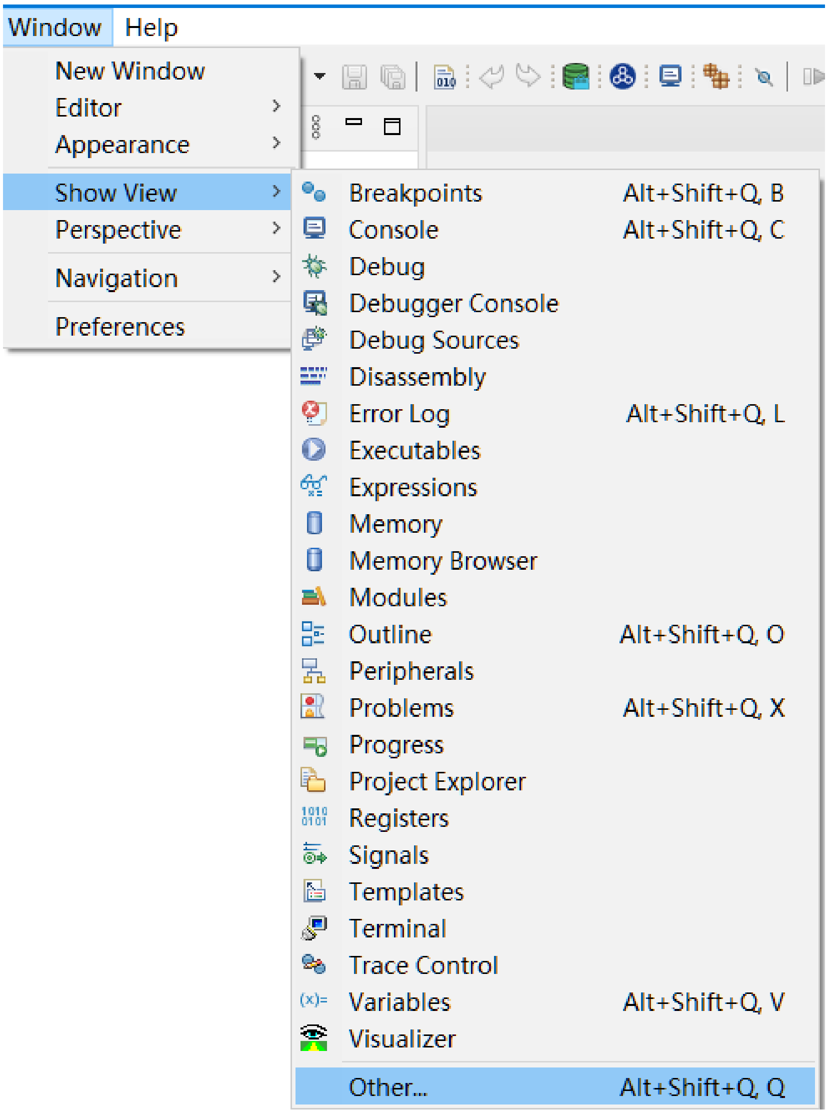


- 修改后右击工程选择 Clean Project 再选择 Build Project 即可。

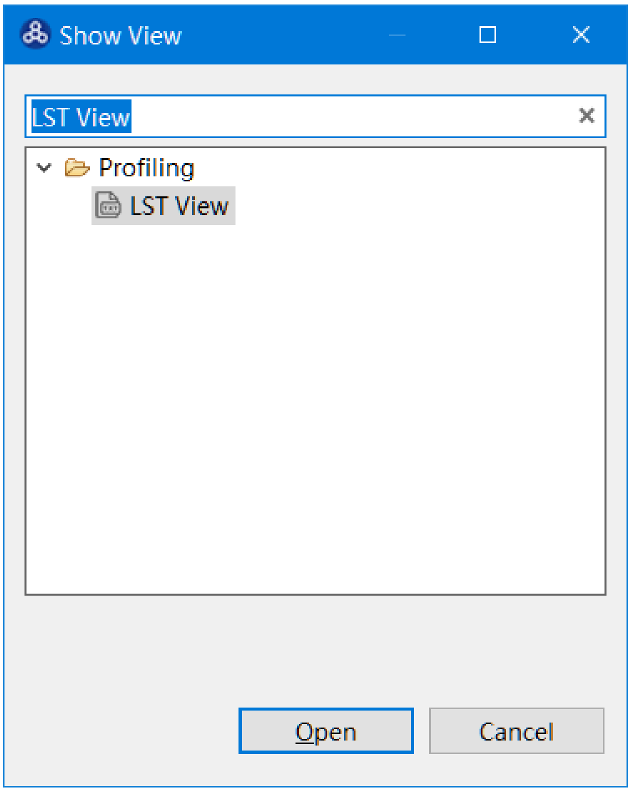
2.10.2 LST View

在 Nuclei Studio 2024.06 版本中，集成了 LST View 工具，LST View 可以单独使用，也可以在 Trace 工具或 GProf 工具中被呼起，主要功能，是帮助用户方便的查看 LST 文件，并实现 LST 文件与源码的联动。

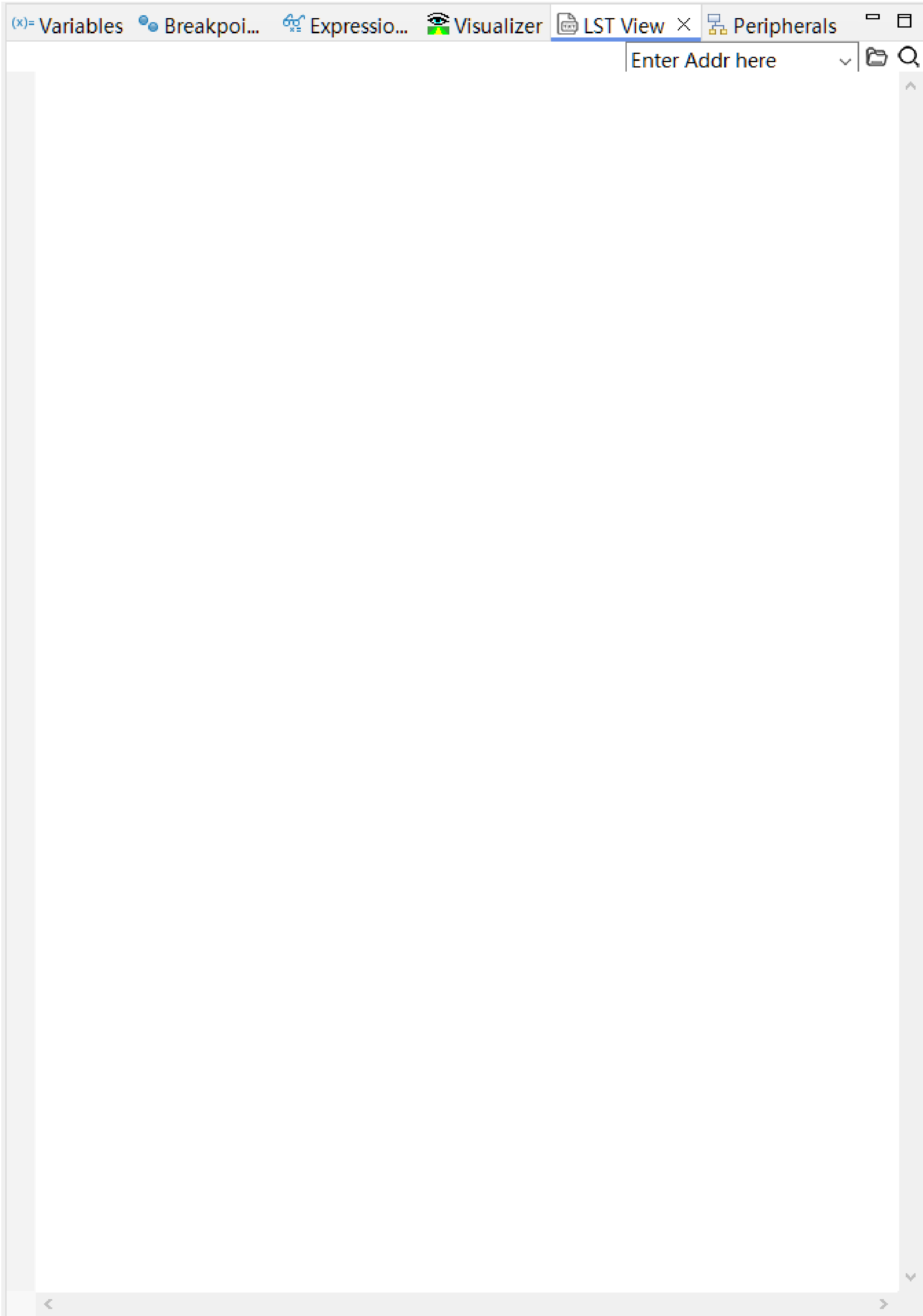
在 Nuclei Studio 中依次 `Window -> Show View -> Other`，在弹出的 Show View 中搜索 `LST View`。



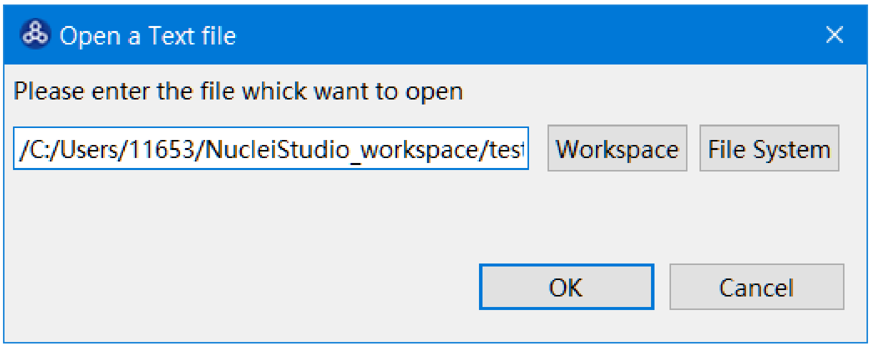
打开 LST View 工具。



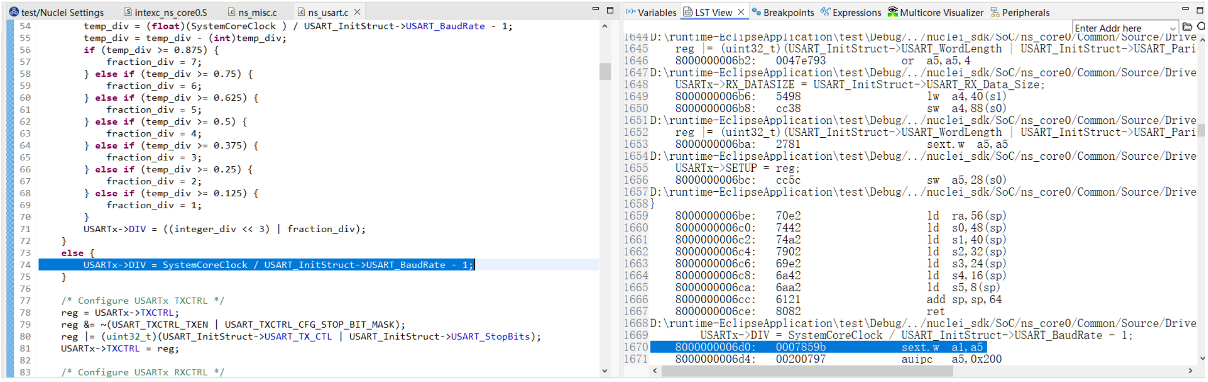
LST View 的顶部有一个工具栏，在工具栏中有搜索下接框，可以输入您想要搜索的 Addr；Open File 菜单，点击会弹出一个文件选择器，可以选择想要打开的 `.lst` 文件；Find 菜单，可以查找任意您想从 LST View 中查找的内容。



Open File 菜单，在弹出的文件选择器中，找到我们想要查看的 lst 文件，一般在工程编译后的 Debug 目录。



打开文件后，可以进行查找操作，同时，当我们选中某行文字，并且文字中包含一个正确的 Addr 时，LST View 会通过这个 Addr 定位到对应的源码所在的文件及行数，并通过程序打开对应的源码文件，并将光标定位到对应的行，通过 lst 文件反定位的源文件，实现两种文件的联动查看。



2.10.3 Code Coverage 和 Profiling 功能

在 Nuclei Studio 2023.10 版以上版本中，集成了Eclipse Linux Tools[8]，并对Eclipse Linux Tools[9]工具进行了部分优化，使其可以支持 Nuclei Studio 工程使用 Code Coverage 和 Profiling 相关功能。在 Nuclei Studio 2024.06 版本中对Eclipse Linux Tools[10]的功能做了进一步的优化和升级，使其更容使用。

关于 Coverage、Profiling 和 Call Graph 的使用教程请查看*Coverage*、*Profiling* 和 *Call Graph* 使用 (page 203)。

关于 Eclipse Linux Tools 的详细参见 Eclipse Linux Tools[11]

在使用过程，如有问题，可以查看 https://github.com/Nuclei-Software/nuclei-studio 相关内容，也可以向我们提交相关 issue。

Note

在 芯来科技视频号中有 如何在 **Nuclei Studio** 中使用 **Code Coverage** 和 **Profiling** 功能的视频，您可以在微信中搜索 芯来科技视频号点击查看相关内容。

[8] https://github.com/eclipse-linuxtools/org.eclipse.linuxtools/blob/master/RELEASE_NOTES.md#eclipse-linux-tools-release-notes
[9] https://github.com/eclipse-linuxtools/org.eclipse.linuxtools/blob/master/RELEASE_NOTES.md#eclipse-linux-tools-release-notes
[10] https://github.com/eclipse-linuxtools/org.eclipse.linuxtools/blob/master/RELEASE_NOTES.md#eclipse-linux-tools-release-notes
[11] https://github.com/eclipse-linuxtools/org.eclipse.linuxtools/blob/master/RELEASE_NOTES.md#eclipse-linux-tools-release-notes

关于 Code Coverage 功能

Nuclei Studio 中的 Code Coverage 功能是借助于 gcc 编译器提供 gcov 工具来查看指定源码文件的代码覆盖率，可以帮助开发人员确定他们的测试用例是否足够充分，是否覆盖了被测代码的所有分支和路径。

在 Nuclei Studio 中，通过给工程中的文件或者文件夹添加 `--coverage` 或者 `-coverage` 编译选项编译，在实际开发板上运行时，可以配合 semihost 功能实现文件读写到主机电脑上，就可以收集到需要的 coverage 文件 (gcda/gcno 文件)，或者通过 Nuclei SDK 提供的 profiling 库[12] 来实现将 coverage 数据打印到串口上，然后通过 IDE 来解析并保存到主机上。

> **Note**
>
> 注意：此处只需要将编译选项 `--coverage` 或者 `-coverage` 加到特定的应用目录或者源码文件上，而不能加到整个工程，否则在程序运行时将会消耗大量内存，导致运行失败。

- `.gcno` 文件是在使用 GCC 编译器的 `-ftest-coverage` 选项编译源代码时生成的。它包含了重构基本块图和为块分配源代码行号的信息。
- `.gcda` 文件是在使用 GCC 编译器的 `-fprofile-arcs` 选项编译的目标文件运行时生成的。每个使用该选项编译的目标文件都会生成一个单独的 `.gcda` 文件。它包含了弧转移计数、值分布计数以及一些摘要信息。

而一般情况下直接使用 `--coverage` 或者 `-coverage` 选项就可以让指示编译器产生上述文件，注意 `*.gcda` 文件是运行时产生的，也就是说需要实际运行的环境支持文件的读写才可以产生这样的文件，这里我们采用的是 semihost 技术，通过 openocd 的 semihost 功能，将文件写到主机上。

> **Note**
>
> 注意：进行 coverage 的时候，建议是使用 `O0` 编译，这样 coverage 的信息才会尽可能的准确。

关于 Code Coverage 的功能详细参见

- Gcov Intro (Using the GNU Compiler Collection (GCC))[13]
- Gcov Data Files (Using the GNU Compiler Collection (GCC))[14]
- Code Coverage for Embedded Target with Eclipse, gcc and gcov | MCU on Eclipse[15]

关于 Profiling 功能

Nuclei Studio 中的 Profiling 功能是借助于 gcc 编译器和 binutils 中的 gprof 工具，来查看指定文件中函数的运行时间和调用次数，以及调用关系。gprof 可以用来确定程序的瓶颈，以便进行性能优化。gprof 通过在程序运行时收集数据来工作，然后生成一个报告，该报告显示每个函数在程序中占用 CPU 时间的百分比以及函数之间的调用关系。

在 Nuclei Studio 中，通过带特定的编译选项 `-pg` 编译指定源码文件，在实际开发板上运行时，可以配合 semihost 功能实现文件读写到主机电脑上，就可以收集到需要的 coverage 文件 (gcda/gcno 文件)，或者通过 Nuclei SDK 提供的 profiling 库[16] 来实现将 coverage 数据打印到串口上，然后通过 IDE 来解析并保存到主机上。

> **Note**

[12] https://github.com/Nuclei-Software/nuclei-sdk/tree/master/Components/profiling
[13] https://gcc.gnu.org/onlinedocs/gcc/Gcov-Intro.html
[14] https://gcc.gnu.org/onlinedocs/gcc/Gcov-Data-Files.html
[15] https://mcuoneclipse.com/2014/12/26/code-coverage-for-embedded-target-with-eclipse-gcc-and-gcov/
[16] https://github.com/Nuclei-Software/nuclei-sdk/tree/master/Components/profiling

注意：此处只需要将编译选项 -pg 加到特定的应用目录或者源码文件上，而不能加到整个工程，否则在程序运行时将会消耗大量内存，导致运行失败。

产生这个 gmon.out 文件需要配合编译器并且实际上板运行，并且运行环境支持文件的读写，才可以进行有效的 Profiling 功能。

关于 Profiling 的功能详细参见

- Introduction (GNU gprof)[17]
- Using GNU Profiling (gprof) With ARM Cortex-M - DZone[18]

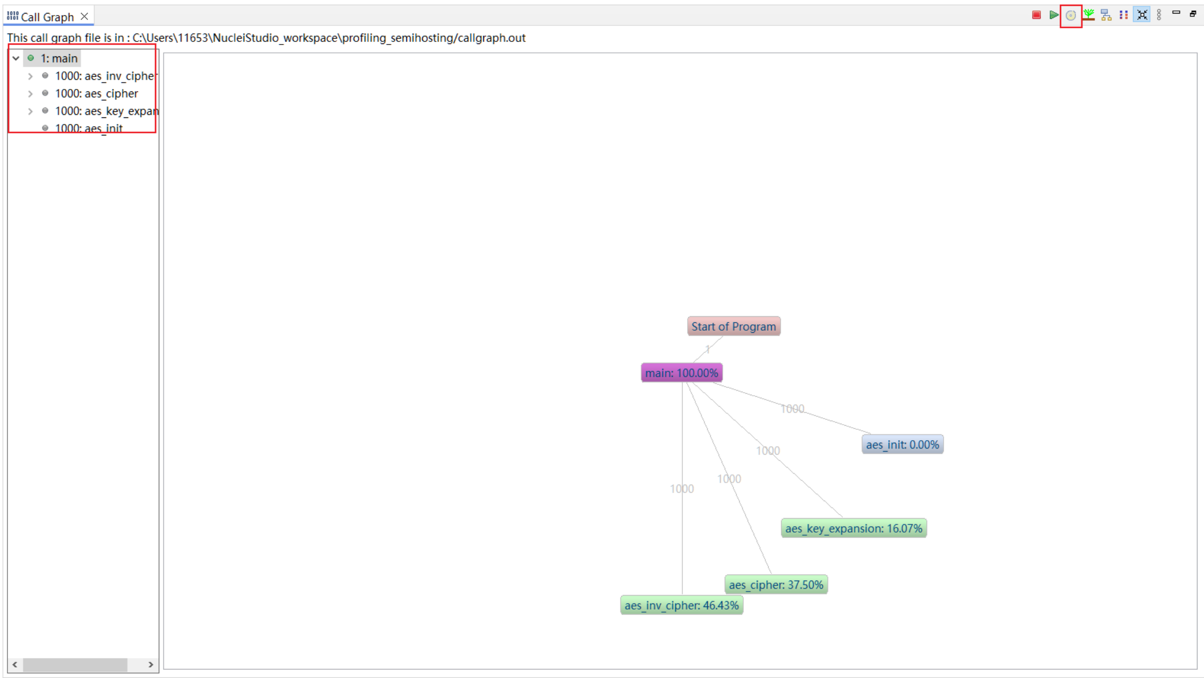
关于 Call Graph 功能

Call Graph（调用图）是一个强大的工具，它允许开发人员直观地理解程序中函数或方法之间的调用关系。通过 Call Graph，开发人员可以迅速识别出哪些函数被频繁调用，哪些函数是关键的入口点，以及函数之间的依赖关系。Nuclei Studio 中 Call Graph 主要是通过分析 Profiling 的数据，来获取到程序的调用关系。

在 NucleiStudio 中依次 Window -> Show View -> Other，在弹出的 Show View 中搜索 Call Graph，打开 Call Graph 工具。Call Graph 工具中提供了多处视图，其中常用到的视图有以下几个。

Radial View

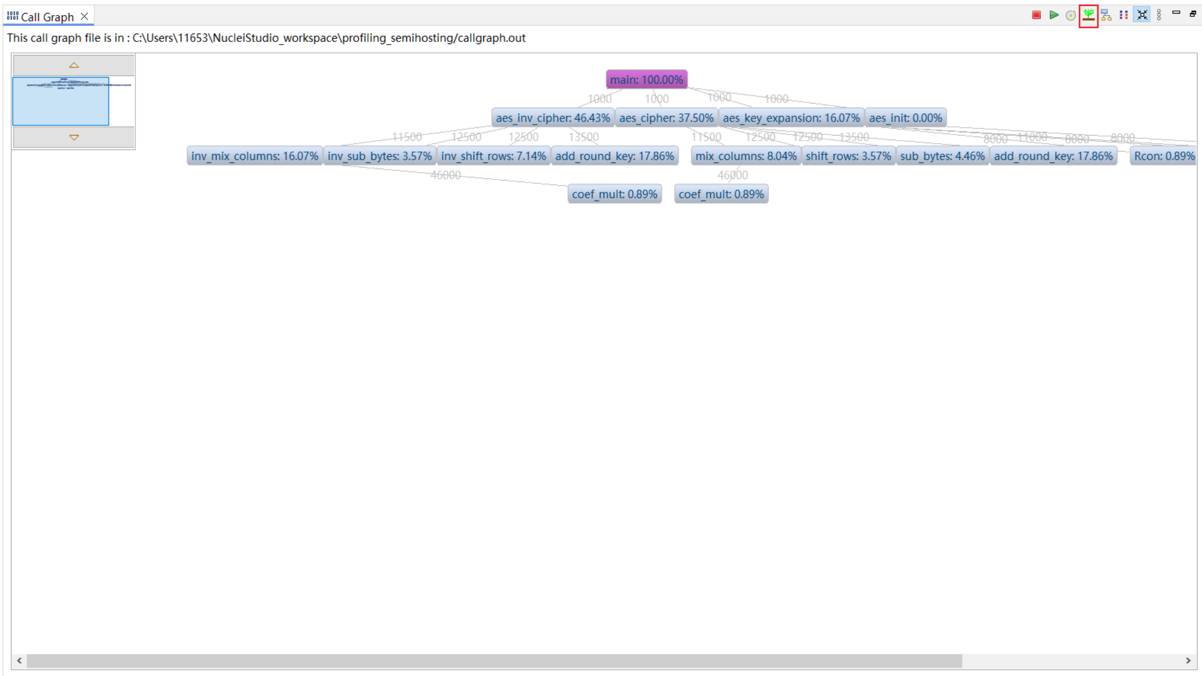
本视图中展示了程序的调用关系，在左侧的菜点中，双击选中某个父节点，在右侧的区域将显示以这个父节点开始的所有的调用关系，也可以通过菜单在其他视图中以不同的方式查看所选中的调用关系。



17 https://sourceware.org/binutils/docs/gprof/Introduction.html
18 https://dzone.com/articles/using-gnu-profiling-gprof-with-arm-cortex-m

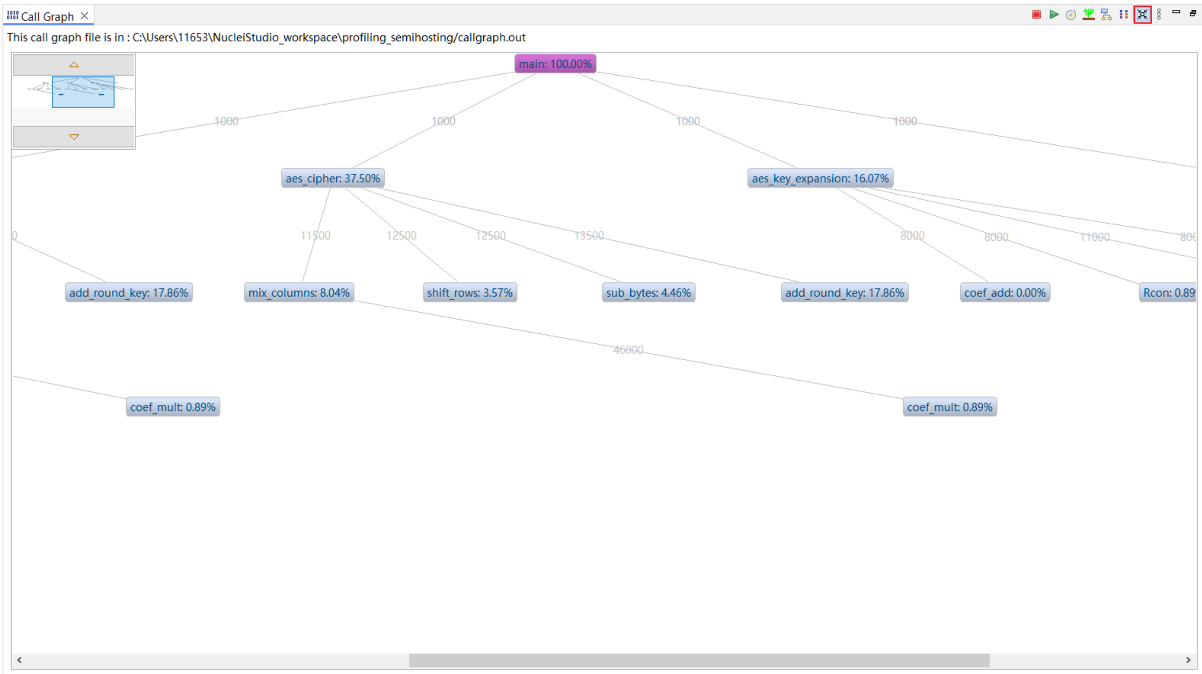
Tree View

展示了 Radial View 中所选中的程序的调用关系、耗时所占比率、调用次数等信息；选中某一个函数，可以查看到它的父节点以及子节点等信息。



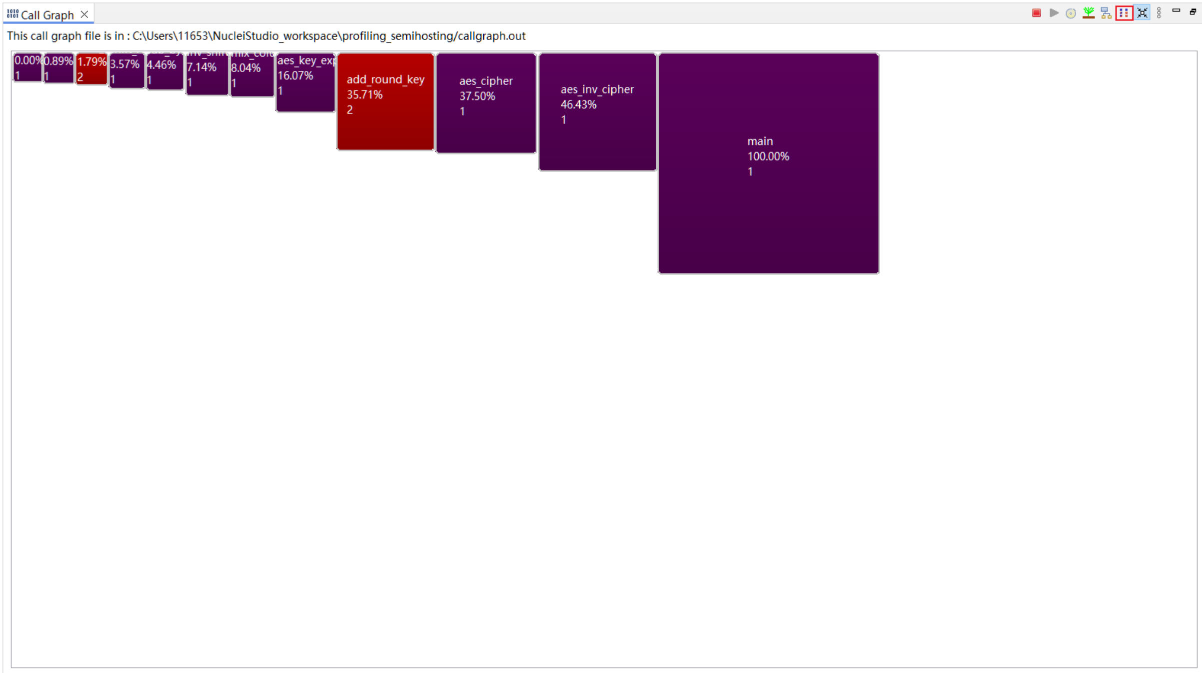
Level View

与 Tree View 有点类似，展示了程序的调用关系以及调用次数。



Aggregate View

以方图的方式，非常直观的展示了程序的耗时关系。



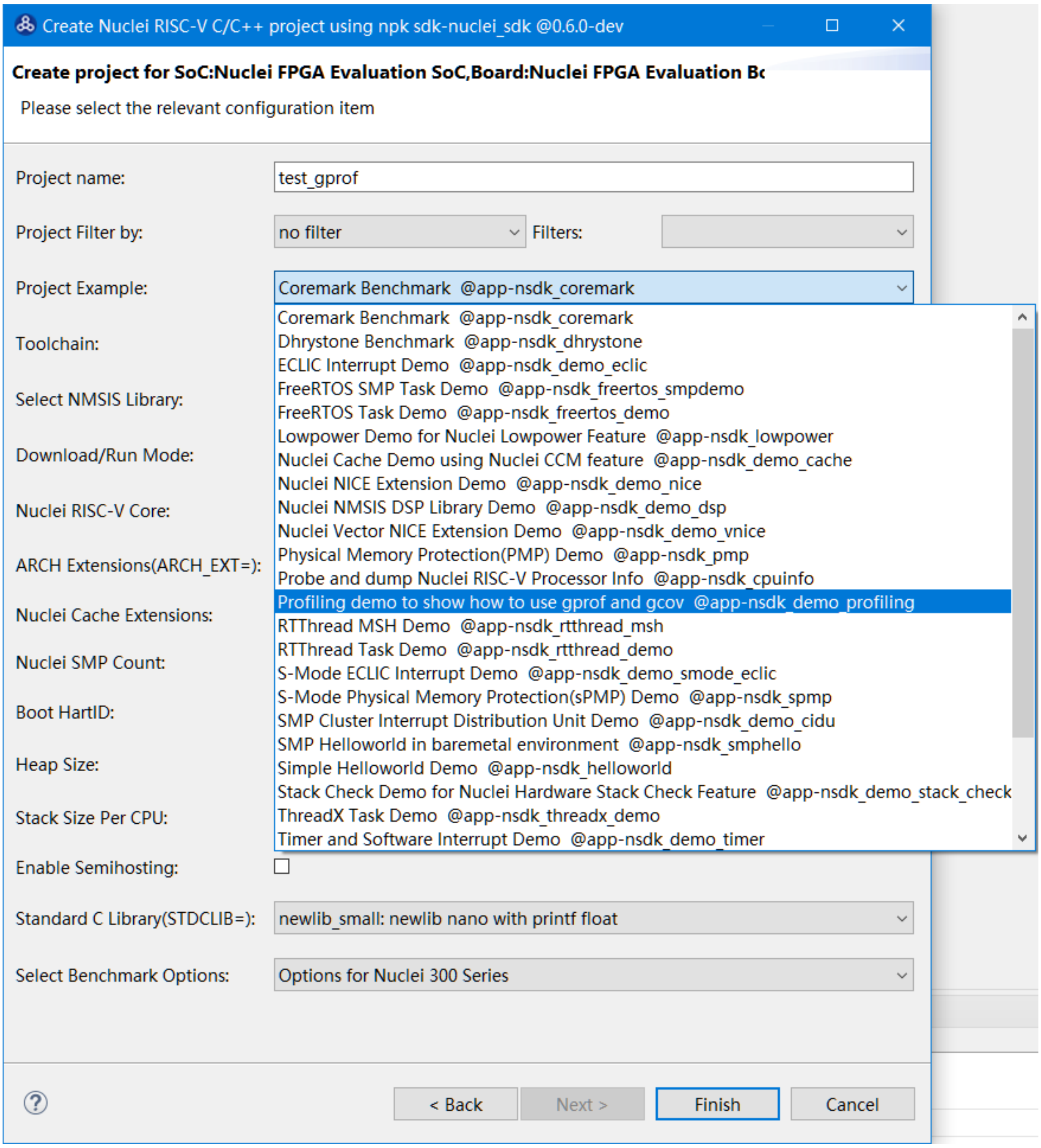
Coverage、Profiling 和 Call Graph 使用

在 NucleiSudio 2024.06 版中使用 Coverage、Profiling 和 Call Graph 方法很简单，下面以 NucleiSudio 2024.06、nuclei_sdk 0.6.0 为例，通过两种方式分别演示如何使用 Coverage、Profiling 和 Call Graph 工具。

通过串口使用

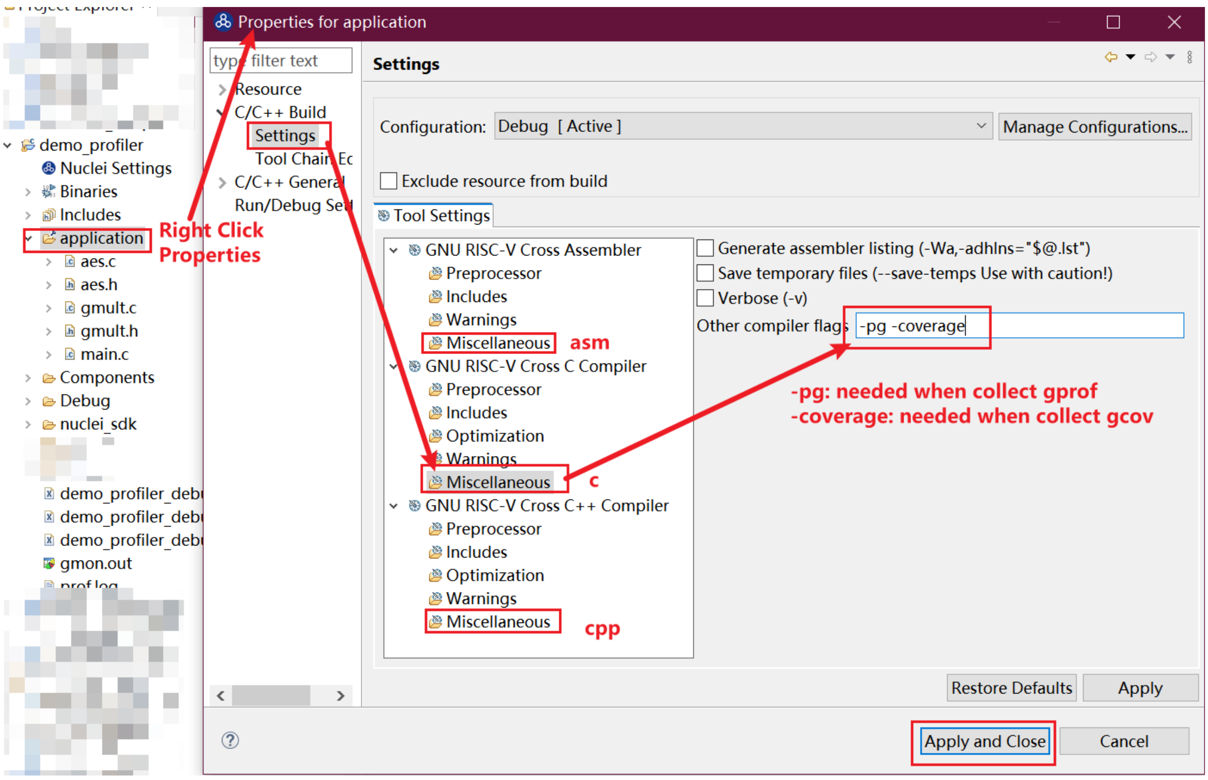
nuclei_sdk 0.6.0 及以上版本的 nuclei_sdk 中，包含一个 `Profiling demo to show how to use gprof and gcov` 测试工程，在 NucleiSudio 安装了 nuclei_sdk 0.6.0 后，可以创建此测试工程。关于 `Profiling demo to show how to use gprof and gcov` 测试工程, 可参考 demo_profiling[19]。

[19] https://doc.nucleisys.com/nuclei_sdk/design/app.html#demo-profiling

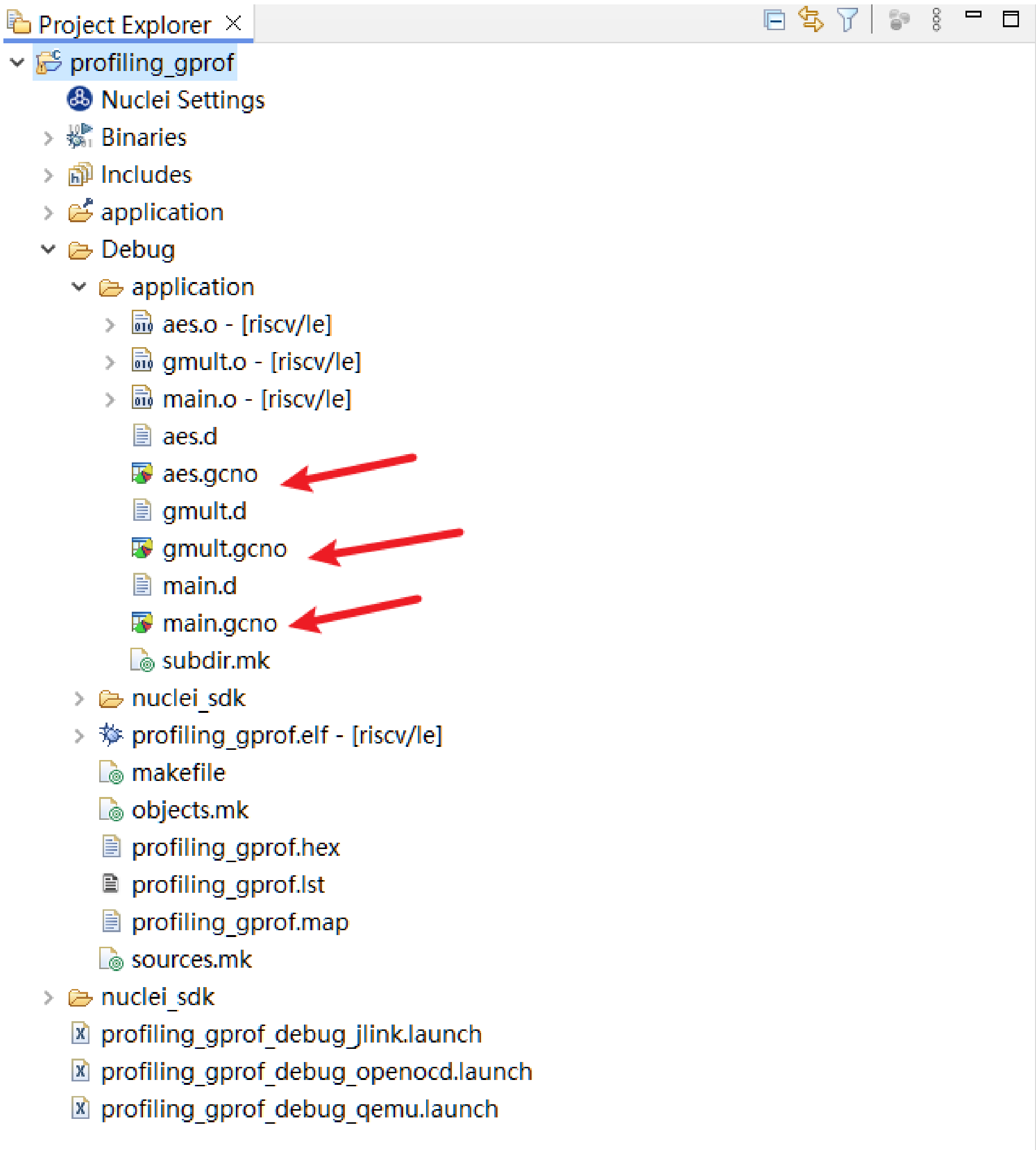


工程创建后，需要对想要进行代码分析的文件或文件夹设置一个 -pg 、--coverage 或者 -coverage 的编译选项，然后编译工程。

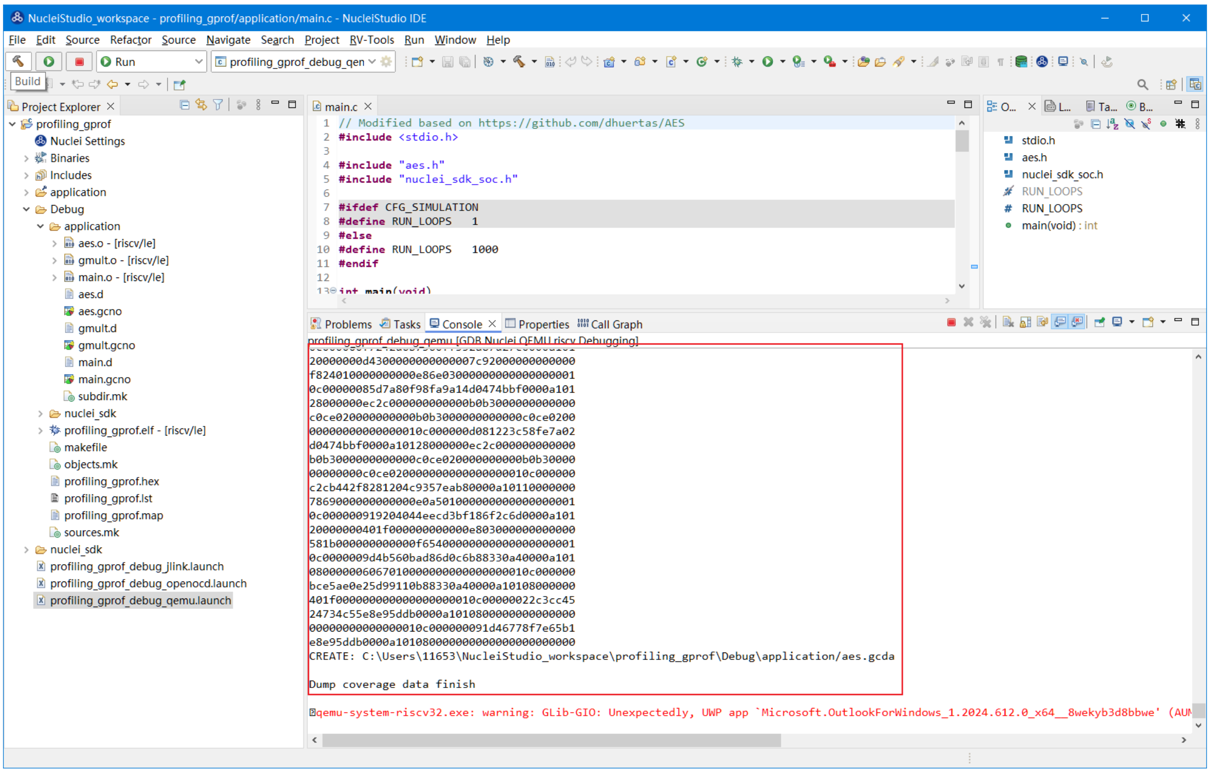
> **Note**
>
> 注意：此处只需要将编译选项 -pg 、--coverage 或者 -coverage 加到特定的应用目录或者源码文件上，而不能加到整个工程，否则在程序运行时将会消耗大量内存，导致运行失败。



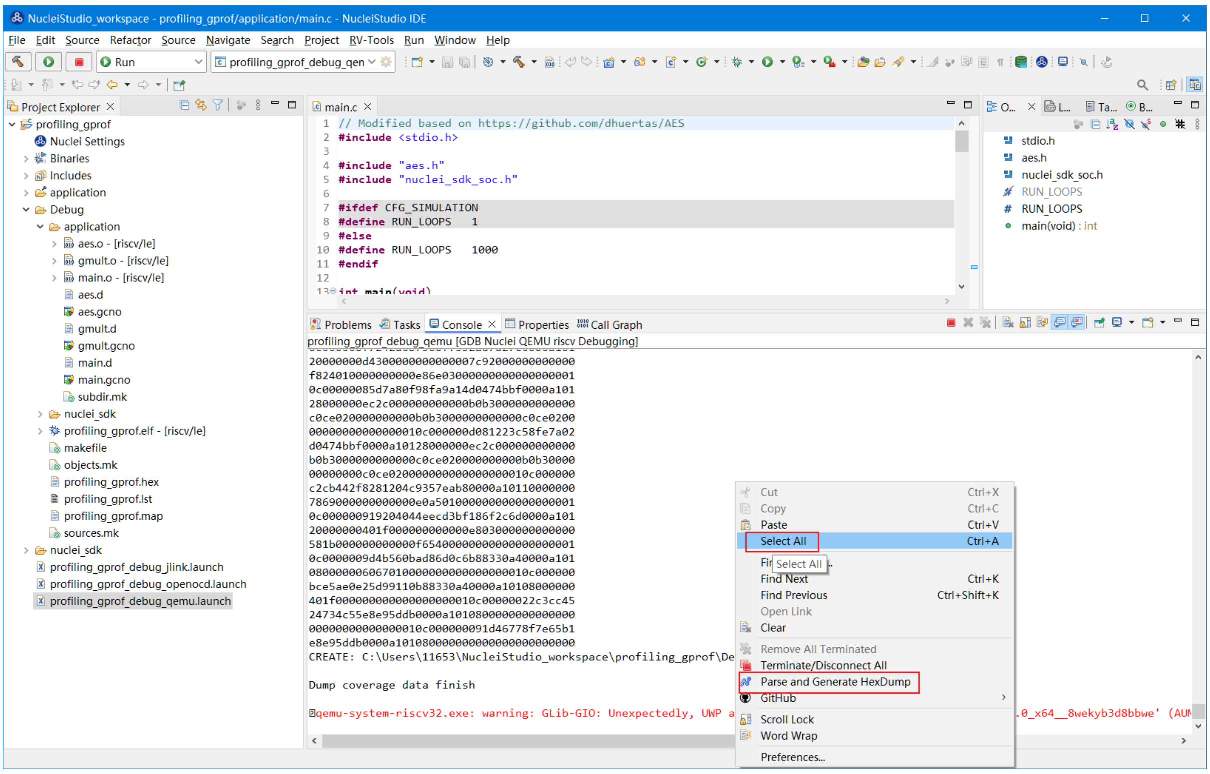
在编译通过的工程的 Debug 目录中，可以看到，已经生成了几个 .gcno 的文件。



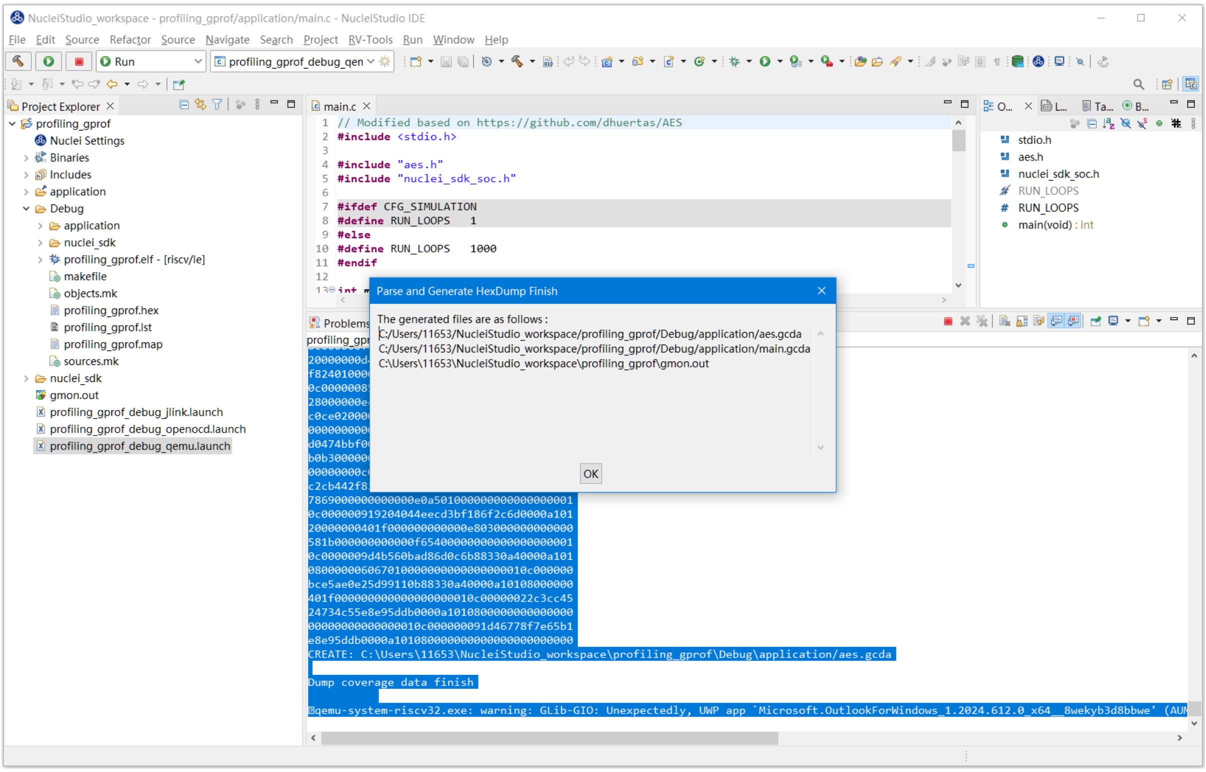
工程编译完后，可以运行或调试工程，我们可以选择在 QEMU 下进行，也可以调试实际的开发板。本例以 QEMU 为例进行运行程序，在 NucleiStudio 的 Console 窗口中可以看到 Profiling 信息输出，如果是在开发板上调试，则是在串口输出中可以找到 Profiling 信息输出。



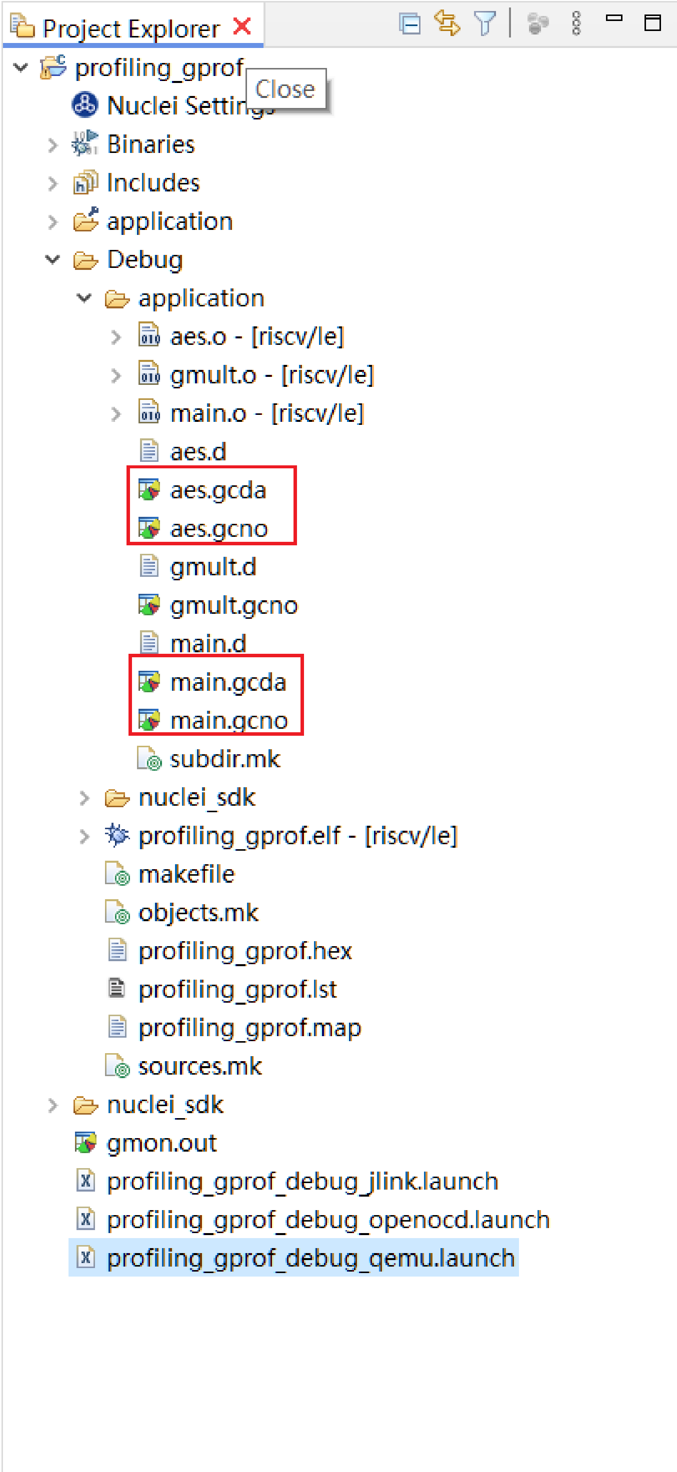
输出的 Profiling 信息需要解析后 NucleiStudio 才可以正确读取，在 Console 框内点击鼠标右键，然后在弹出的菜单中点击 Select All，来选中所有输出，再次击鼠标右键，在弹出菜单中选择 `Parse and Generate HexDump` 菜单。



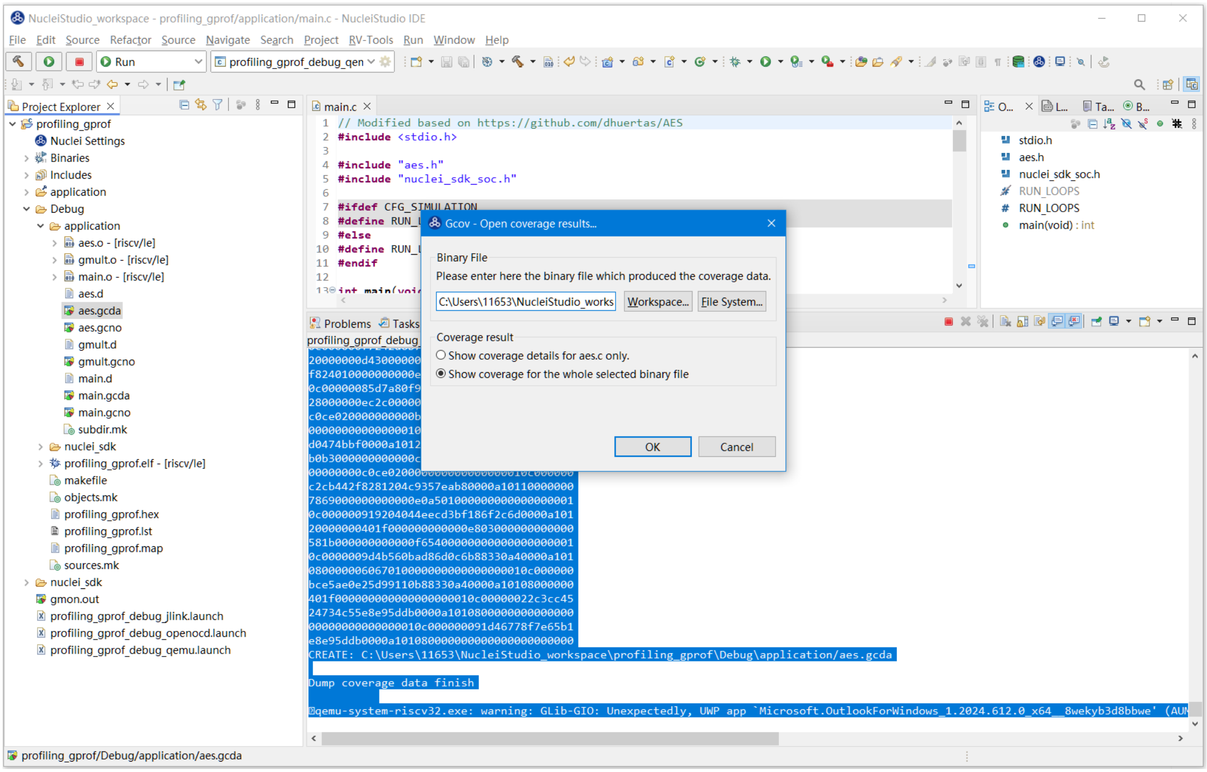
此时 NucleiStudio 会对输出的文件进行分析，并将结果分别存放在对应的文件中。



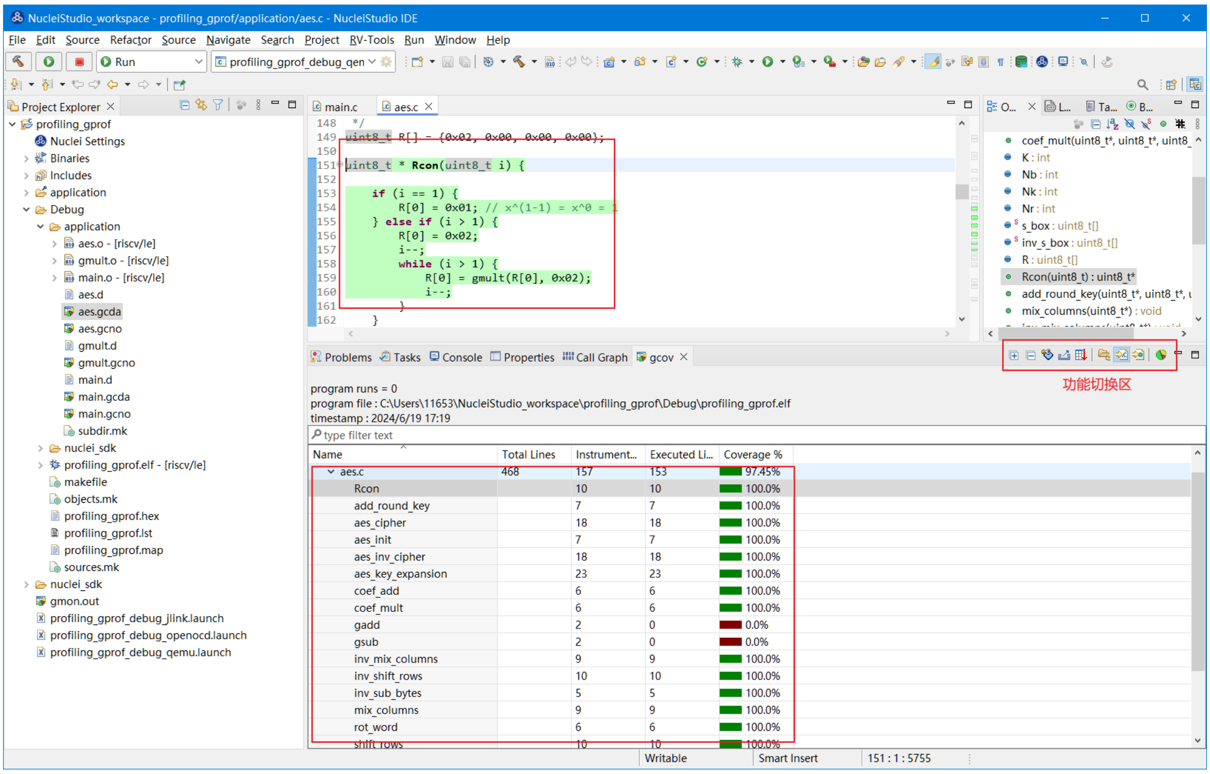
再次查看工程的 Debug 目录，可以看到产生了对应的 .gcda 文件。



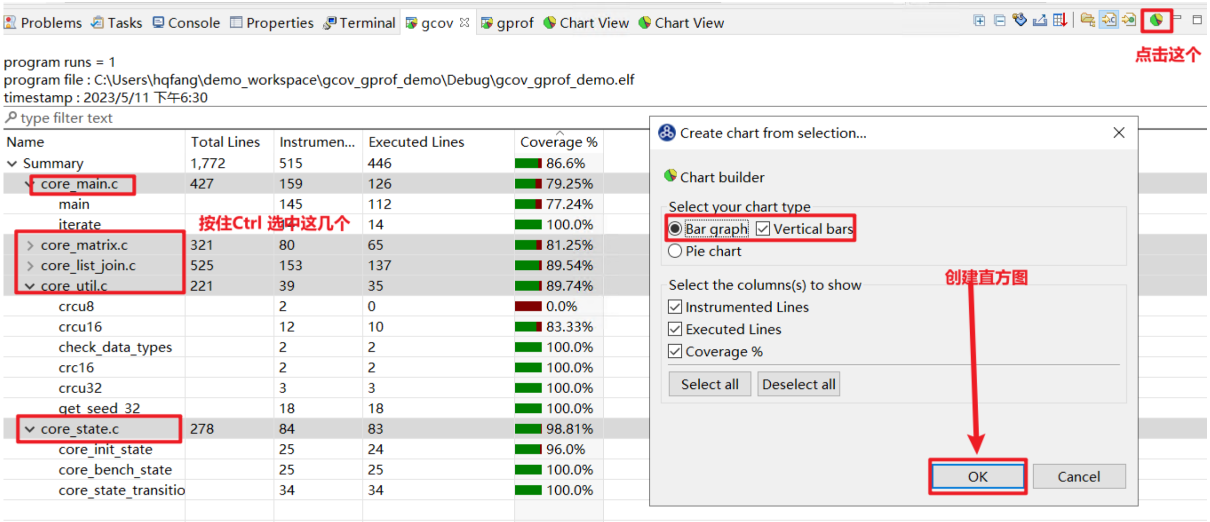
双击 `.gcda` 文件，打开 Gcov 工具，就可以看到对应用程序的分析结果，在结果中显示了某个文件或某个方法在程序执行过程中是否执行到，以及代码执行覆盖比等数据。



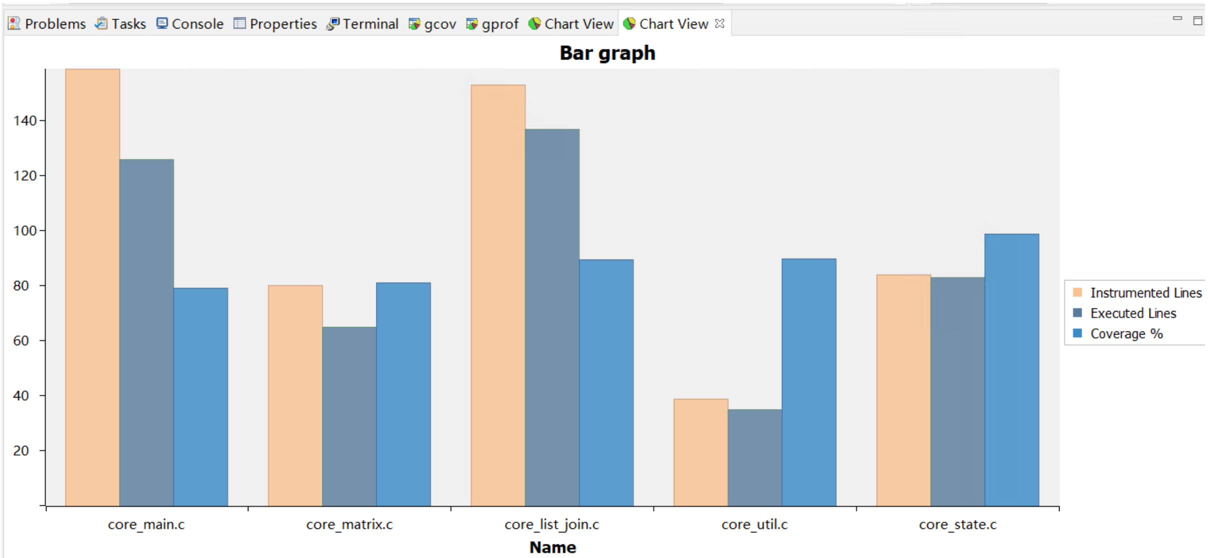
双击 Gcov 中的某一行，NucleiStudio 就会自动打开对应的文，并对文件中的代码着色，绿色表示在程序执行过程中有执行到，红色代表在程序过程中没有被执行到。开发者可以参考 Gcov 的结果，并对代码做出相应的优化。



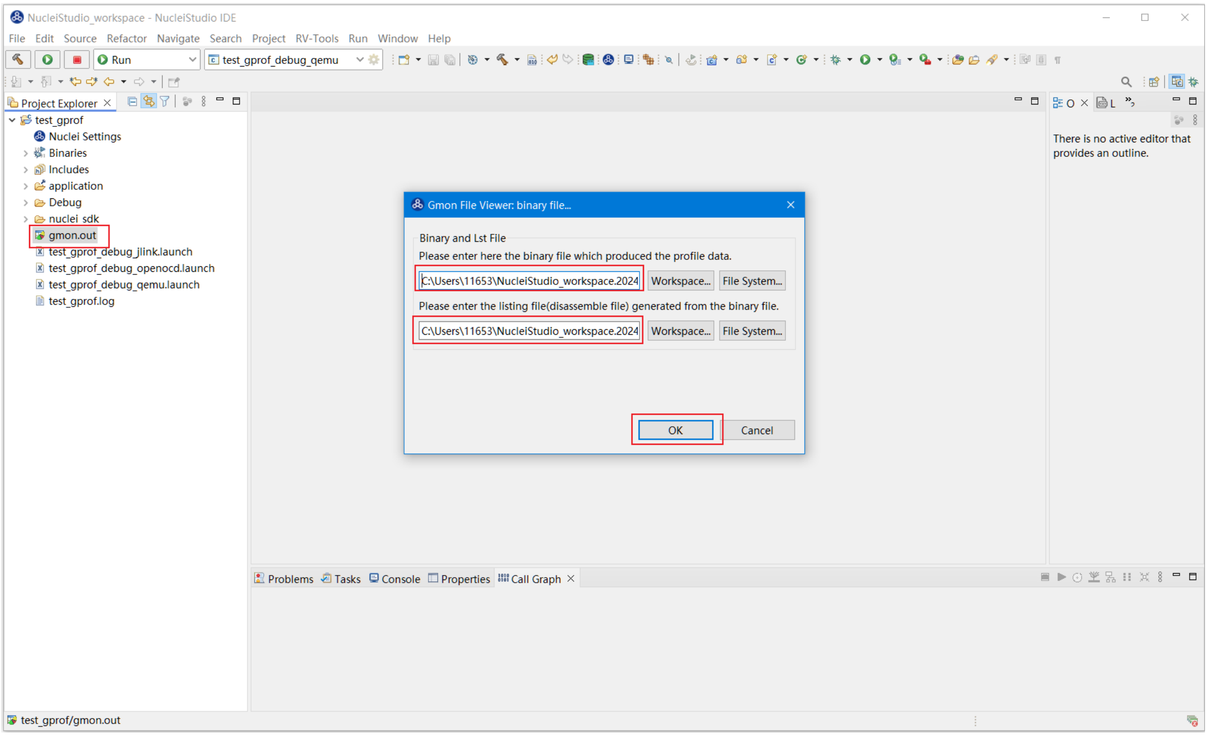
code coverage 也提供了以直方图的方式查看数据，选中想要查看的数据项，点击菜单中的直方图菜单，并按需求配置。



就可以在 Nuclei Studio 中查看 code coverage 直方图信息了。

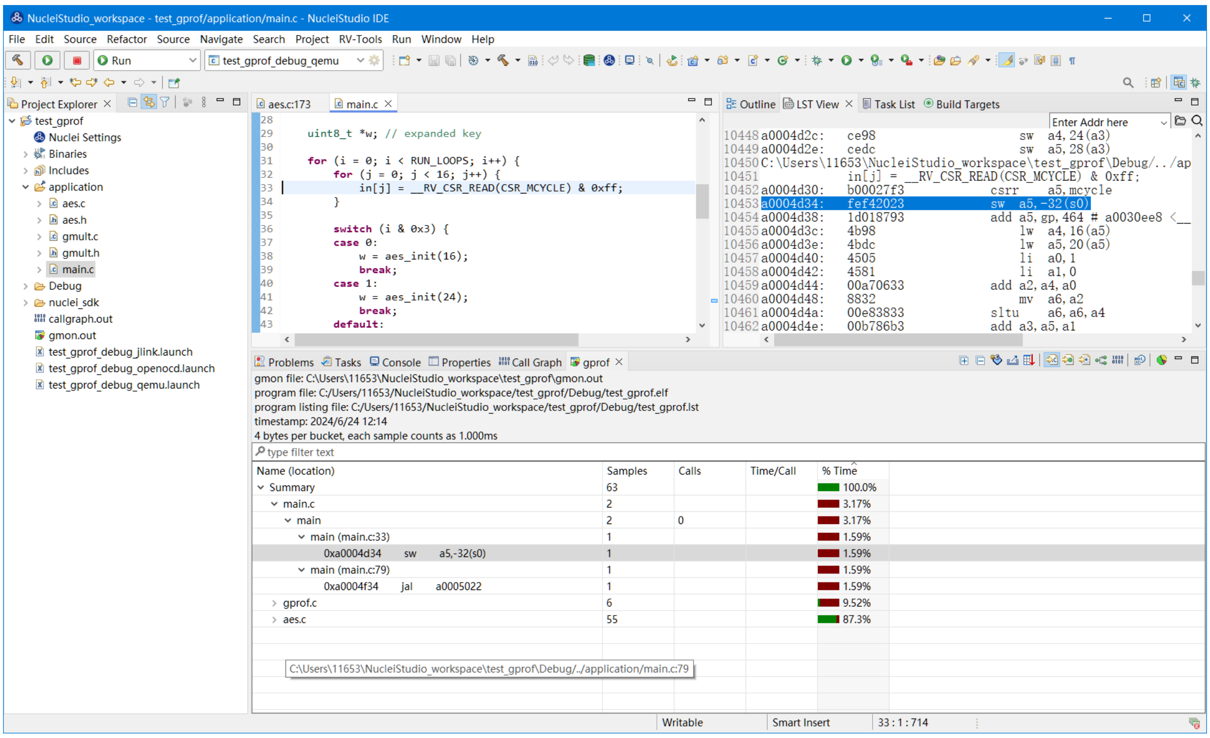


双击 gmon.out 文件，弹出一个文件选择框，提示填写与选中与 gmon.out 文件相关的 elf 文件和 *.lst 文件，默认会根据当 gmon.out ，自动填入对应的工程内的 elf 文件和 *.lst 文件，点击 OK 按钮。

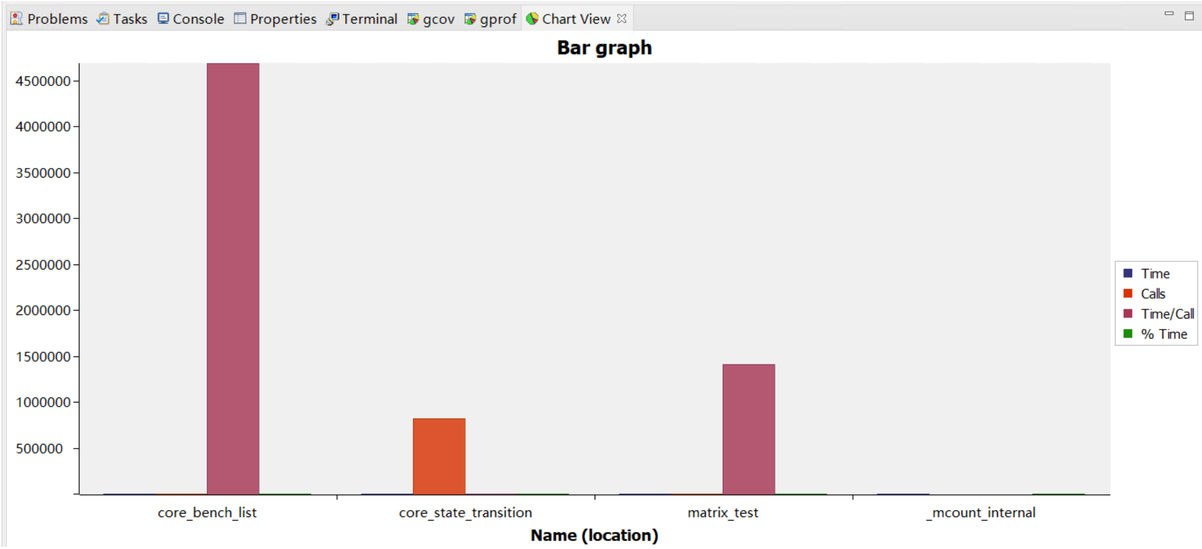


Gprof 工具会启动，就可以看到对应用程序的分析结果，显示了文件、方法的调用关系等。

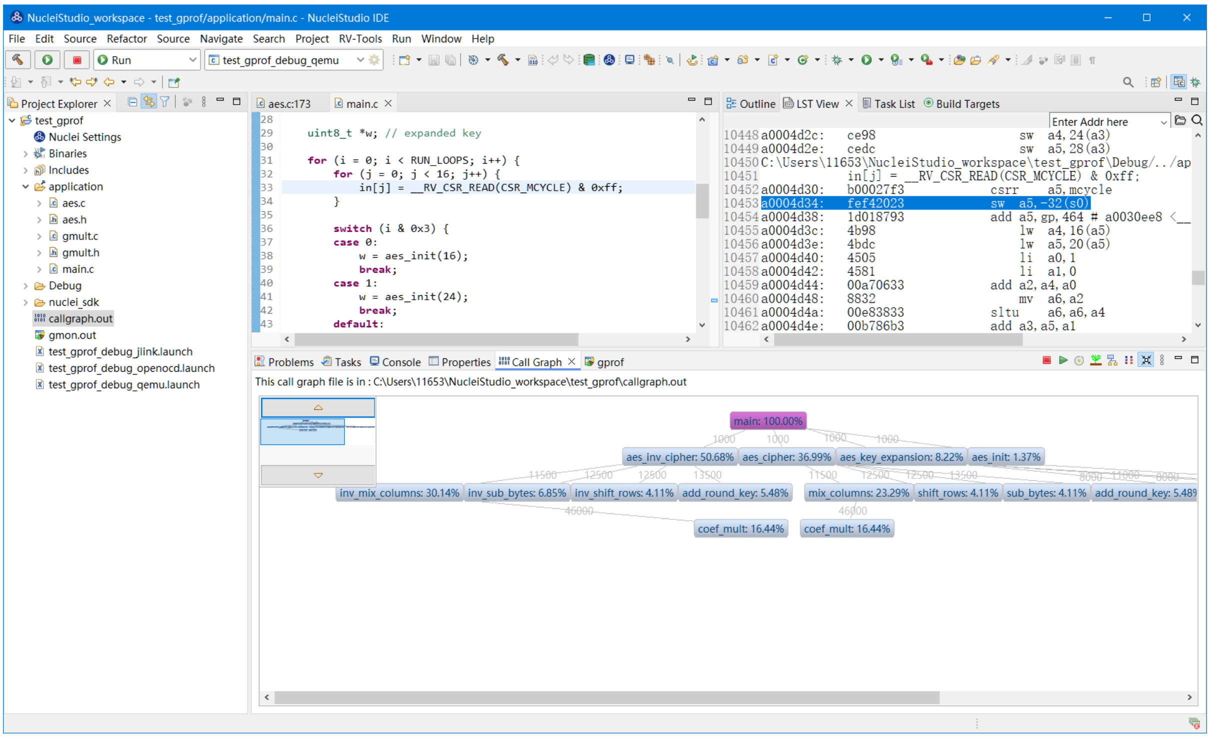
双击 Gprof 中的某一行，NucleiStudio 就会自动打开对应的源文件并定位到对应的行，同时打开 LST View 工具，并根据 addr 定位那一行，实现 Gprof、源代码、反汇编码的联系，帮用户快速了解程序结构及调用关系。



同样在 Nuclei Studio 中，可以查看 profiling 数据的直方图信息。



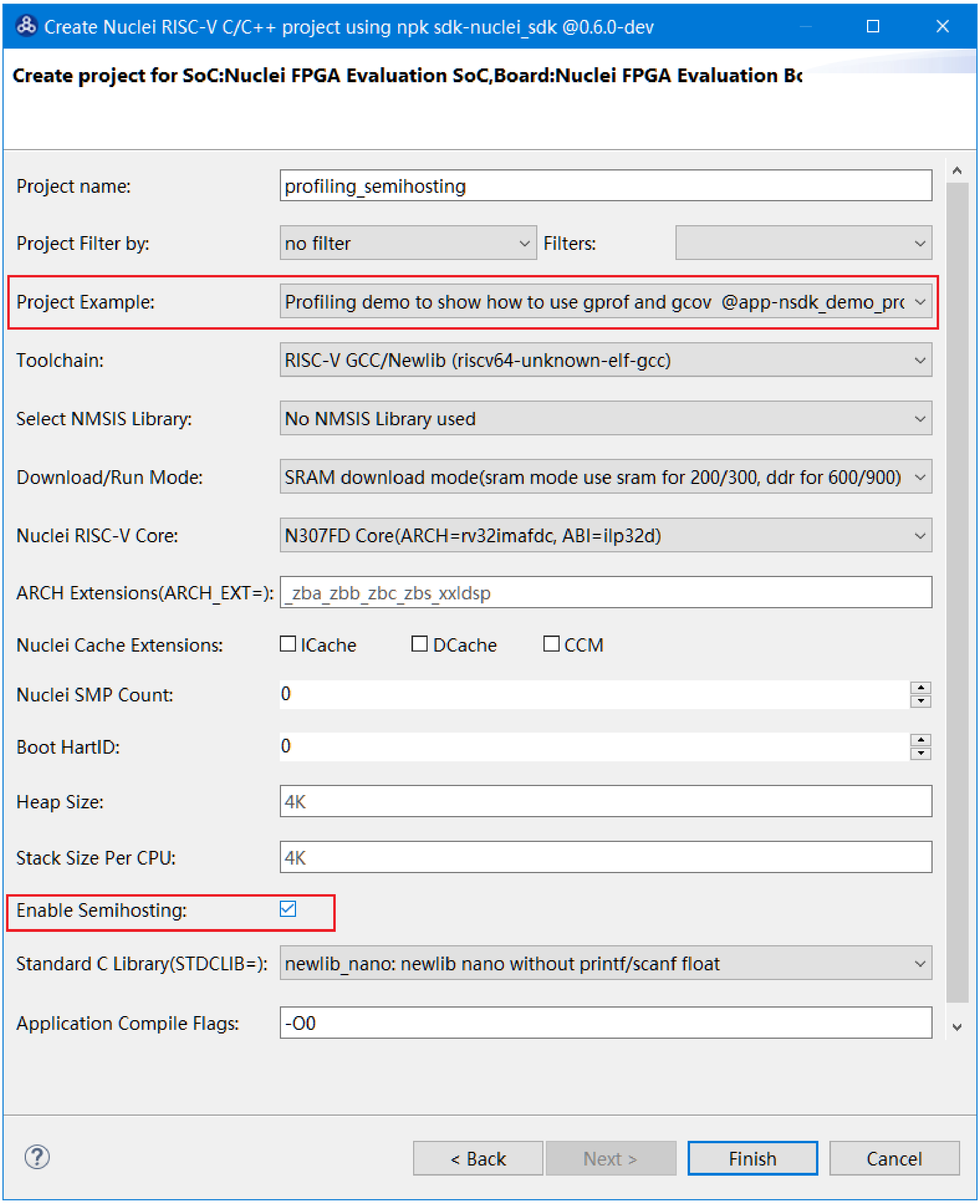
打开 Gprof 的同时，NucleiStudio 会根据 gmon.out 文件解析出程序的 Call Graph 并生成 `callgraph.out` 文件。双击 `callgraph.out` 文件，也可以点击 Gprof 工具的菜单栏中 `Open Call Graph View` 按钮，来启动 Call Graph 工具。关于 Call Graph 的具体使用，可以参考关于 *Call Graph* 功能 (page 201)。



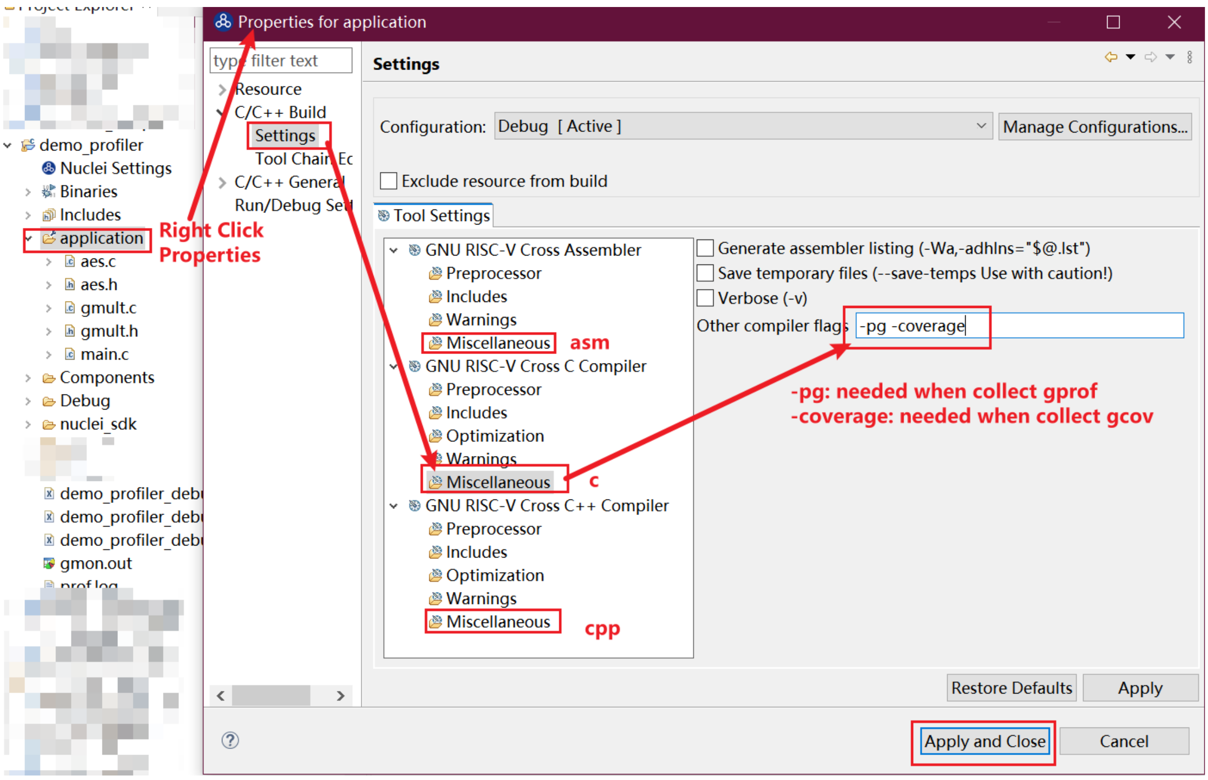
通过 Semihosting 使用

NucleiSudio 安装了 nuclei_sdk 0.6.0 后，可以创建一个 `Profiling demo to show how to use gprof and gcov` 的测试工程，此时需要选中 `Enable Semihosting`。关于 `Profiling demo to show how to use gprof and gcov` 测试工程，可参考 demo_profiling[20]。

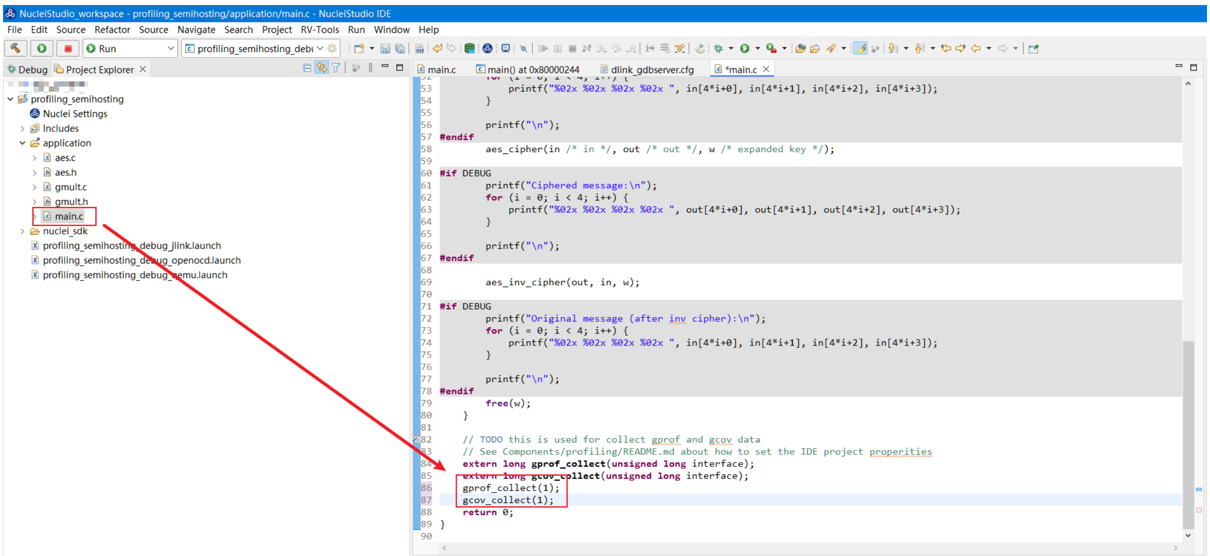
[20] https://doc.nucleisys.com/nuclei_sdk/design/app.html#demo-profiling



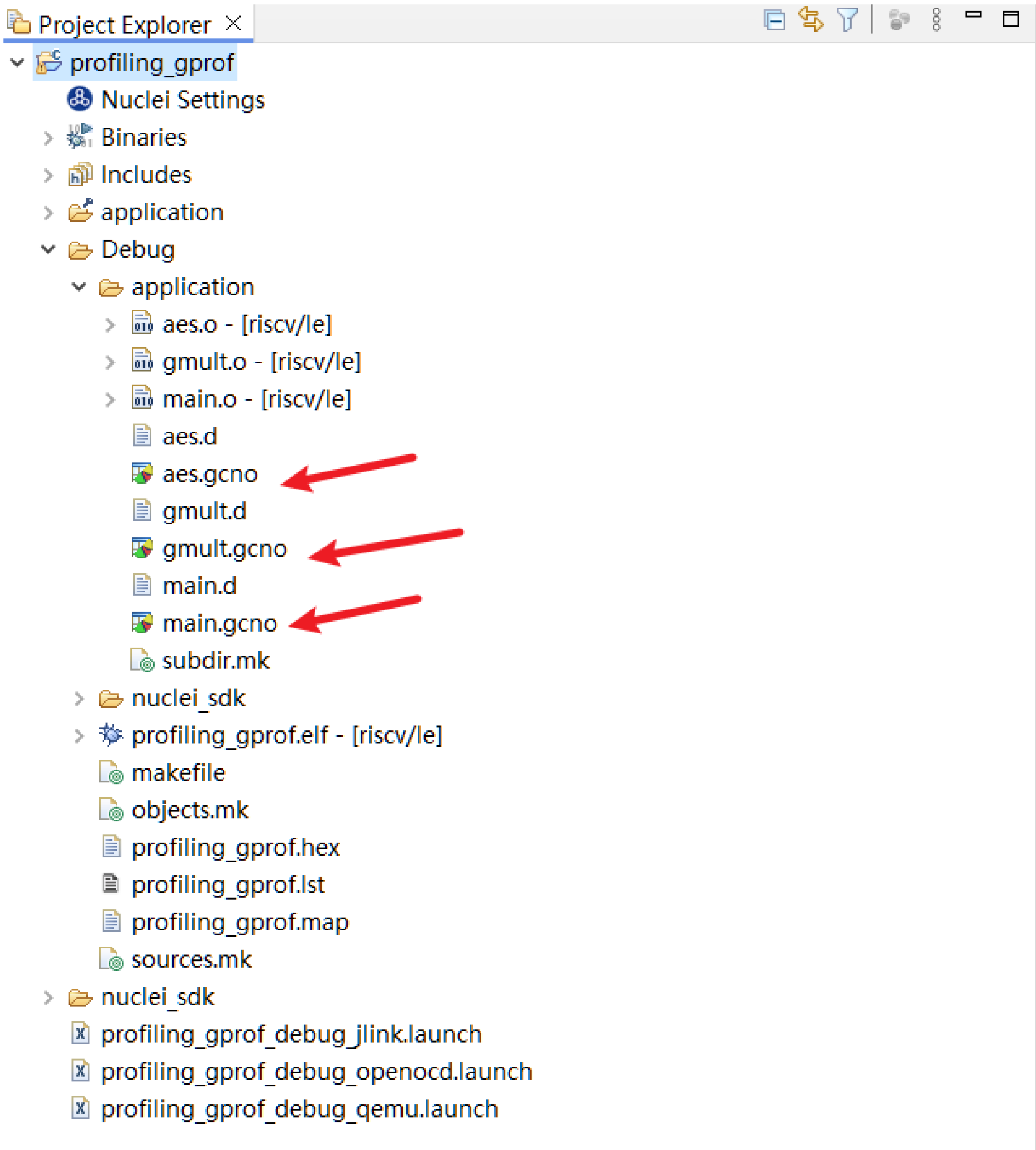
工程创建后，需要对想要进行代码分析的文件或者文件夹设置一个 `-pg --coverage` 的编译选项，然后编译工程。



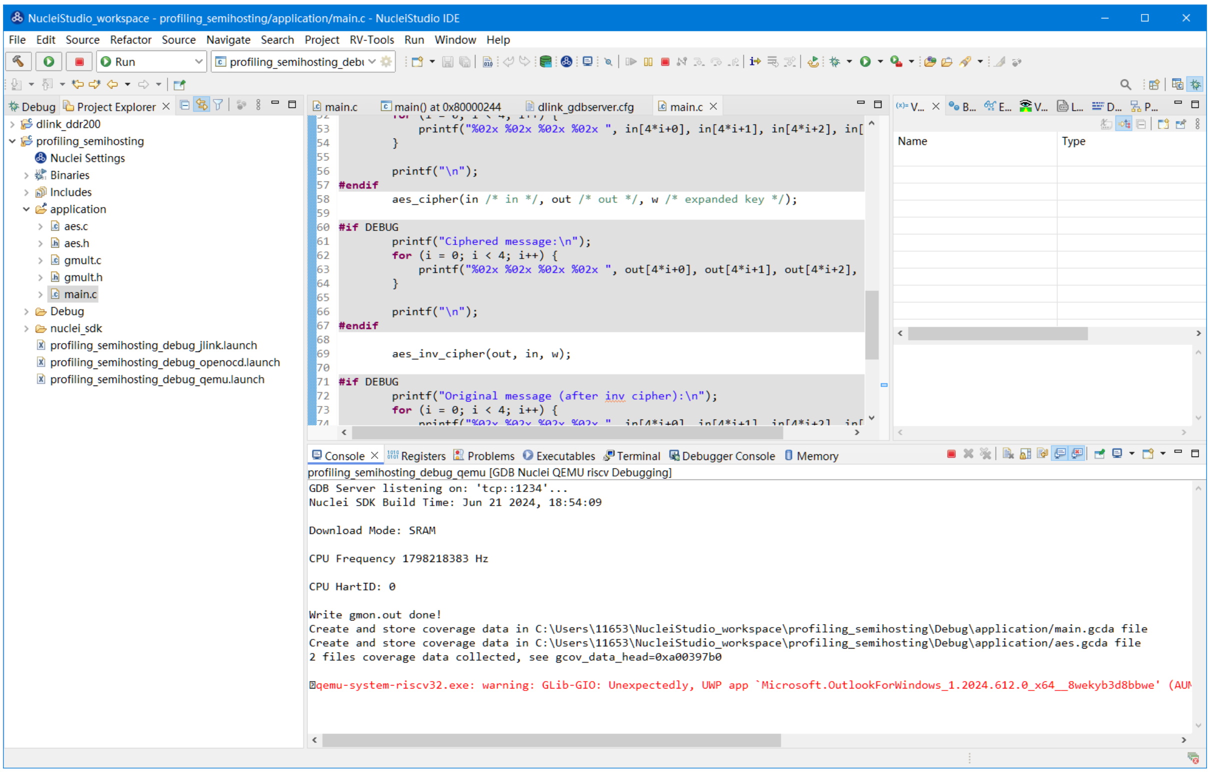
同时，需要修改程序中 gprof_collect(2); 为 gprof_collect(1); 、gcov_collect(2); 为 gcov_collect(1); （测试工程中在 main 函数的最后），则在运行过程中，将会通过 Semihosting 将结果输出为文件。



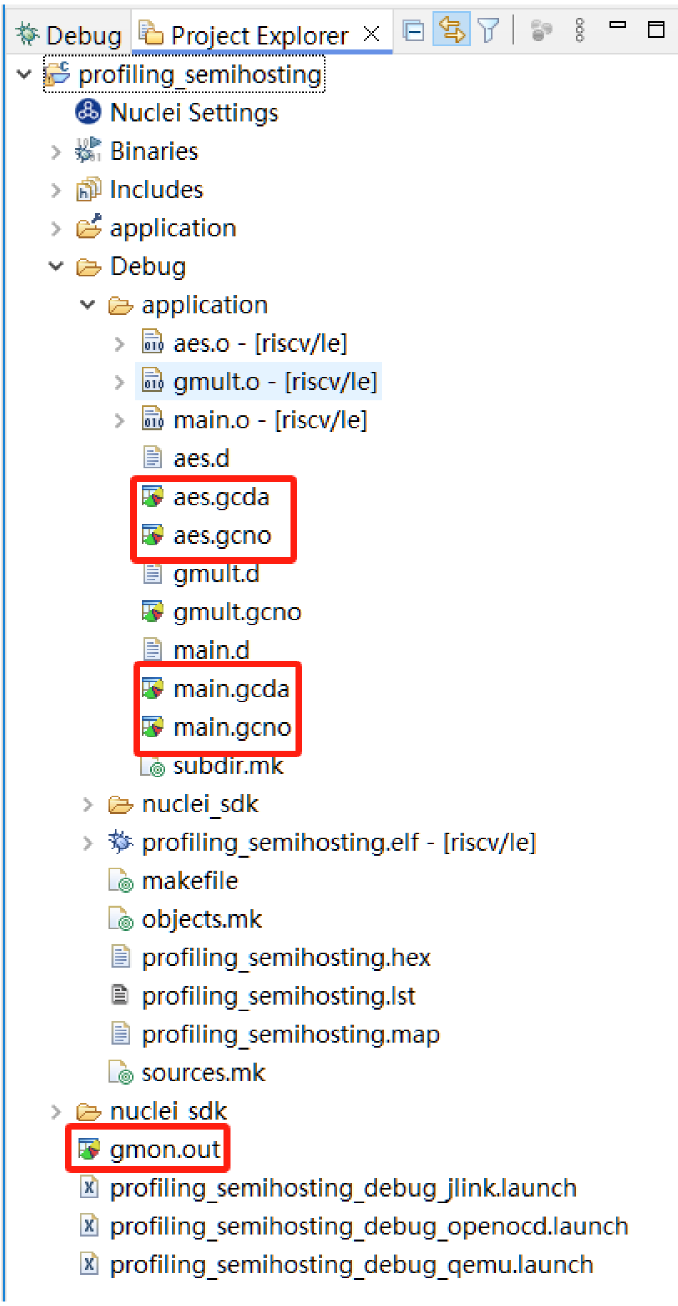
开始编译工程，在编译通过的工程的 Debug 目录中，可以看到，已经生成了几个 .gcno 的文件。



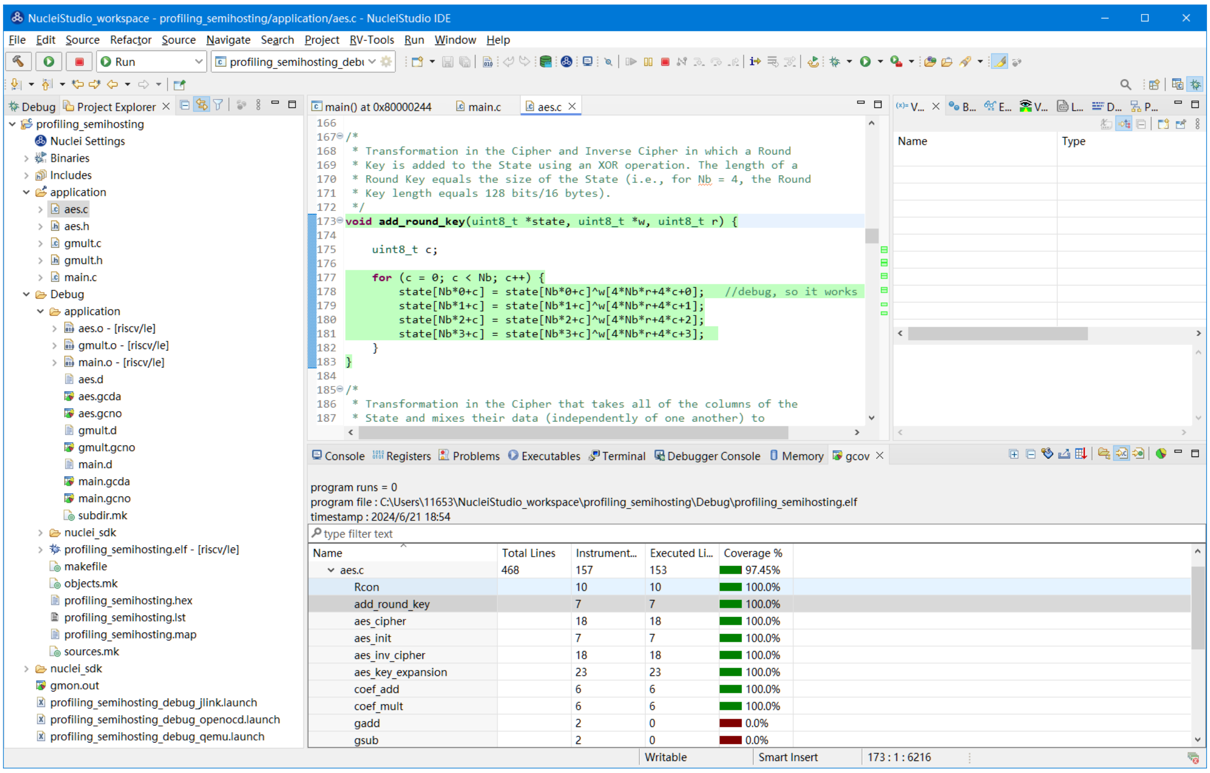
工程编译完成后，可以运行或调试工程，我们可以选择在 QEMU 下进行，也可以调试实际的开发板。



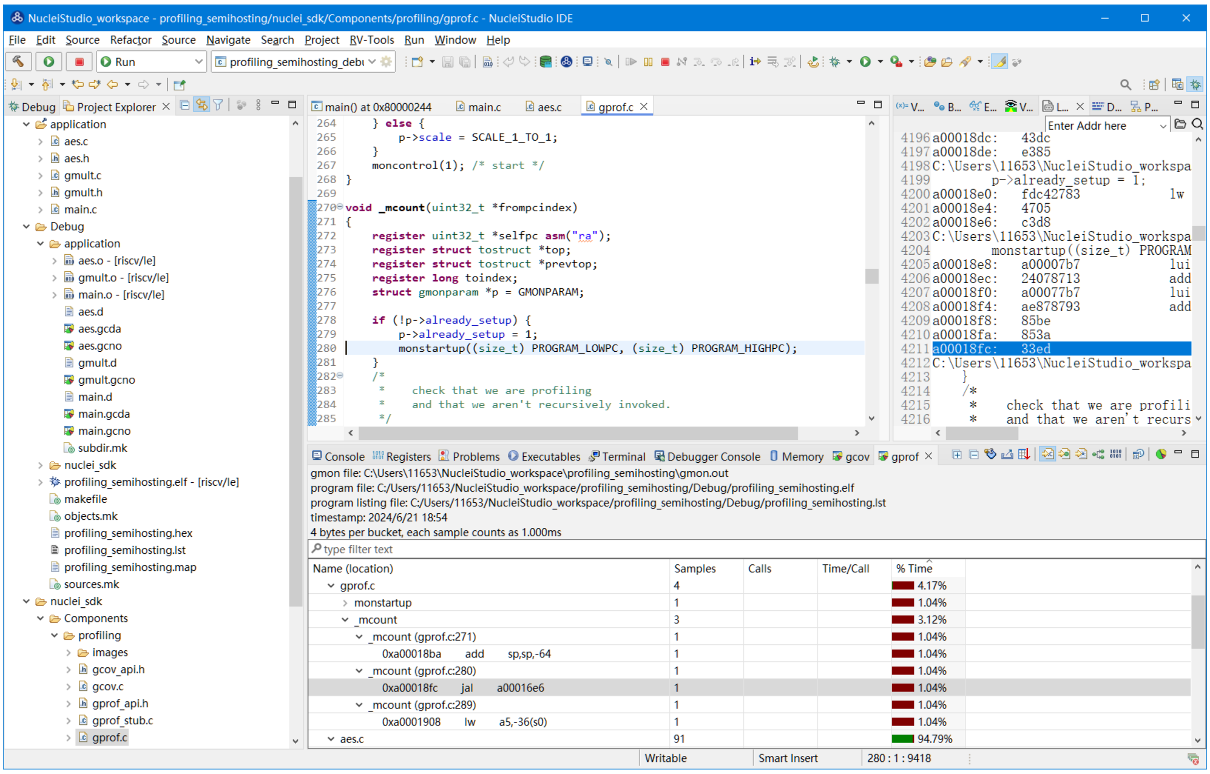
本例以 QEMU 为例进行运行程序，程序运行结束后，刷新工程，可以看到工程下多出了几个文件，*.gcda 文件以及 *.out 文件。至此，后面查看结果与上面类似。



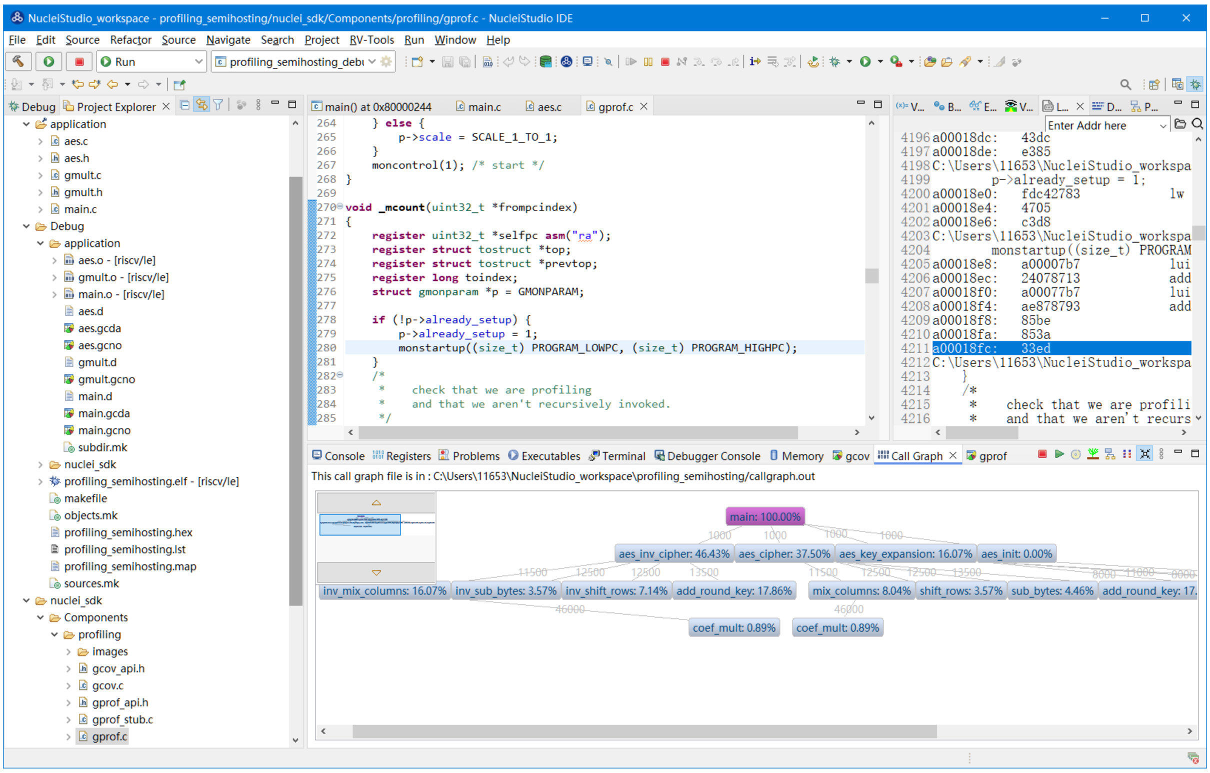
在 Nuclei Studio 中通过 gcov 工具查看应用程序的 Code Coverage 信息。



在 Nuclei Studio 中通过 gprof 工具查看应用程序的 Profiling 信息。



在 Nuclei Studio 中通过 Call Graph 查看调用关系信息。



2.10.4 Trace 功能的使用

Trace 技术是一种强大的调试工具，它能够帮助开发人员跟踪和记录程序执行过程中的关键信息，从而有效地诊断问题、优化性能和提升系统的稳定性。

Nuclei Studio 集成了 Trace 工具，结合相应硬件和 Nuclei OpenOCD，用户在工程 Debug 时可查看 Trace 日志，并结合源码进行问题排查。

> **Note**
>
> 在 芯来科技视频号中有 如何在 **Nuclei Studio** 中使用 **Trace** 功能的视频，您可以在微信中搜索 芯来科技视频号点击查看相关内容。

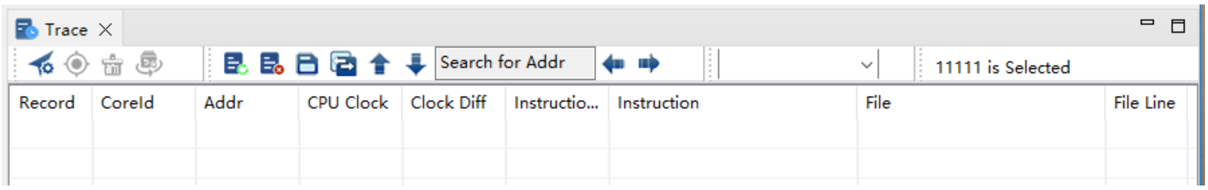
> **Note**
>
> 关于 OpenOCD 的 Nuclei ETrace 的一些命令，请参加 OpenOCD 下的 openocd.pdf 手册。

在使用过程，如有问题，可以查看 https://github.com/Nuclei-Software/nuclei-studio 相关内容，也可以向我们提交相关 issue。

Trace 界面介绍

Trace View

在 Nuclei Studio 中，通过菜单 `Window->Show View->Other` 打开 View 管理器，在里面找到 RV Trace->Trace 菜单，点击打开 Trace 菜单。



Trace 的视图分两部分，上面为 Trace 工具栏，下面是 Trace 记录表格。Trace 工具栏的介绍和功能分别如下：

- **Trace setting**

trace 的配置信息，在这里配置 Trace ATB2AXI Config Addr、Trace Buffer Base Addr、Trace Buffer Size in Bytes、Trace Wrap

- **Start trace/stop trace**

设置开始/停止 trace 操作。

- **Trace clear**

清空硬件上的所有的 trace 设置。

- **Dump trace file**

从硬件上 Dump trace 文件。

- **Reload trace file**

本地重新加载 trace 记录表内容。

- **Clear viewer**

清空 trace 记录表内容，以及 Trace Decode 相关的配置，如 HartID 和 Thread 的关系等。

- **Save trace log**

将 trace 记录表保存为 csv 表格。

- **Toggle instruction stepping**

当选种某条记录时，可以打开并定位到该条记录所对应的源码和反汇编码。

- **step into previous line**

当选种某条记录时，跳转到该条记录的上一条记录，并定位到所对应的源码和反汇编码。

- **step into next line**

当选种某条记录时，跳转到该条记录的下一条记录，并定位到所对应的源码和汇编码。

- **Search for Addr**

搜索框，可以通过 Addr 搜索到对应的那一行 trace 记录。

- **search backward**

搜索结果的记录是多条时，可以查看上一条搜索结果。

- **search forward**

搜索结果的记录是多条时，可以查看下一条搜索结果。

- **Page**

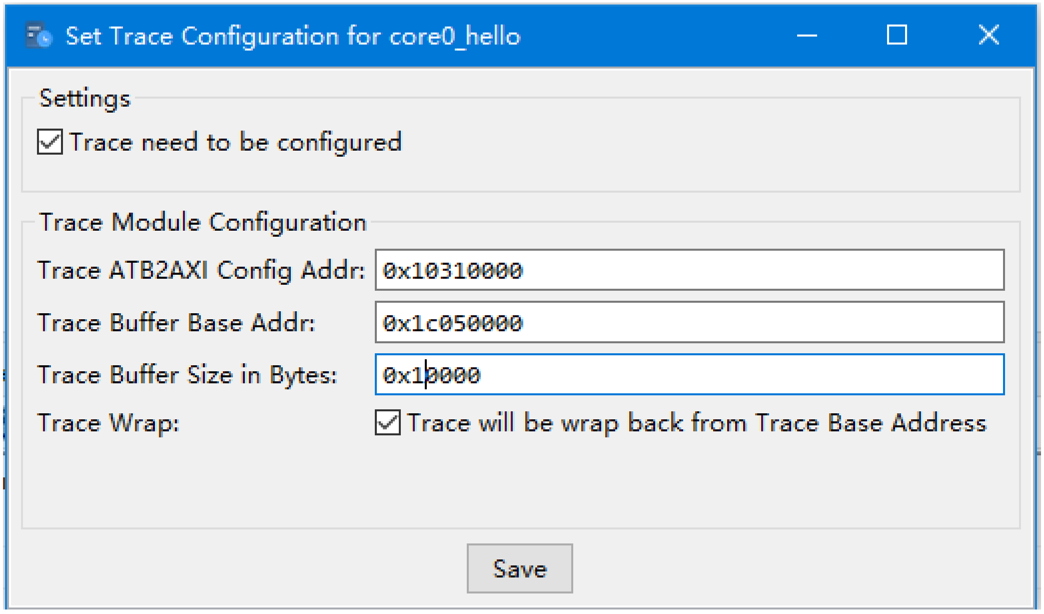
多页的翻页，trace 如果条数很多时，为了方便查看，会采用多页显示。

Trace 记录表格，是 Nuclei Studio 将 dump 到的 trace 文件进行解密之后，生成的记录进行展示，并且当用户点击某条记录时，会自动定位到对应的源代码和反汇编代码的行数。

- **Record**：记录 id
- **CoreId**： Coreid，主要是在多核时可以用于区分不同的 Core
- **Addr**：指令地址
- **CPU Clock**：时钟 Cycle 计数
- **Clock Diff**：时钟 Cycle 差
- **Instruction Code**：十六进制表示的指令码
- **Instruction**：指令码
- **File**：指令码对应的源码所在的文件
- **File Line**：指令码对应的源码所在的文件的行数

Trace Configuration

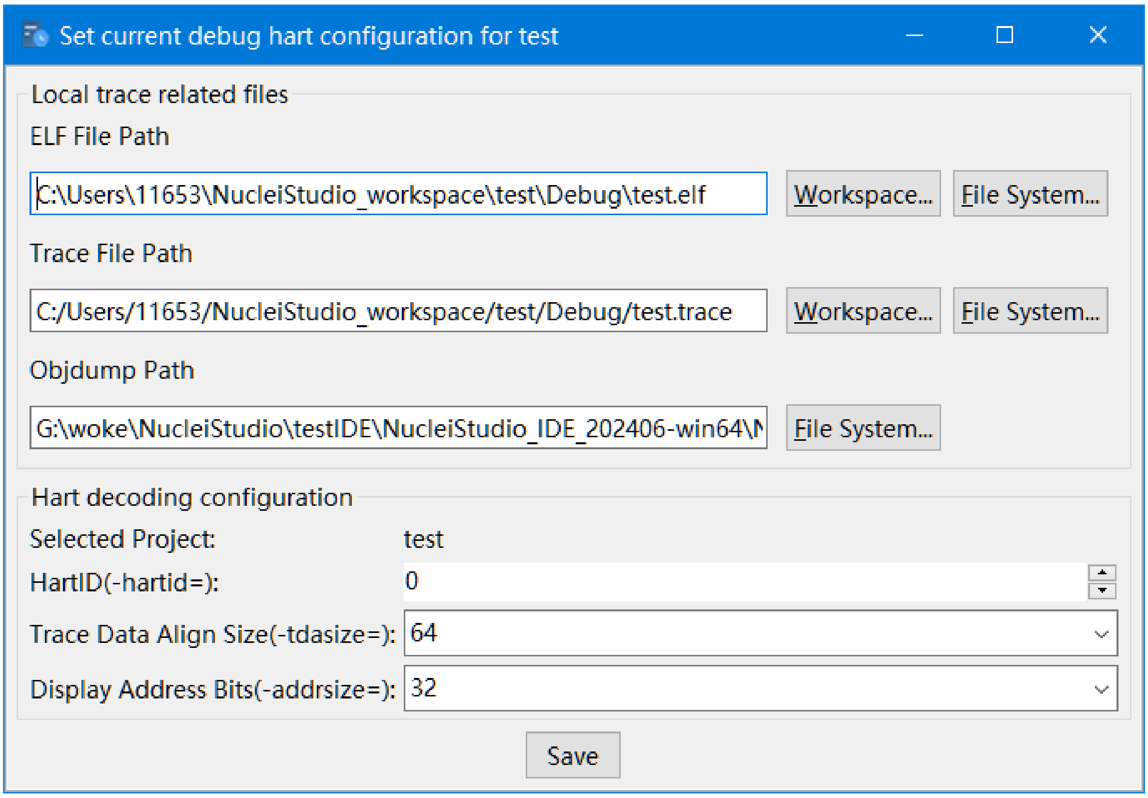
用户可以在这里配置 Trace 的 Trace ATB2AXI Config Addr、Trace Buffer Base Addr、Trace Buffer Size in Bytes、Trace Wrap。具体的信息，根据不同的硬件而不同。



- **Trace need to be configured:** 如果需要配置 Trace 模块就勾选，如果其他地方已经配置过了，就千万不要勾选了，例如多核 SMP/AMP 的情况下，SoC 上只有一个 Trace 模块，假设其中一个核心已经勾选配置了，其他的核心就不能勾选了，或者是配置是在 C 代码中或者其他地方做了，也千万不要勾选。
- **Trace ATB2AXI Config Addr**： ATB2AXI 模块控制器的基地址。
- **Trace Buffer Base Addr**：存放 trace 记录的开始地址，例如：针对某个 SoC, 举例如下在 flashxip 模式，使用 ilm（0x1c000000）作为缓存 buffer；在 sramxip 模式，使用 dlm（0x08010000）作为缓存 buffer。
- **Trace Buffer Size in Bytes**：存放 trace 记录的 Buffer 大小，单位为字节。
- **Trace Wrap**：是否允许自动复盖，允许则在 Buffer 满时，将再次从头开始覆盖记录。

Trace Decoder Configuration

Set Current Debug hart Configuration 弹框中，用户可以自定义 trace decoder 的参数，具体如下。



- **ELF File Path**： trace 生产时执行的 elf 文件的地址。
- **Trace File Path**： 需要解析的 trace 文件的地址。
- **Objdump Path**： trace decode 过程中，需要用到 objdump 工具，所以这里需要指定所使用到的 objdump 工具的地址。
- **HartID**： trace decode 时需要指定当前需要查看的 trace 对应的 HartID，单核工程默认 HartID=0。
- **Trace Data Align Size**：跟踪数据对齐大小，一般与硬件的 trace 输出位宽对齐，默认有 8、32、64。
- **Display Address Bits**： trace decode 后显示地址的位数，一般是 32、64、128 位。

Trace 的使用

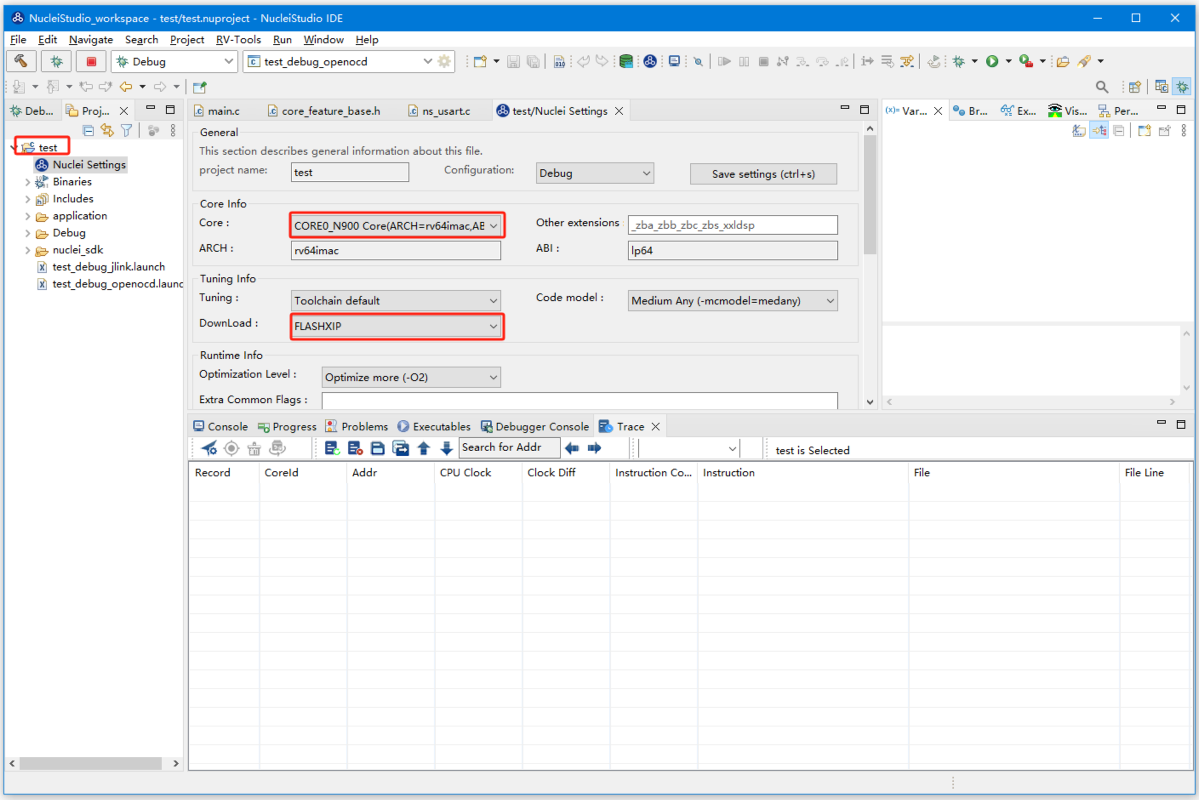
在使用 trace 功能时，必须在工程 Debug 时，通过 Nuclei OpenOCD 或者 Dlink 将 Trace 命令下发到硬件，目前通过 OpenOCD，可以实现在单核、多核 SMP 和多核 AMP 应用下进行 Trace 记录，而 Dlink 仅支持在单核应用下的 trace 记录。

下面我们以 OpenOCD 为例，演示如何使用 Trace 功能。

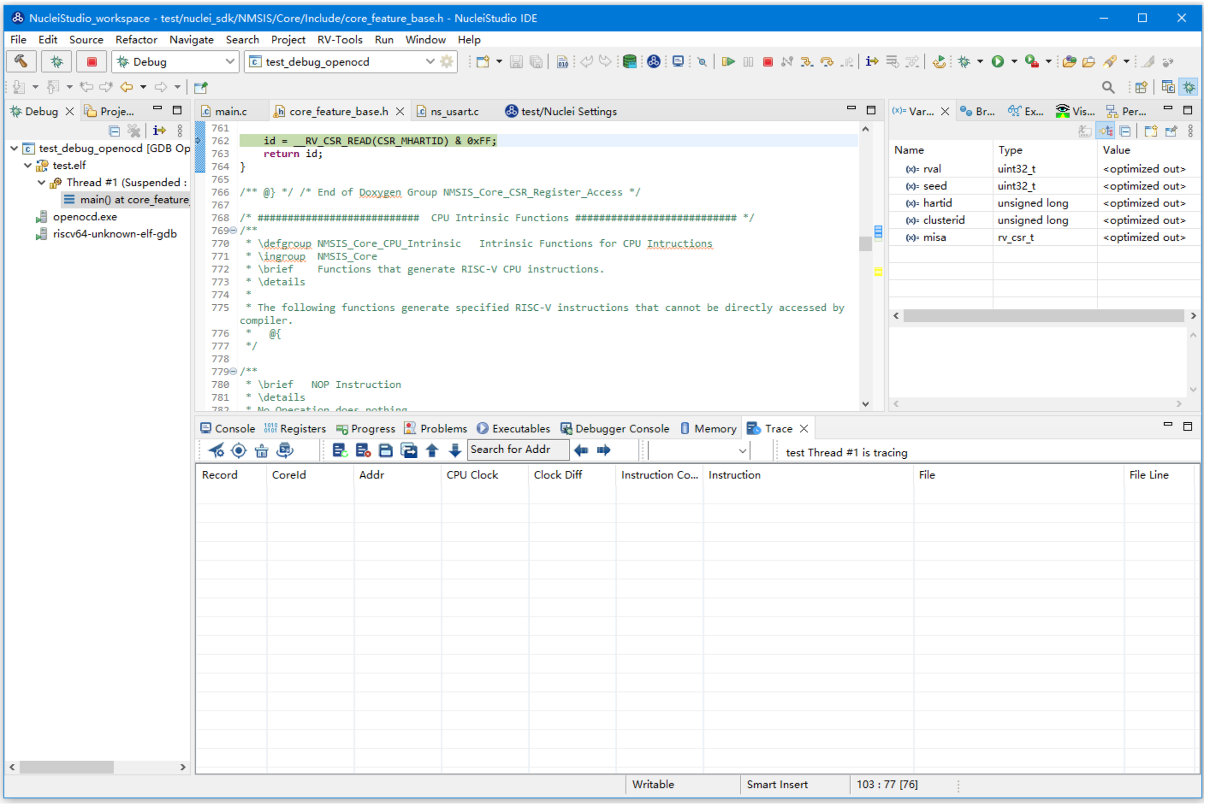
在单核应用中使用 Trace

如果您已获取到芯来授权的 CPU 和相关配套硬件并准备好硬件环境，这里不详细说明。然后创建好对应工程并确保它能在硬件上运行和调试。以下示例是在我们自己构建的一个测试环境上的流程举例说明。

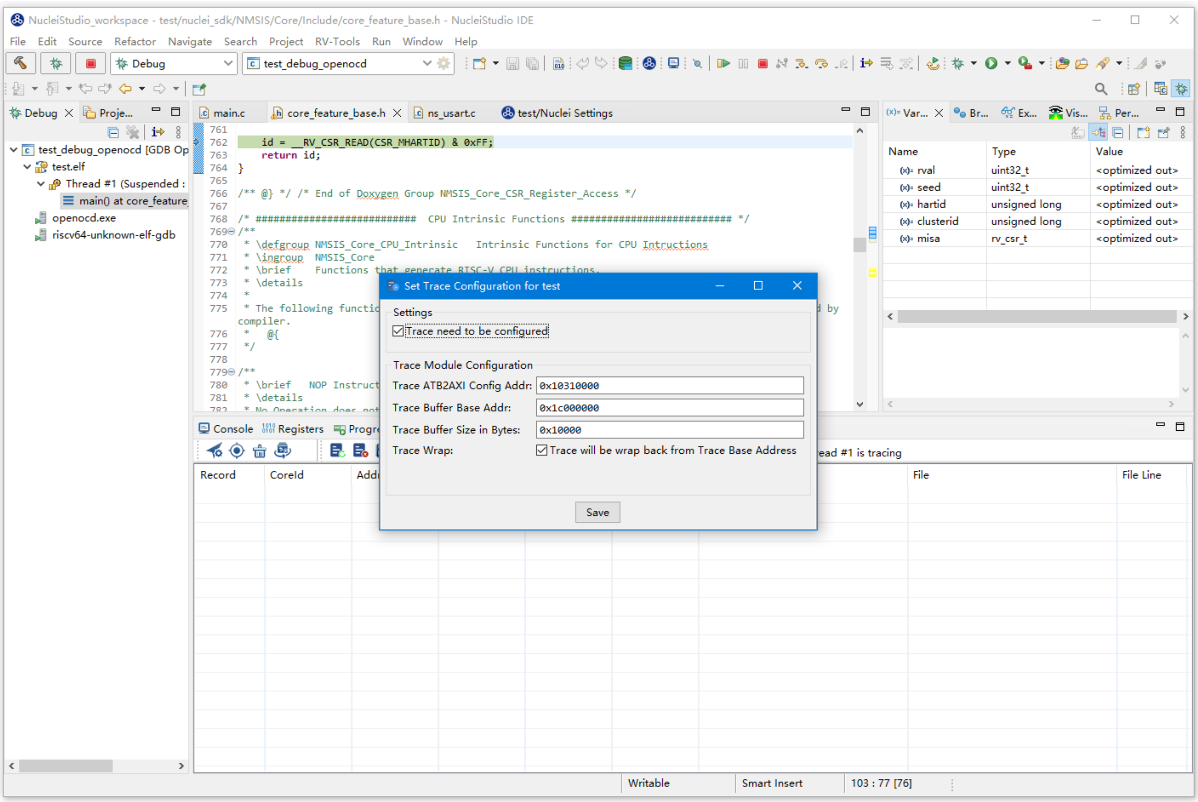
我们在这里创建了一个 N900 的单核应用 helloworld，并让它跑在 FLASHXIP 模式下。



我们可以记录整个应用运行完的 trace，也可以记录某一段 Debug 断点之间的 trace。进入 Debug 模式后，打开 Trace 视图。

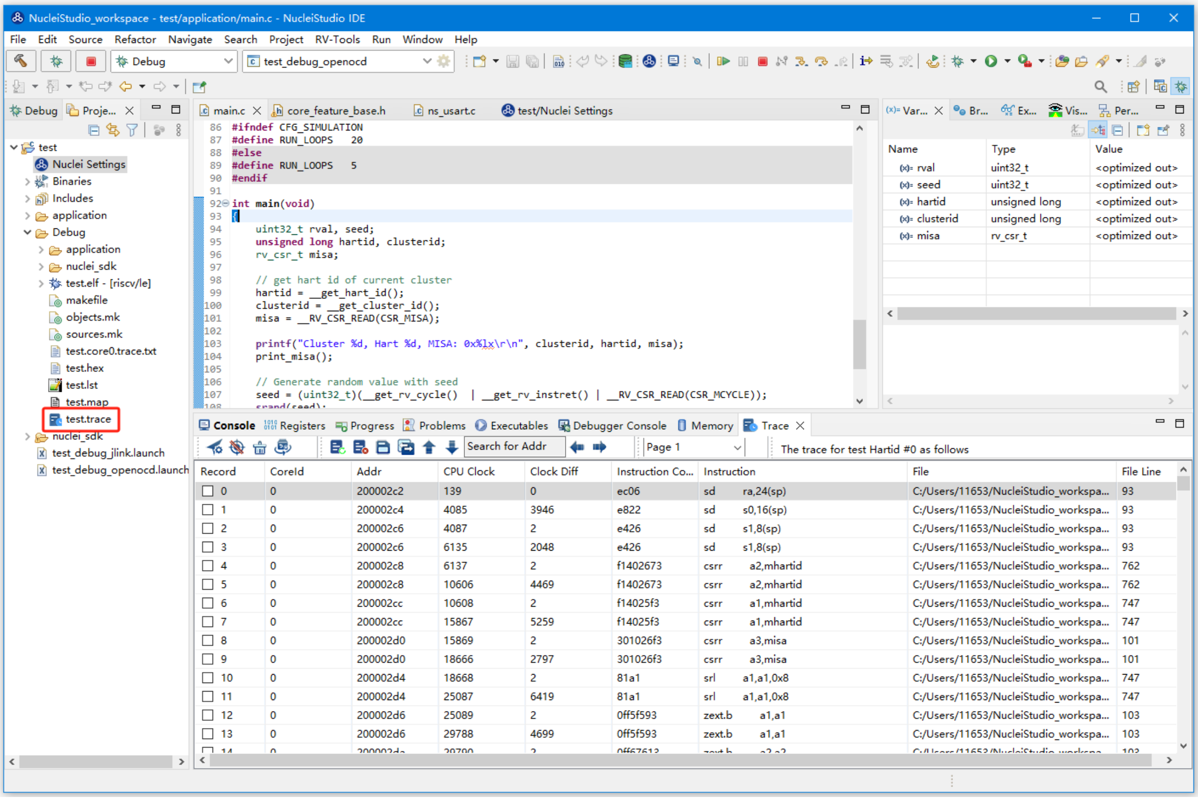


设置 Trace Configuration，设置 trace 配置信息并保存 (Save)，如果不想保存，就关闭窗口。

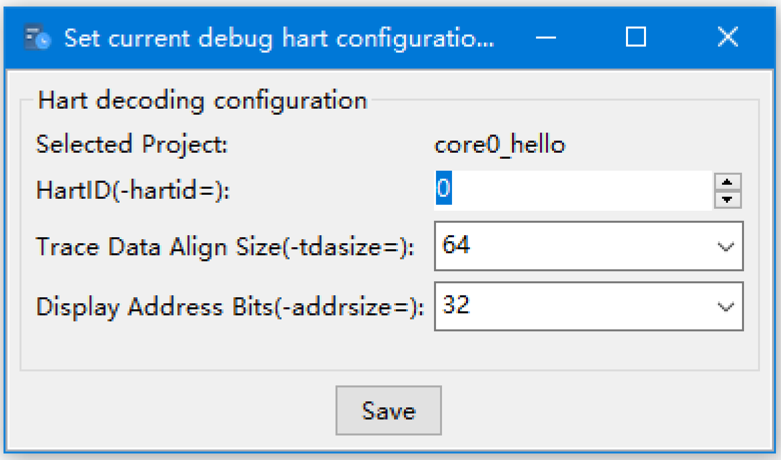


Trace 配置完毕后，可以设置两个断点，一个断点用于 Trace 开始点，一个断点用于 Trace 结束点，在开始点断点停下后就可以点击 `start trace` 按钮，就可以继续 debug 操作 (如单步或者运行等) 了，在结束点断电停下后，就可以点击 `stop trace` 按钮来结束 Trace。上面只是 Start/Stop Trace 的一种使用示例，也可以更灵活一些，请根据自己需要进行使用。当 trace 结束时（多核情况下请确保每个 CPU 的 Trace 都结束了），就可以点 `Dump trace file` 按钮，将 trace 文件从硬件上下载到本地，默认下载的

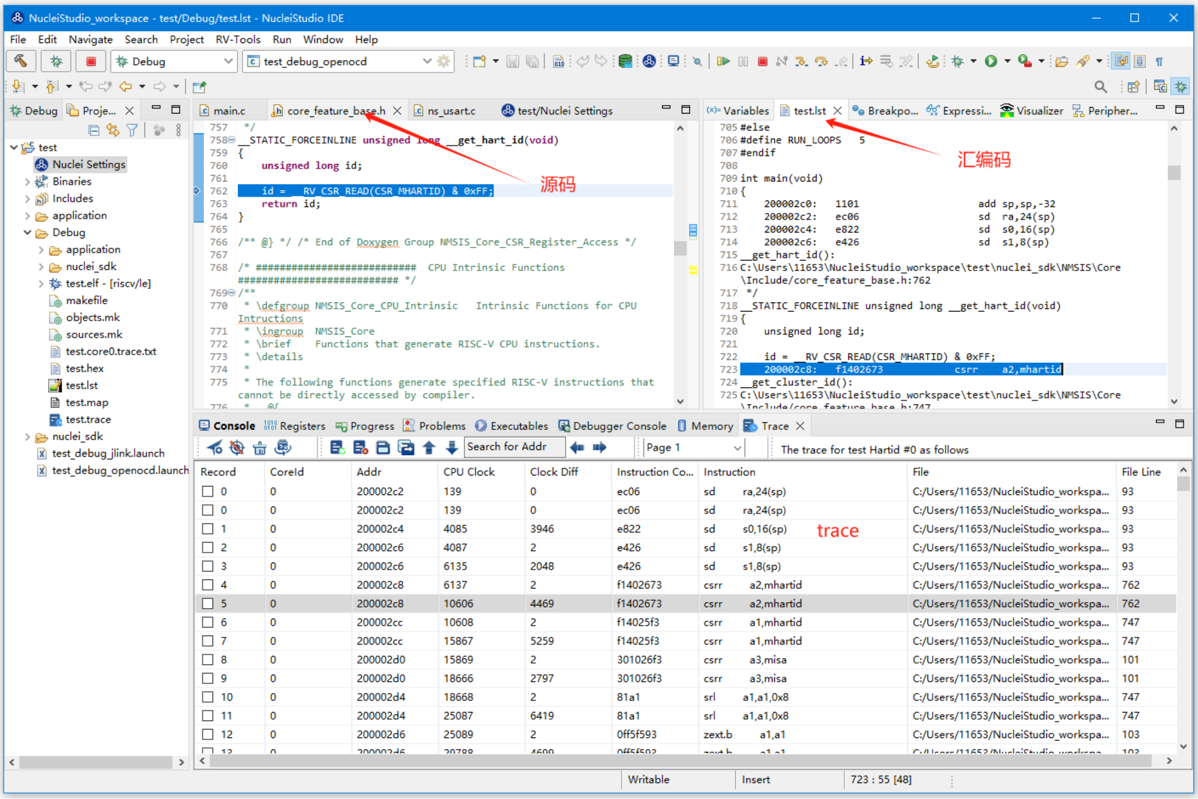
trace 文件存在工程目下的 debug 目录下，有一个 工程名.trace 的文件。



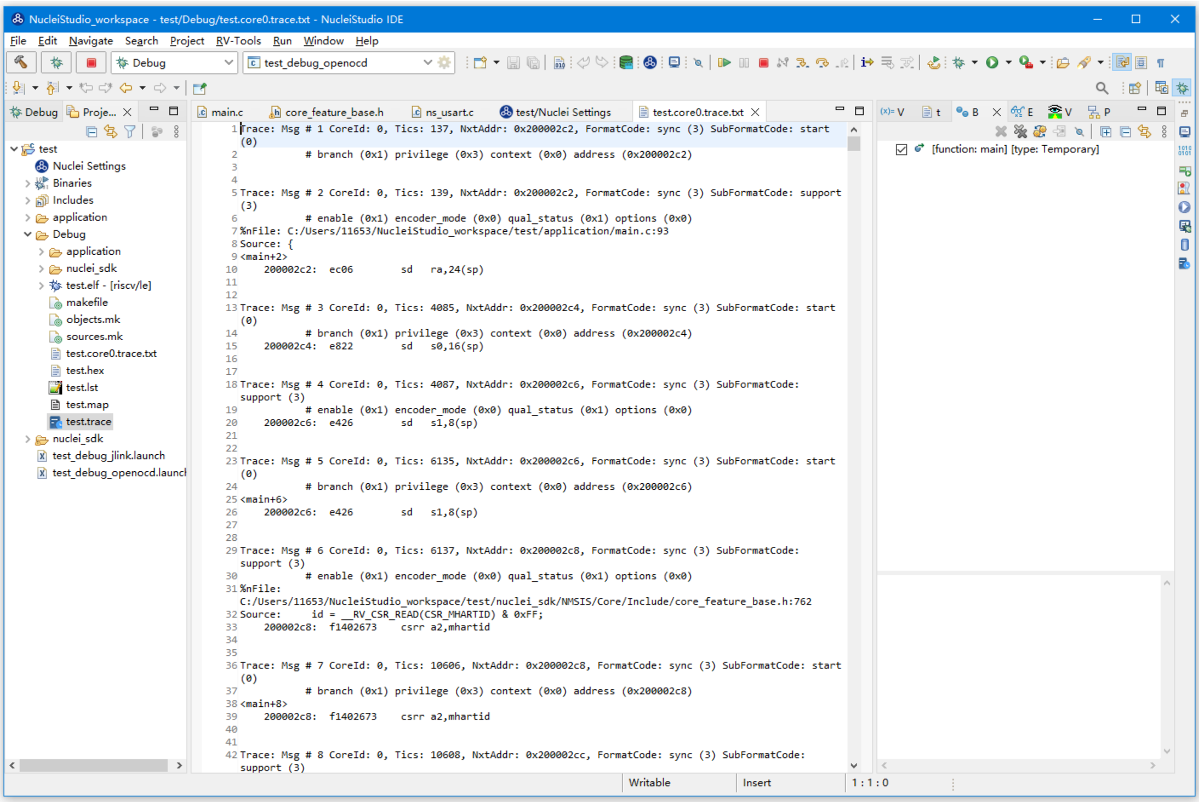
Trace 文件下载完后，Nuclei Studio 会弹出一个 Set current debug hart configuration 框。



在框中填写确正信息（这里的 HartID 指的是对应的 Thread 的 hartid，请不要填错了）并确认，Nuclei Studio 对 trace 文件开始解析，并生成 trace 记录表格。在 trace 记录表格，选中任意一条记录，Nuclei Studio 会自动找到源码和反汇编码，并定位那对应的那一行（因反汇编码与源码在同一个视图中打开，需要用户自己把反汇编码移到另一个视图中）。

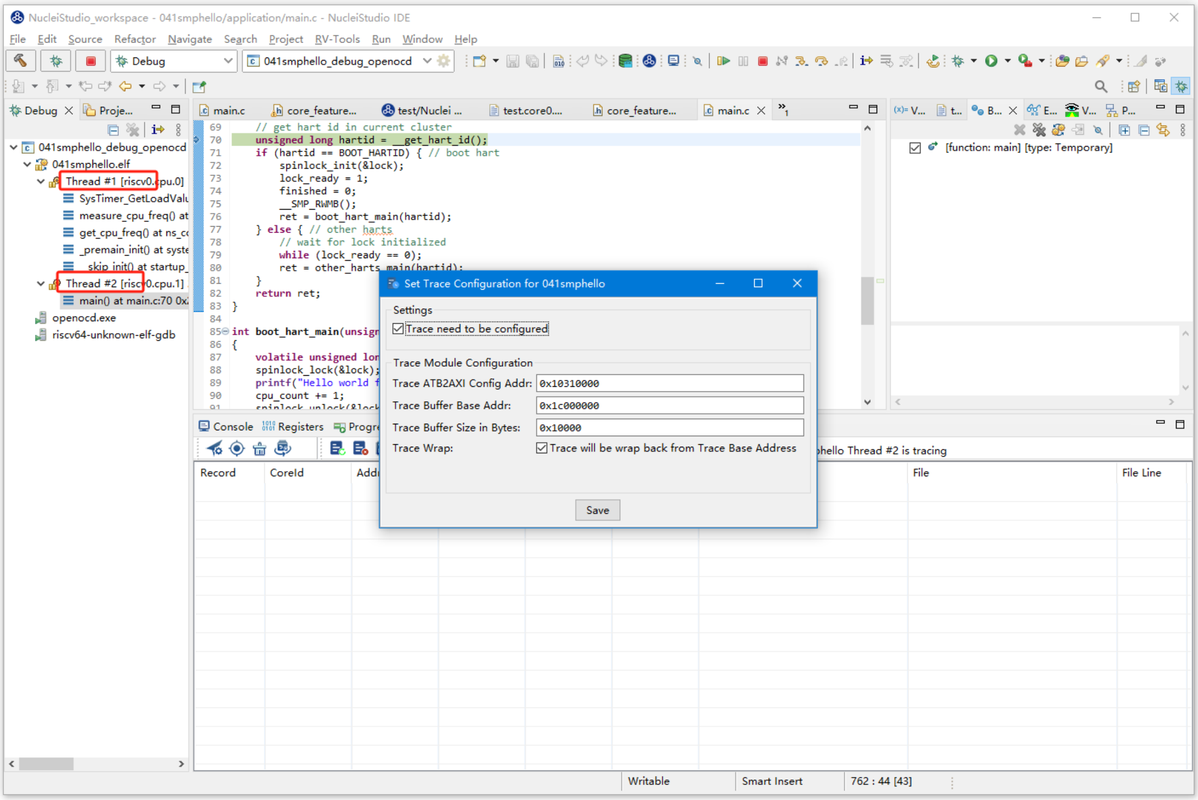


也可以双击 工程名.trace 文件，以文本的方式查看 trace 文件。

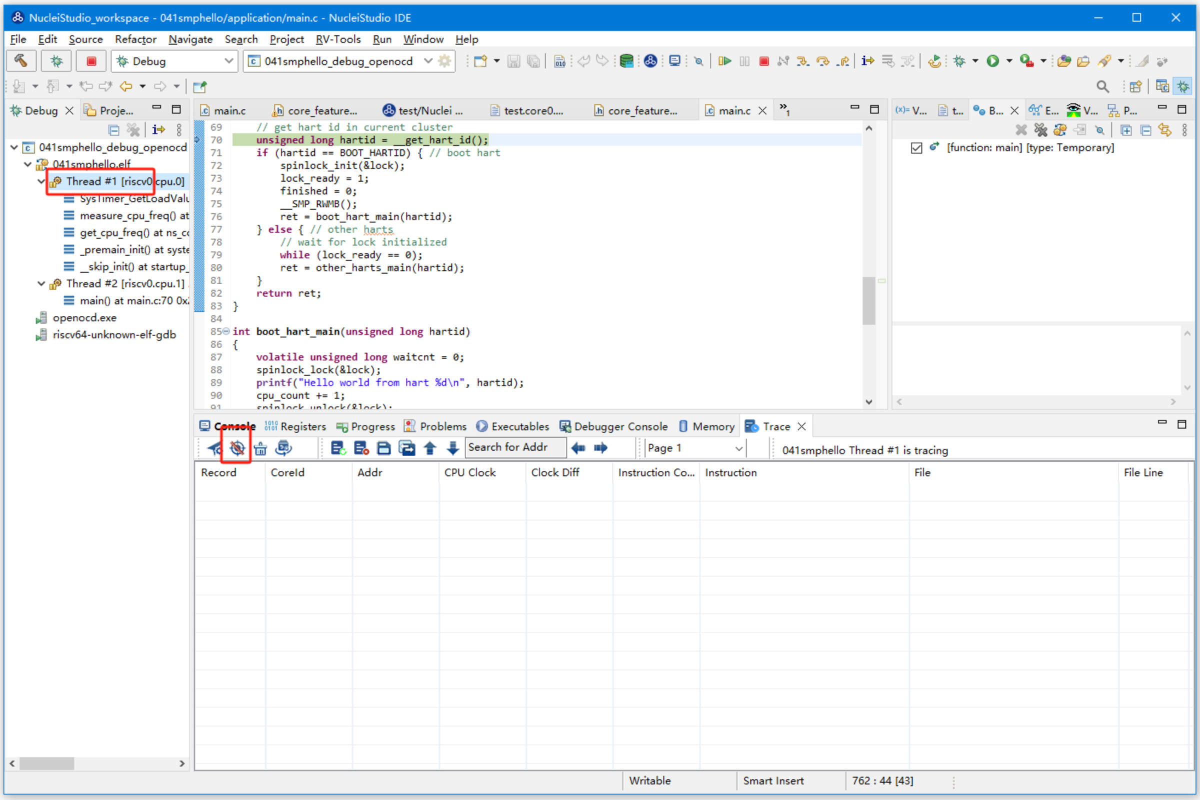


在 SMP 多核应用中使用 Trace

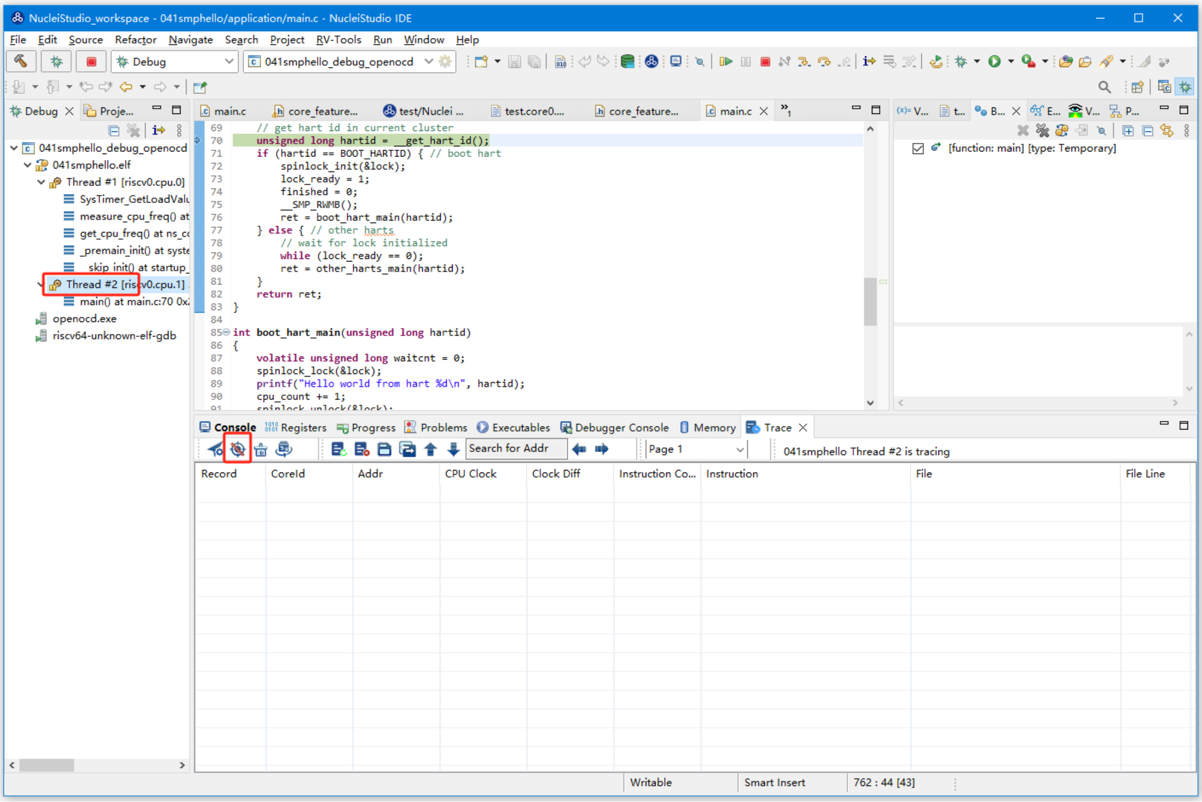
在 SMP 多核应用中使用 trace 与单核大体相似，差别在于 SMP 多核在 Debug 时，不同的 thread 共用一个 Trace Configuration，且需要通过选择不同的 Thread 来对不同的 CPU Hart 核心单独 `start trace/stop trace`。在 Debug 视图中，点击任意一个 Thread, 然后点击 Trace 工具栏中的 `trace setting` 来设置 Trace Configuration。



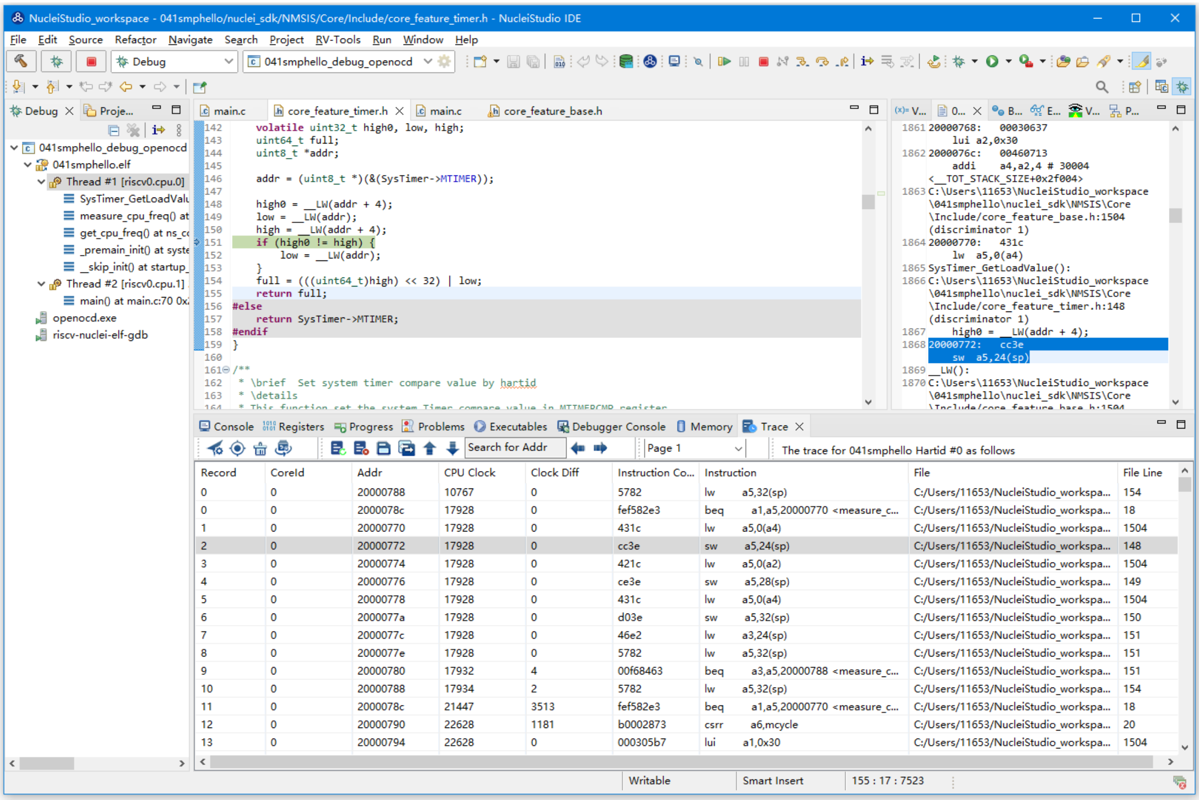
在 Debug 视图中，可以通过点击不同的 Thread, 来切换不同的 Core, 如下图点击 Thread #1 或者 Thread #1 下对应的函数名来选中对应的是 SMP 多核应用中的 Core 0, 可以对 Core 0 开启或者关闭 Trace，在 SMP 多核应用中，只要有一个 Core 在完成 start trace 操作时,Trace Configuration 中的信息就会在硬件中设置好，其他的 core 在 `start trace` 操作时，就不会重复设置 trace Configuration。



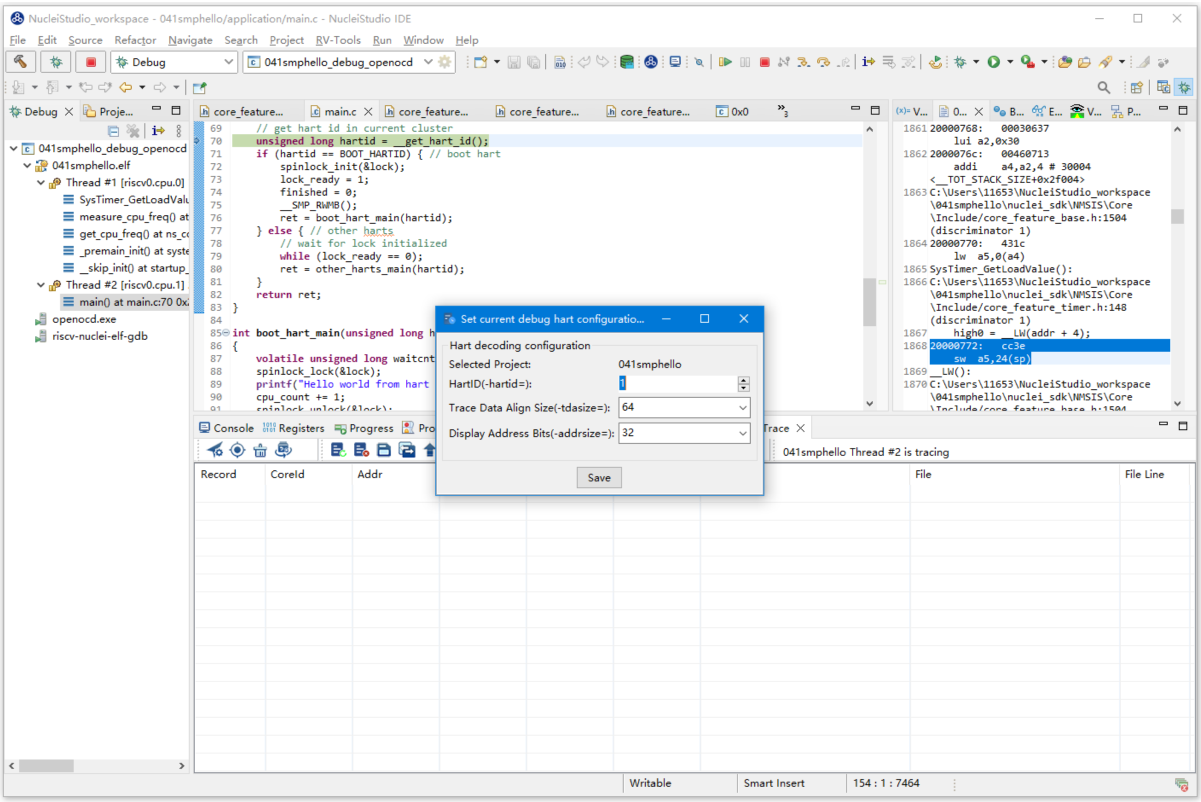
同理，在 Debug 视图中点击 Thread #2 或者 Thread #2 下对应的函数名，来切换到 Core 1 上进行 `start trace/stop trace` 的操作。



在 `dump trace file` 操作时，在 SMP 多核应用中，只有当所有的 Core 都 `stop trace`，才可以执行 `dump trace file` 的指令并成功下载 Trace 文件。Trace 文件的下载，在 SMP 多核应用中，只需要下载一份，在对 trace 文件进行 decode 时，注意设置 Hart ID，就可以解析出不同的 trace 记录表，如下图，当 `HardID=0` 时，就可以查看到 Core 0 对应的 Trace 记录。

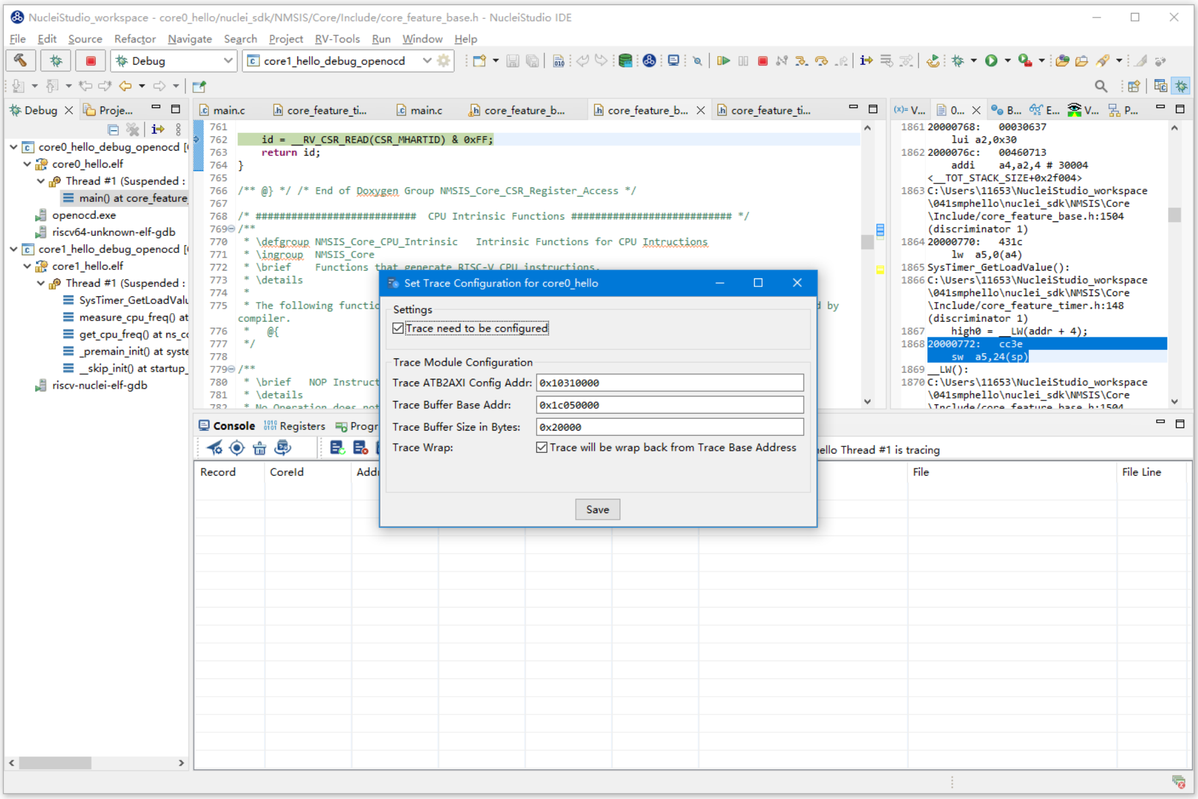


同理当 HardID=1 时，就可以查看到 Core 1 对应的 Trace 记录。

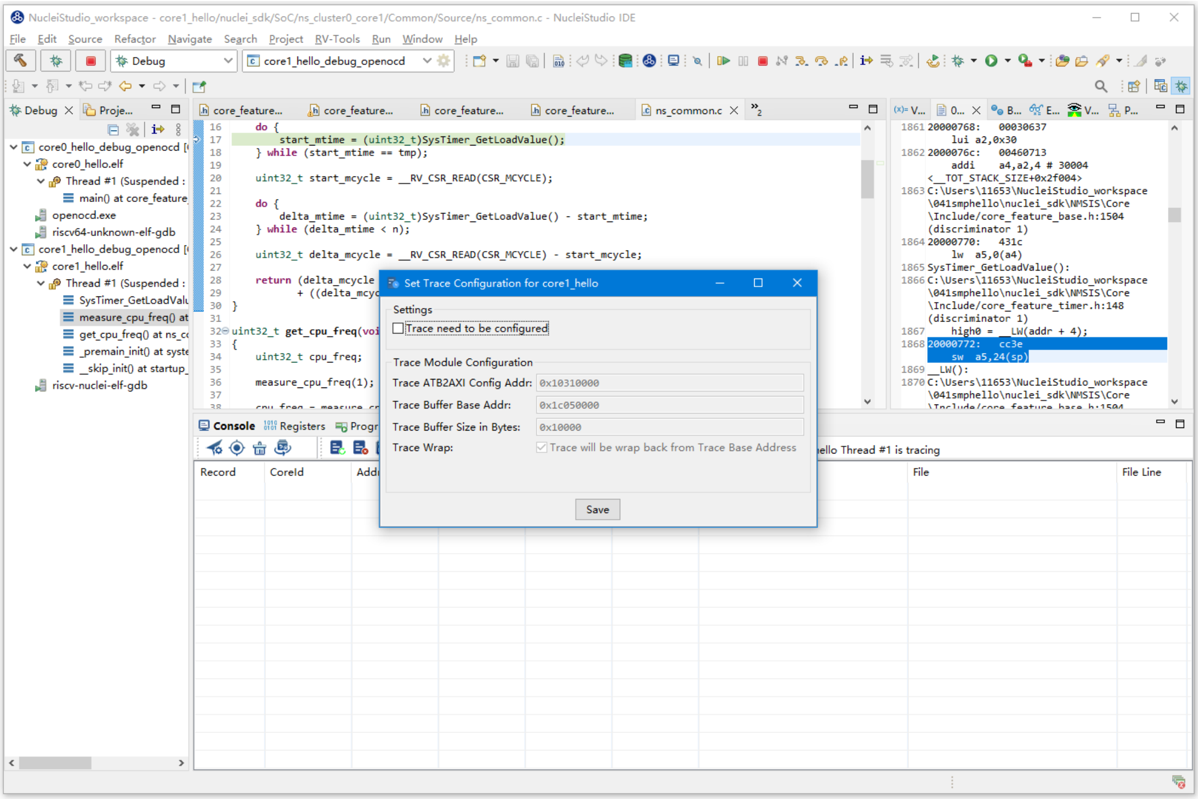


在 AMP 多核应用中使用 Trace

在 AMP 多核应用中使用 trace 也类似，trace 配置也是共享。不同的 thread 共用一个 trace configuration，但可以通过不同的 thread，对不同的核单独 `start trace/stop trace` 。如下图，在 Debug 视图，点击 `Thread #1` 或者 `Thread #1` 下的函数名，切换到 AMP 多核应用中 `Core 0` ，然后点击 Trace 工具栏中的 `trace setting` 来设置 Core 0 对应的 Trace Configuration 。



在 Debug 视图，点击 `Thread #2` 或者 `Thread #2` 下的函数名，切换到 AMP 多核应用中 `Core 1` ，然后点击 Trace 工具栏中的 `trace setting` 来设置 `Core 1` 对应的 Trace Configuration，因为在 AMP 多核应用中 trace 配置是共用，所以此处设置需要将 `Trace need to be configured` 的勾去掉，表示可以使用 trace 功能，但不需要有任何设置。

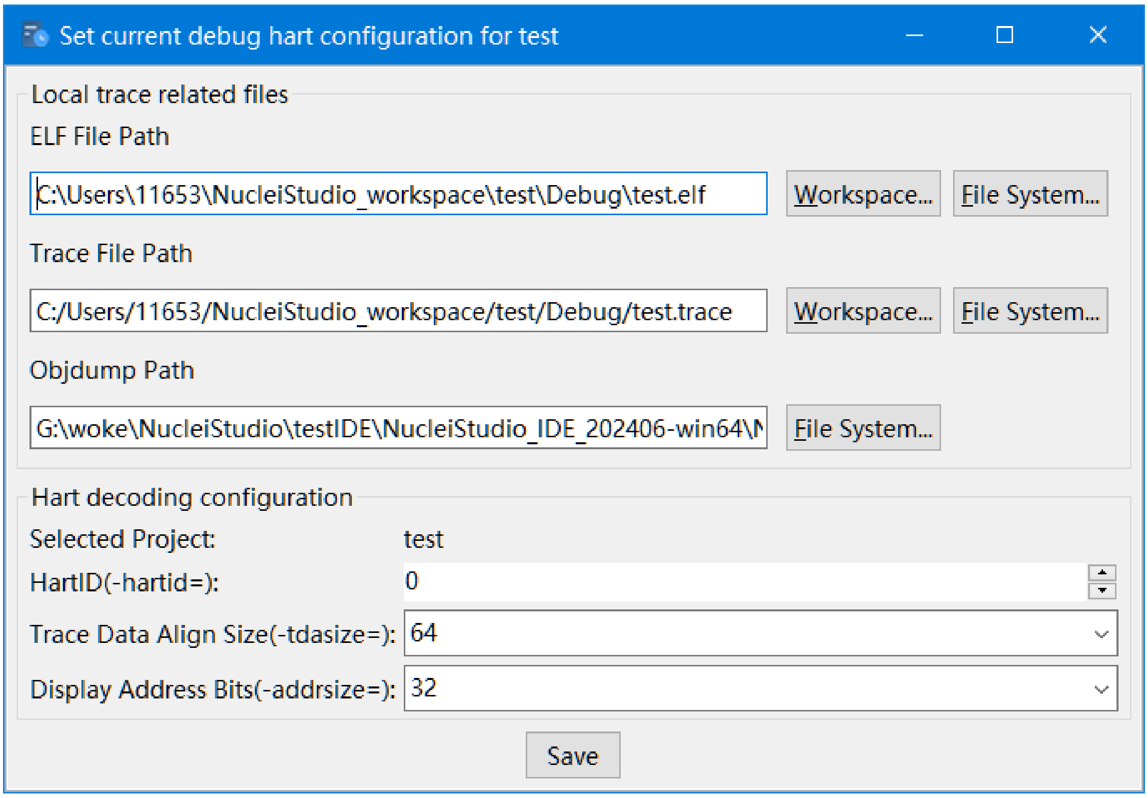


Trace Configuration 设置完成后，同样的通过 Debug 视图的 Thread 来切换不同的 Core，进行 `start trace/stop trace/dump trace file` 操作, 注意，设置了 Trace Configuration 的 Core 需要优先于其它 Core 开始 `start trace`，并将 Trace Configuration 的信息设置好，其他的 Core 才可以正常的 `start trace/stop trace/dump trace file` 操作。

在 `dump trace file` 操作时，在 AMP 多核应用中，请确定所有的 Core 都 `stop trace`，才执行 `dump trace file` 的指令，否则可能在某一下 Core 在 `dump trace file`，其他的 Core 还在记录 trace，最后得到的 Trace 文件并与预期不符。Trace 文件下载，在 AMP 多核应用中，需要每一个工程应用单独 dump 一份 trace 文件，其实 dump 到的 trace 文件内容是一样的，在对 trace 文件进行 decoder 时，同样需要注意设置 `Core Hart ID`，就可以解析出对应的 trace 记录表。其他操作与上文内容中所述类似。

查看脱机 Trace

在某些场景下，用户可能通过命令行或其他方式，得到了一个 trace 文件，这时只需打开 `Set Current Debug hart Configuration`，并按要求配置好参数，即可通过 NucleiStudio 的 trace 工具解析这个 trace 文件了。



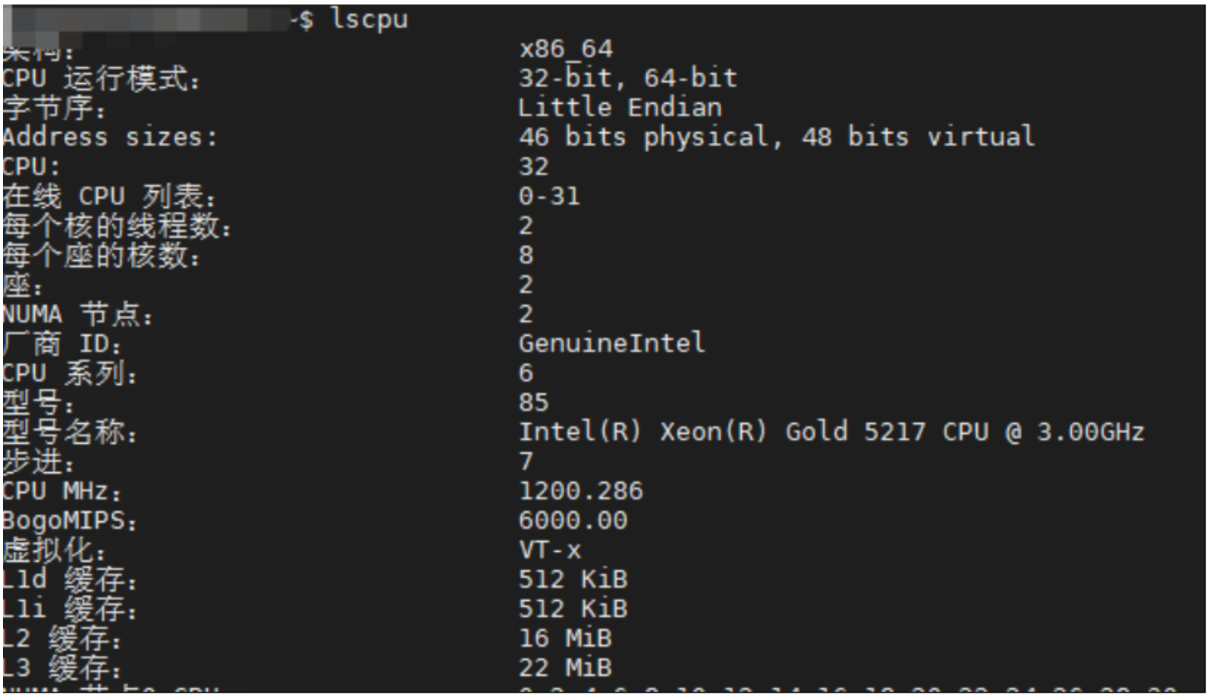
2.10.5 RVProf 功能的使用

RVProf 是芯来科技针对 cpu cycle model 开发的性能分析工具，Nuclei Studio 在 2024.02.dev 版本中，完成对 RVProf 的支持。在实际使用中，RVProf 功能分三步完成，首先通过 Cycle model 工具，运行代码，产生 `.rvtrace` 文件，然后 RVProf 工具，将 `.rvtrace` 解析成对应的 `.json` 文件，最后通过 google 的开源工具 Perfetto Trace Viewer 对 `.json` 文件进行解析并展示。因为 cpu cycle model 当前仅提供了 linux 版本，所以本文档均是在 linux 环境下演示此功能。

在使用过程，如有问题，可以查看 https://github.com/Nuclei-Software/nuclei-studio 相关内容，也可以向我们提交相关 issue。

测试环境

cpu cycle model 在运行过程中，对硬件环境的性能要求较高，在实际使用，四核及以上的系统中运行效果较好，一般不建议在虚拟机环境下使用。为了较好的体验效果，本测试在工作站上进行。



```
[redacted]-$ lscpu
架构:                            x86_64
CPU 运行模式:                    32-bit, 64-bit
字节序:                          Little Endian
Address sizes:                   46 bits physical, 48 bits virtual
CPU:                             32
在线 CPU 列表:                   0-31
每个核的线程数:                  2
每个座的核数:                    8
座:                              2
NUMA 节点:                       2
厂商 ID:                         GenuineIntel
CPU 系列:                        6
型号:                            85
型号名称:                        Intel(R) Xeon(R) Gold 5217 CPU @ 3.00GHz
步进:                            7
CPU MHz:                         1200.286
BogoMIPS:                        6000.00
虚拟化:                          VT-x
L1d 缓存:                        512 KiB
L1i 缓存:                        512 KiB
L2 缓存:                         16 MiB
L3 缓存:                         22 MiB
```

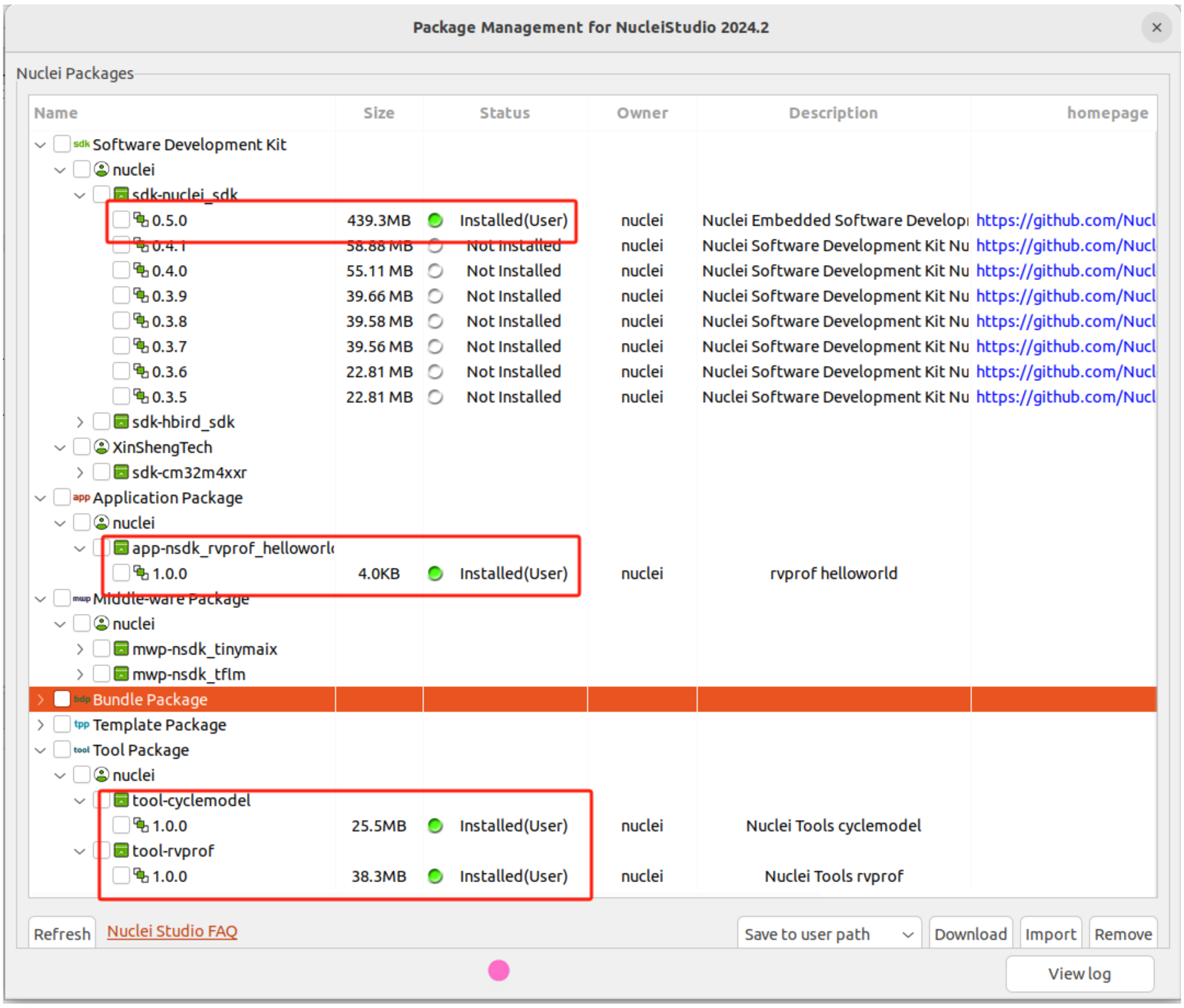
准备测试 NPK 软件或者工具包

目前此功能仅提供测试用的 NPK 包，将相关的包安装到 Nuclei Studio 中，关于安装 NPK 包，可以查看 Nuclei Studio 手册中相关章节，因为 RVProf 测试包没有公开，请联系我们索取。

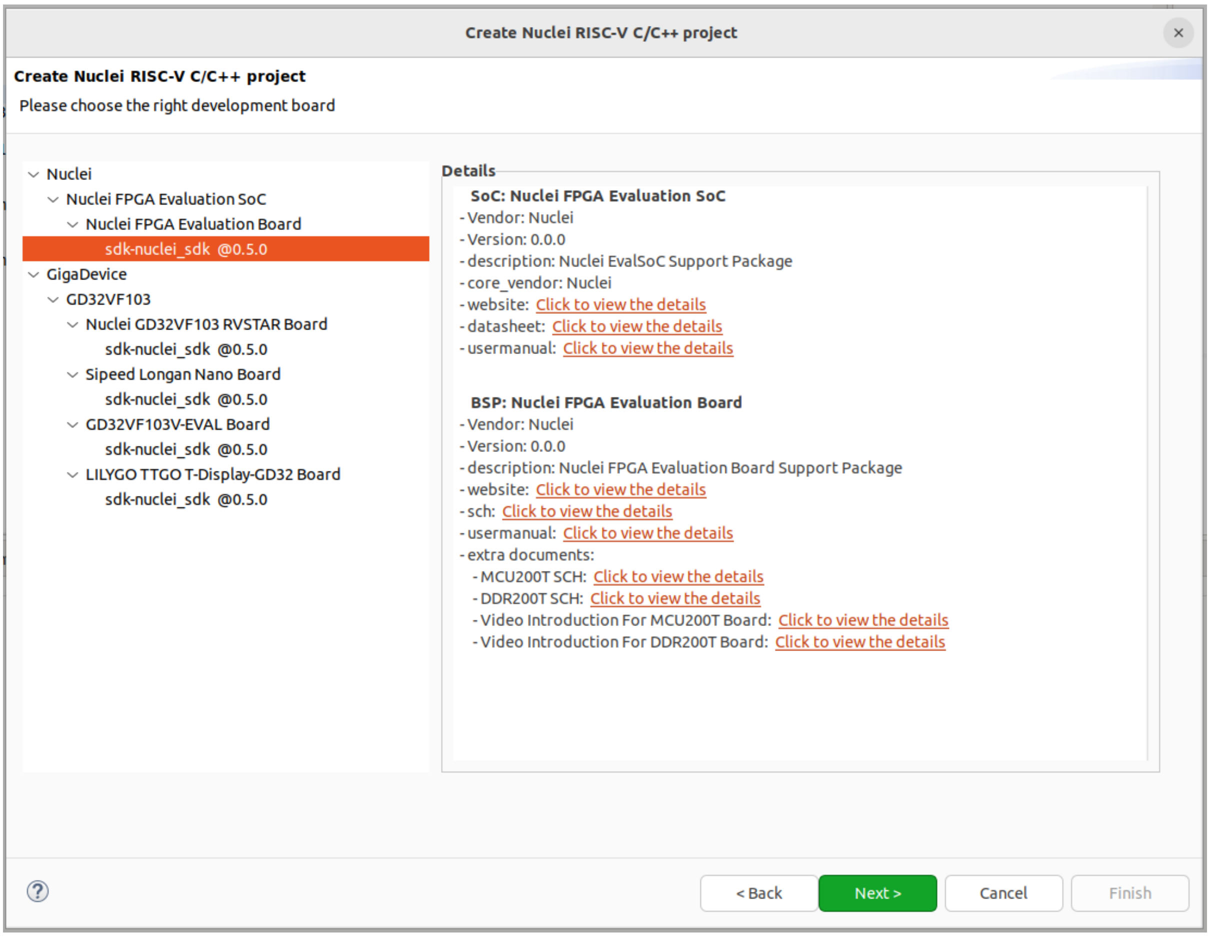
- cymodel.zip cymodel 的 NPK Tools 包
- rvprof.zip RVProf 的 NPK Tools 包
- Rvprof helloworld.zip 测试 demo NPK App 包

创建 rvprof 测试工程

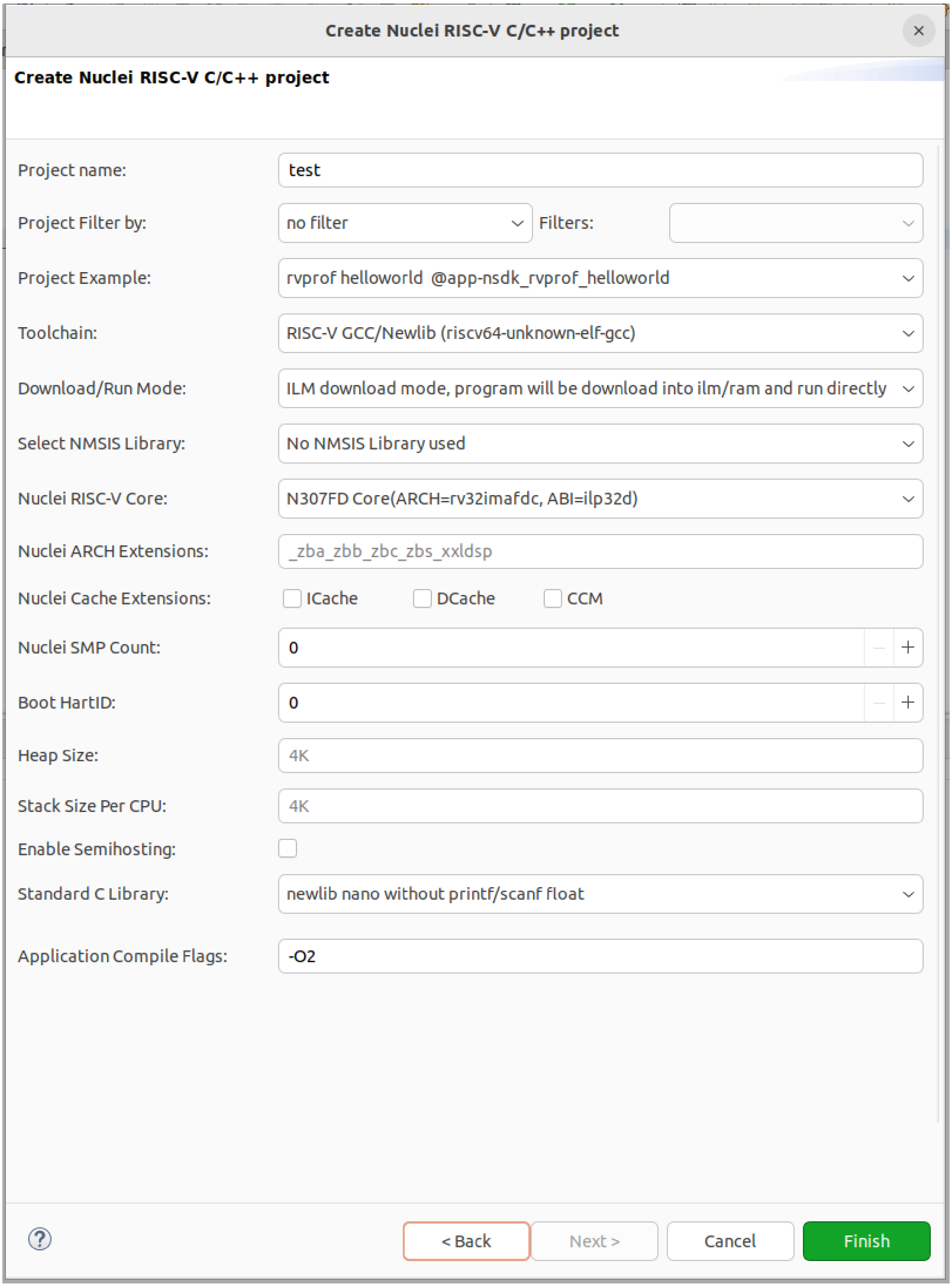
创建工程前，先查看 Nuclei Package Management 中 NPK 是否安装正确，因为测试 demo 是依赖于 nuclei_sdk，所以也要先安装 sdk-nuclei_sdk，具体如下：



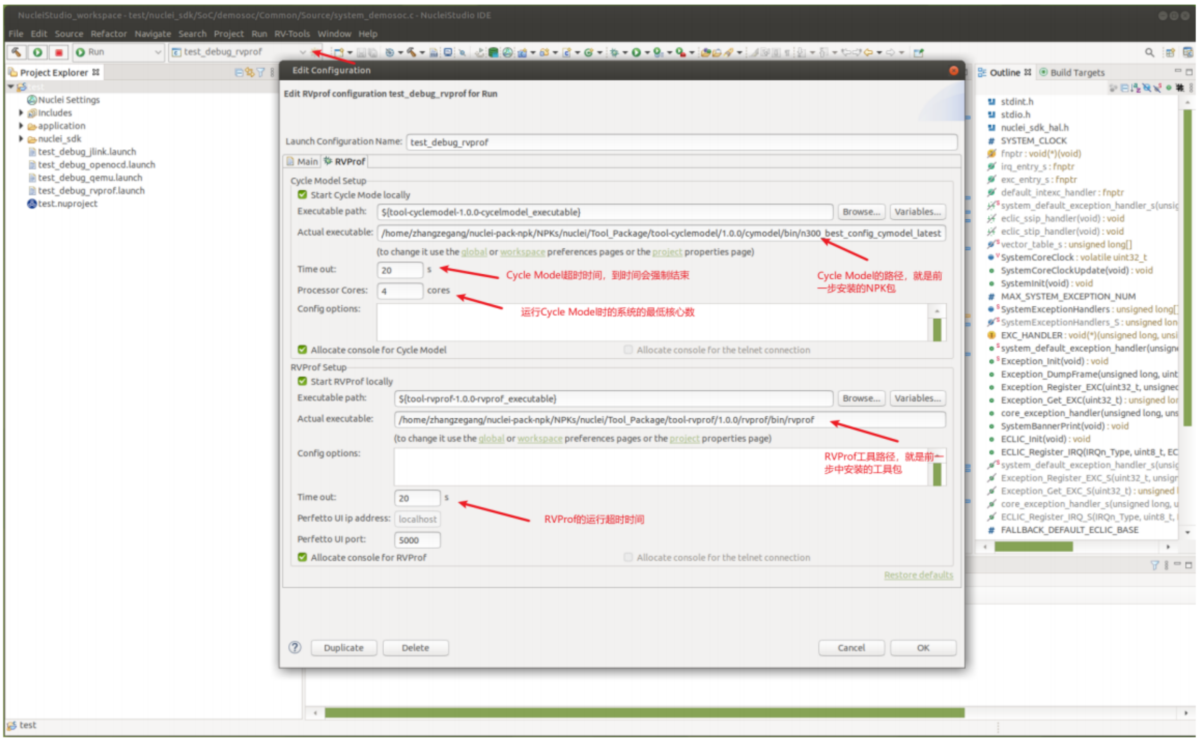
然后创建一个 test 测试工程, 在创建工程的向导中，依次 `New Nuclei RISC-V C/C++ Project -> sdk-nuclei_sdk@0.5.0 -> next`，在工程配置页面，依次填写工程名、选择 Project Example : `rvprof helloworld@app-nsdkrvprof_helloworld` ,Nuclei RISC-V Core: `N307FD` （这里的 code 要跟 cpu cycle model 对应）。



在 Project Example 可以看到我们导入的 demo NPK App 中的 Rvprof helloworld 工程，选择此工程，然后下一步，完成工程的创建。



在创建的 test 工程中，可以看到多了一个 `test_debug_rvprof.launch` 文件，rvprof 相关的配置在此文件中，可以查看内容如下。其中 Cycle Model 的 time out 时间，用来设置 Cycle Model 超时时间，因为 Cycle Model 运行时比较耗时，如果工程比较简单，可以设置一个较短的起始时间，到时间后，可以及时中断 Cycle Model 的运行；RVProf 中的超时时间的功能也是类似。

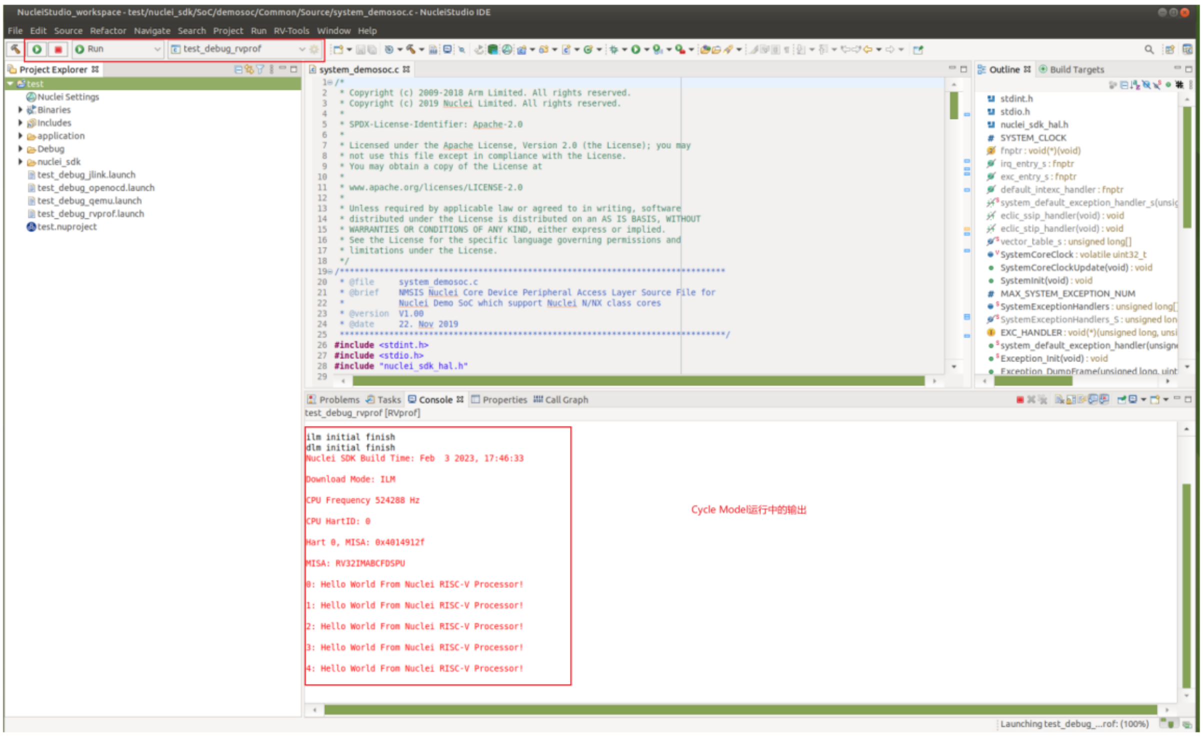


查看 rvprof 的结果

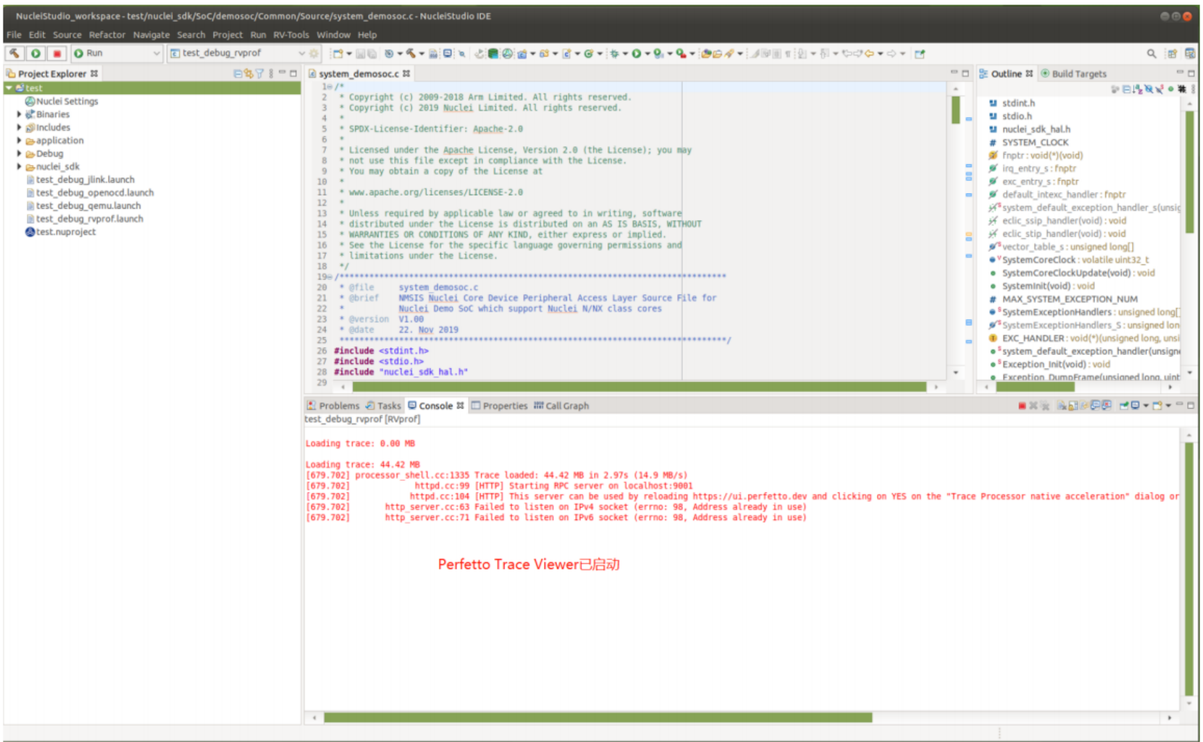
创建完工程后，在 Nuclei Studio 的 launch bar 上，选中 `test_debug_rvprof.launch`，并点击工具栏中的运行按钮，Nuclei Studio 依次完成以下任务，并将最终的结果在在 Perfetto Trace Viewer 中展示。

- 编译工程代码
- 启动 Cycle Model 并产生 trace 文件
- 启动 RVProf 解析 trace 文件生成 json 文件
- 启动 Perfetto Trace Viewer 展示结果

Cycle Model 启动及 log 输出

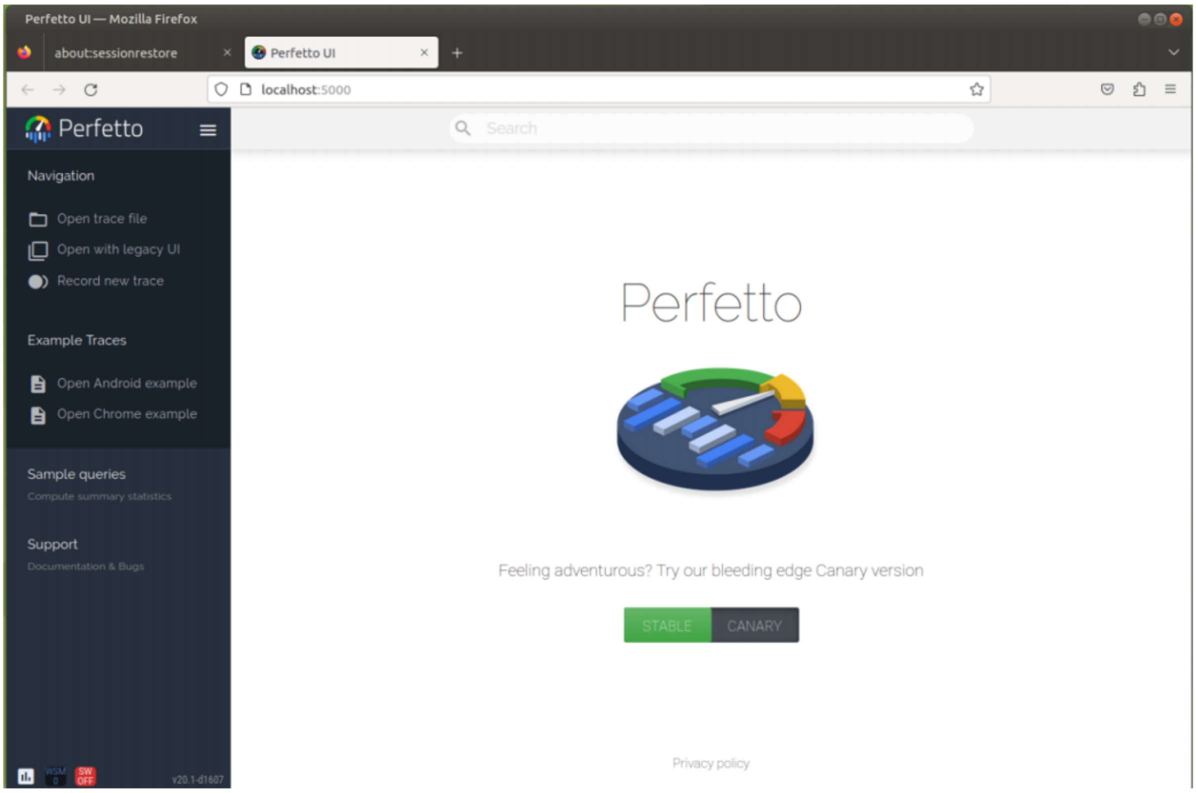


perfetto 启动本地服务

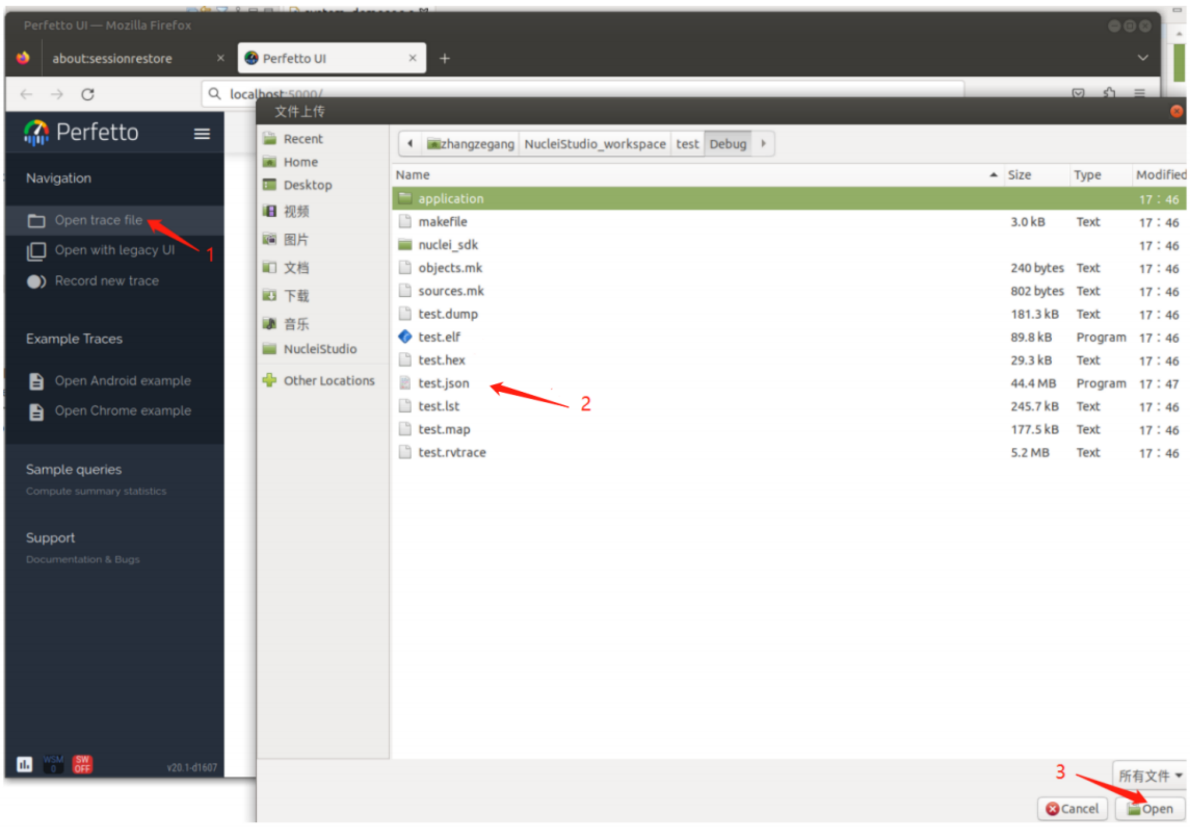


Perfetto Trace Viewer 的官方地址是 https://ui.perfetto.dev/ 。Nuclei Studio 默认会尝试打开 https://ui.perfetto.dev/ ，同时自动载入 json 文件并解析。如果因为网络原因（国外服务器）打开失败，Nuclei Studio 会在本地启一个 Perfetto Trace Viewer 本地服务，并自动打开本地 localhost:5000/，此时需要用户手动载入工程目录下的 `Debug/test.json` 文件。在 Perfetto Trace Viewer 中可以看到 trace 的展示结果。

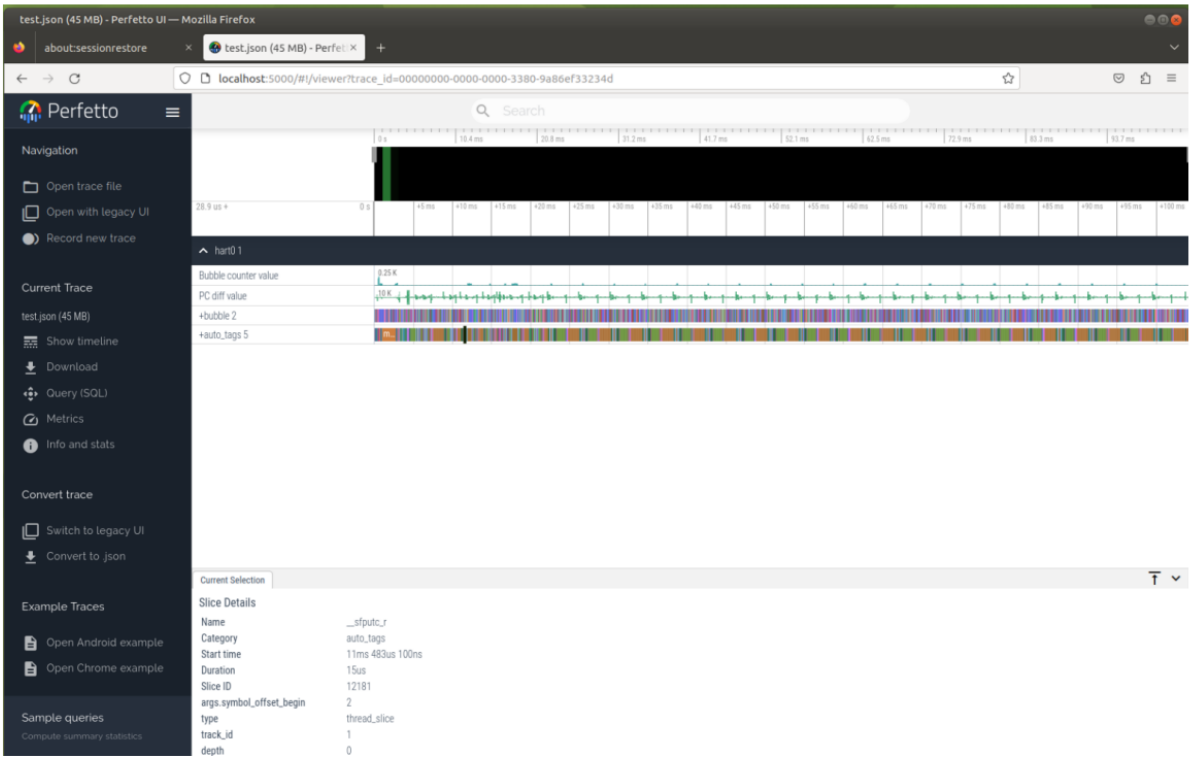
Nuclei Studio 会在本地启一个 web 服务，同时打开 Perfetto Trace Viewer。



点击 Open trace file，找到工程中生成的 json 文件，手动将 json 文件 load 到 Perfetto Trace Viewer 中。



些时，在 Perfetto Trace Viewer 就可以查看到 rvprof trace 结果展示了，用户可以通过键盘的 `W/A/S/D` 按键查看更详细的信息。



2.10.6 Nuclei NICE Wizard

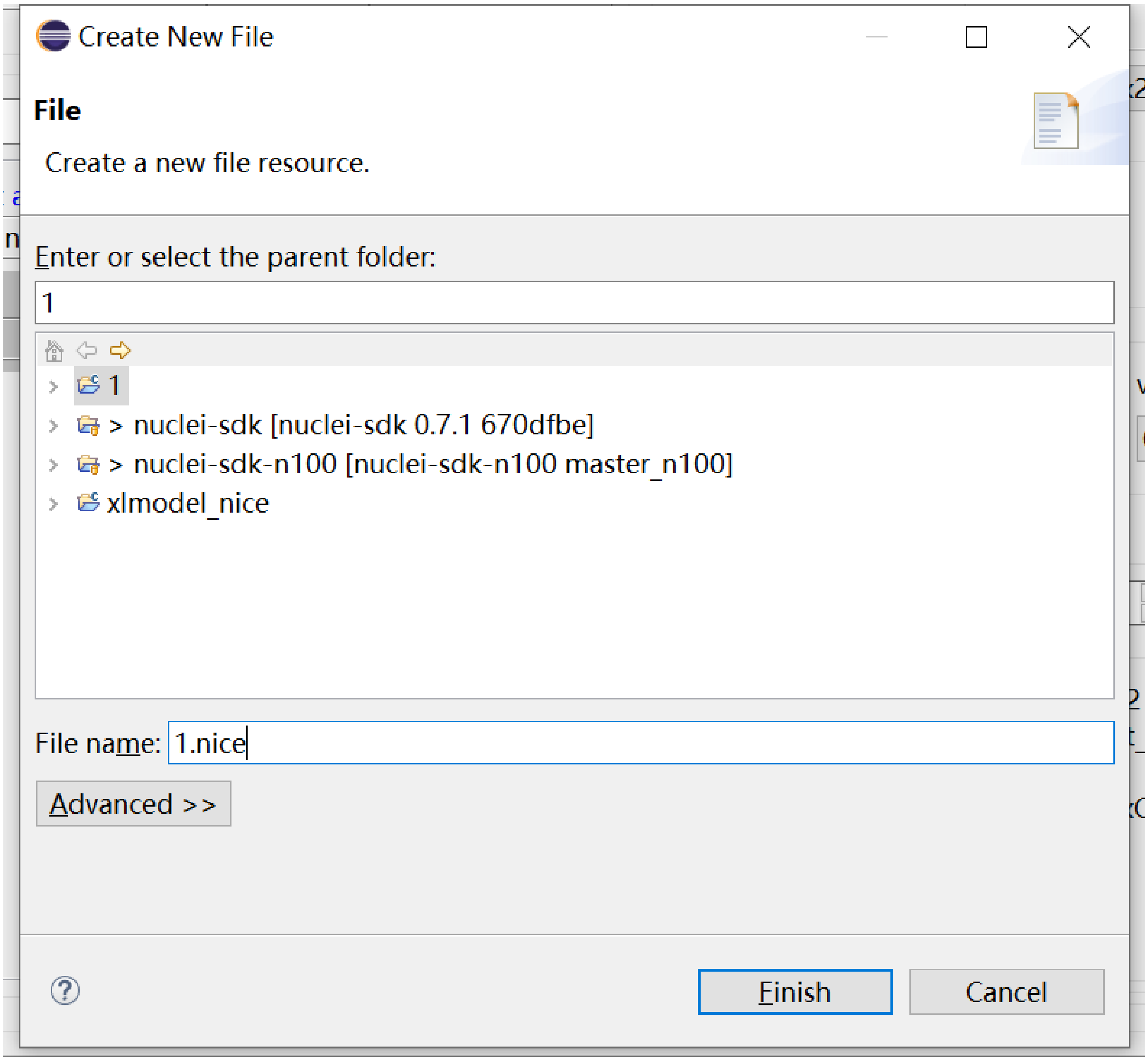
> **Note**
>
> 在 芯来科技视频号中有 **Nuclei NICE Wizard** 的视频，您可以在微信中搜索 芯来科技视频号点击查看相关内容。

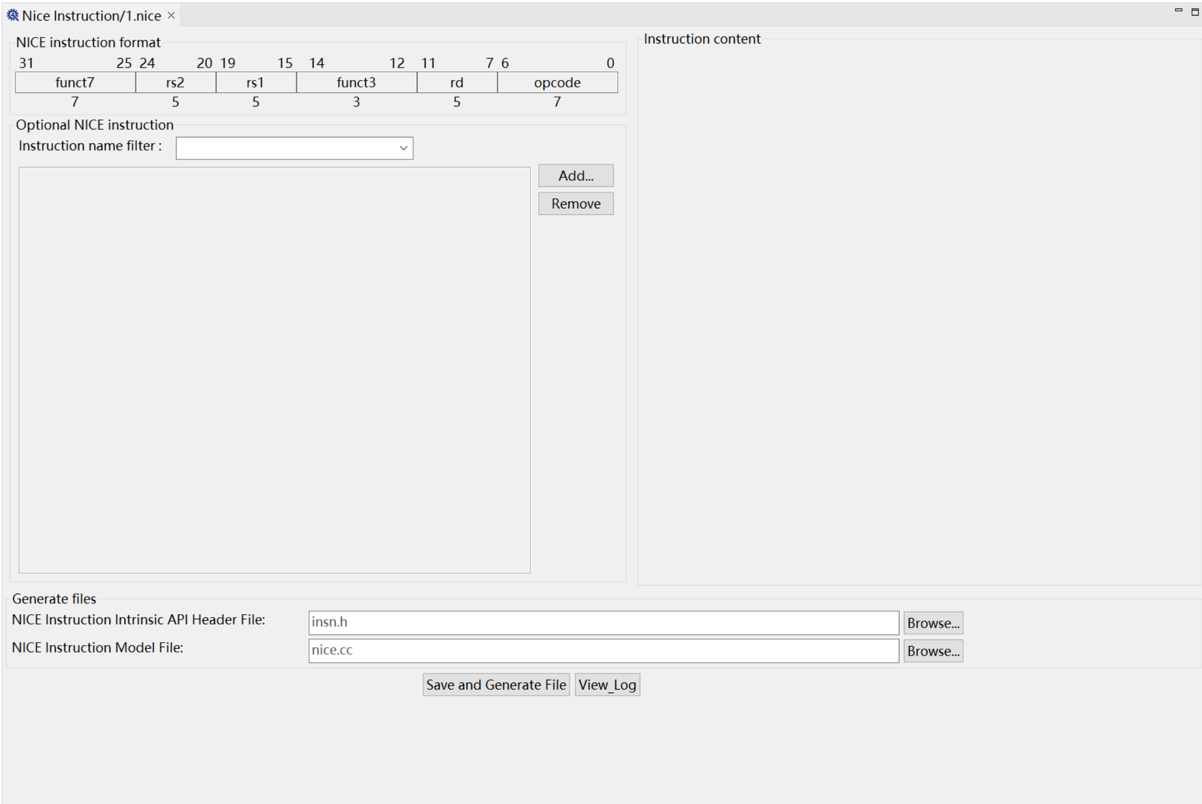
Nuclei NICE Wizard 是一个集成在 Nuclei Studio 上的工具，旨在简化和加速 NICE (自定义指令扩展) 和 VNICE (向量化自定义指令扩展) 指令的创建过程。它允许用户通过图形界面快速配置并生成自定义指令所需的代码框架，从而实现对特定应用算法的硬件加速。具体来说：

- 简化开发流程：减少从构思到实现自定义指令的时间。
- 提高效率：通过生成优化后的指令代码，提高应用程序的执行效率。
- 易于集成：生成的代码可以直接整合到现有项目中，减少了额外的工作量。

创建.nice 文件，打开 Nuclei NICE Wizard

在 Nuclei Studio 中打开目标工程，并在项目根目录下创建一个 `*.nice` 文件（例如 aicc.nice），双击打开 Nuclei NICE Wizard。

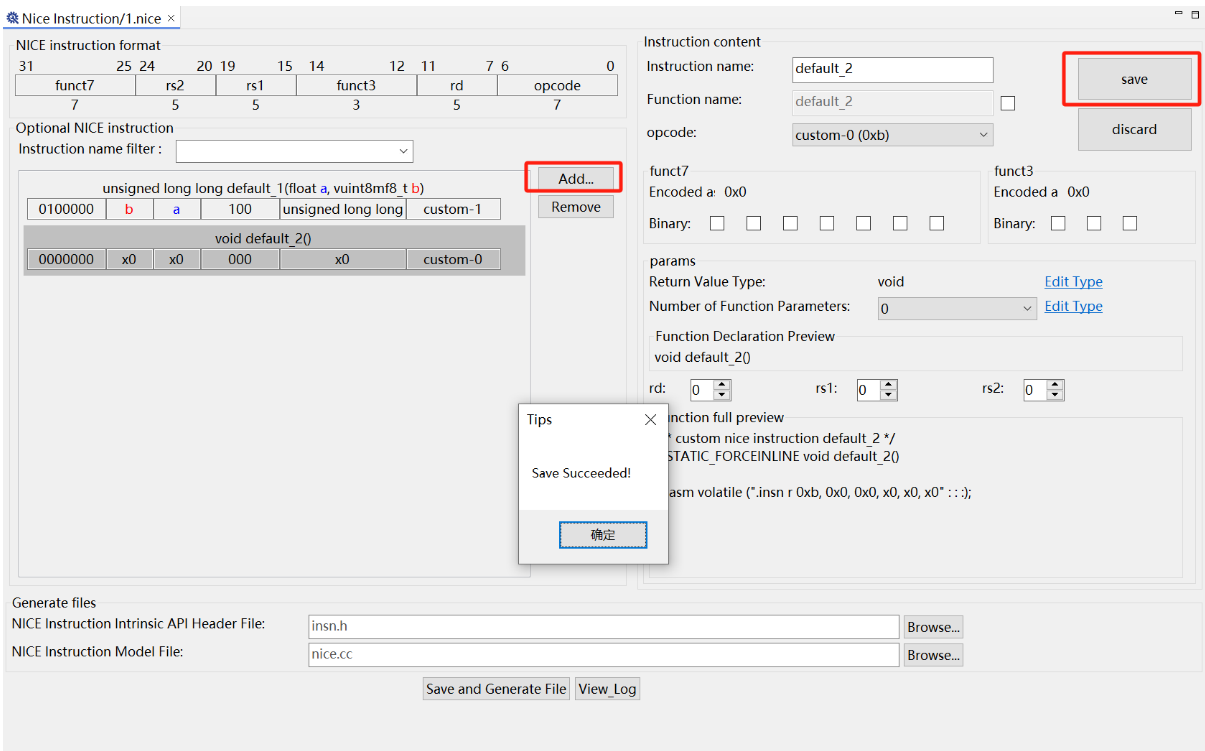




新增指令

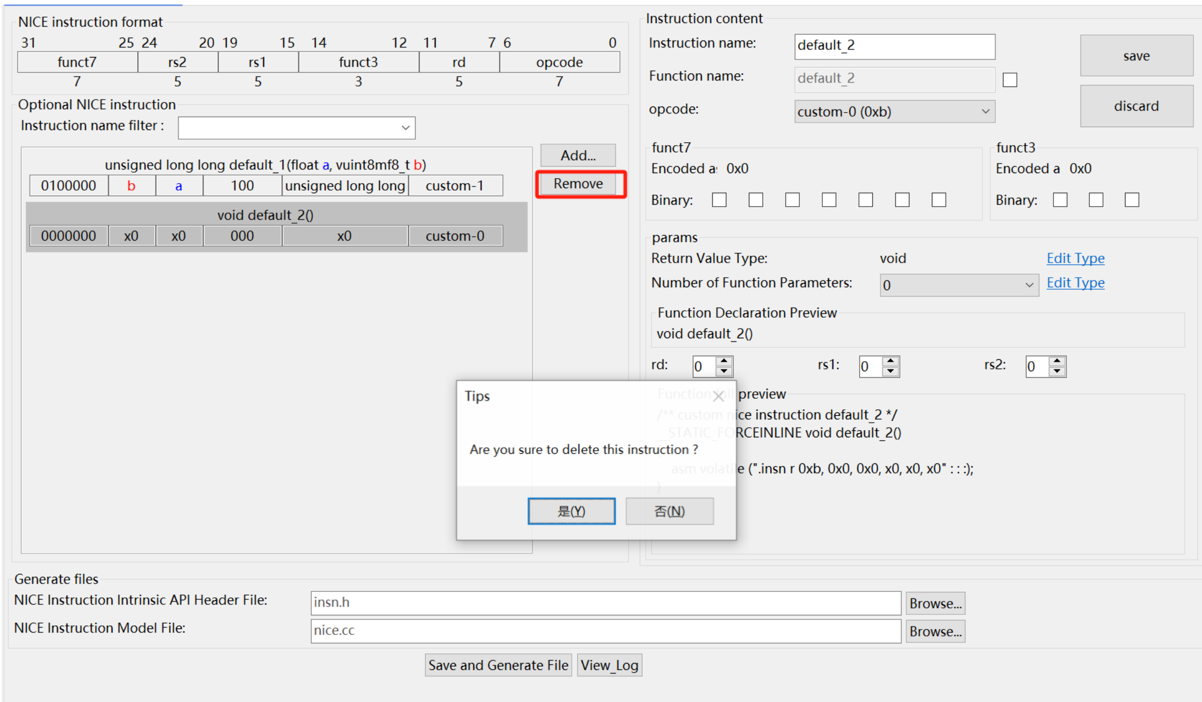
点击 Add...，根据需要修改指令内容后，点击右上角 save 即可。

这里举例先创建两条指令，同时左侧被选中的指令会变灰，对应内容显示在右侧。



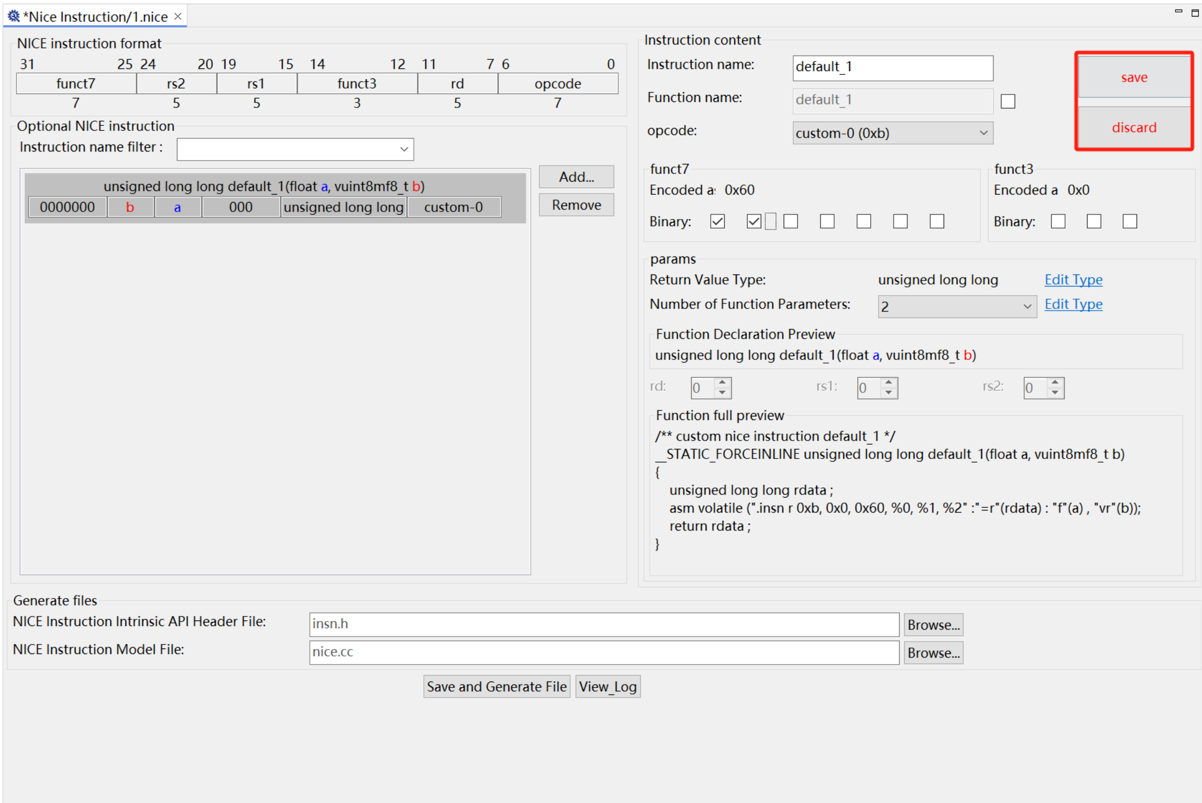
删除指令

左侧选择对应指令，点击 Remove，确认后删除对应指令。



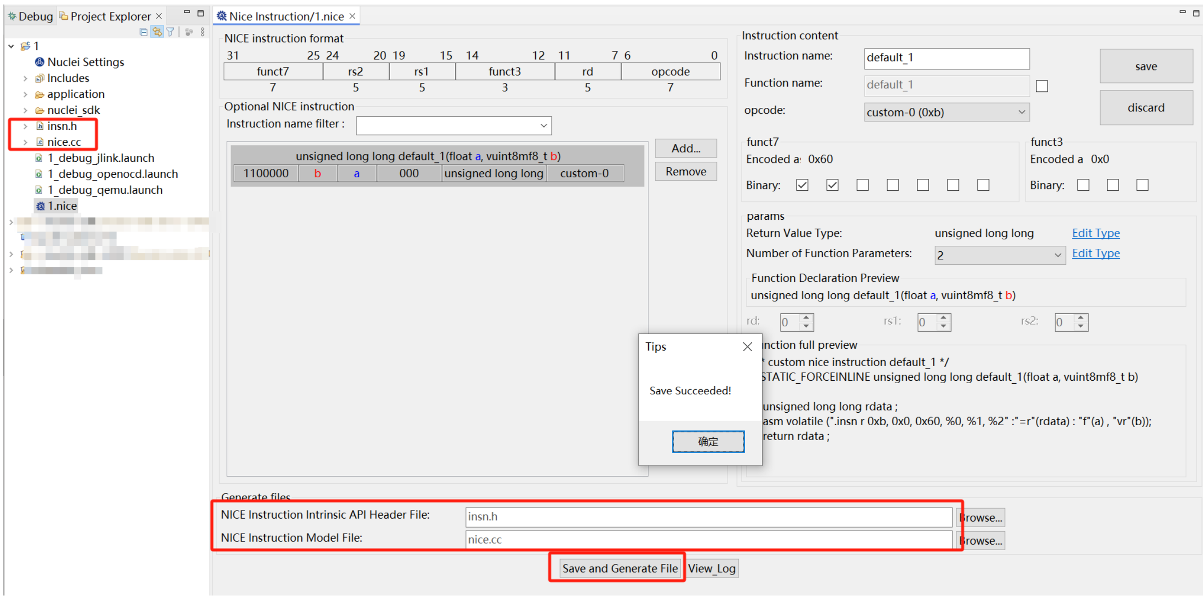
修改指令

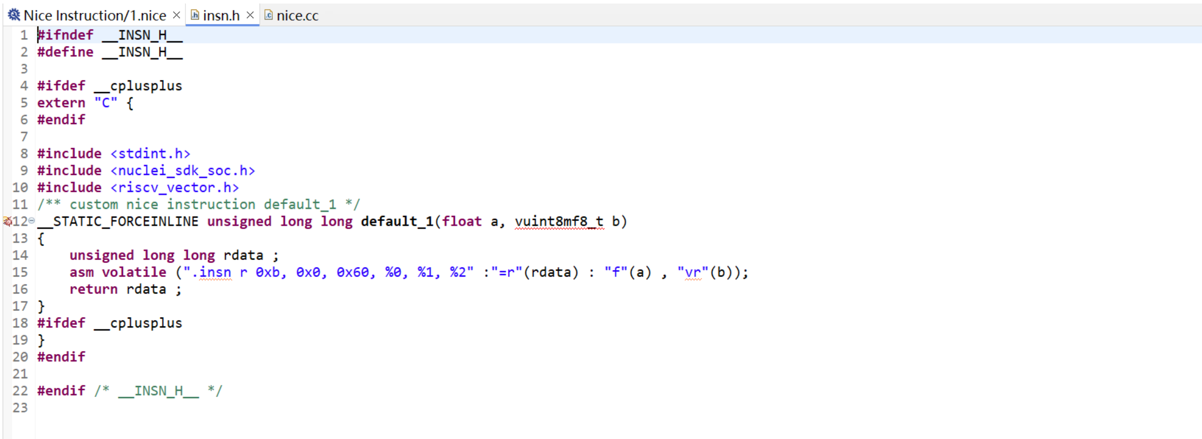
左侧选择对应指令，修改指令内容后，右上 save 和 discard 按钮变红，可保存修改或放弃修改。



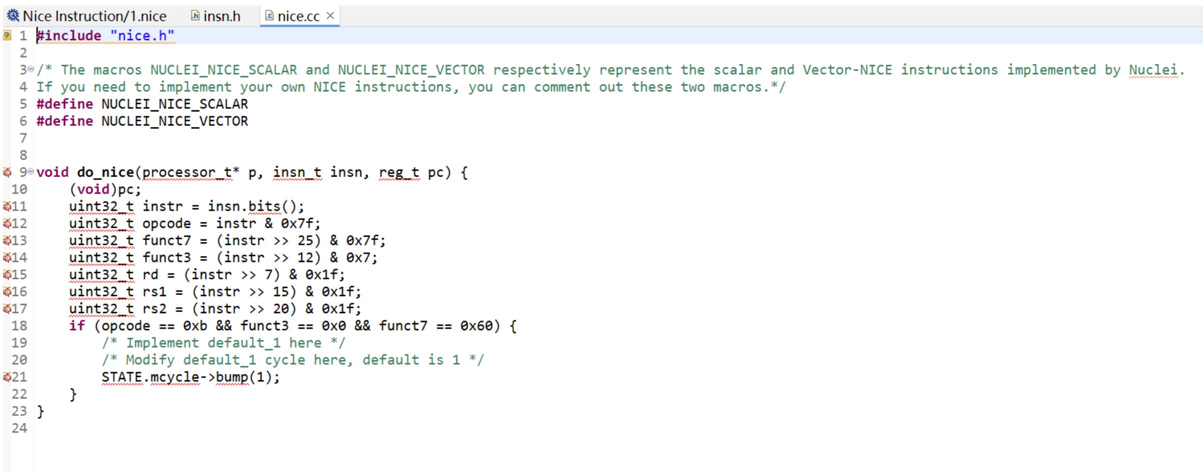
文件生成

可定义 insn.h（包含内嵌汇编头文件）和 nice.cc（包含指令实现逻辑）文件的保存地址，点击 Save and Generate File，会生成对应文件。



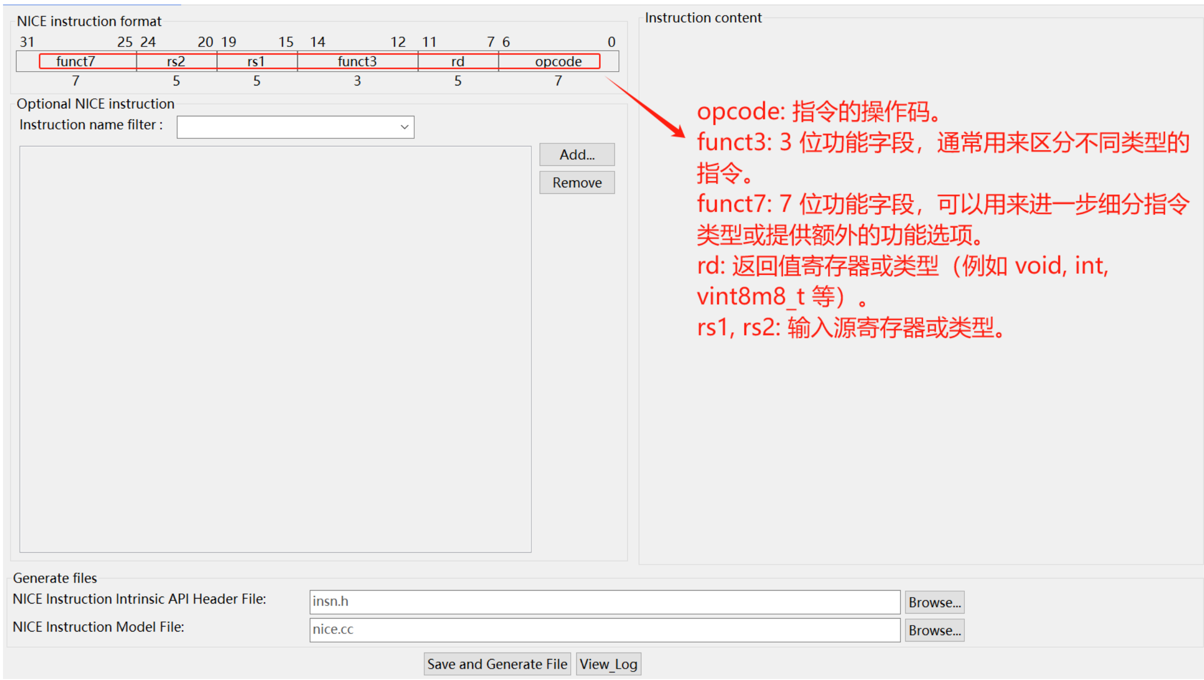


```
#ifndef __INSN_H__
#define __INSN_H__

#ifdef __cplusplus
extern "C" {
#endif

#include <stdint.h>
#include <nuclei_sdk_soc.h>
#include <riscv_vector.h>
/** custom nice instruction default_1 */
__STATIC_FORCEINLINE unsigned long long default_1(float a, vuint8mf8_t b)
{
    unsigned long long rdata ;
    asm volatile (".insn r 0xb, 0x0, 0x60, %0, %1, %2" :"=r"(rdata) : "f"(a) , "vr"(b));
    return rdata ;
}
#ifdef __cplusplus
}
#endif

#endif /* __INSN_H__ */
```



```
#include "nice.h"

/* The macros NUCLEI_NICE_SCALAR and NUCLEI_NICE_VECTOR respectively represent the scalar and Vector-NICE instructions implemented by Nuclei.
If you need to implement your own NICE instructions, you can comment out these two macros.*/
#define NUCLEI_NICE_SCALAR
#define NUCLEI_NICE_VECTOR


void do_nice(processor_t* p, insn_t insn, reg_t pc) {
    (void)pc;
    uint32_t instr = insn.bits();
    uint32_t opcode = instr & 0x7f;
    uint32_t funct7 = (instr >> 25) & 0x7f;
    uint32_t funct3 = (instr >> 12) & 0x7;
    uint32_t rd = (instr >> 7) & 0x1f;
    uint32_t rs1 = (instr >> 15) & 0x1f;
    uint32_t rs2 = (instr >> 20) & 0x1f;
    if (opcode == 0xb && funct3 == 0x0 && funct7 == 0x60) {
        /* Implement default_1 here */
        /* Modify default_1 cycle here, default is 1 */
        STATE.mcycle->bump(1);
    }
}
```

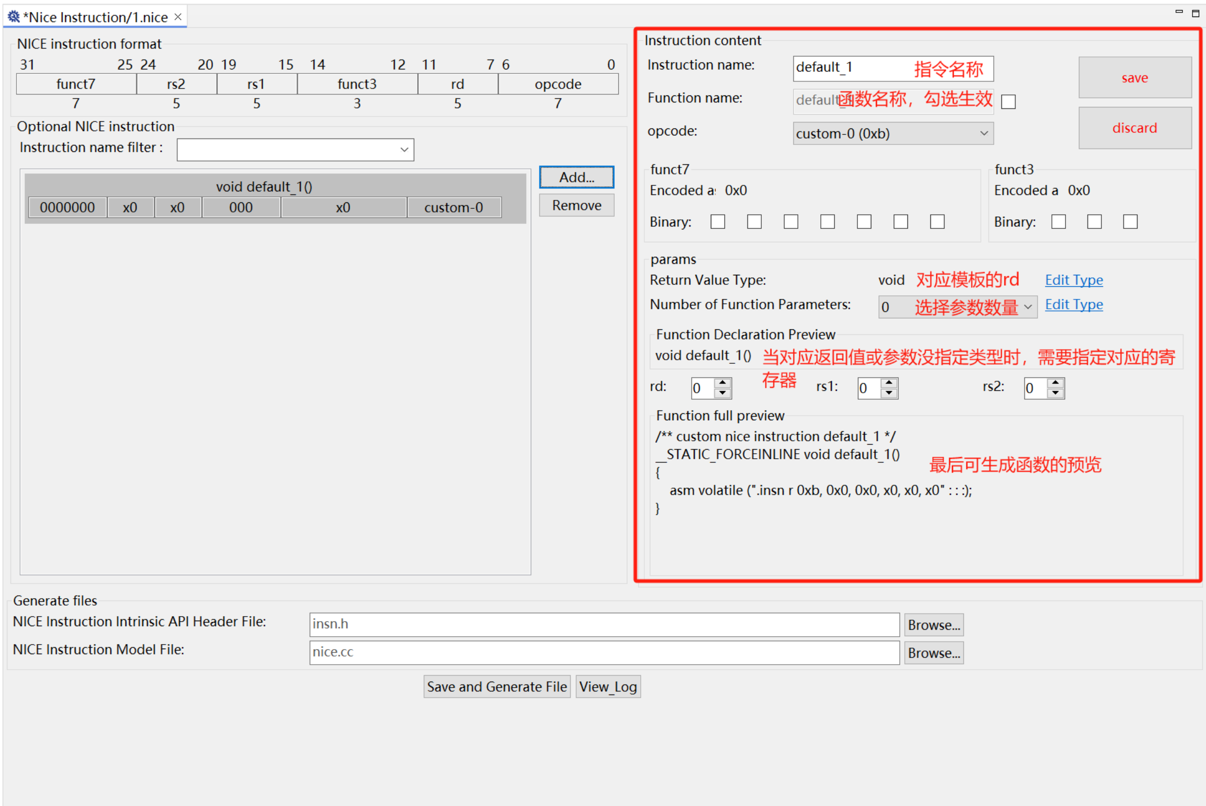
NICE 指令模板说明



单个指令模板如上图所示，

- opcode: 可选 custome-0,custome-1,custome-2,custome-3
- funct3: 3 位功能字段，通常用来区分不同类型的指令。
- funct7: 7 位功能字段，可以用来进一步细分指令类型或提供额外的功能选项。
- rd: 返回值寄存器或类型（例如 void, int, vint8m8_t 等）。
- rs1, rs2: 输入源寄存器或类型。

指令内容编辑说明



如上图，Instruction content 显示默认内容。

- **Instruction name** ：指令名称，具体定义规范如下
  – 字母和数字：函数名可以包含字母（`A-Z`，`a-z`）和数字（`0-9`），但是不能以数字开头。
  – 下划线：函数名中可以使用下划线 `_` 来提高可读性，尤其是在多单词组合的情况下。例如，`get_user_name` 是一个有效的函数名，`<` ，`>` ，`…`，`?` ，`/` 都不允许出现在函数名中。
  – 特殊字符：除了下划线以外，其他特殊字符如 `!` ，`@` ，`#` ，`$` ，`%` ，`^` ，`&` ，`*` ，`(` ，`)` ，`{` ，`}` ，`[` ，`]` ，`\` ，`:` ，`;` ，。
  – 关键字：函数名不能是 C 语言的关键字或保留字，比如 `int` ，`char` ，`float` ，`double` ，`if` ，`else` ，`while` ，`for` ，`return` 等等。
- **Function name** ：函数名称，在不勾选的情况下生成的对应函数名为指令名称，命名规范与 `Instruction name` 相同。
- **funct7** ：对应模板的 `funct7`，可通过勾选 Binary 对应项设置。
- **funct3** ：对应模板的 `funct3`，可通过勾选 Binary 对应项设置。
- **Return Value Type** ：对应模板的 rd，可点击 Edit Type 进行设置，如果 rd 为 void。
- **Number of Function Parameters** ：参数个数，可设置传入参数 rs1、rs2 以及 rs3（rs3 既为参数也为返回值）的对应类型。
  – 参数为 0 时，Edit Type 不可设置，rs1 和 rs2 可在下方指定寄存器，如 rd 为 void 类型，rd 也可在下方指定寄存器。
  – 参数为 1 时，Edit Type 可设置 rs1 类型，rs2 可在下方指定寄存器，如 rd 为 void 类型，rd 也可在下方指定寄存器。
  – 参数为 2 时，Edit Type 可设置 rs1、rs2 类型，如 rd 为 void 类型，rd 也可在下方指定寄存器。
  – 参数为 3 时，Edit Type 可设置 rs1、rs2、rs3 类型。

2.10.7 Nuclei Model 功能的使用

芯来科技为 Nuclei Near Cycle Model 开发了专门的运行工具——Model。自 Nuclei Studio 2024.06 版本起，Nuclei Near Cycle Model 最初是通过 RVProf 工具运行的。随着 Nuclei Near Cycle Model 的不断迭代和发展，为了提供更简洁高效的用户体验，我们在 RVProf 的基础上进行了功能简化，推出了新的 Model 工具。

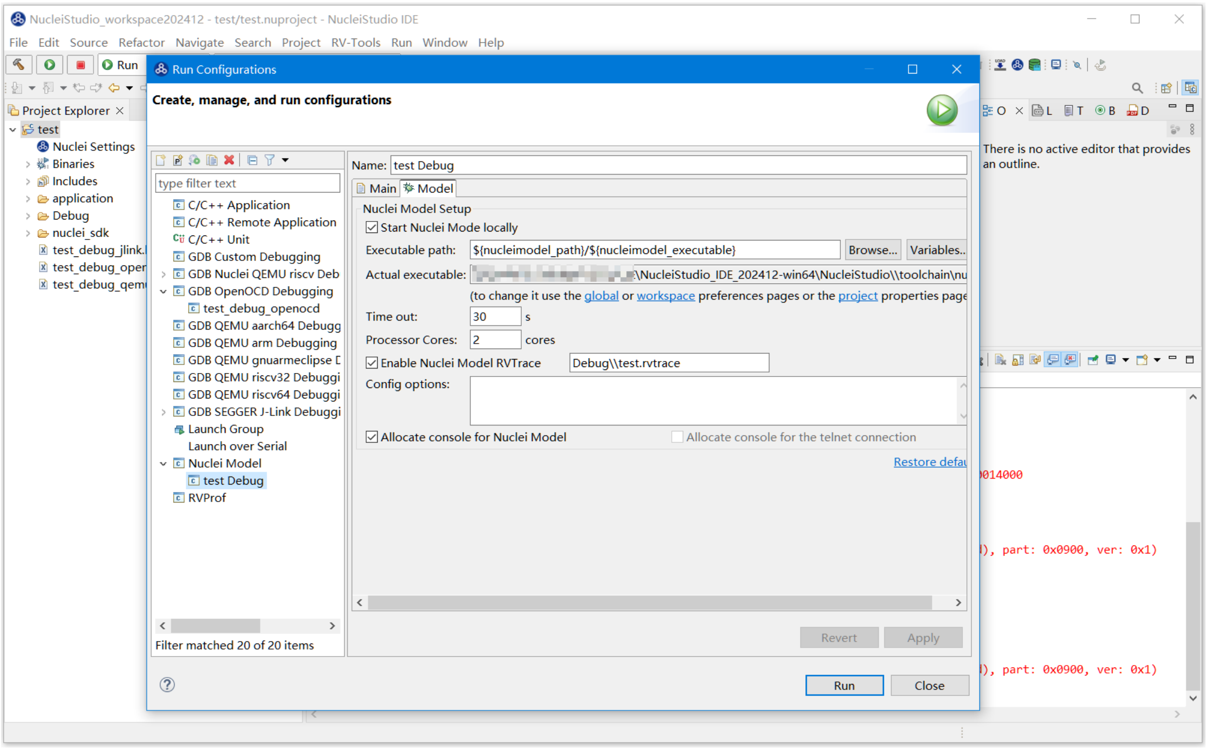
新工具的主要特点包括：

简化功能：移除不必要的复杂功能，使用户能够更专注于 Nuclei Near Cycle Model 的核心功能。

提升效率：优化操作流程，减少用户配置和使用的时间成本。

兼容性好：确保与现有工作流无缝集成，同时支持最新的 Nuclei Near Cycle Model 特性。

通过这些改进，用户可以更加高效地利用 Nuclei Near Cycle Model 进行开发和调试。通过 Nuclei Studio 菜单 `Run -> Run Configuration` 打开 Run Configuration，然找后到 `Nuclei Model`，双击 `Nuclei Model` 菜单，就会生成对应工程的配置。



关于 Nuclei Model 的使用，将在 Nuclei Near Cycle Model 章节中详细介绍。

2.10.8 Nuclei Near Cycle Model

在 Nuclei Studio 2024.06 版中，集成了 Nuclei Near Cycle Model，它是由芯来科技自主研发的仿真测试和性能分析工具，可以帮助研发人员在项目初期进行一些必要的仿真测试和程序性能分析。

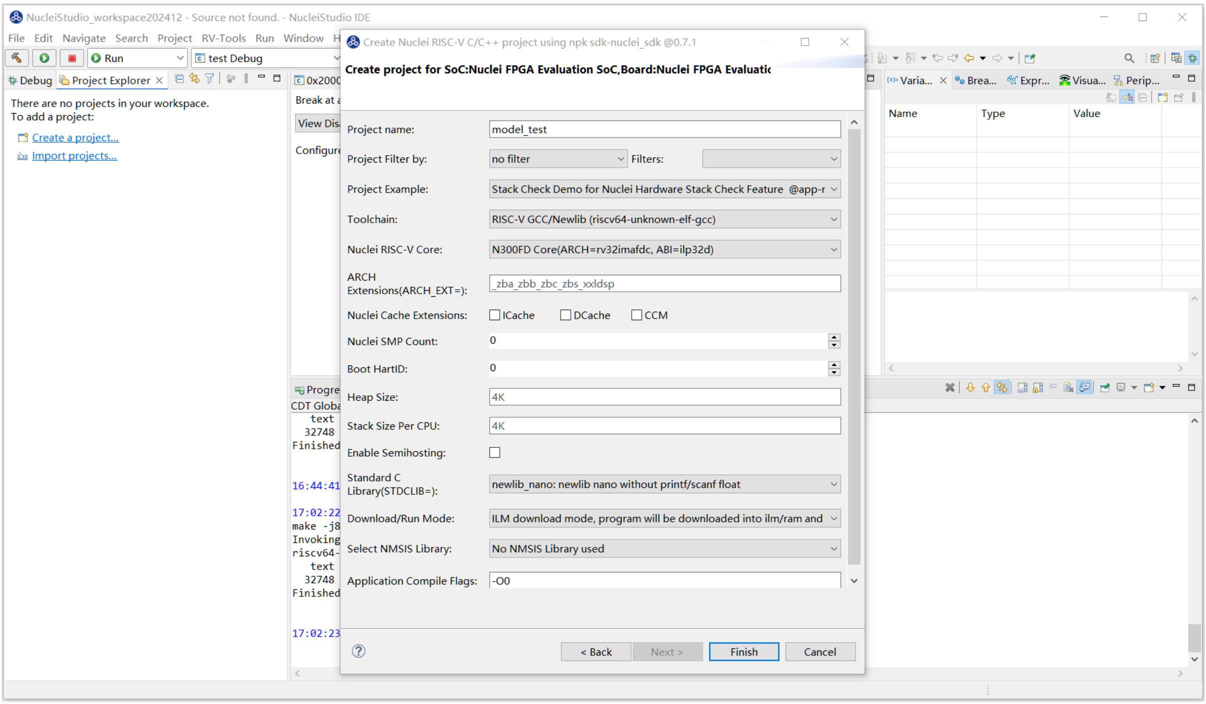
Nuclei Near Cycle Model 在 Nuclei Studio 2024.06 版中只有 Linux 版本，从 2025.02 版开始，已实现对 Windows 的支持。其具体介绍和命令行上使用参见 （https://doc.nucleisys.com/nuclei_tools/xlmodel/intro.html ），下面将在 Nuclei Studio 上演示如何使用 Nuclei Near Cycle Model 进行仿真和性能分析。

> **Note**
>
> Nuclei Near Cycle Model 已支持 Windows/Linux 版本，此文档测试都是基于 Nuclei Studio IDE 2025.02 的 Windows 版本完成的。

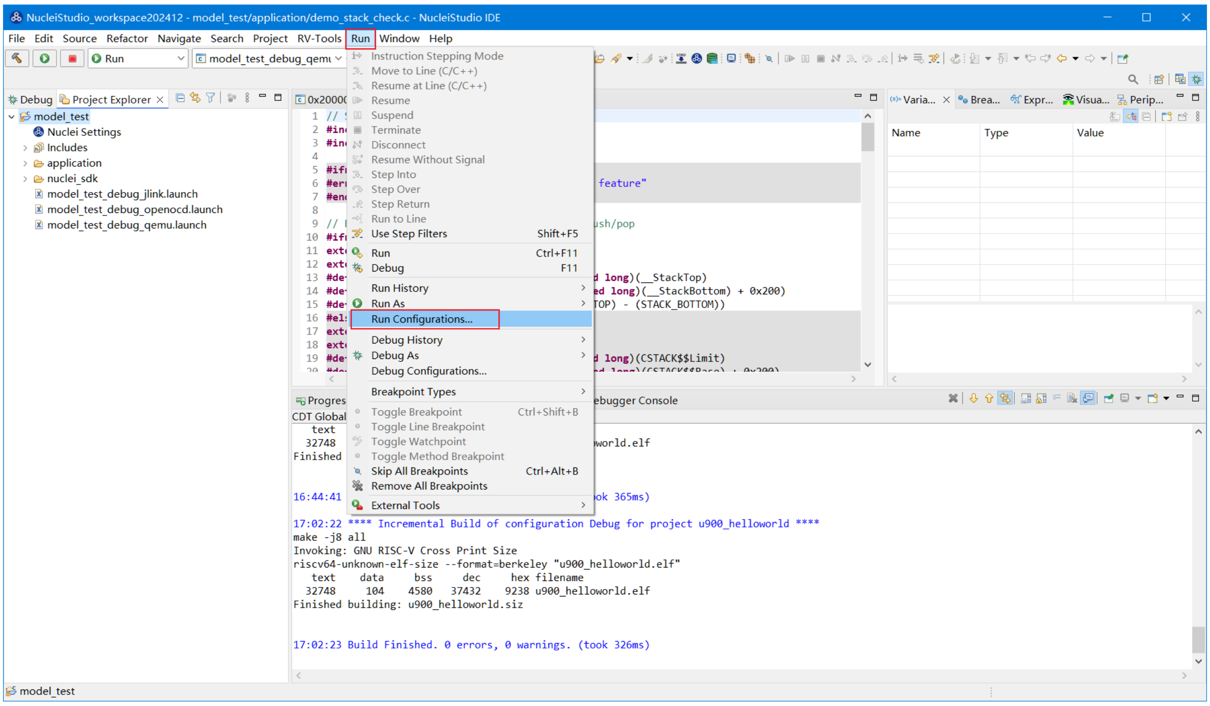
在使用过程，如有问题，可以查看 https://github.com/Nuclei-Software/nuclei-studio 相关内容，也可以向我们提交相关 issue。

创建测试工程

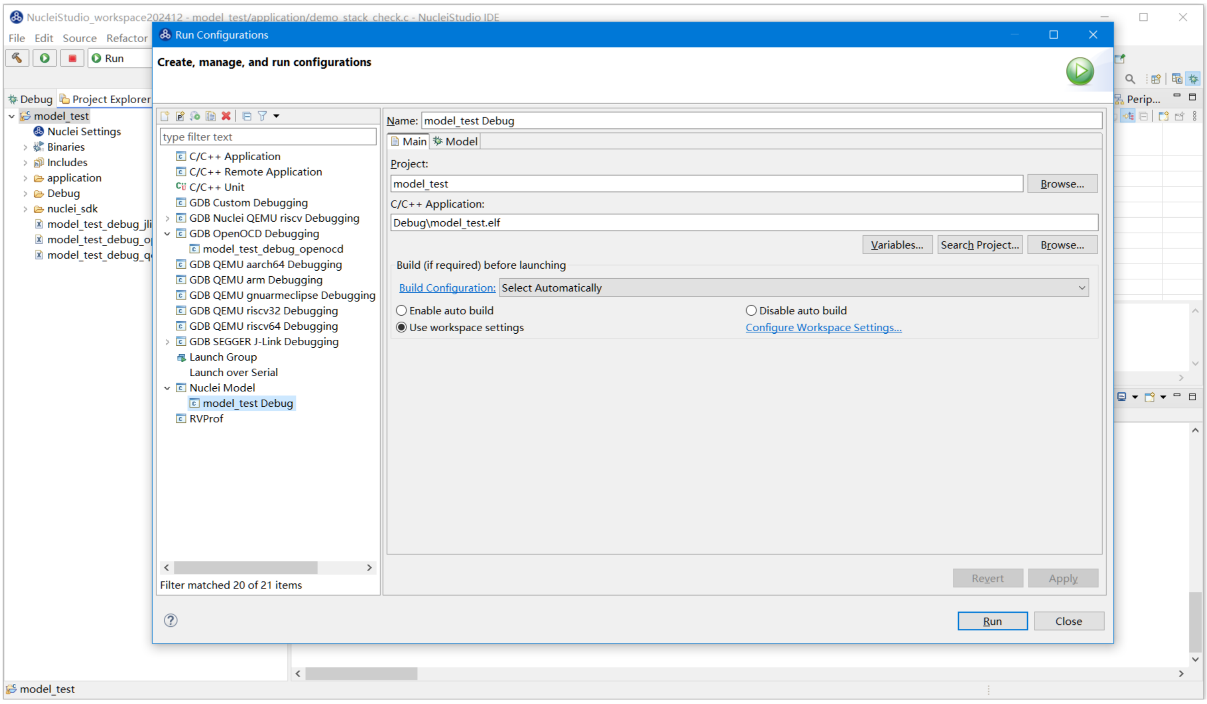
Nuclei Near Cycle Model 对芯来全类型的 Core 都有支持，可以创建任意一个 demo 工程并编译。创建任意一个 demo 工程并编译。



Nuclei Near Cycle Model 采用 Nuclei Studio 中的 Model 运行配置来进行运行测试，选中编译好的测试工程，然后打开 NucleiStudio 的 Run Configurations。



并创建一个 Nuclei Near Cycle Model 的配置，具体的配置及参数说明如下。



在演示示例的 Config options 中配置了 `--trace=1` `--gprof=1` `--logdir=Debug` `--cpu=n300fd`，`--trace=1` 表示开启 rvtrace，`--gprof=1` 表示开启 gprof 功能，`--logdir=Debug` 则表示最终生成的 `.rvtrace` 文件、`.gmon` 文件存放的路径为当前工程下的 Debug 目录，`--cpu=n300fd` 表示当前模拟的 cpu 核是 n300fd。

Note

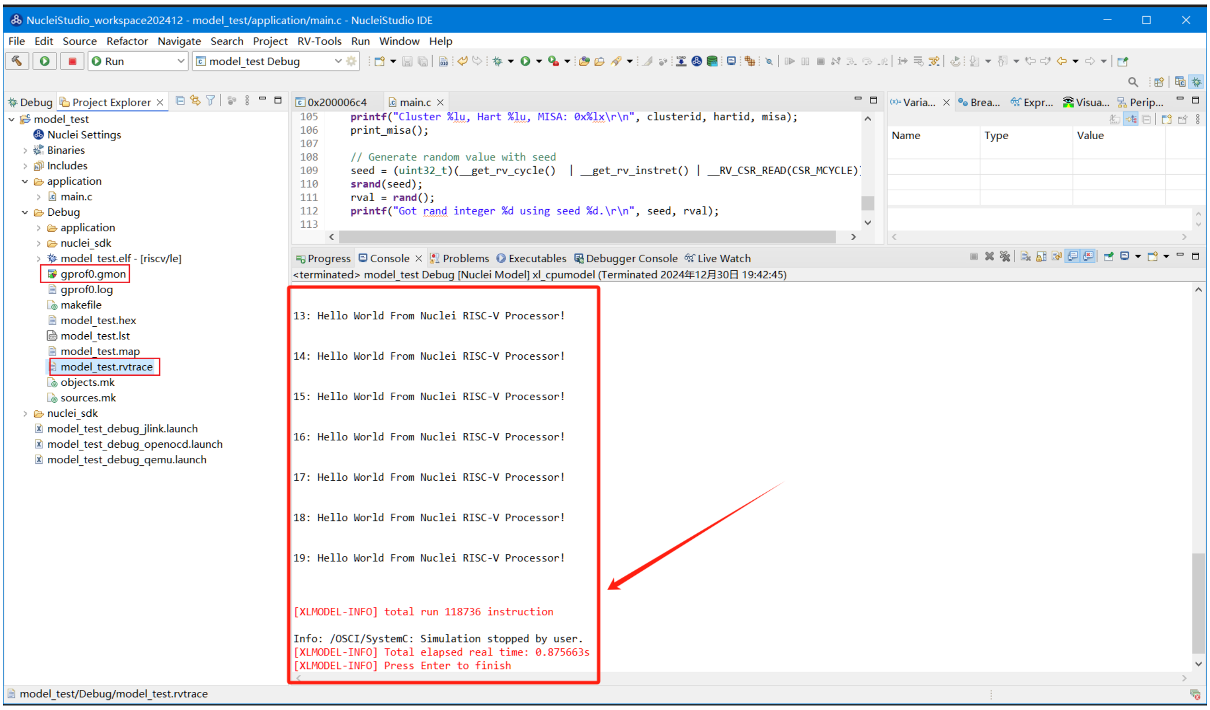
`--cpu=<core type>` 必须配置且与 Nuclei Setting 中配置的 Core 的值一致。

`--ext=<extension type>` 与 Nuclei Setting 中配置的 Other extensions 的值一致。

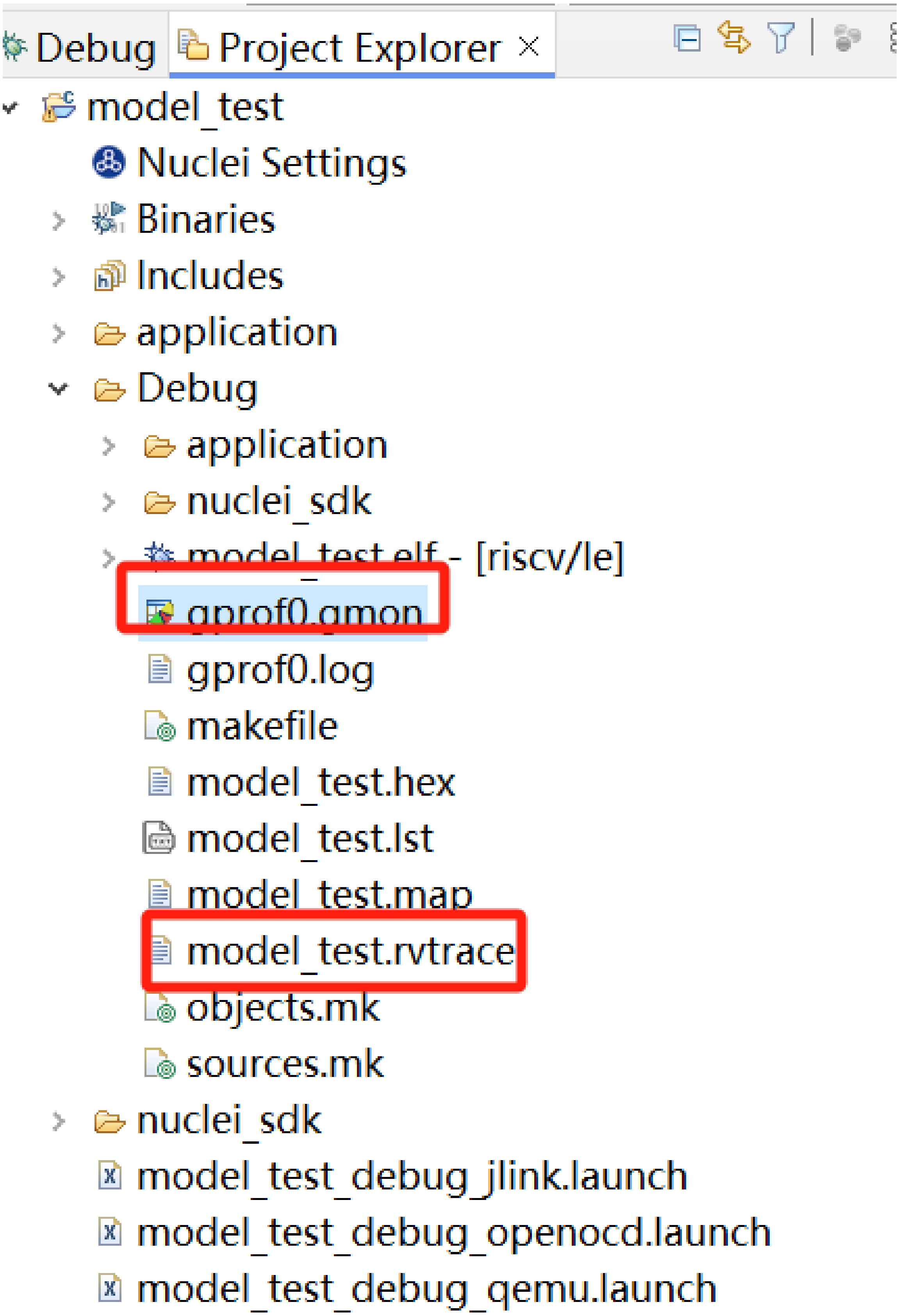
关于 Nuclei Near Cycle Model 的参数具体说明，请参见*Description of Parameters* (page 667)。

运行工程并生成性能分析结果

点击 Run 按钮，开始运行程序。程序在 Nuclei Near Cycle Model 中成功执行，输出了对应的 Log 信息。

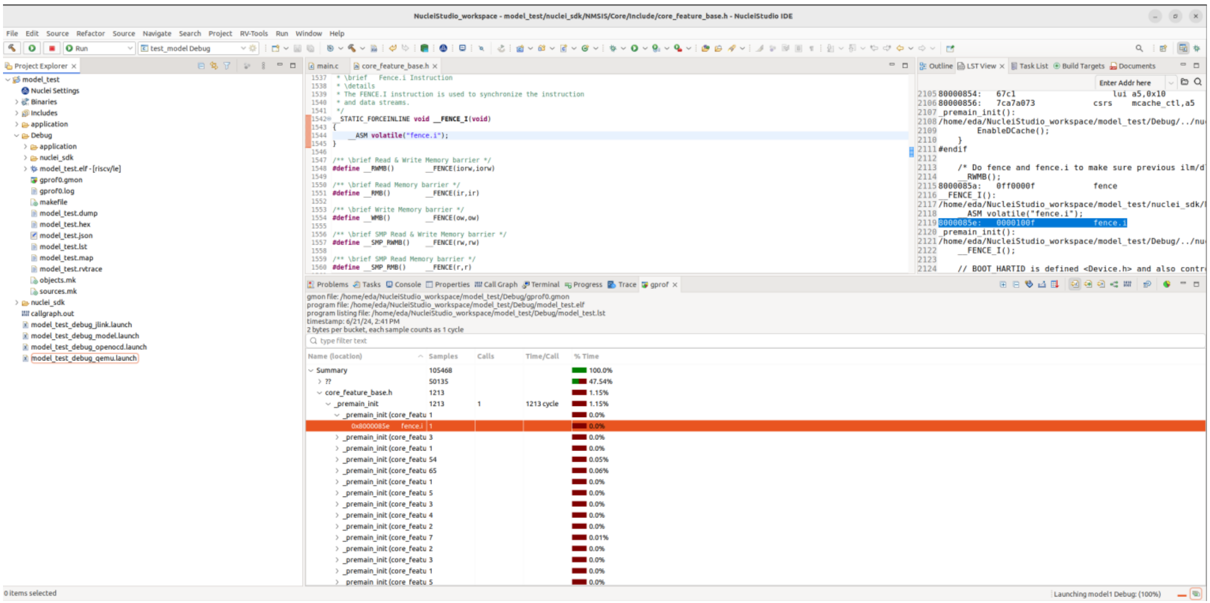


在工程的 Debug 目录中可以查看到已经生成 .rvtrace 文件、.gmon 文件。

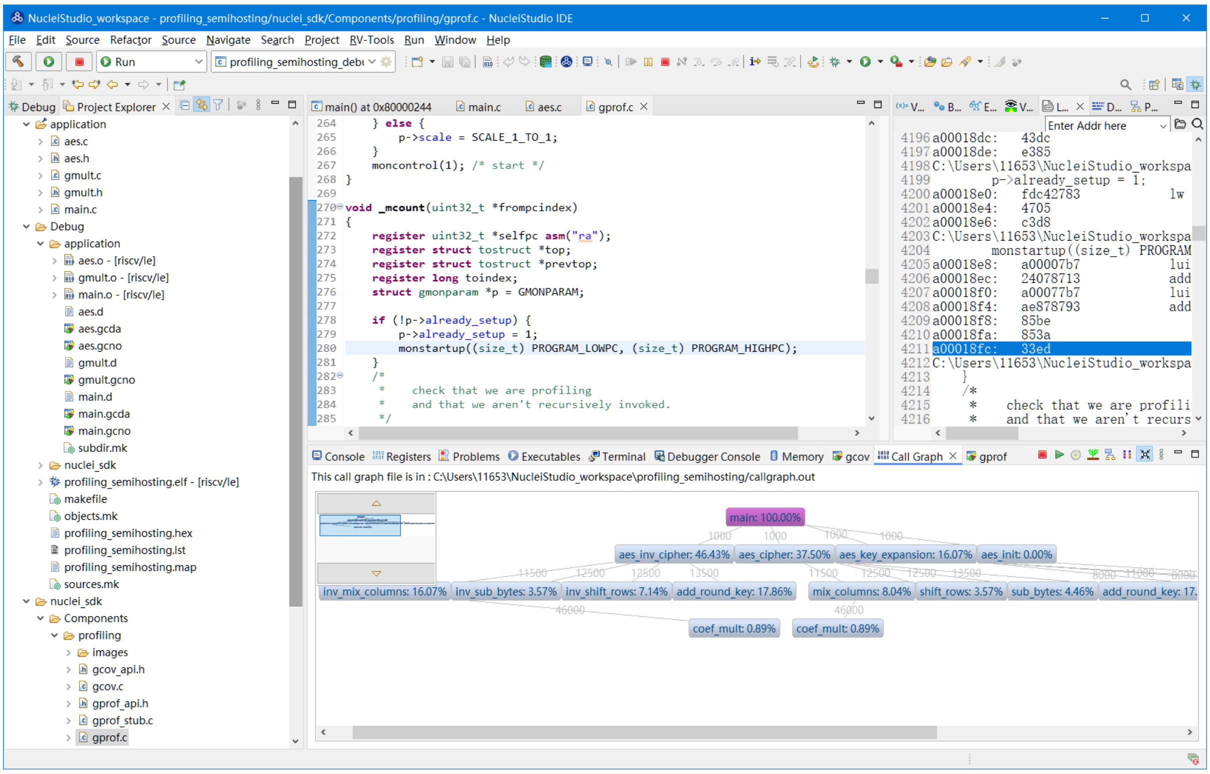


Nuclei Near Cycle Model 中支持通过 gprof 来分析程序，所以当我们配置了 `--gprof`，在程序运行时，也会在 Debug 目录（`--logdir=XX` 所配置的目录）下同步产生一个 `.gmon` 文件，双击 `.gmon` 文件，将调用 gprof 工具来分析程序执行所消耗的 cycle 数及调用关系；同时也会产生对应的 `callgraph.out` 文件，双击 `callgraph.out` 文件，调用 Call Graph 查看程序的调用关系。

调用 gprof 工具，可以查看生成的 .gmon 文件中的内容。



gprof 工具在查看 .gmon 文件的同时，会根据其内容，解析出程序的调用关系，并生成 callgraph.out 文件，双击 callgraph.out 调用 Call Graph 工具查看。



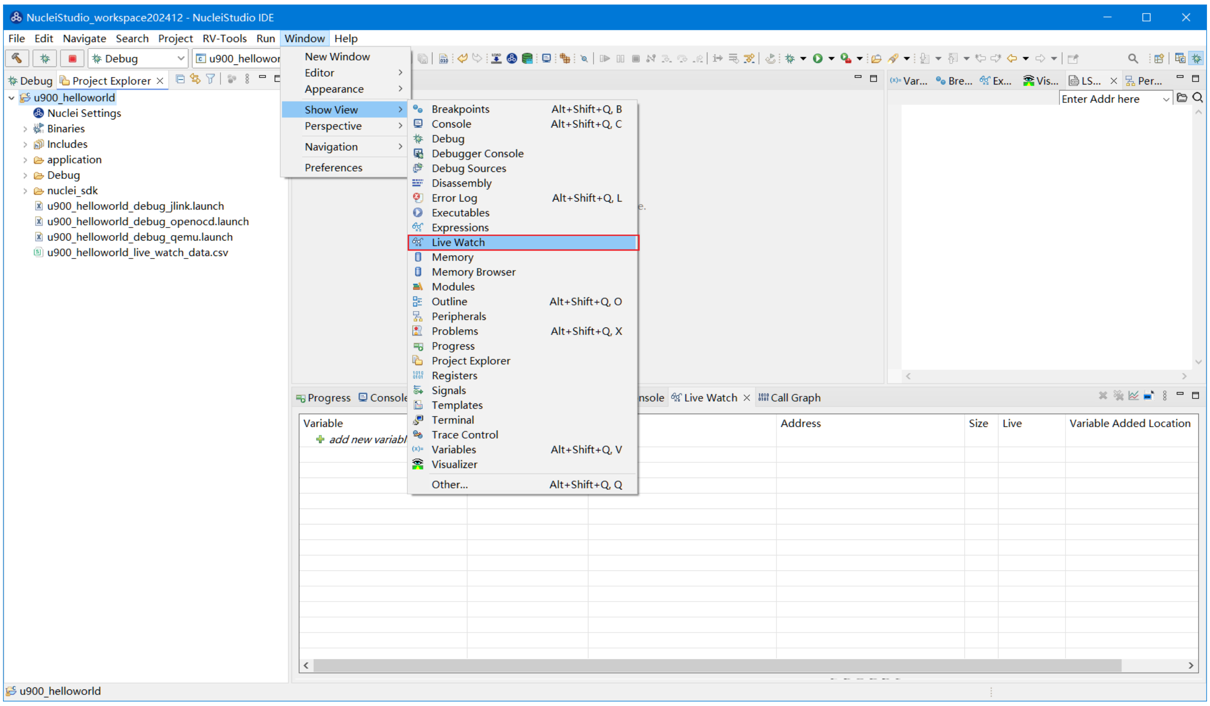
2.10.9 Live Watch 功能的使用

Live Watch 是一款强大的实时监控工具，专为开发者设计，旨在帮助您更高效地调试和优化代码。通过 Live Watch，您可以即时查看程序运行过程中变量的变化情况，无需打断执行流程或手动添加日志语句。在 Nuclei Studio 2025.02 版中实现了 Live Watch 功能，它支持自动刷新变量值，确保始终看到最新的数据变化。直观的图形化界面，能轻松管理需要监控的变量。

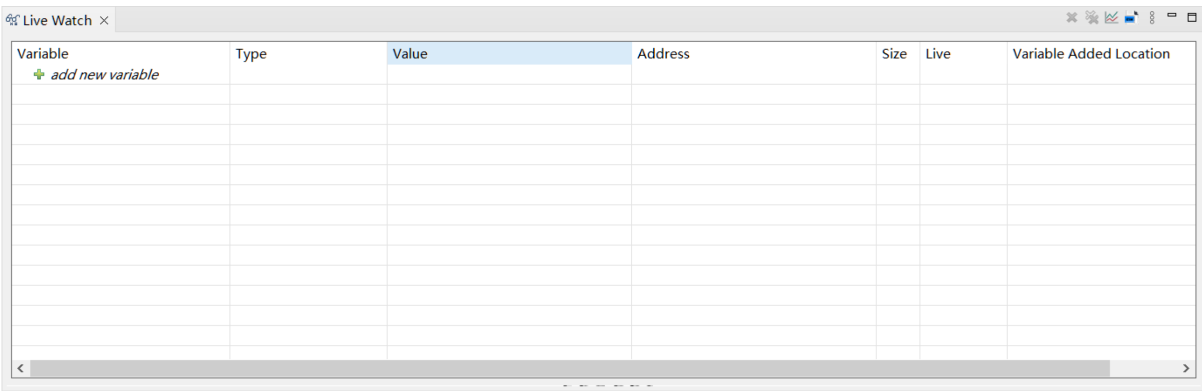
> **Note**
>
> Live Watch 功能依赖 Nuclei OpenOCD >= 2025.02 版本。仅支持 Nuclei CPU 配置了 RISC-V SBA 功能。

Live Watch 功能介绍

通过 Nuclei Studio 菜单 `Window -> Show View -> Live Watch` 可以打开 Live Watch 视图。



Live Watch 视图提供了一系列功能菜单，帮助用户更高效地管理和监控变量：



Remove

- 删除 Live Watch 视图中指定的变量行。

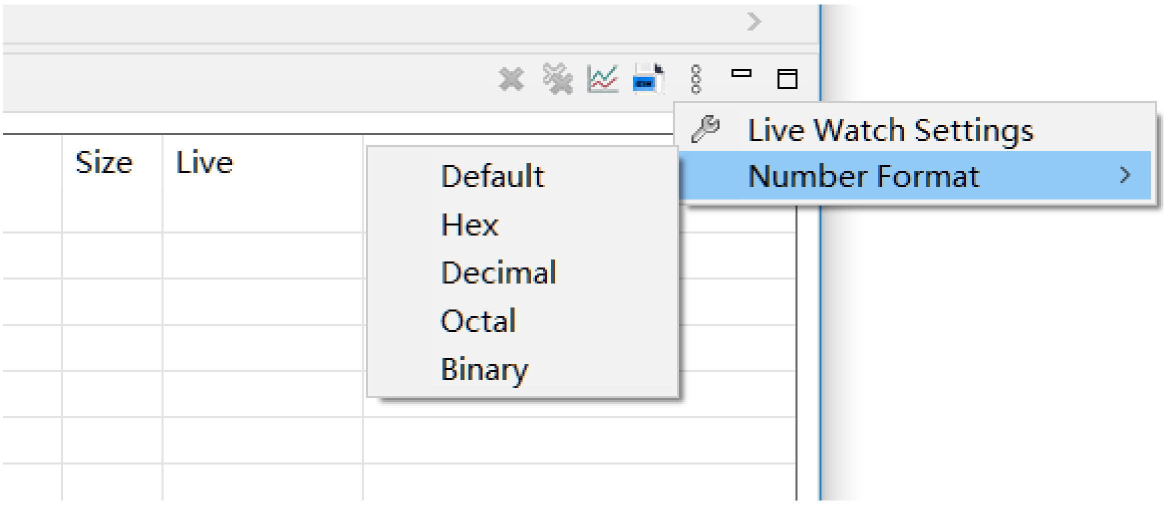
Remove All

- 清除 Live Watch 视图中所有添加的变量。

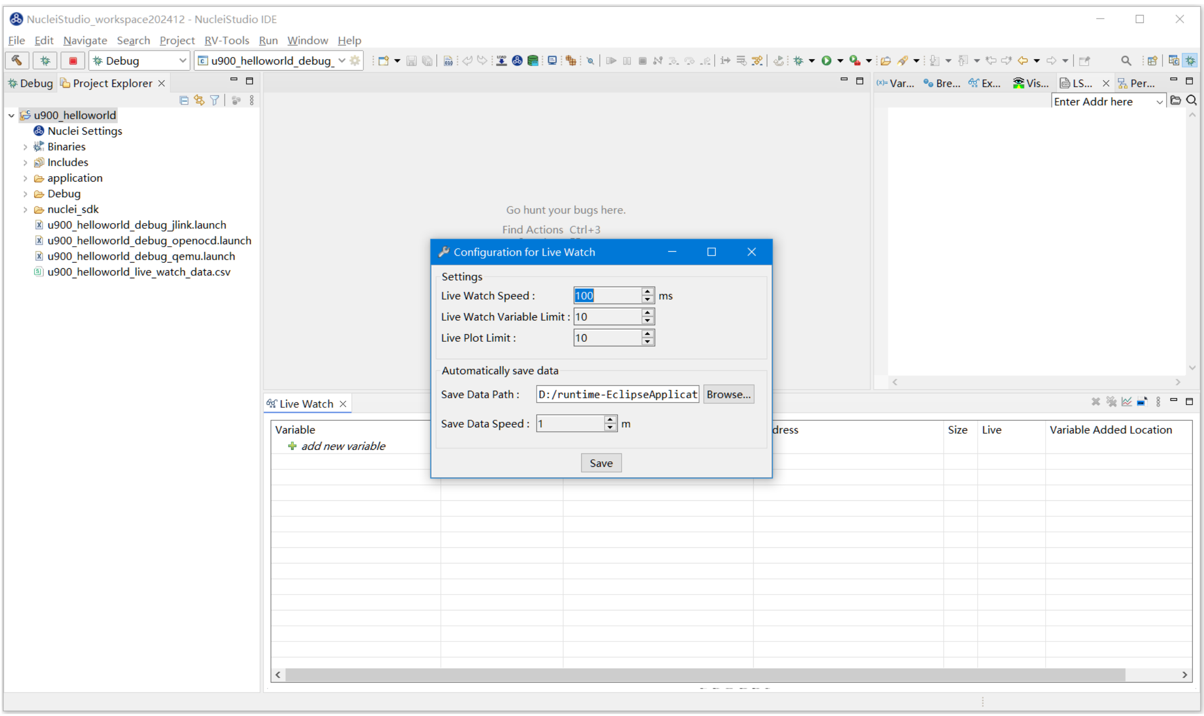
Show Live Plot

- 显示 Live Plot 视图，用于对采样的数据进行实时绘图。

在隐藏的菜单栏中，有两个设置菜单用于配置全局属性：



Live Watch Settings



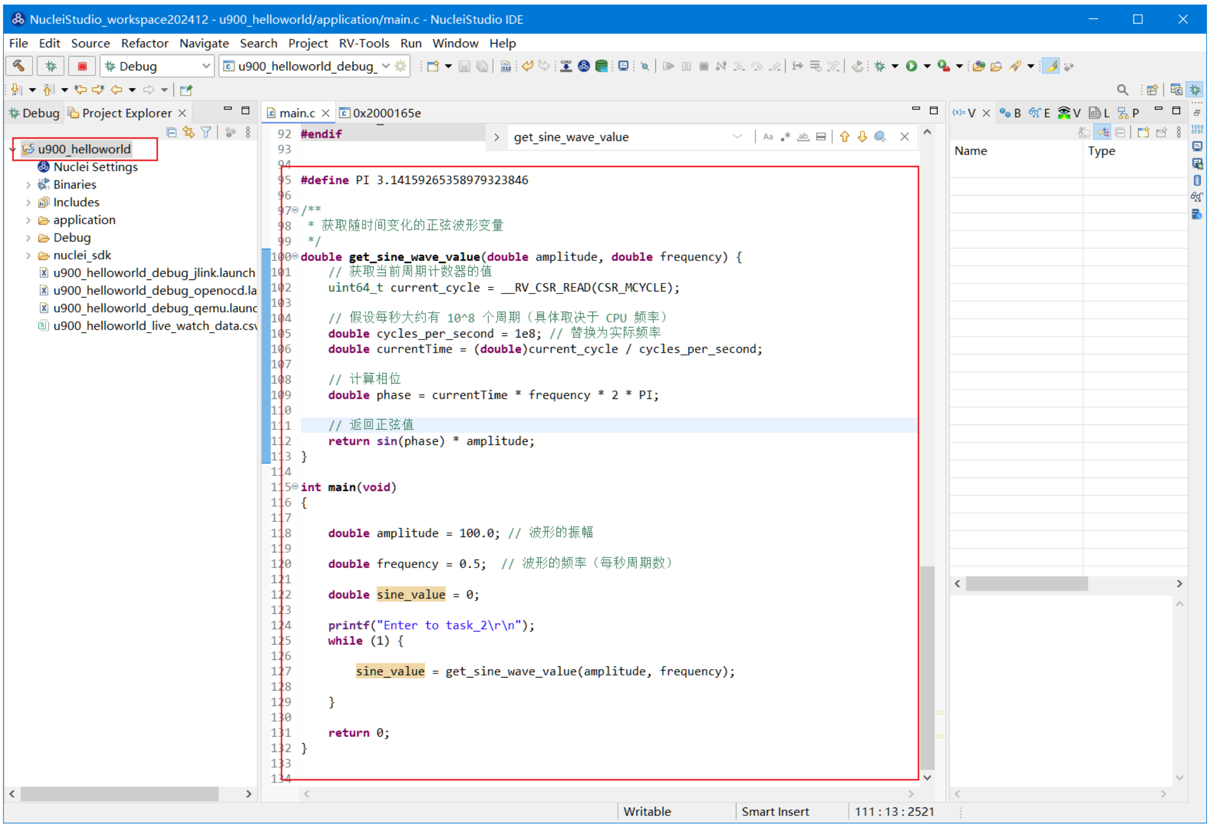
- Live Watch 中的一些常用设置，包含：
  - 包含以下常用设置：
    - Live Watch Speed : 设定 Live Watch 的采样频率, 最快为 100 ms 每次。
    - Live Watch Varible Limit : 限制同时采样的变量数量，最多为 10 个。
    - Live Plot Limit : 设定 Live Plot 同时绘制的最大样本数，最多同时绘制 10 个样本。
    - Save Data Path : 指定 Live Watch 采样的数据自动保存路径，供后续分析使用。
    - Save Data Speed : 设定 Live Watch 数据自动保存的频率，默认为每 10 分钟保存一次。

Number Format

- Live Watch 视图变量的值的显示方式。

Live Watch 使用演示

创建一个测试工程，并在工程内实现一个正弦计算。打开 Live Watch 视图，找到 Live Watch Settings 并根据需要设置相关参数（无可不设置，直接使用默认值）。



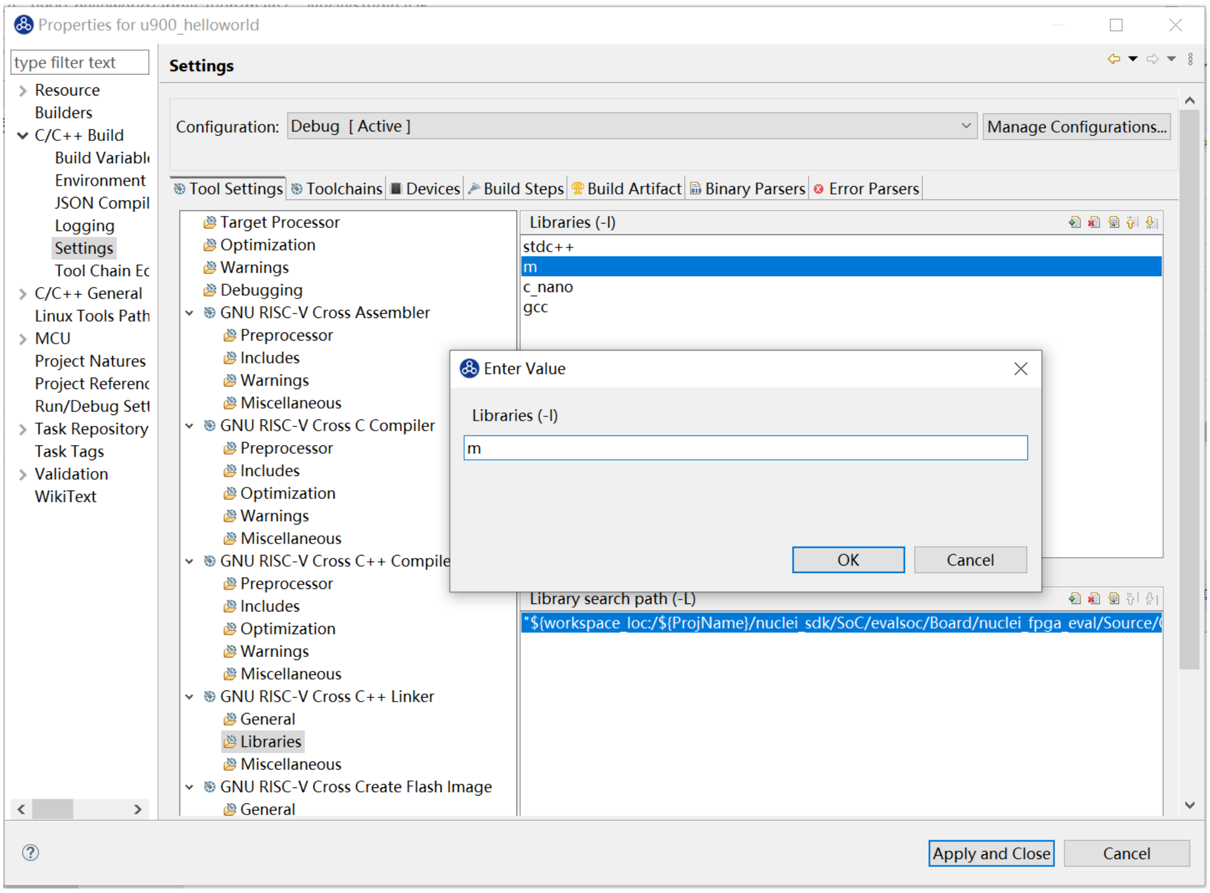
```c
#include <stdio.h>
#include "nuclei_sdk_soc.h"
#include <math.h>

#define PI 3.14159265358979323846
/**
* 获取随时间变化的正弦波形变量
*/
double get_sine_wave_value(double amplitude, double frequency) {
    // 获取当前周期计数器的值
    uint64_t current_cycle = __get_rv_cycle();

    // 计算当前时间 (单位: 秒)
    double currentTime = (double)current_cycle / SystemCoreClock;

    // 提前计算频率相关的因子
    double omega = frequency * 2 * PI;

    // 计算相位
    double phase = currentTime * omega;

    // 返回正弦值
    return sin(phase) * amplitude;
}

int main(void)
```

(continues on next page)

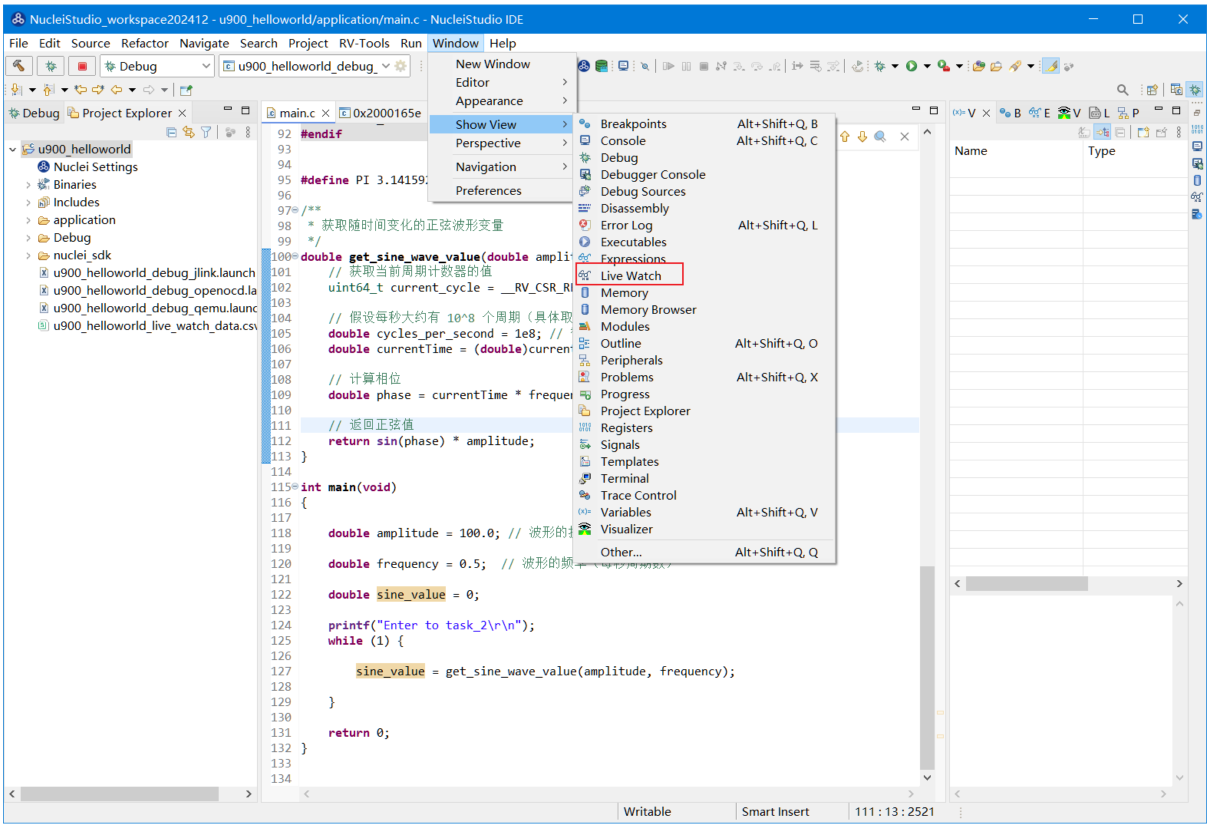
(continued from previous page)

```
{
    double amplitude = 100.0;  // 波形的振幅
    double frequency = 0.1;   // 波形的频率（每秒周期数）
    double sine_value = 0;

    printf("Enter to task_2\r\n");
    while (1) {
        sine_value = get_sine_wave_value(amplitude, frequency);
    }

    return 0;
}
```

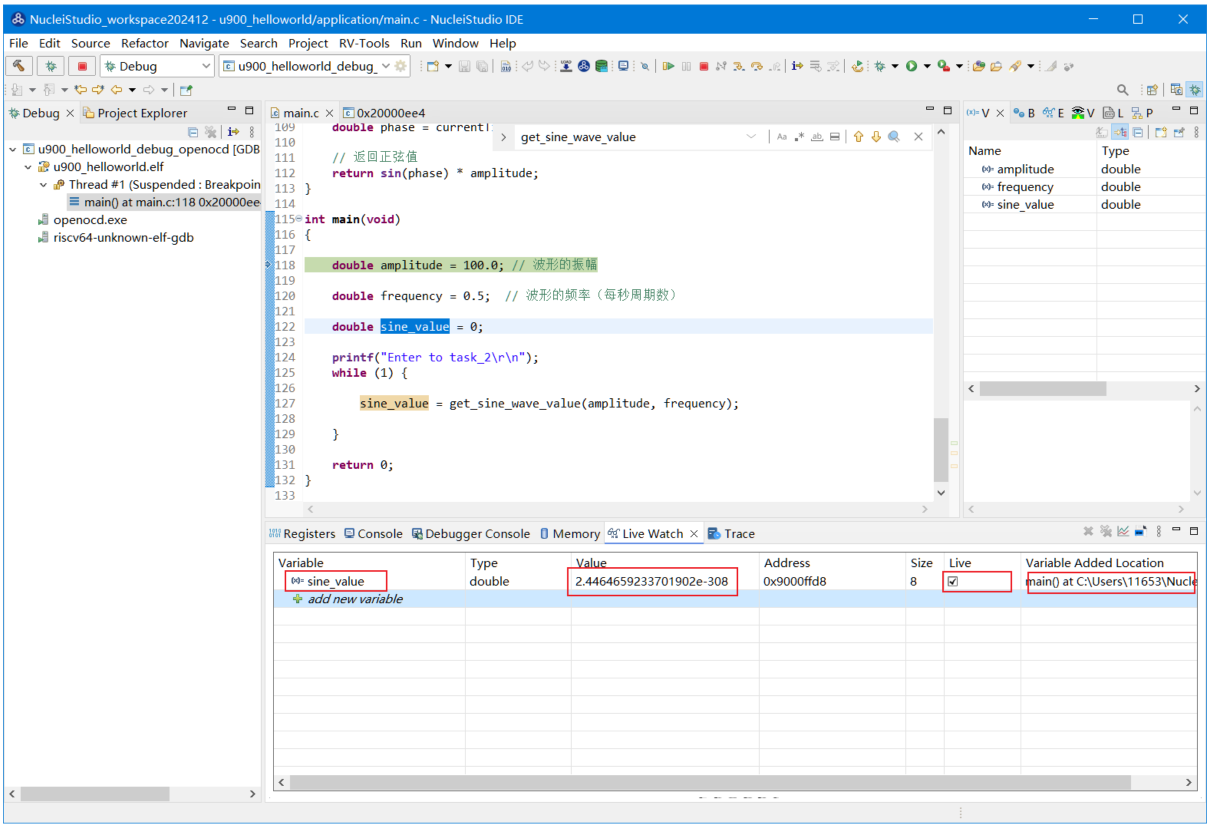
测试 demo 中运用了数学函数，所以需要在编译选项中添加 `-lm`，在工程的属性中，找到 `Settings -> GNU RISC-V Cross C++ Linker -> Libraries`，并添加 `-lm`。



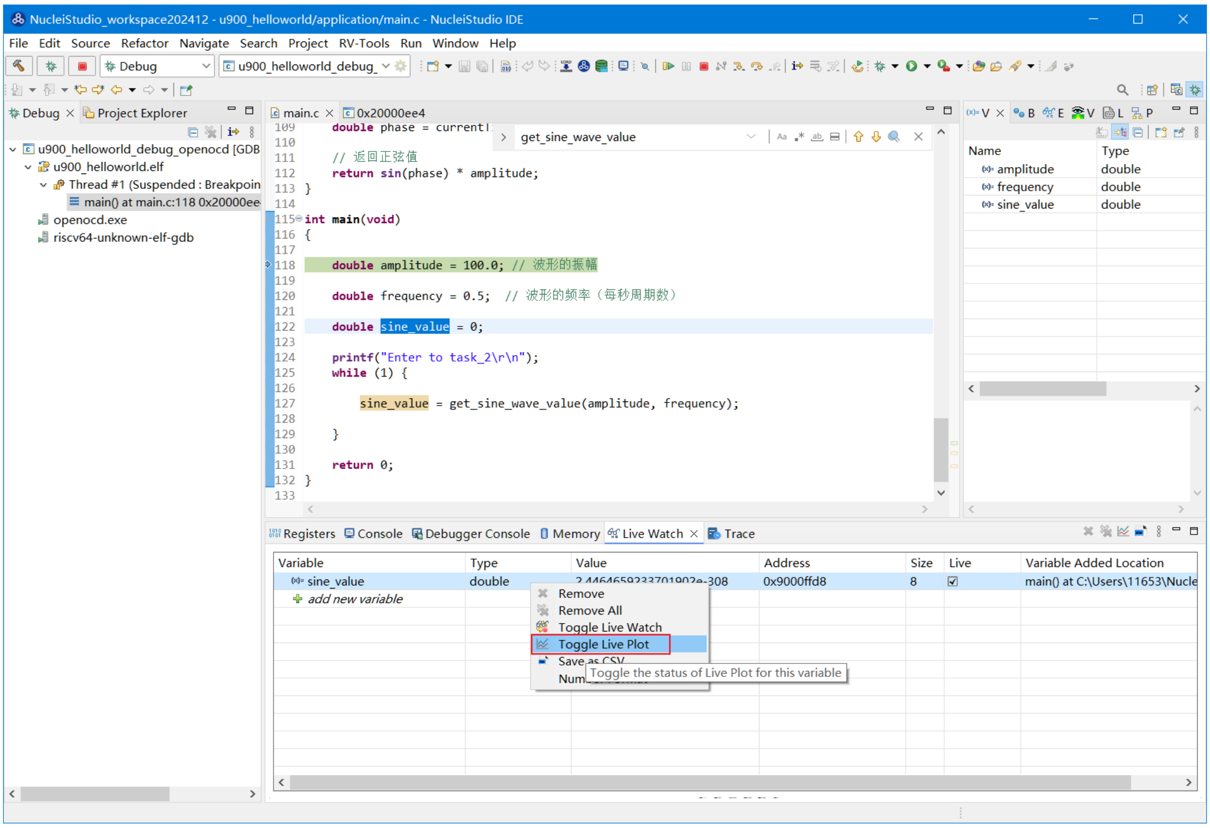
通过菜单 `Windows -> Show VIew -> Live Watch`，打开 Live Watch 视图。



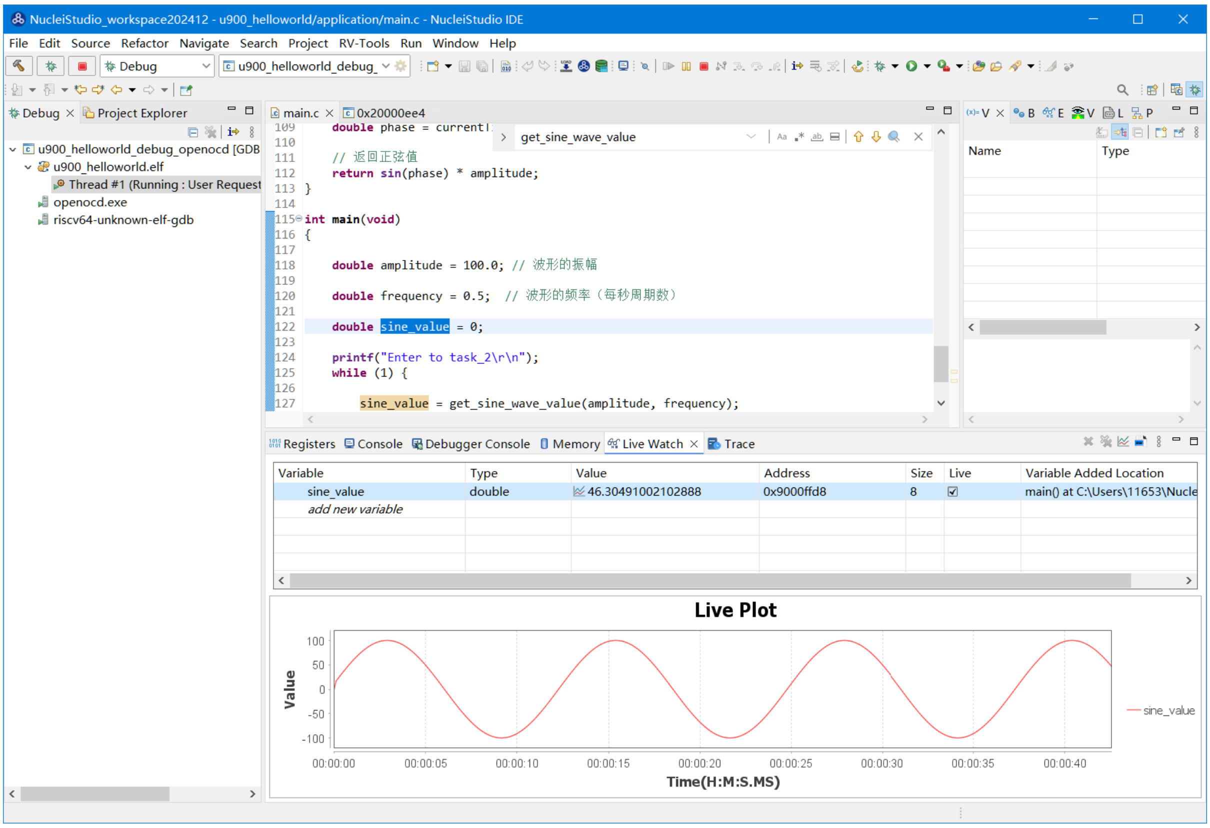
编译工程后，Debug 运行程序，在 Live Watch 视图中添加需要查看的变量。



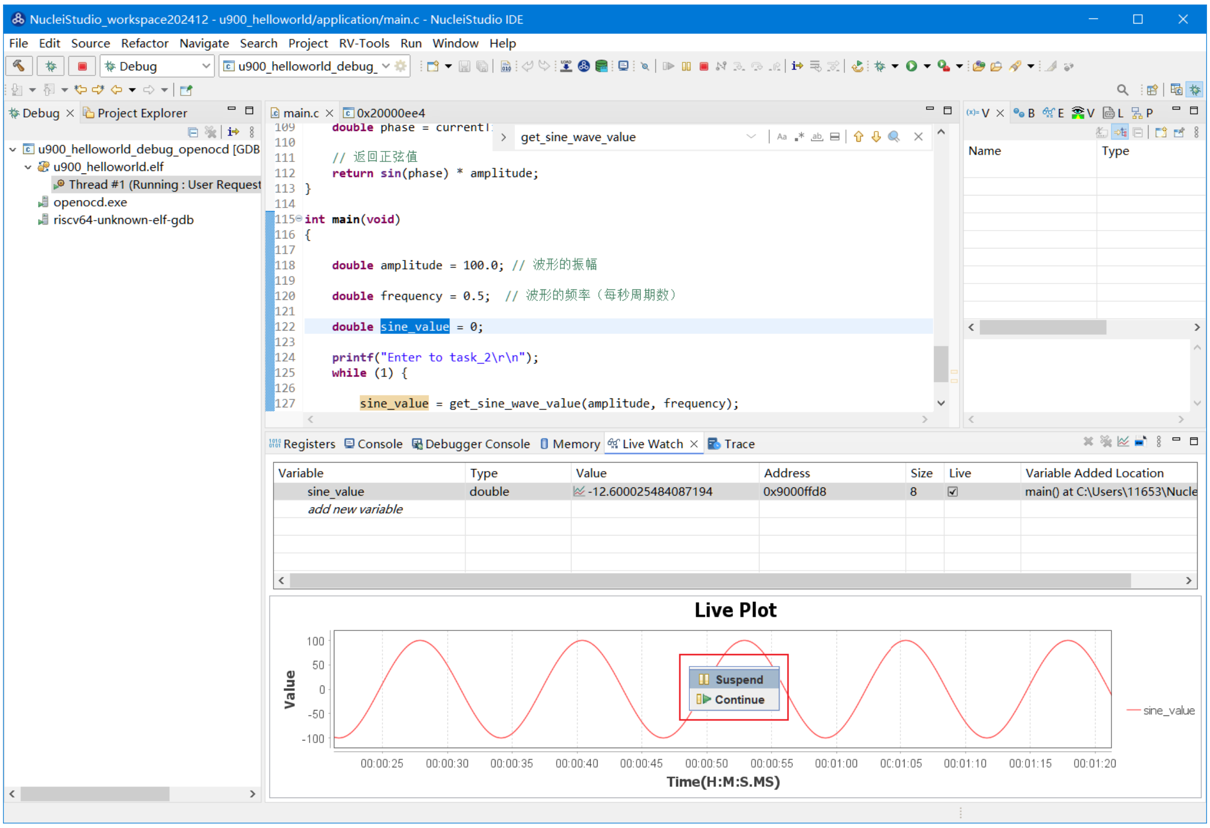
让工程全速运行时，可以看到变量的值，以设定的 Live Watch Speed 变化，如果想要通过 Live Plot 查看变量的变化曲线，可以选中该条记录，并点击鼠标右键，在弹出的菜单中选中 `Toggle Live Plot` ,Live Plot 工具就会弹出，并适应的画出变量的变化曲线。



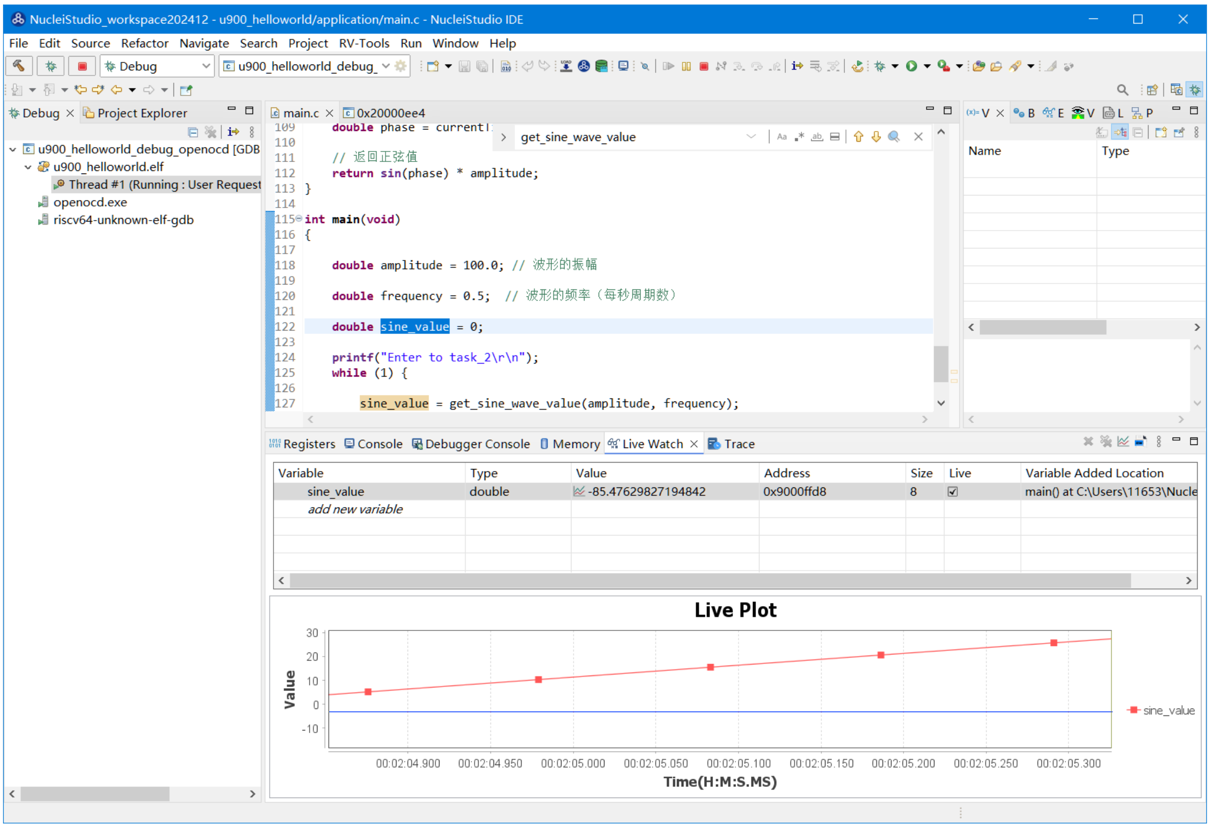
Live Plot 绘制的曲线图如下



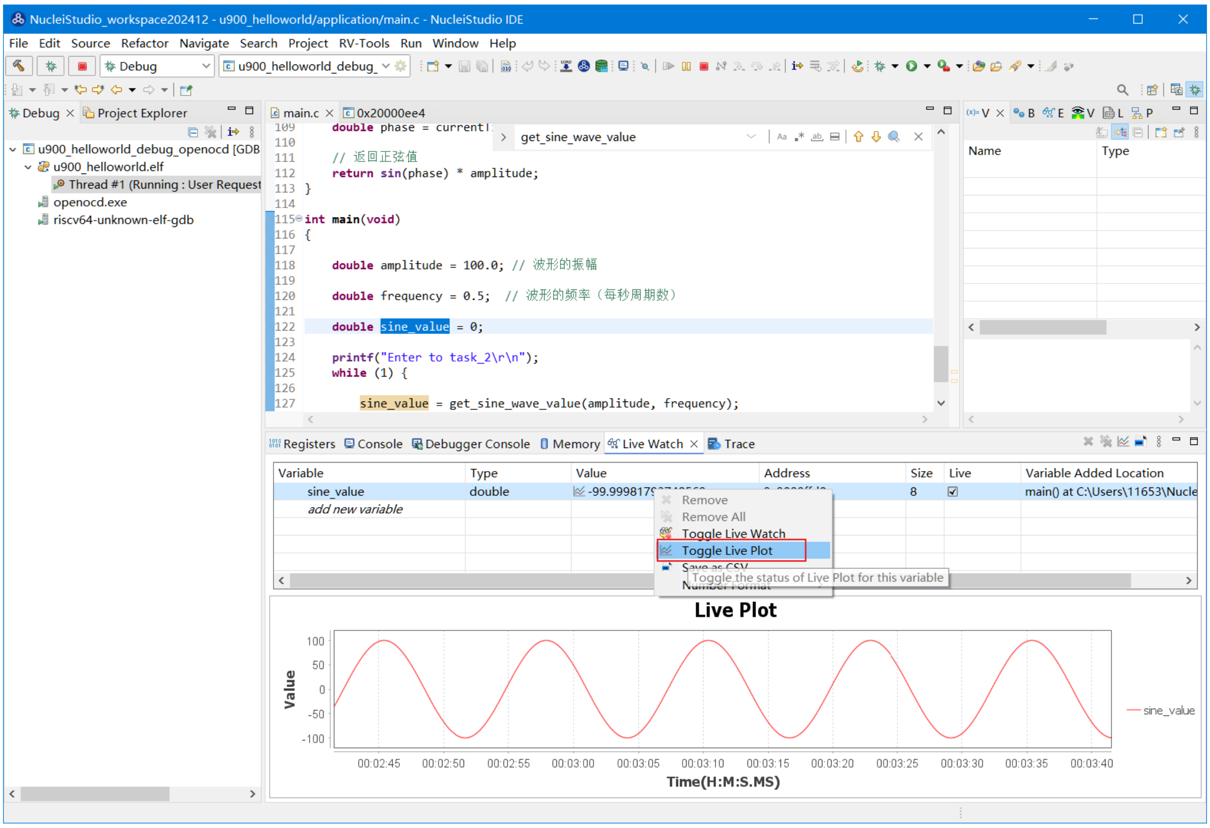
在 Live Plot 中点击鼠标右键弹出菜单，有 Suspend、Continue 两个功能菜单，点击 Suspend，Live Plot 会暂停画图。



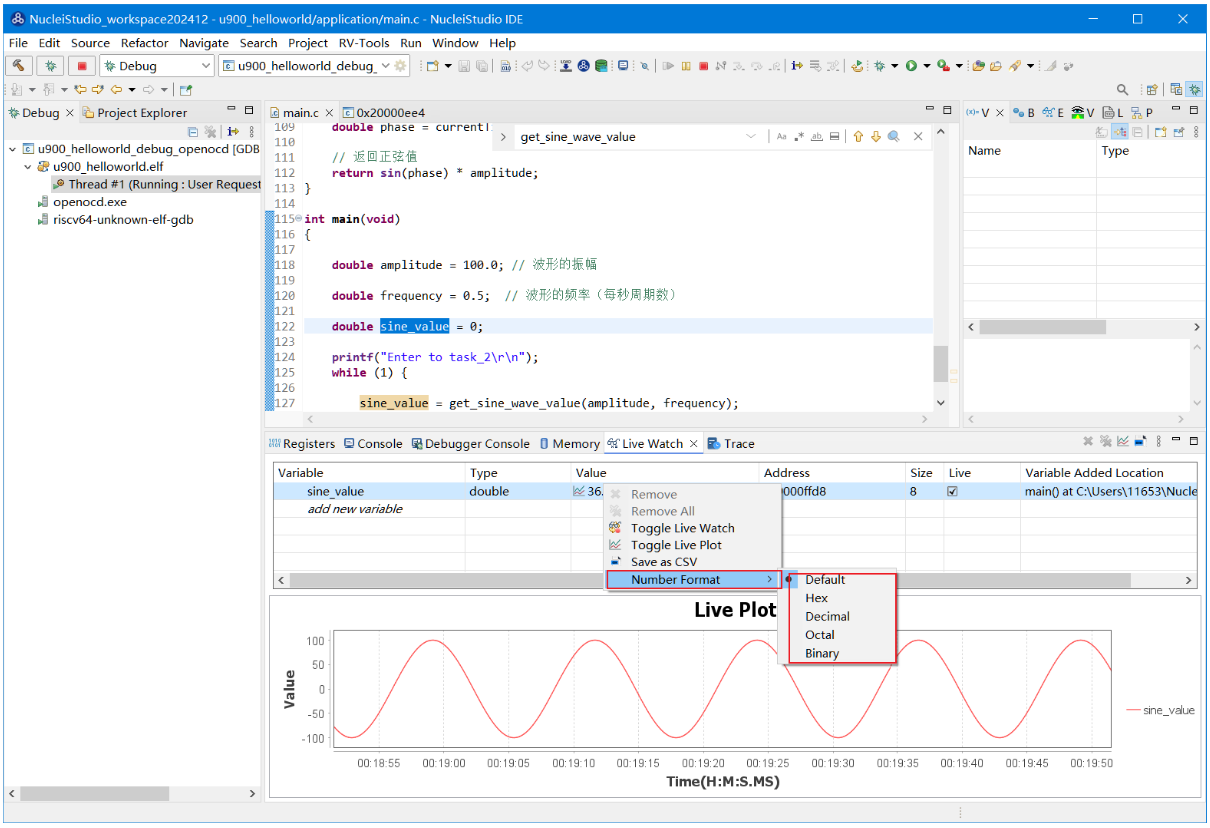
用户可以通过滚动鼠标放大曲线，查看数据详情；点击 `Continue` Live Plot 会继续绘制曲线。



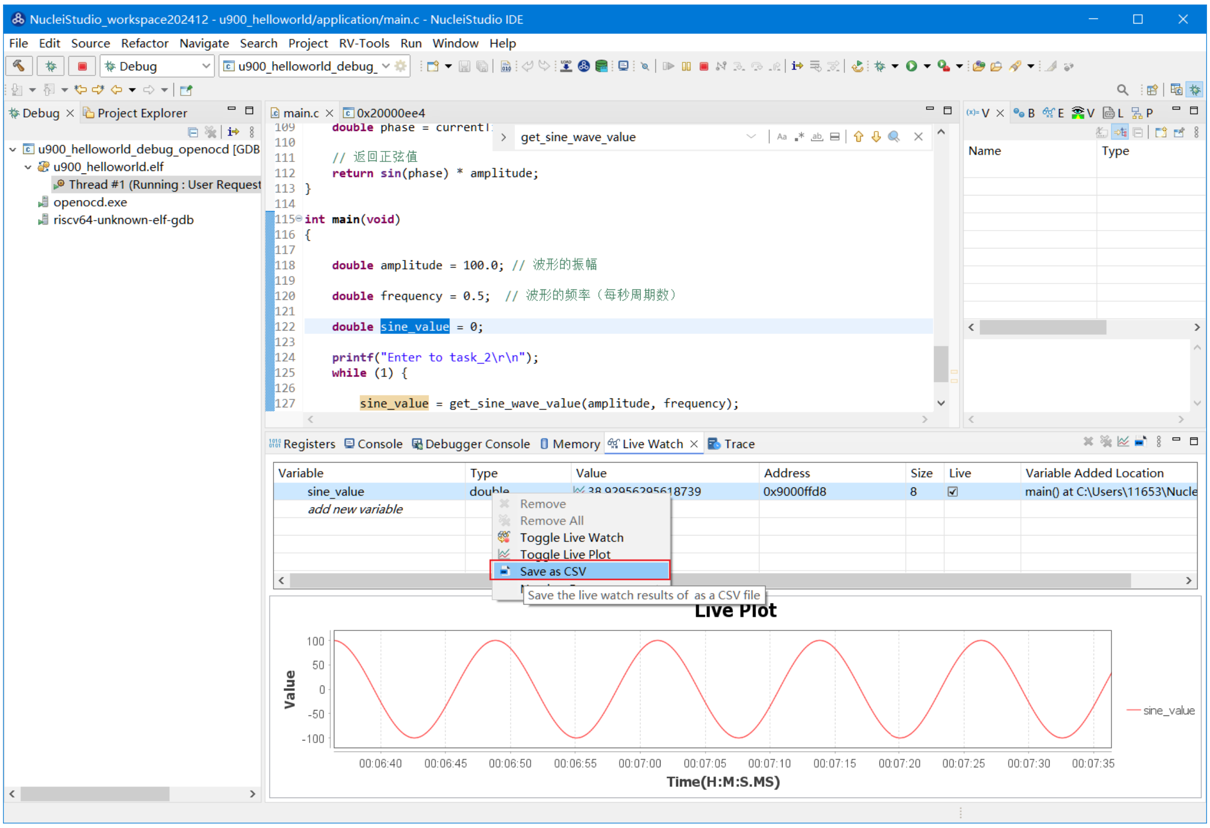
如果不想查看该变量的变化曲线，可以再次点击 `Toggle Live Plot`，将该变量从 Live Plot 踢除。



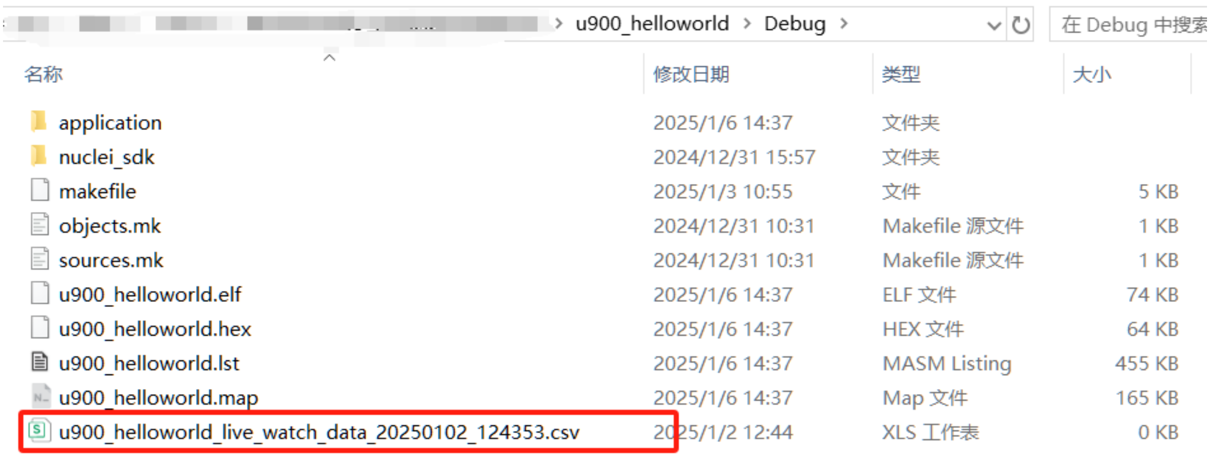
Live Watch 视图中的某个变量，点击鼠标右键，可以修改数据显示的格式。



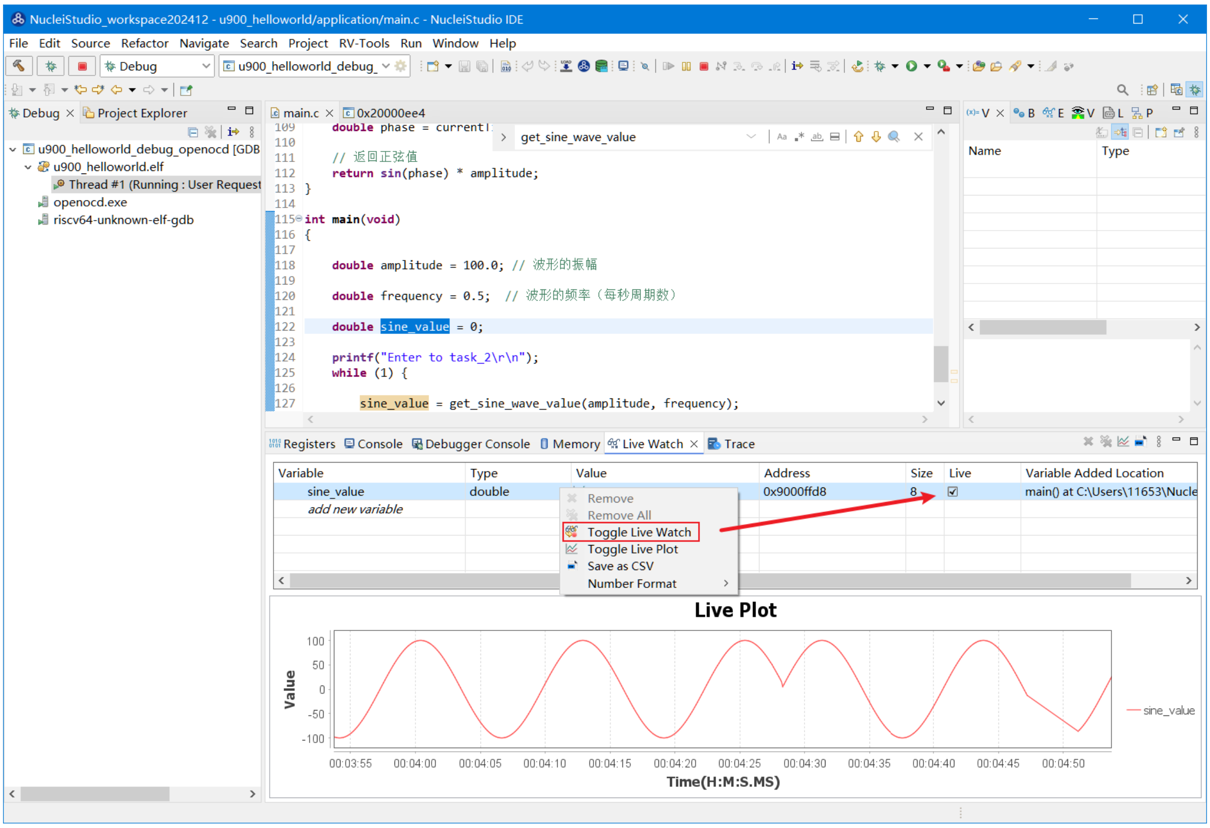
Live Watch 视图中的某个变量，点击鼠标右键，将该变量的结果存存为 CSV 格式文件，方便查阅和使用。



Live Watch 也会自动将查询到的数据结果保存到 `Save Data Path` 中，用户可以在 Save Data Path 找到对应的 CSV 格式的数据文件。



如果不想继续查看该变量的值，也可以选中该条记录，并点击鼠标右键，在弹出的菜单中选中 `Toggle Live Watch`，Live Watch 就不再适时查询该变量的值。

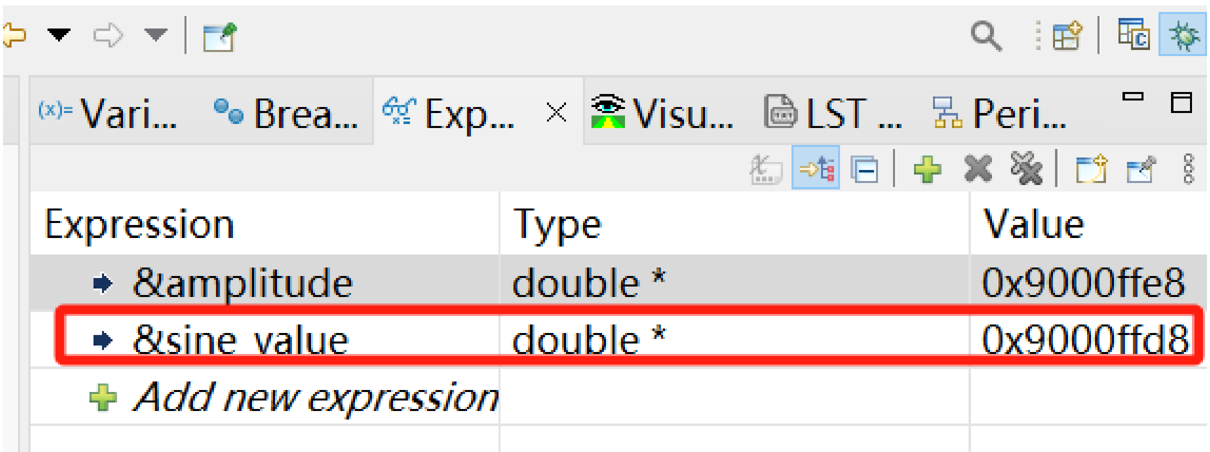


Live Watch 使用时的一些问题总结

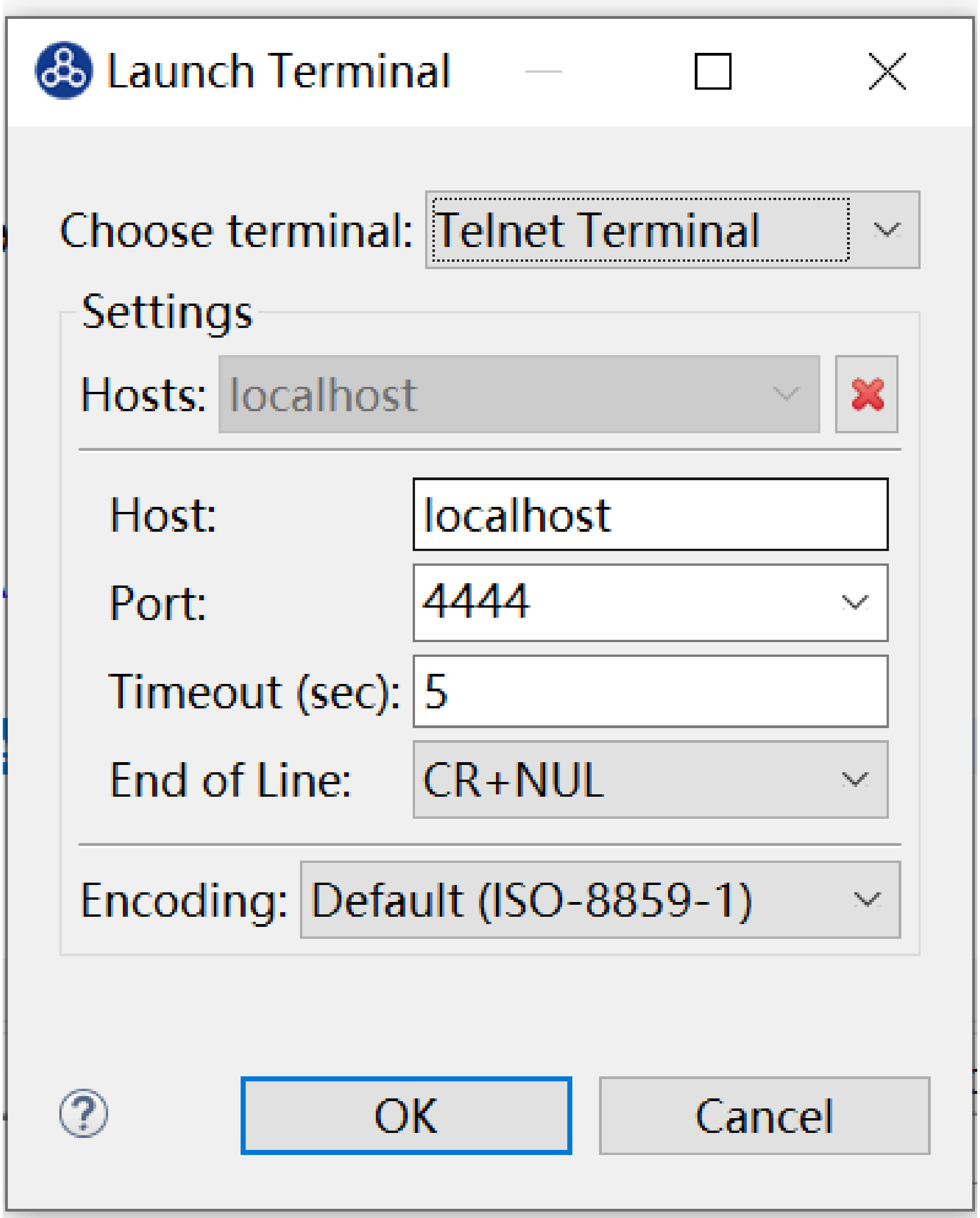
对 RISC-V SBA 的依赖

前文已经提到，Live Watch 对 RISC-V SBA 的依赖，如果在使用 Live Watch 时，无法获取到对应的值，有可能是当前 CPU 不支持 RISC-V SBA，如何确认当前 CPU 是否支持 RISC-V SBA，可以尝试以下方法。

在工程处理 Debug 状态时，获取到将要在 Live Watch 查看的变量的地址。可以参考如下图。



打开 telnet 工具，配置并连接。如无法连接，请检查 OpenOCD 服务是否启功正常，telnet 端口是否正常开启。



依然是在 Debug 状态下，在 telnet 工具中，使用命令 `riscv set_mem_access sysbus` 将 riscv 的 memory access 设置为 sysbus(SBA) 访问模式，并且尝试读取变量的值。

```
> riscv set_mem_access sysbus
> mdw 0x9000ffd8 4
0x9000ffd8: 000ab156 404e8df0 9999999a 3fb99999
```

如果上述操作能读取到变量的值，那么在全速运行程序的状态下，再次尝试读取变量的值，如果没有报错并且能读取到变量的值，说明当前 CPU 是支持 SBA 功能。

```
> mdw 0x9000ffd8 4
0x9000ffd8: 000ab156 404e8df0 9999999a 3fb99999
> mdw 0x9000ffd8 4
0x9000ffd8: d0b53424 c0510557 9999999a 3fb99999
> mdw 0x8000ffd8 4
0x8000ffd8: 00000000 00000000 00000000 00000000
```

(continues on next page)

(continued from previous page)

```
> mdw 0x9000ffd8 4
0x9000ffd8: b169d356 c0580176 9999999a 3fb99999
> mdw 0x9000ffd8 4
0x9000ffd8: a69861c1 c056547d 9999999a 3fb99999
> mdw 0x9000ffd8 4
0x9000ffd8: 4dbf49f2 404022d3 9999999a 3fb99999
> mdw 0x9000ffd8 4
```

优化级别太高 **Live Watch** 读取不到变量的值

当工程优化级别较高时，Live Watch 可能无法获取到某些变量的值，有 demo 如下。

```
int test()
{
   uint32_t live_watch = 0;
   while(1){
      live_watch += 1;
      delay_ms(100U);
   }
}
```

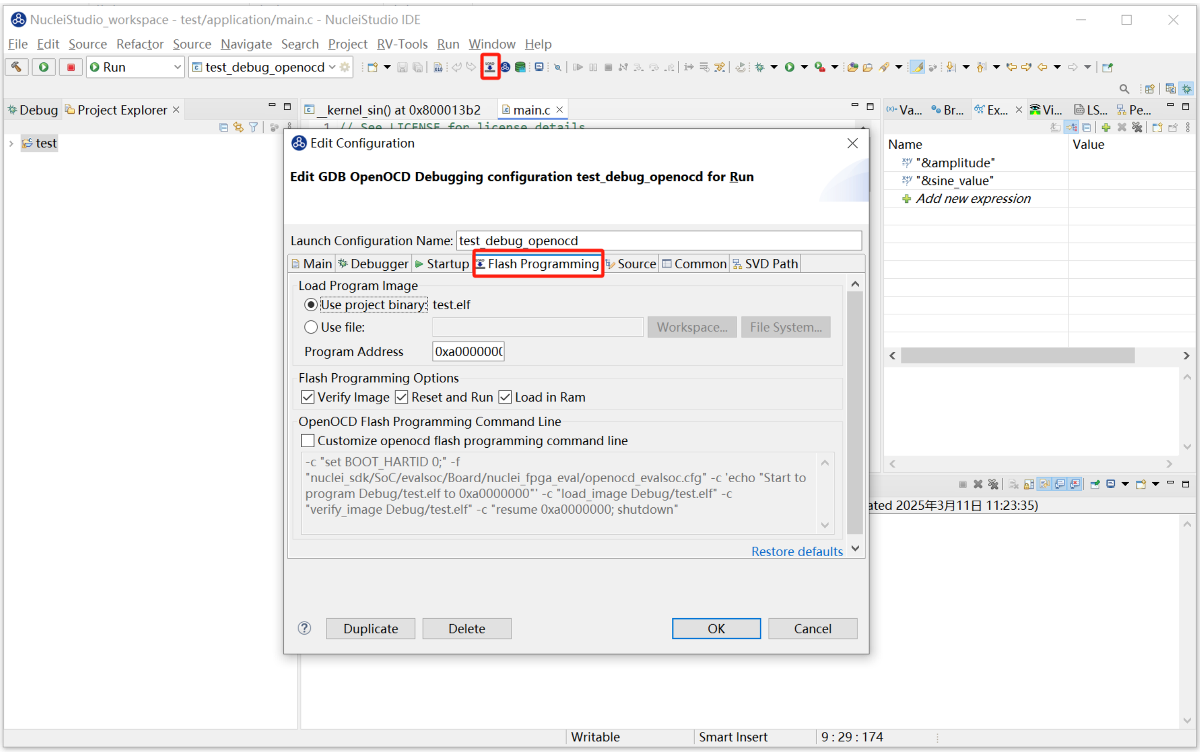
当工程的优化级变为 O3 或其他更高级别的，Live Watch 无法获取到 `live_watch` 的值。可以调整工程的优化等级；也可以将需要监控的变量设置为全局变量或者 static 静态变量，且需要避免被编译器优化，工程代码修改如下。

```
int test()
{
   static uint32_t live_watch = 0;
   while(1){
      live_watch += 1;
      delay_ms(100U);
   }
}
```

2.10.10 Flash Programming

为了满足用户将编译好的二进制文件直接下载到硬件开发板的需求，Nuclei Studio 新增了 Flash Programming 功能。该功能允许用户快速、便捷地将编译好的二进制文件直接下载到硬件开发板中，极大提升了开发和调试的效率；简化操作流程，用户只需点击一次即可完成二进制文件的下载。工程编译好后，找到 Flash Programming，并点击，即可完成二进制文件的下载。

具体参见*Flash Programming 功能* (page 137) 。



2.11 Nuclei Studio 升级更新

一般情况下，如果 Nuclei Studio 没有大的版本变动，Nuclei Studio 升级工作可以由用户下载最新的 GNU 工具链以及其他相关工具和升级 IDE Plugins 的方式来完成。

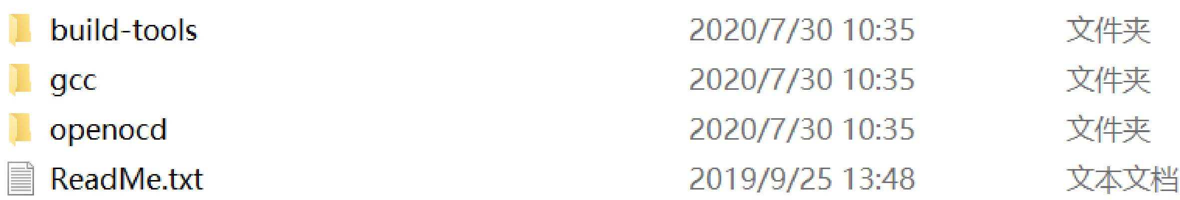
2.11.1 工具链的安装

Nuclei Studio 已经把工具链集成在 IDE 内部，工具链存放在 Nuclei Studio_IDE_XXX 文件夹内，路径为：`Nuclei Studio_IDE_XXX\\NucleiStudio\\toolchain` 。IDE 默认使用此路径的工具链，所以请不要移动 `toolchain` 此文件夹，使用 IDE 创建工程后也不需要进行工具链的相关配置。

由于工具链升级的速度会比 IDE 快，所以后期需要用户手动升级工具链，工具链升级方法如下：

- 首先删除需要更新的 GCC 或者 OpenOCD 文件内的内容，然后在芯来科技官网下载最新版本的 GCC 或者 OpenOCD。
- 将 GCC 或者 OpenOCD 的内容复制到对应的文件夹下，即可完成 IDE 工具链的更新，IDE 自带的工具链的版本号记录在 ReadMe.txt 中。

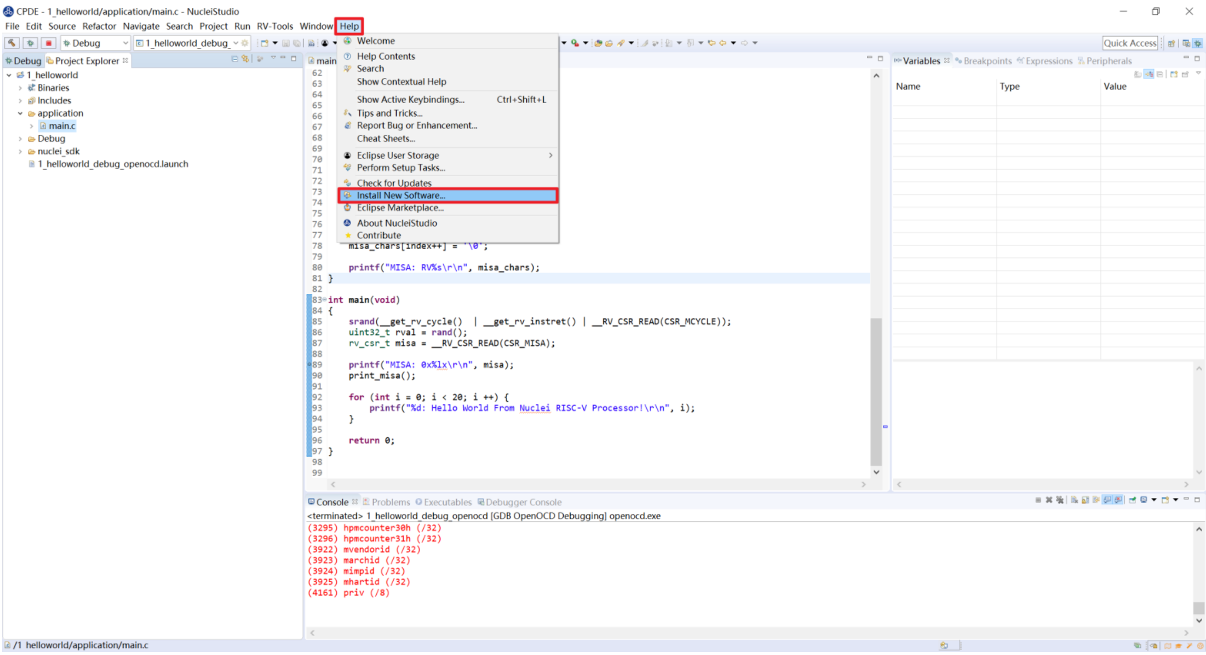
> **Note**
>
> 注意：要保证 gcc 所在的这一级目录文件夹名字不变，替换后保证 bin 文件夹所在层级是图中文件夹的子目录，中间不要有其他文件夹。



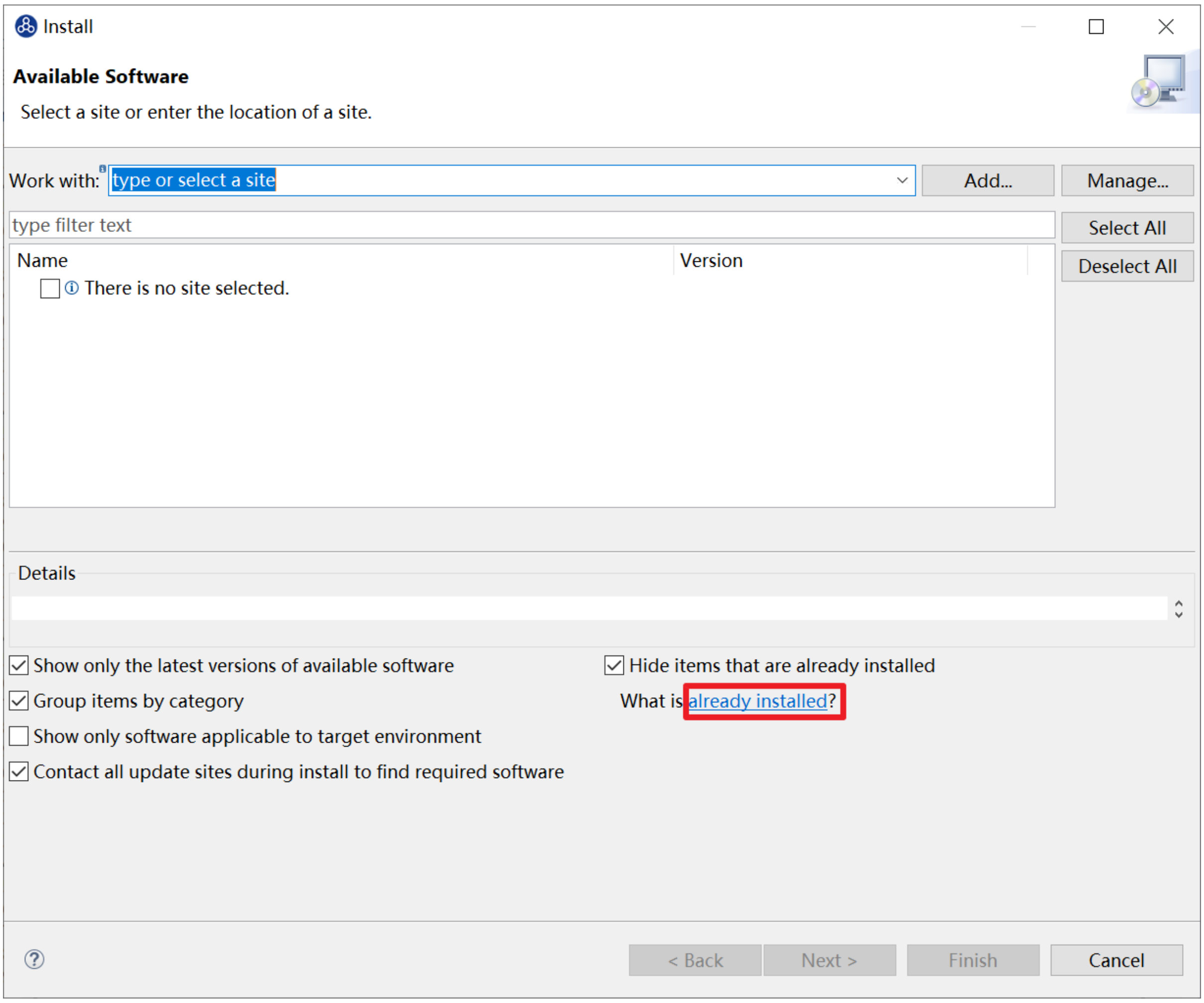
2.11.2 IDE Plugins 升级

Nuclei Studio 支持 IDE 内在线升级，步骤如下。

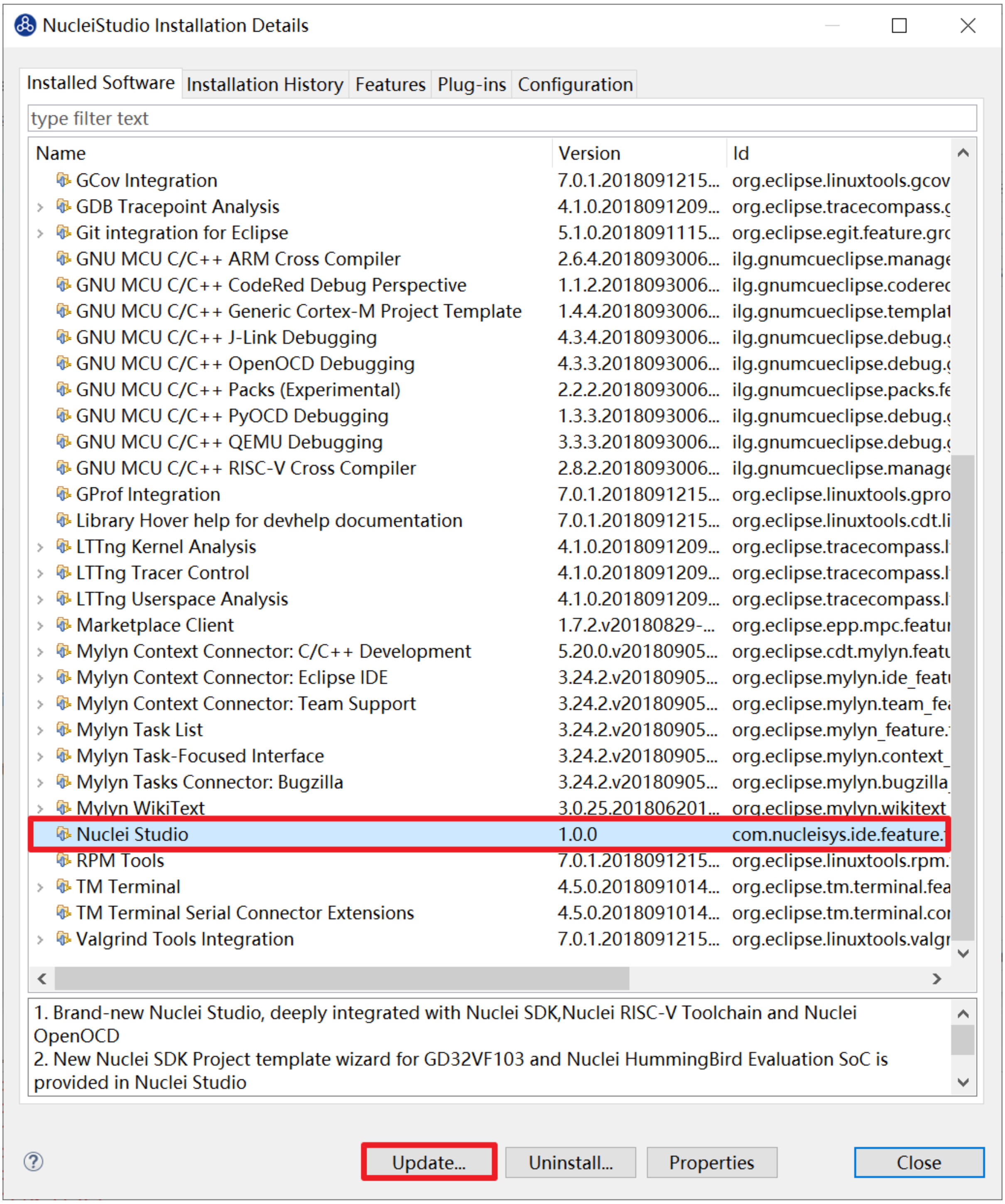
打开安装软件弹窗，在菜单栏选择 `Help ` Install New Software`。



点击 `already installed` 打开已安装界面。



选择 Nuclei Studio，点击 update 即可自动完成在线更新。



2.12 Nuclei Studio 常见问题

2.12.1 Nuclei Studio 启动慢

Nuclei Studio 是基于 eclipse 进行开发，继承了 eclipse 的可扩展性的特点的是同时，也存在着启动较慢的问题，第一次启动时，软件需要验证环境；生成常用的缓存文件；创建 Workspace. 在 2021.09 版的 Nuclei Studio 中，我们针对启动做了优化，在电脑性能足够的前提下，第一次启动 Nuclei Studio 能在一分钟内完成。

Nuclei Studio 最佳运行环境：

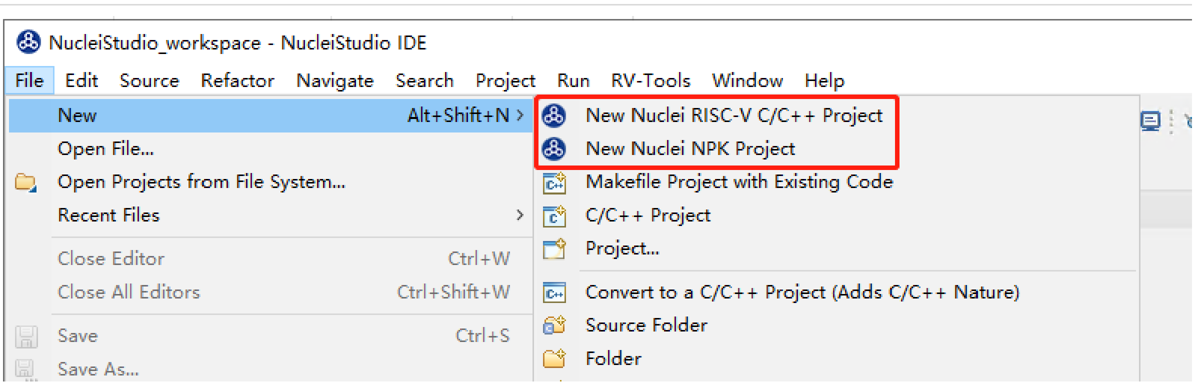
- Windows 10 操作系统
- 4G 以上内存（2G 可以被 Nuclei Studio 分配使用）

2.12.2 Nuclei Studio 编译程序很慢

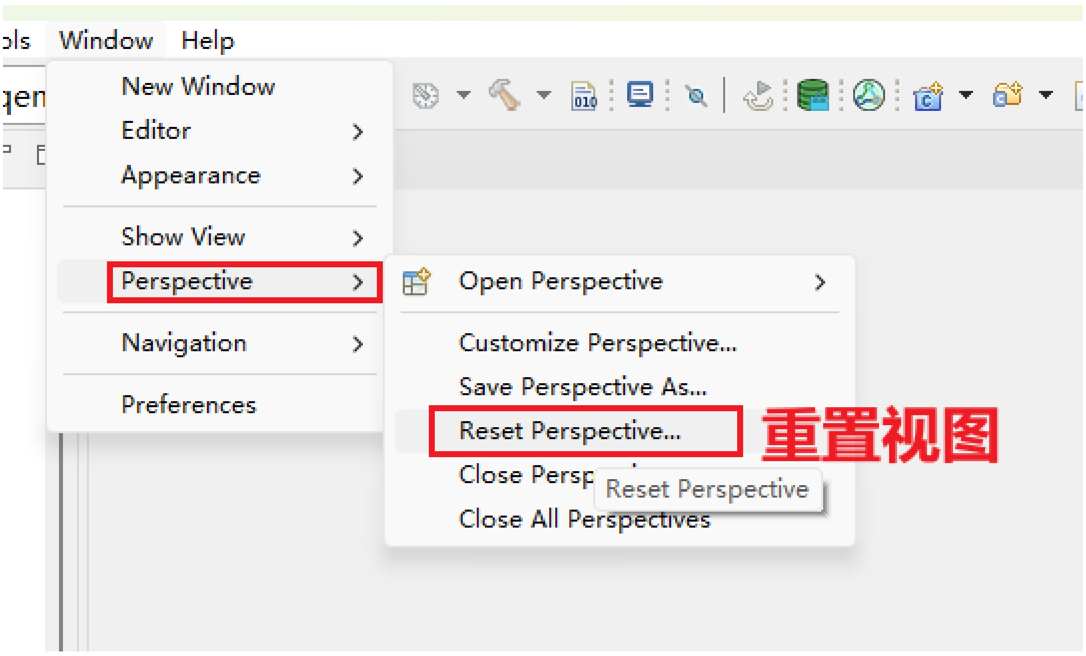
在测试使用过程中，我们发现 Nuclei Studio 在编译项目时会出现很慢的现像，经过排查发现，当电脑安装 360 杀毒软件时，编译指令执行很慢，退出 360 杀毒软件时，编译指令可以按正常速度执行。为了保障 Nuclei Studio 的正常使用，建议在使用时退出 360 等杀毒软件。

2.12.3 找不到 New Nuclei RISC-V C/C++ Project 菜单

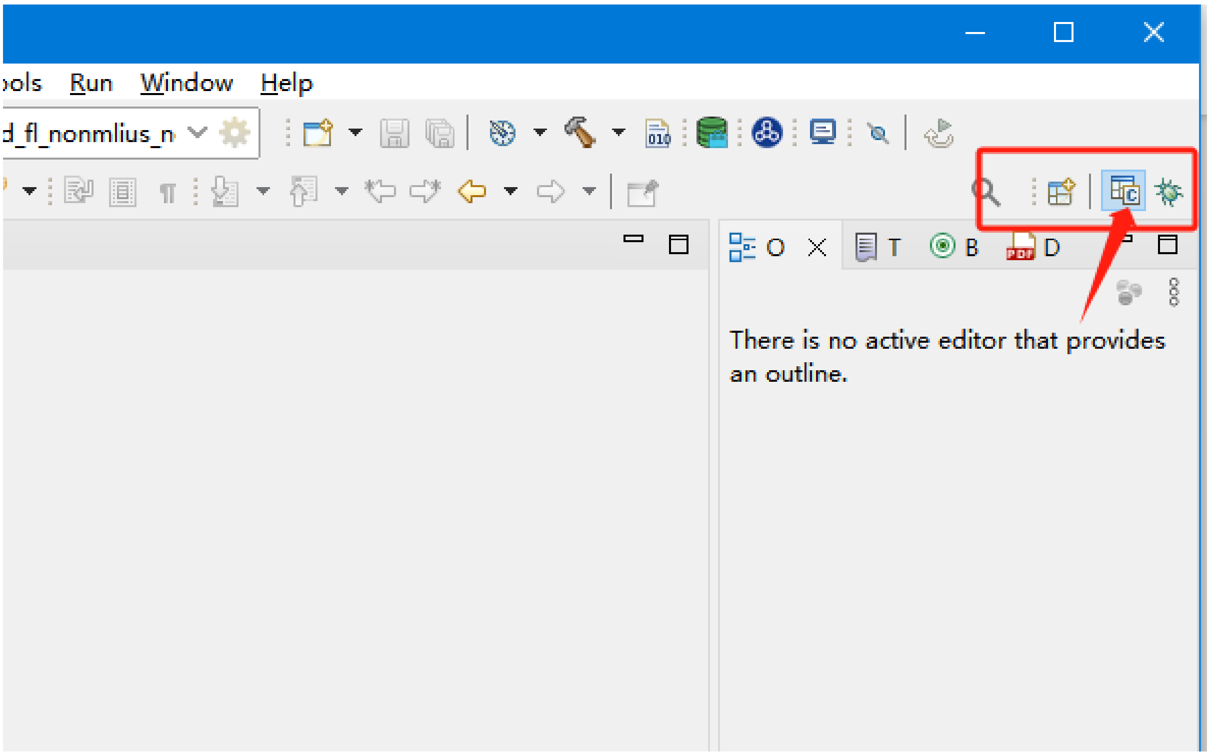
New Nuclei RISC-V C/C++ Project 是创建工程的快捷入口，有时候如果找到不到 New Nuclei RISC-V C/C++ Project 菜单。



可以在菜单 `Window->Perspective->Reset Perspective` 里进行视图重置，`New Nuclei RISC-V C/C++ Project` 就会重新出现。

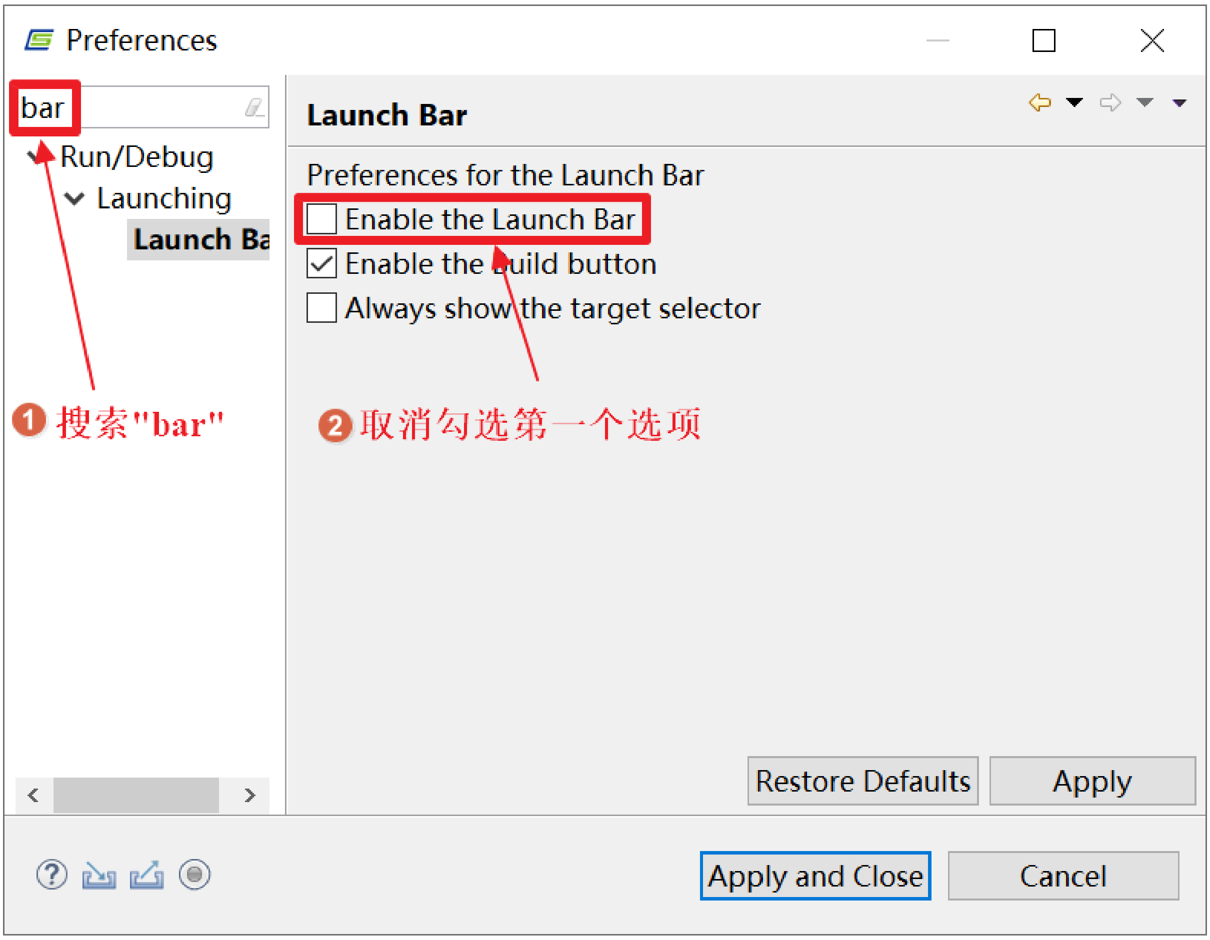


或者将当前视图切换到 C/C++ 视图。



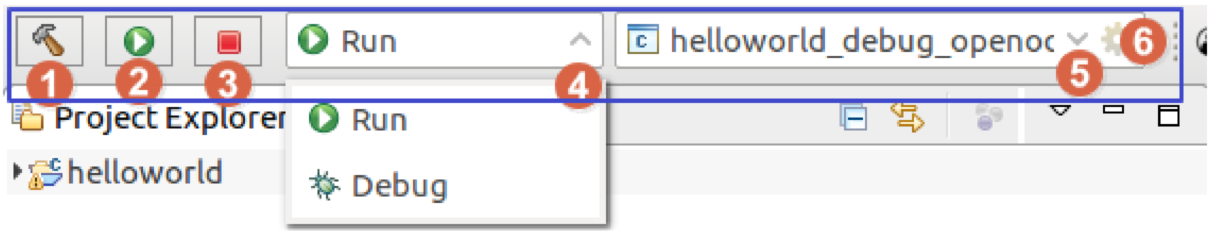
2.12.4 打开/关闭 Launch Bar

Nuclei Studio2022.12 及以后的版本中，默认是开户启了 Launch Bar 功能。打开菜单栏 `Window -> Preferences`，搜索 `bar`，取消勾选第一个选项即可关闭 Launch Bar。同理，想要使用 Launch Bar 功能，请勾选此选项。



2.12.5 使用 Launch Bar

Nuclei Studio 的 Launch Bar（下图中蓝色框内标注部分）是对后边下拉框中选中的调试配置对应的工程进行编译、调试和下载等操作，如需使用此部分请注意，图标是与下拉框中调试配置内容相绑定。下面详细介绍 Launch Bar 各部分功能：



图标 1：编译选中的调试配置对应的工程。此处选中的工程为 5 号下拉框选中的调试配置对应工程，并非 Project Explorer 中选中的工程。

图标 2：运行/调试选中的调试配置对应的工程。此处图标会根据下拉框 4 的内容变化。

图标 3：退出下载/调试模式。

下拉框 4：切换选择下载/调试模式。切换后会改变图标 2 的显示内容。

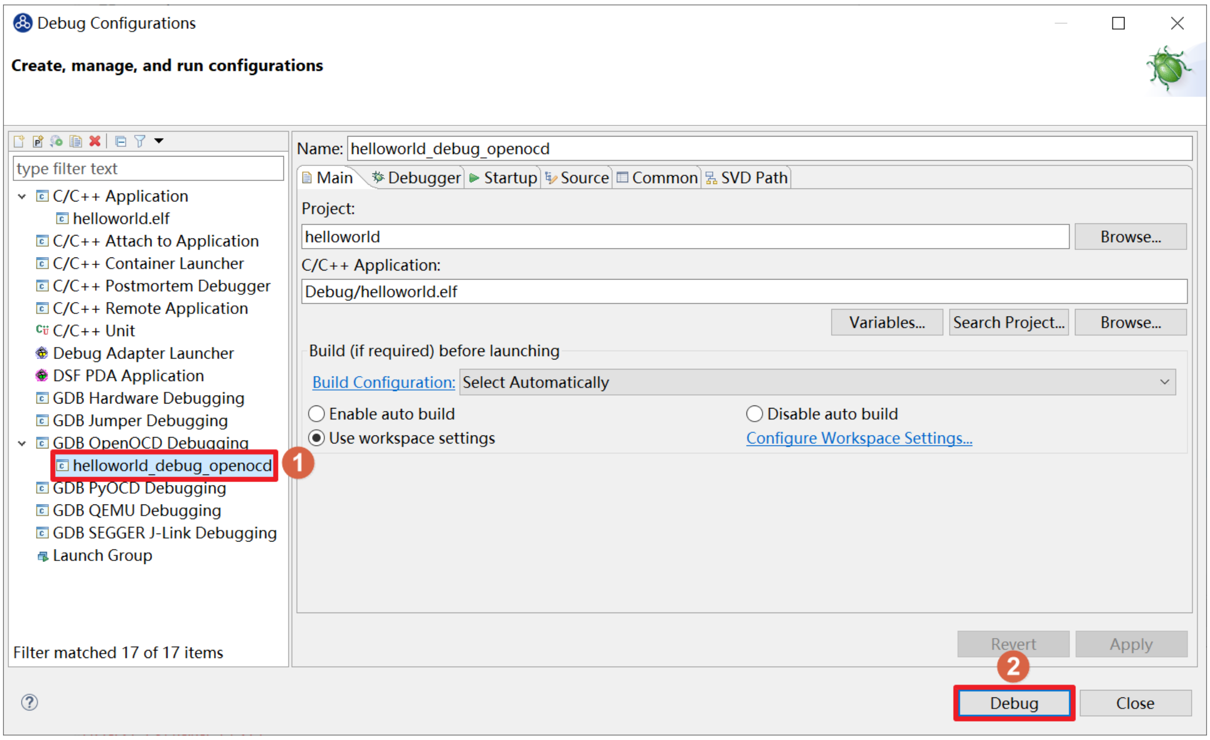
下拉框 5：选中调试配置文件。此处的内容会影响图标 1、2、3 对应的工程。

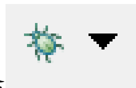
齿轮图标 6：打开当前选中的调试配置页面，可直接进行修改保存。

2.12.6 不使用 Launch Bar 进行运行/调试

对于初次新建的工程，如果不使用 Launch Bar 功能，可以右键工程，选择 `Debug As -> Debug Configurations` 。在弹窗中选择工程对应的设置选项，这里以 helloworld 工程为例，选择 `helloworld_debug_openocd` ，之后选择 `Debug` 开始调试。

同理，初次运行需要右击工程，选择 `Run As -> Run Configurations` 。在弹窗中选择工程对应的设置选项，之后选择 `Run` 开始运行。

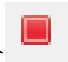


若当前工程已按以上方式进行过调试或运行操作，可在上方按键中选择  的下拉选项，快速选择对应的工程设置进行调试。对于运行，可以在  的下拉选项，快速选择对应的设置。

2.12.7 Debug 页面查看寄存器

进入 Debug 模式后，可能有些页面并不能第一时间找到，这里以查看寄存器栏目为例。在菜单栏选择 `Window -> Show View -> Registers` 打开寄存器栏目，在 Register 页面可以通过 `Ctrl F` 来查找寄存器。同样，如果想查看内存，可以选择 `Window -> Show View -> Memory` ，想打开调试控制台，可以选择 `Window -> Show View -> Debugger Console` ，其他页面此处不做过多介绍，请用户自行体验。

2.12.8 Debug 页面结束进程

在 Debug 页面要退出调试，可点击  退出调试。为防止打开多个进程导致报错，在 Debug 页面，右击 Debug 栏目中的进程，选择 `Terminate / Disconnect All` 关闭所有 Debug 进程。

2.12.9 Nuclei Studio 工具栏中各按键功能

在 Nuclei Studio 中，常见的工具栏图标如下，本节将依次介绍各按键的功能。

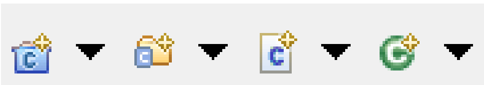
新建按键 ，包括新建各种文件和工程。

保存当前文件  和保存所有未保存文件 。

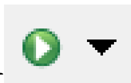
切换编译设置 ，默认只有 Debug 和 Release，用户可自行添加，这里不做介绍。

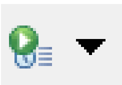
编译工程 ，可以在下拉选项中编译 Project Explorer 中选中的工程。

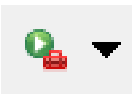
编译所有工程 ，一次性编译所有 Project Explorer 中的工程。

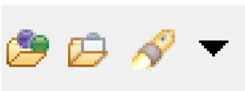
新建功能 ，包含工程创建，目录创建，文件创建和类创建。

调试 ，可根据下拉选项选择不同调试设置来调试不同的工程。

运行 ，可根据下拉选项选择不同的设置来选择不同的工程。

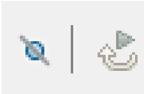
Profile ，使用 Profile 功能，暂不支持，此处不做介绍。

用户自定义工具 ，此处不做介绍，用户可自己深入研究。

搜索工具 ，可搜索任务，文本等。

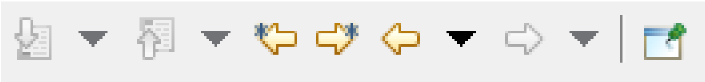
文本阅读工具 ，可以展开或折叠文本等功能。

控制台工具 ，可选择打开各种控制台或串口等。

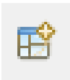
Debug 用，前者跳过所有断点，后者实现 Restart 功能。

查看 CMSIS，eclipse 自带功能，暂不支持。

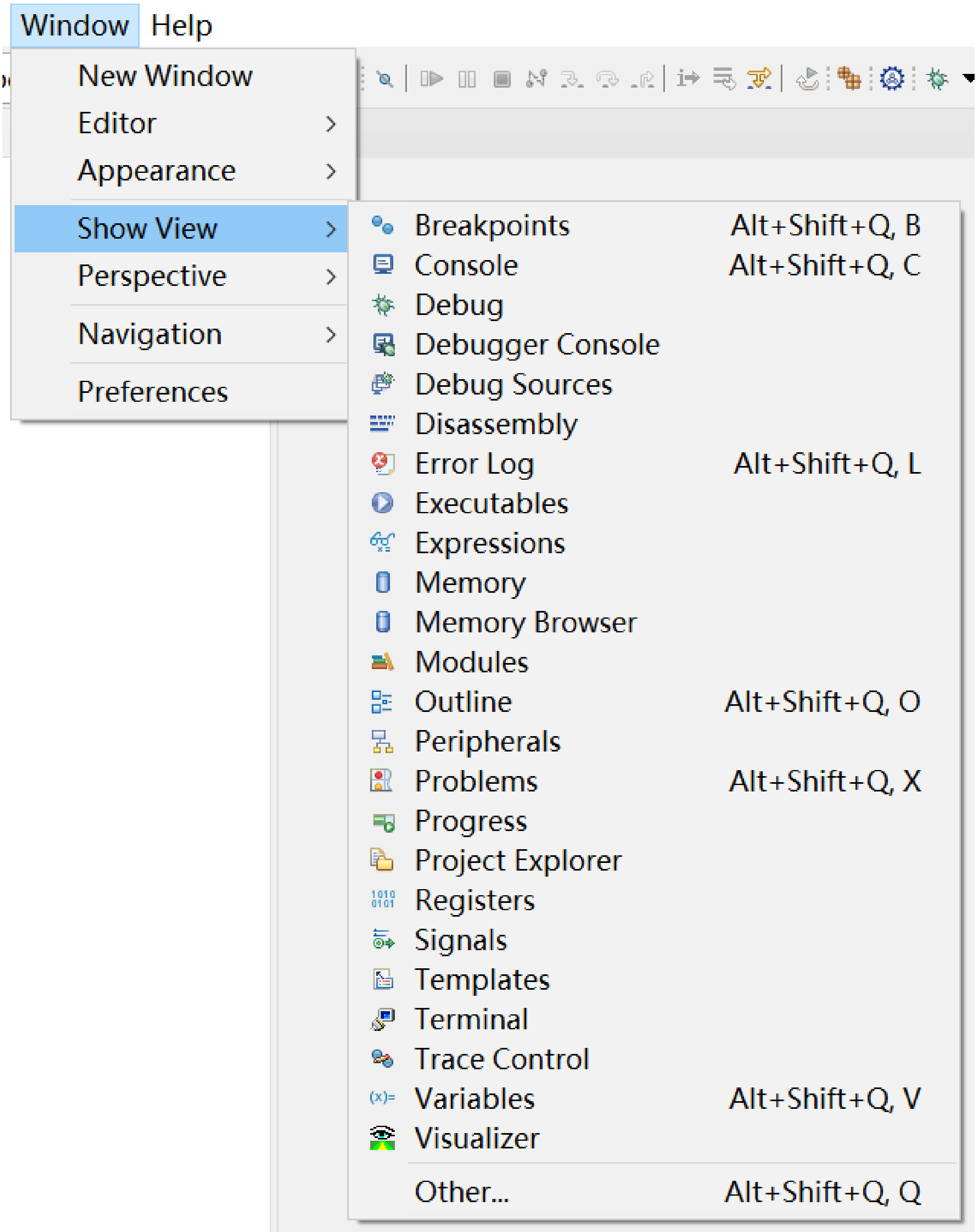
Nuclei SDK 工程设置工具，参见 6.6.1。

文本阅读工具，可设置快速跳转，具体功能此处不作过多介绍。

2.12.10 显示其他窗口

在 IDE 的右上角按键可以切换不同的显示模式，常用的是 C/C++ 页面和 Debug 页面。其中，在进入 Debug 模式时，若不在 Debug 页面，会有弹窗提示，此时点击 `Switch` 选项可以切换至 Debug 页面。在菜单栏中选择 `Window -> Perspective -> Open Perspective` 可以快速切换显示窗口。

在不同的窗口下，`Window -> Show View` 显示的内容也不尽相同。在 Debug 窗口中可以选择显示如寄存器（Registers），内存（Memory）和调试控制台（Debugger Console）。

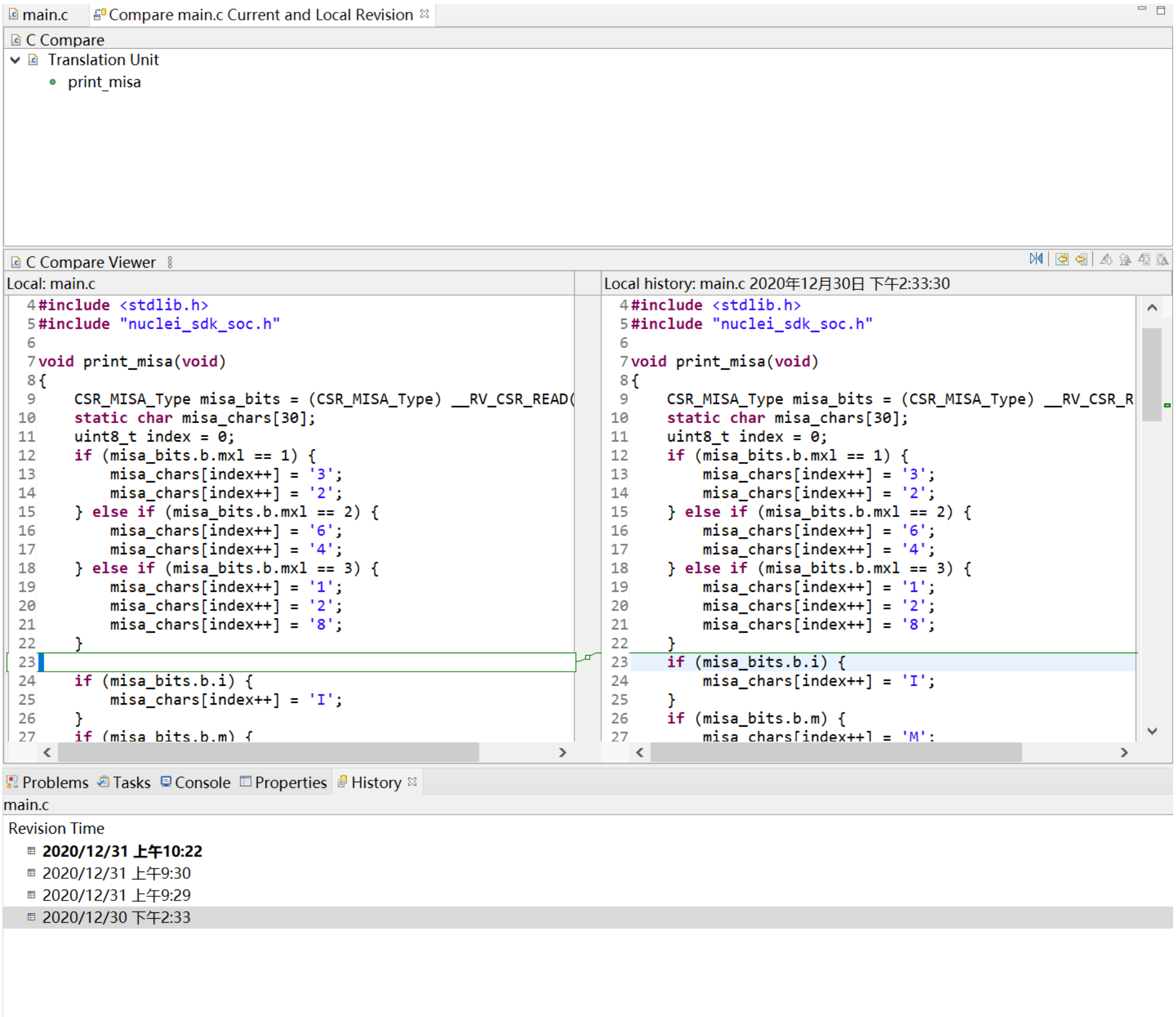

Window
Help
New Window
Editor
Appearance
Show View
Perspective
Navigation
Preferences
Breakpoints Alt+Shift+Q, B
Console Alt+Shift+Q, C
Debug
Debugger Console
Debug Sources
Disassembly
Error Log Alt+Shift+Q, L
Executables
Expressions
Memory
Memory Browser
Modules
Outline Alt+Shift+Q, O
Peripherals
Problems Alt+Shift+Q, X
Progress
Project Explorer
Registers
Signals
Templates
Terminal
Trace Control
Variables Alt+Shift+Q, V
Visualizer
Other... Alt+Shift+Q, Q

2.12.11 恢复默认窗口布局

在菜单栏中选择 `Window -> Perspective -> Reset Perspective`，在弹窗中选择 `Reset Perspective` 可以恢复窗口默认布局。

2.12.12 对比历史文件

右击要查看历史的文件，选择 `Compare With -> Local History` 打开历史记录。在打开的 History 栏目双击选择要对比的文件历史版本，可以查看文件变动历史。



2.12.13 新建工程时可能出现报错

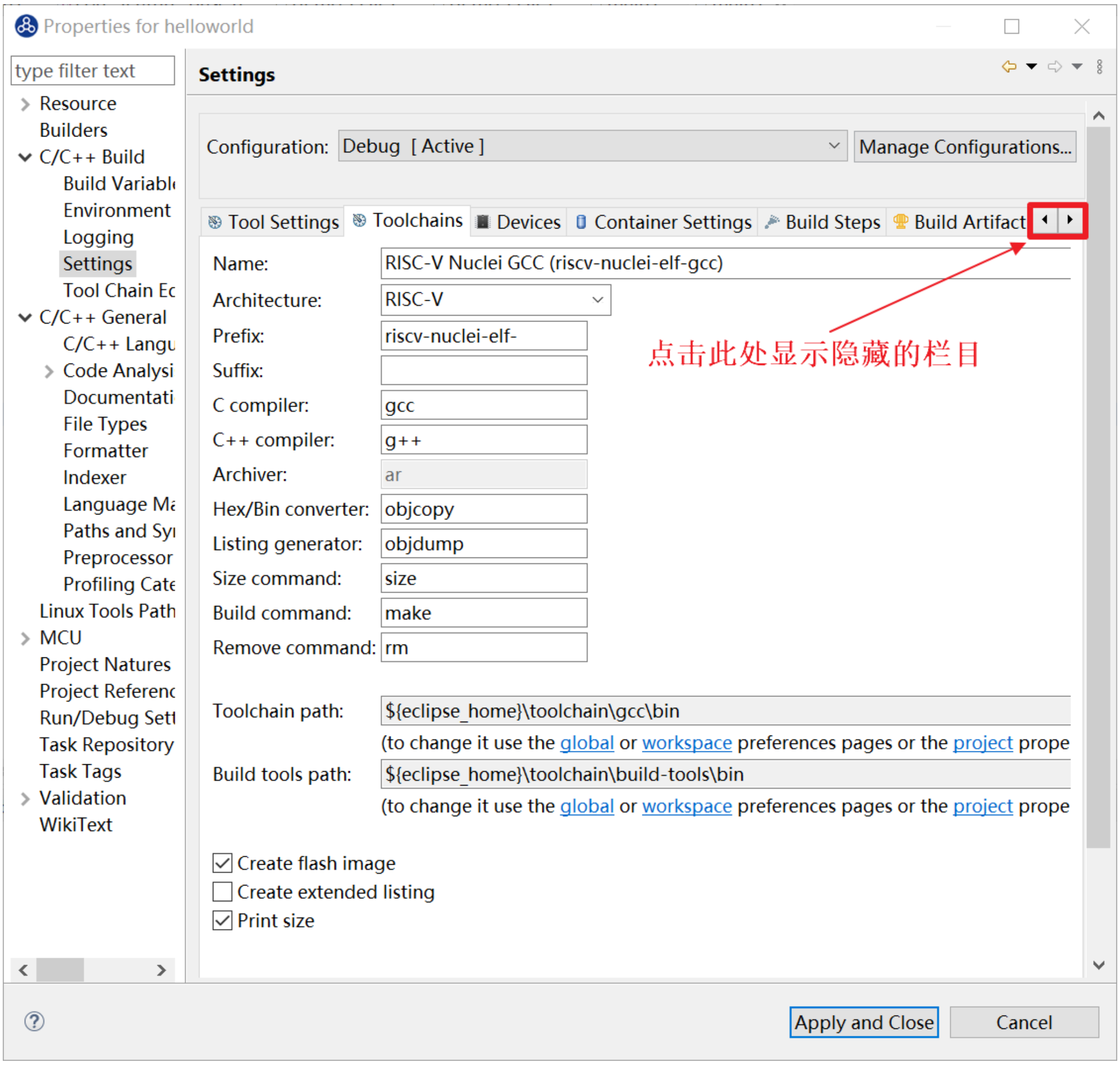
新建工程时 IDE 需要索引整个工程，根据主机性能其所用时间不同。如果主机性能较差，可能会看到索引时产生 error，索引结束后 error 会消失。如出现以上现象，请以编译时是否报错为准。

2.12.14 新增 Include 路径出现缓存

在 include 页面新增路径时，可能会出现缓存的路径内容，此时点击 Workspace 或 File System 选中后可覆盖其内容。

2.12.15 设置页面栏目找不到

有时可能因为弹窗大小，部分设置栏目被隐藏起来，可点击红框中左右方向图标显示隐藏栏目。



2.12.16 开发板下载速度很慢

如果遇到开发板下载速度很慢，甚至出现超时的报错，请切换至使用 USB3.0 接口，如果使用虚拟机开发，也请同时将 USB 接口设置为 3.0。

2.12.17 Linux 环境下多用户使用 Nuclei Studio

如果需要多用户同时使用 Nuclei Studio（不推荐），用户在运行 Nuclei Studio 时首先要打开菜单栏的 `Window->Preferences`。在弹窗中需要修改三个地方：

打开 `MCU->Global OpenOCD Path`，`Executable` 输入 `openocd`，`Folder` 输入 `${eclipse_home}/toolchain/openocd/bin`。

打开 `MCU->Global QEMU Path`，`Executable` 输入 `qemu-system-riscv32`，`Folder` 输入 `${eclipse_home}/tools/qemu`。

打开 `MCU->Global RISC-V Toolchains Paths`，`Default toolchain` 选中 `RISC-V Nuclei GCC`，`Toolchain folder` 输入 `${eclipse_home}/toolchain/gcc/bin`。

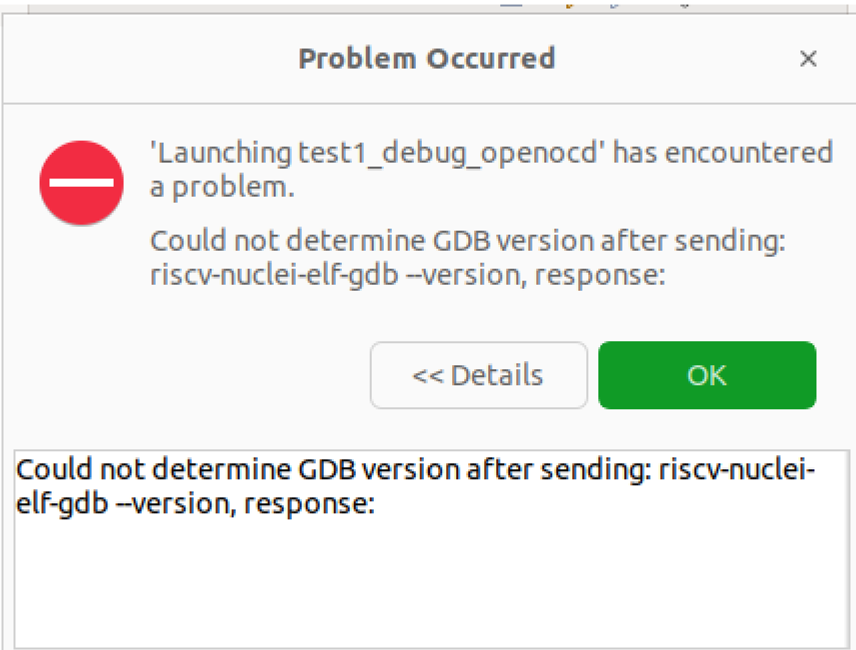
修改后点击 `Apply and Close` 保存并关闭设置弹窗。

2.12.18 设备管理器中识别出两个串口

可能连接设备后在任务管理器中识别出两个串口，其中 COM 编号较大的是串口打印使用，可以使用串口调试助手等工具连接查看打印信息，但是请不要占用 COM 编号较小的串口。

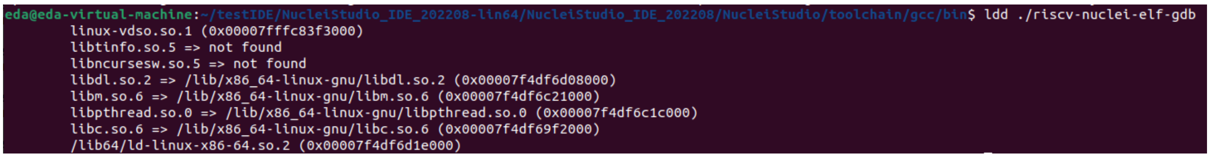
2.12.19 Linux 下使用时报 Could not determine GDB version after sending:riscv-nuclei-elf-gdb version,response: 的错误

第一次在 linux 下使用 Nuclei Studio 时，会报错 `Could not determine GDB version after sending:riscv-nuclei-elf-gdb  version,response:` .



riscv-nuclei-elf-gdb 在 Nuclei studio 2023.10 及之后的版本变更为 riscv64-unknown-elf-gdb。

可以使用命令 `ldd $(which riscv-nuclei-elf-gdb)` 查看，依赖缺失



使用命令 `sudo apt install libncursesw5 libtinfo5` 安装相关依赖后，ide 运行正常，具体可以参考 https://github.com/riscv-mcu/riscv-gnu-toolchain/issues/9

2.12.20 在 linux 下使用 QEMU 时报错

例如在 Ubuntu 20.04 上使用 QEMU 时，可能会报错：

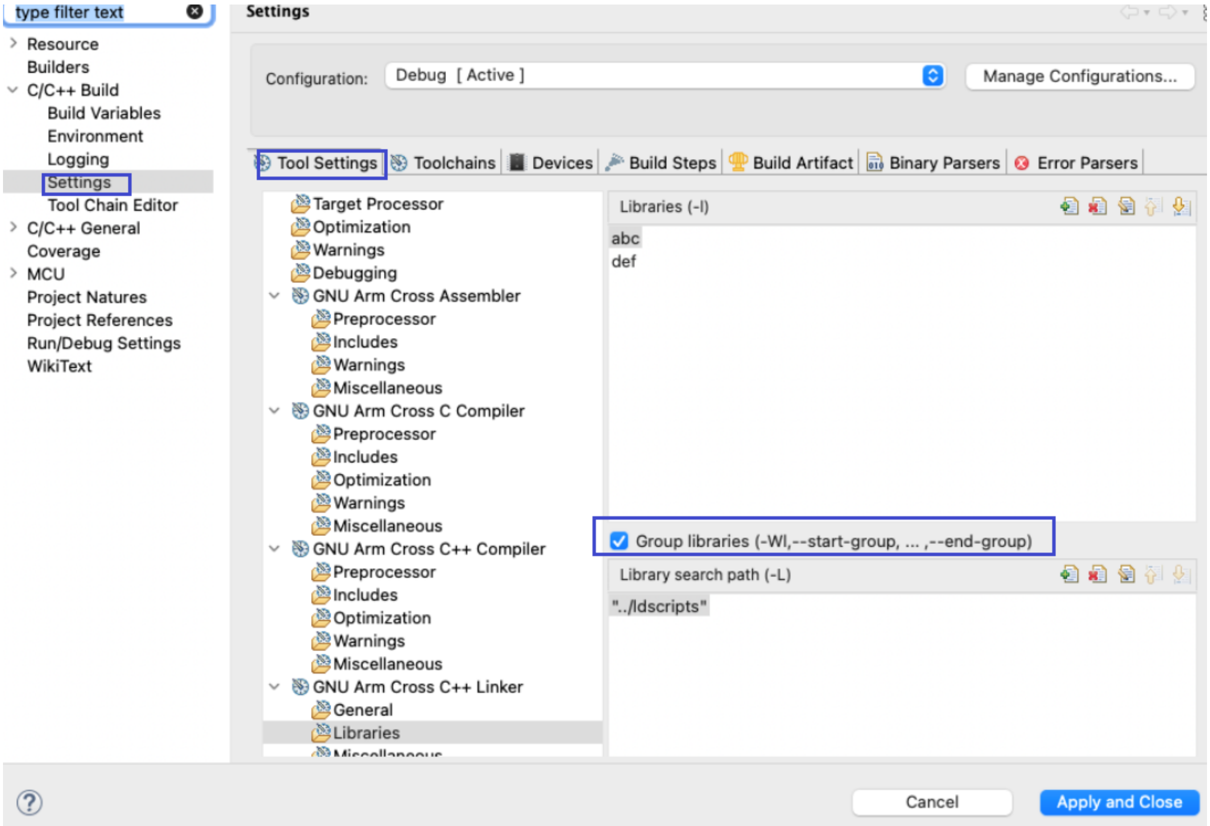
```
error while loading shared libraries: libfdt.so.
↪1: cannot open shared object file: No such file or directory,
```

这是因为缺少 libfdt.so 等依赖所致，只需要在对应版本的 linux 上安装对应的依赖。例如针对 Ubuntu 20.04 可以使用如下命令:

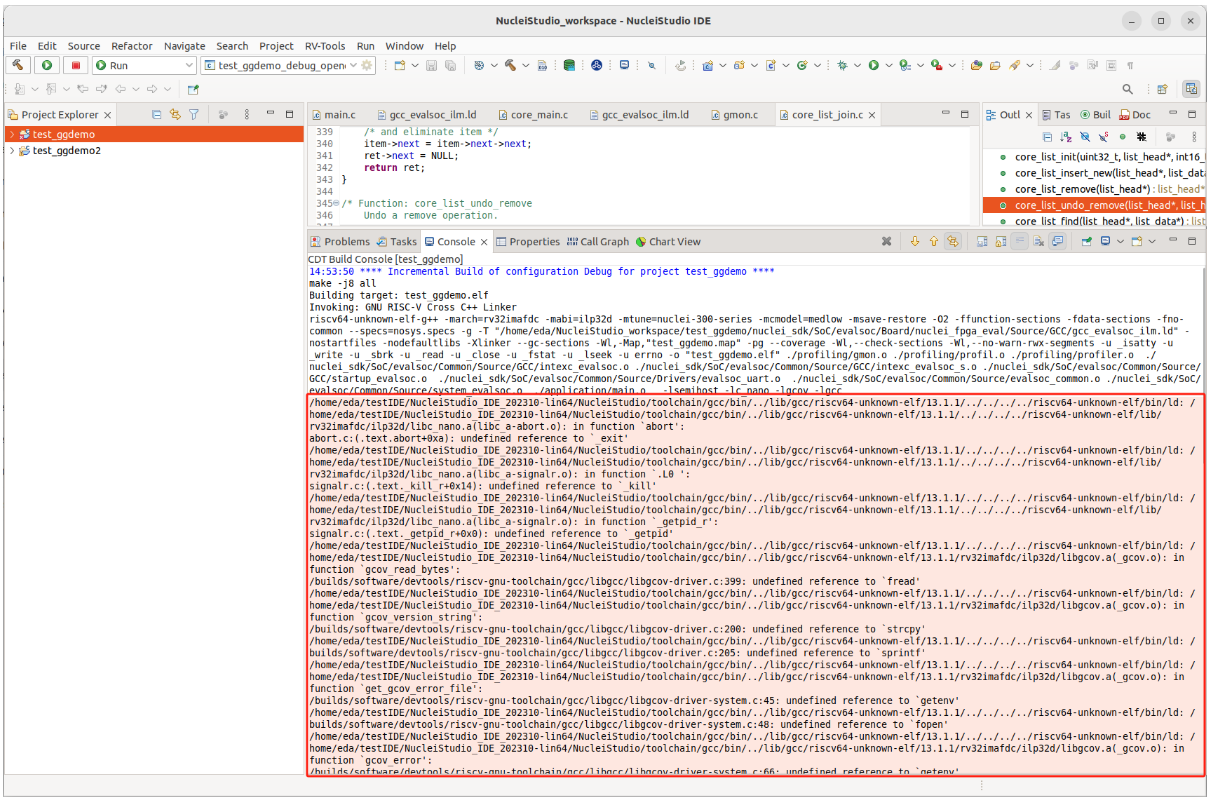
安装 libfdt 等依赖：`sudo apt install libfdt1 libpixman-1-0 libpng16-16 libasound2 libglib2 0-0`

2.12.21 工程编译链接 C 库找不到符号报错

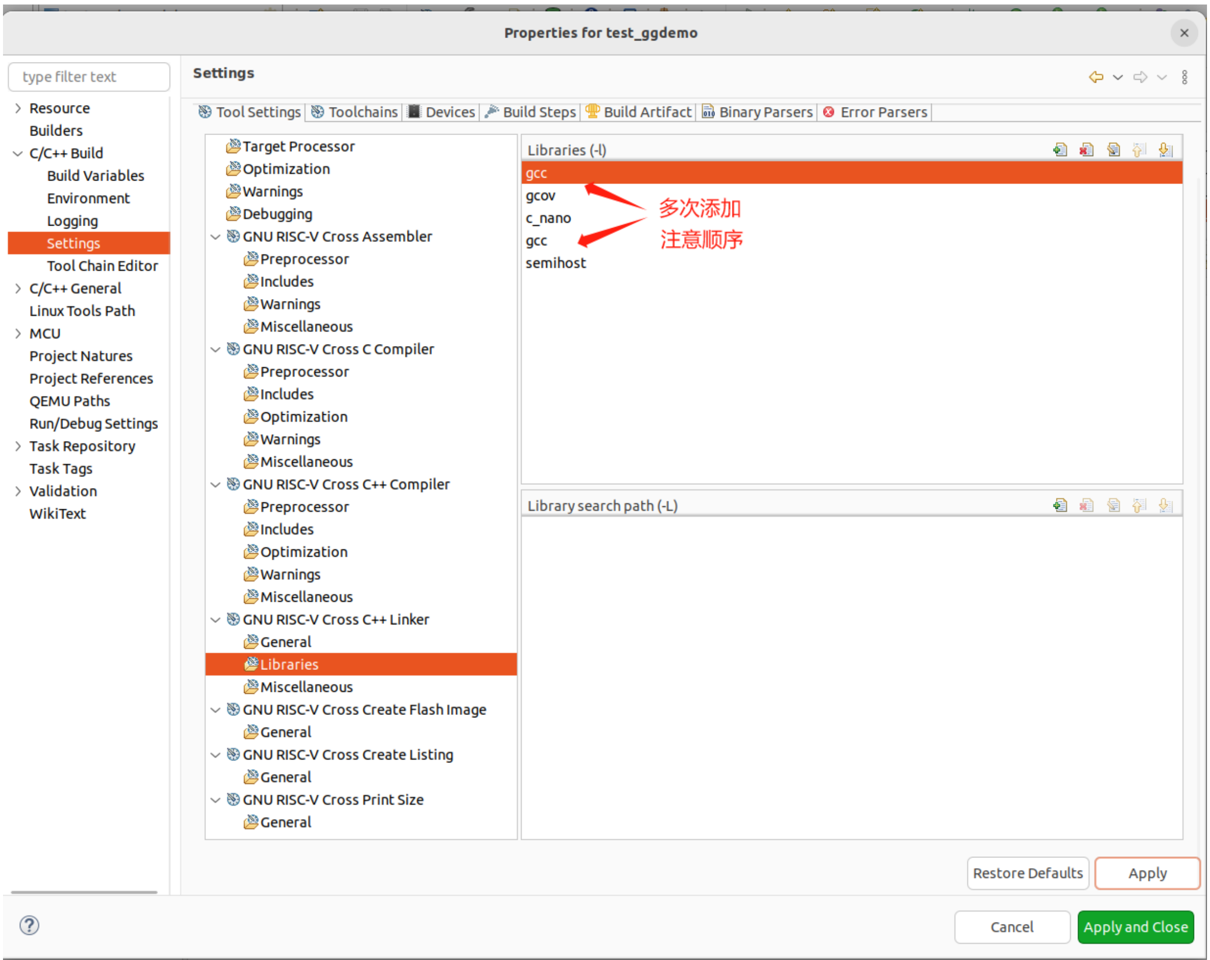
这个问题在 Nuclei Studio 2024.06 可以得到解决，只需要在 Linker -> Libraries 页面勾选 Group Libraries 即可。



因为在对 Libraries 的处理中，没有能很好的处理链接之间的内部依赖关系，当使用的链接之前互相有依赖时，可能会导至编译不通过，详细参见 *https://github.com/eclipse-embed-cdt/eclipse-plugins/issues/592*。解决这个问题，可以有两种办法。



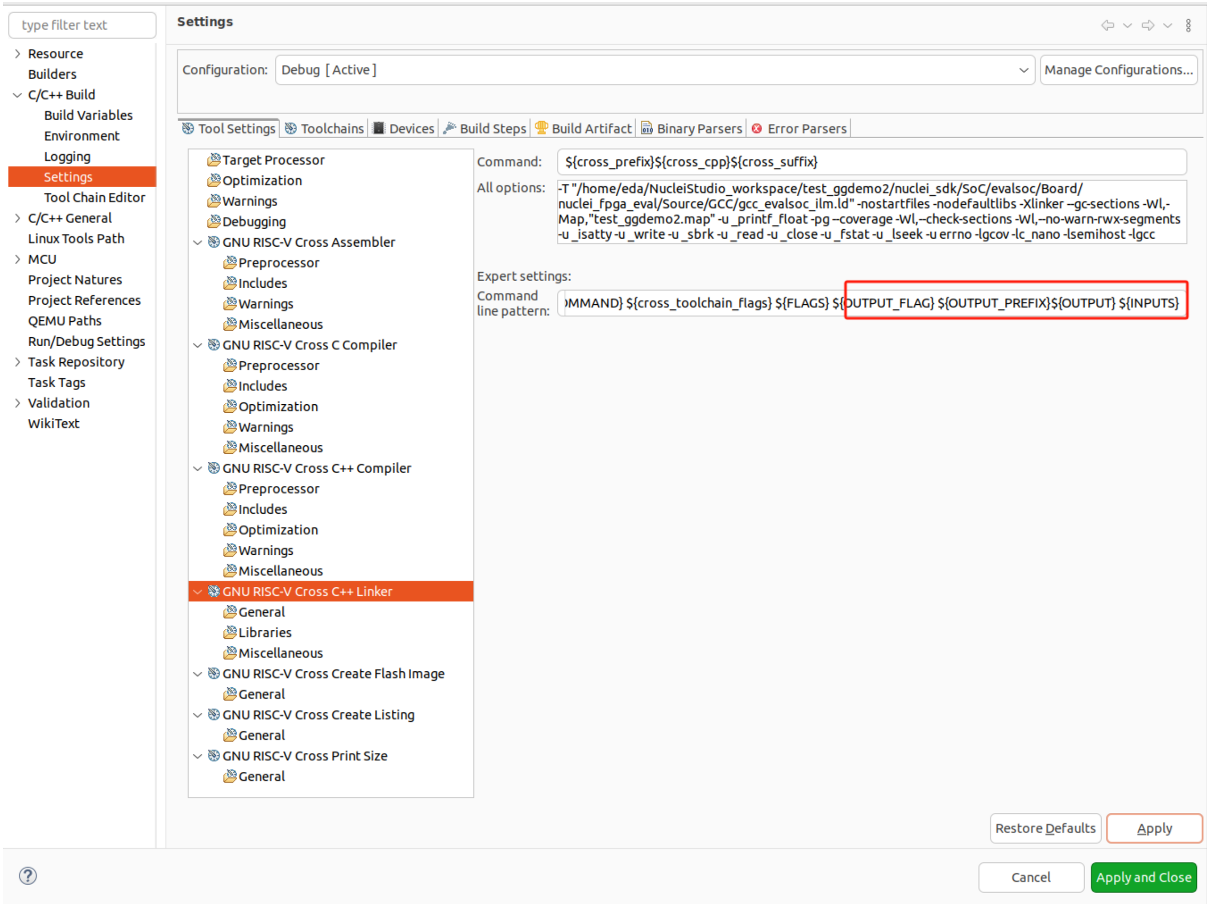
通过调整 Libraries 的顺序或者添加多个链接来实现

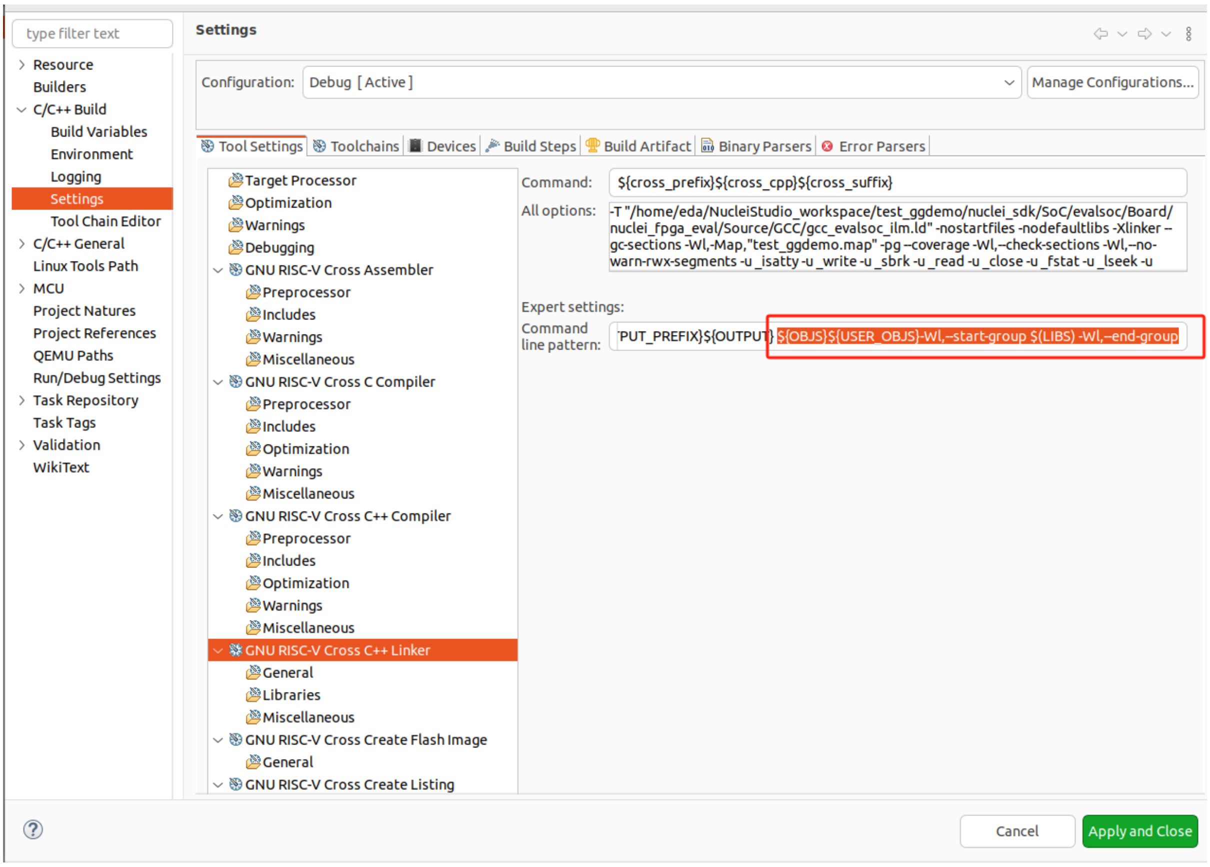


打开 `C/C++ Build -> Settings -> Tool Settings ->GUN RISC-V Cross C++ Linker` , 并修改 Command line paatern 内容，将其修改为

```
${COMMAND}
${cross_toolchain_flags} ${FLAGS} ${OUTPUT_FLAG}
${OUTPUT_PREFIX}${OUTPUT} ${OBJS}${USER_OBJS} -Wl,--start-group
$(LIBS)
-Wl,--end-group,
```

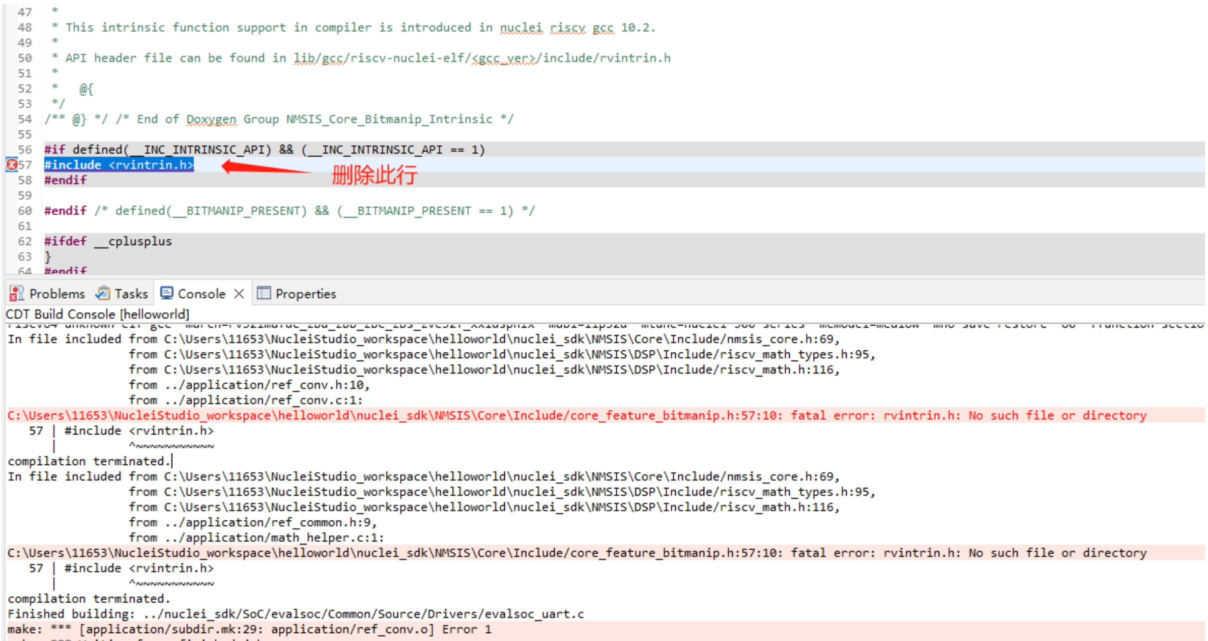
IDE 就会将 Libraries 内的内容以 group 的方式处理。





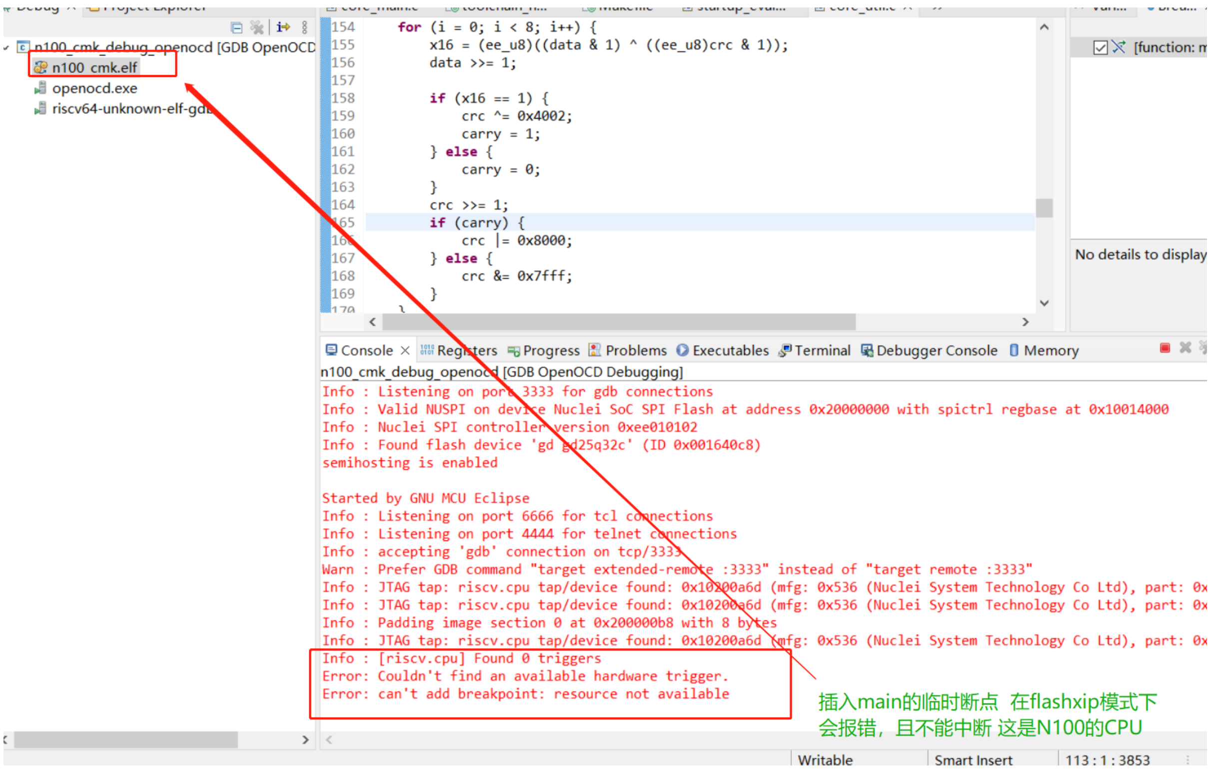
2.12.22 编译工程报错 fatal error: rvintrin.h: No such file or directory

在 Nuclei Studio 2023.10 中，如果使用旧的 sdk 所创建的工程，如果编译时报错 `fatal error: rvintrin.h: No such file or directory`，是因为在 GCC 10 时，工程中 `#include <rvintrin.h>`，而在 GCC 13 中，不需要再 `#include <rvintrin.h>`，只需要删除此行，即可编译通过。



2.12.23 Debug 时报错 Error: Couldn't find an available hardware trigger.

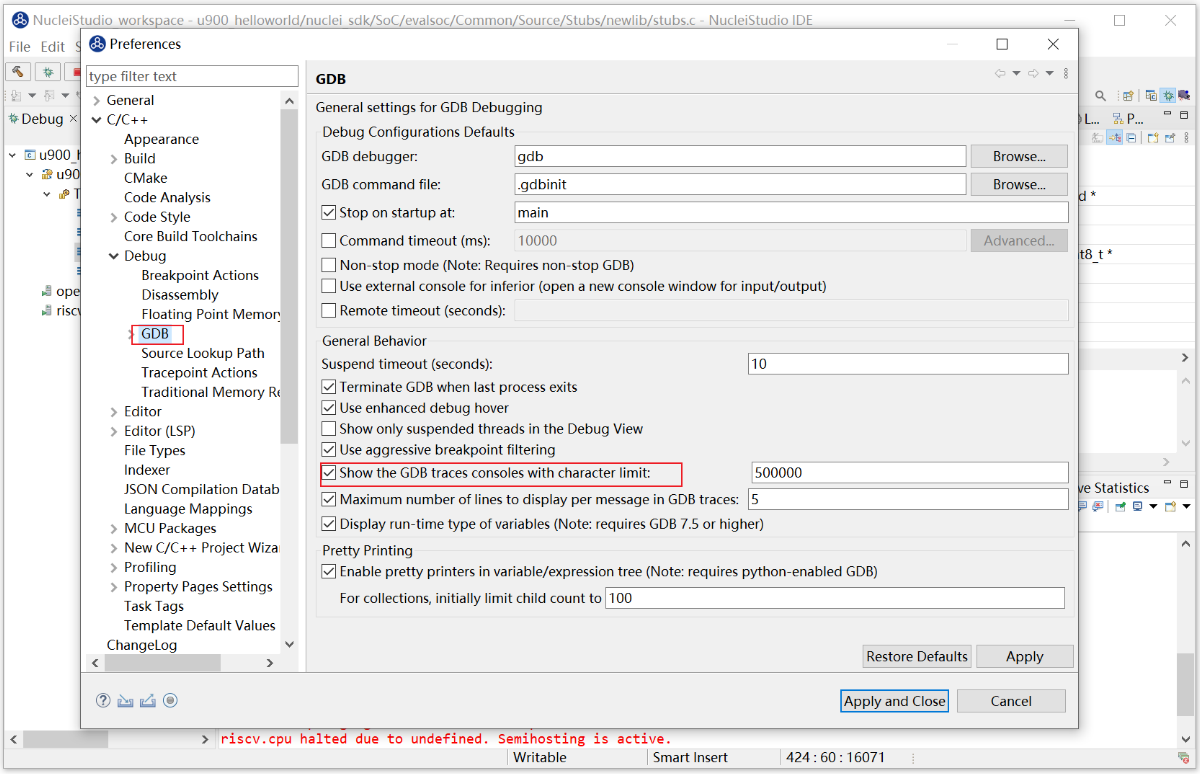
在 Nuclei Studio 环境中，当工程在没有硬件断点的 CPU 硬件上运行时，且选择程序下载到 Flash 中运行以 Nuclei SDK/Nuclei N100 SDK 为例就是 (flash/flashxip DOWNLOAD 模式)，可能会遇到程序 Debug 无法停住的问题，并收到错误提示：`Error: Couldn't find an available hardware trigger`。这是因为程序运行在 Flash 上，软件断点无法被成功写入，而 CPU 上又没有硬件断点可以被使用，从而导致报错。



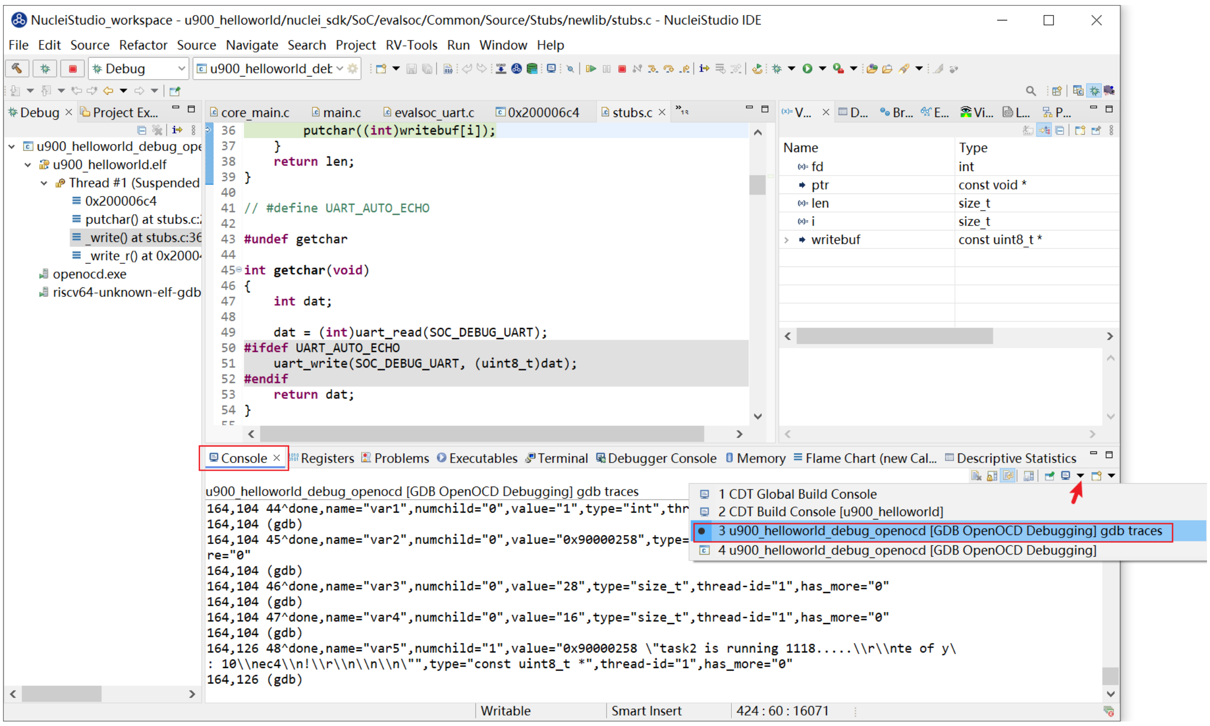
这种情况下需要将程序编译到 RAM 上才可以支持 IDE 上进行调试（软件断点），如果需要调试则暂时只能通过命令行的方式进行调试。

2.12.24 Debug 工程时查看 GDB Trace

在 Nuclei Studio 环境中，GDB Trace 功能详细记录了在调试程序过程中使用的 GDB 命令，默认情况下该功能是关闭，如需要查看 GDB Trace，可以开启该功能。在 NucleiStudio 的菜单中依次点击 `Window -> Preferences`，在弹出的 Preferences 窗口中依次选择 `C/C++ -> Debug -> GDB`，找到 `Show the GDB traces consoles with character limit` 选项并勾选。



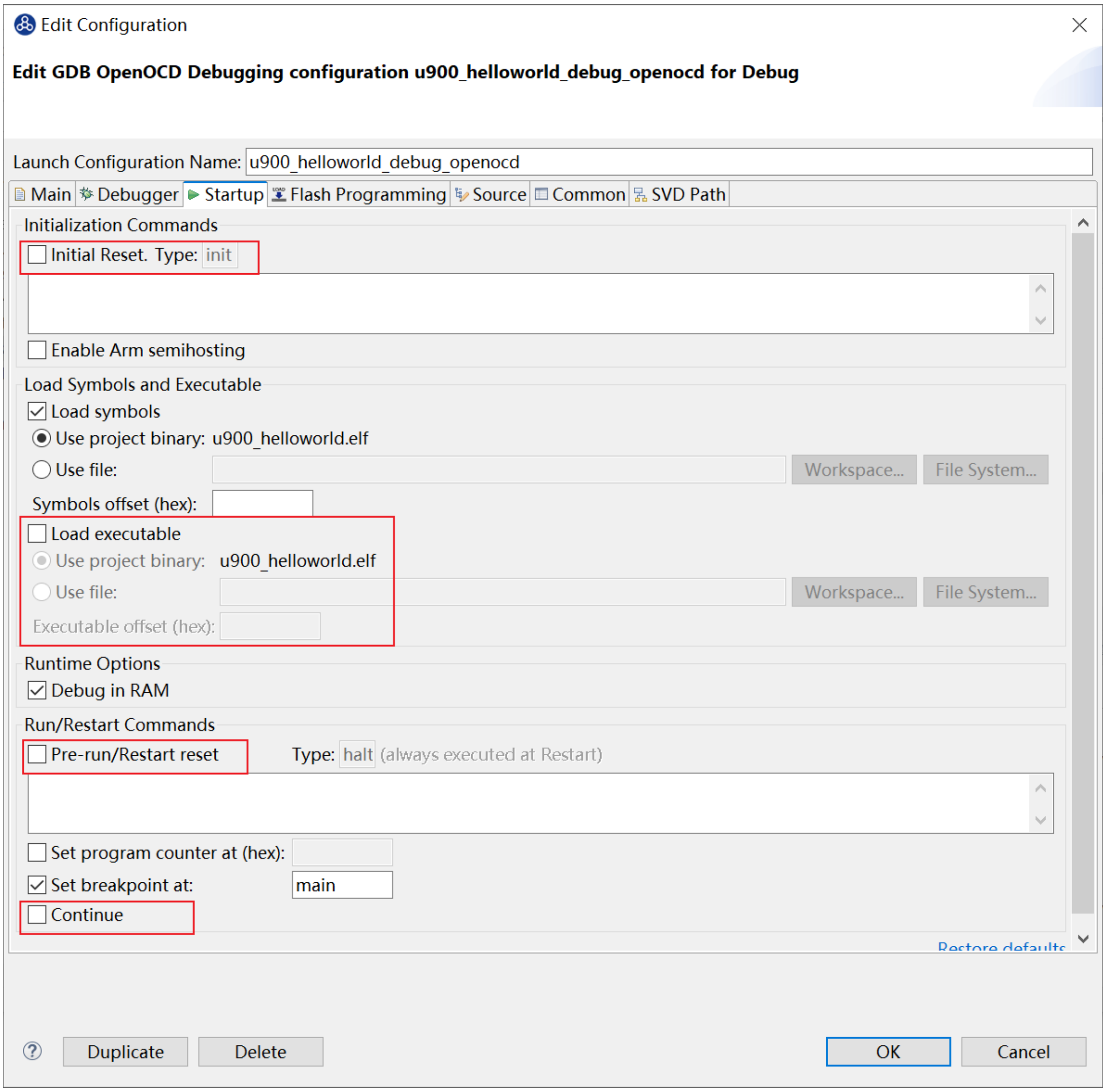
在 Debug 时，可以在 Console 窗口中，点击 `Display Selected Console` 切换到 GDB Trace Console，就可以查看到 GDB Trace，用以帮助分析调试过程中的问题。



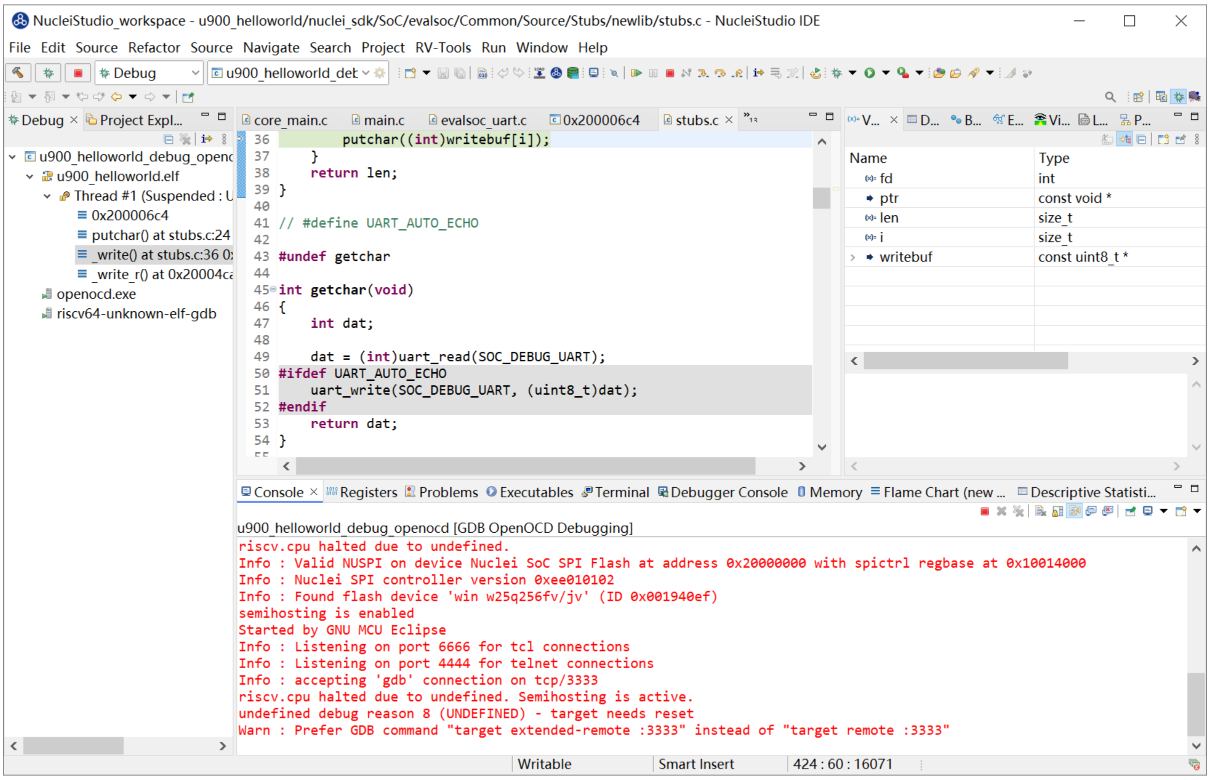
2.12.25 Debug 工程时链接到运行中的 Target

在 Nuclei Studio 环境中使用 OpenOCD Debug 工程时，链接到运行中的 Target，可以通过如下操作实现。打开 Debug_openocd 的配置文件，选中 `Startup` 页并修改配置如下图：

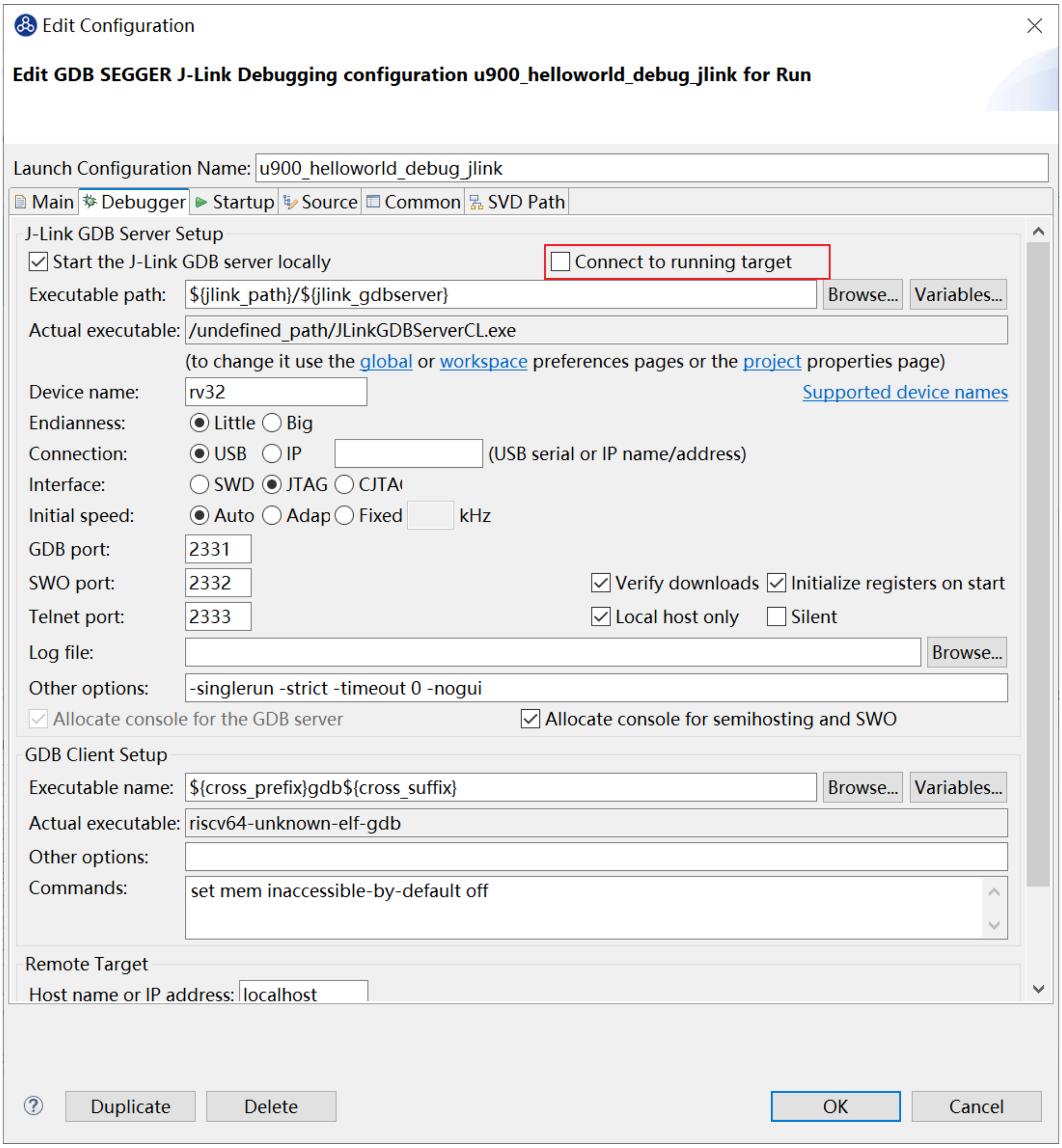
- Init reset 去掉勾选，避免发 monitor reset 之类的命令导致 cpu 复位；
- Load executable 去掉勾选，避免重新下载 elf；
- Pre-run/restart commands 去掉勾选，避免发 monitor reset 之类的命令导致 cpu 复位；
- Continue 去掉勾选，当 OpenOCD 连上去以后，直接进入调试状态且运行暂停。



点击 Debug 按钮，可以看到已经连接到运行中的 Target。

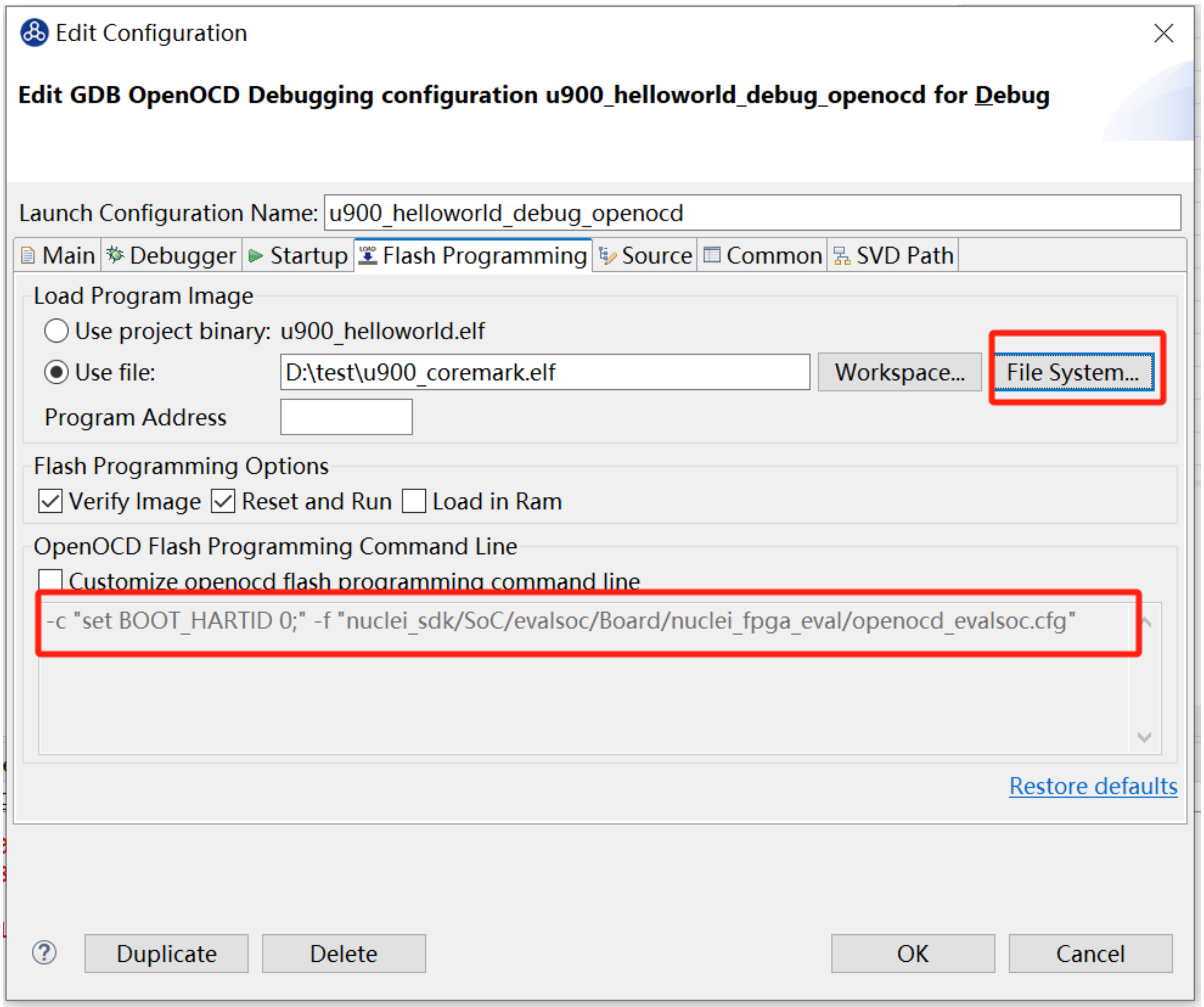


使用 Jlink Debug 工程时，链接到运行中的 target，可以通过如下操作实现。打开 Debug_jlink 的配置文件，选中 `Debugger` 页并修改配置如下图，点击 Debug 按钮，可以看到已经连接到运行中的 Target。

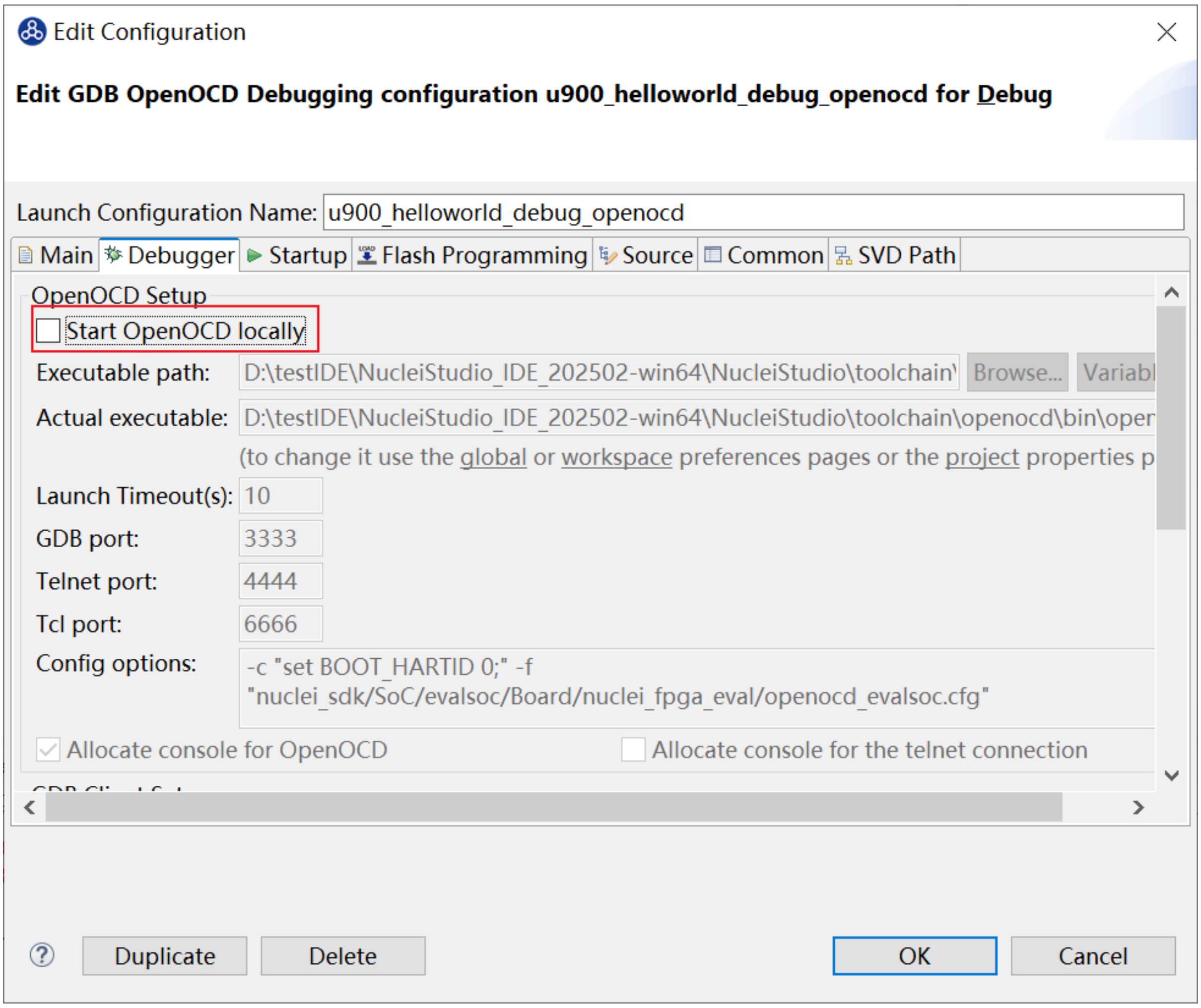


2.12.26 Flash Programming 中已知缺陷

在选中需要使用 Flash Programming 功能时，如果通过 `File System` 选择 Load 的文件，因代码缺陷会导致命令不完整。为规避此类问题，尽量将需要选择的文件放在工程目录下



Flash Programming 依赖 OpenOCD，当在 Nuclei Studio 中配置了不启动 OpenOCD 的前提下使用 Flash Programming，会导致 Nuclei Studio 卡死，如出现上述情况，请确认配置是否正确。



2.12.27 Nuclei Studio 调试工程前的硬件检测

在使用 **Nuclei Studio IDE** 进行软件开发和调试之前，必须确保目标硬件（如基于 FPGA 的 RISC-V 系统）已正确配置并稳定运行。以下步骤可用于验证硬件平台是否具备进行正常软件调试的条件。

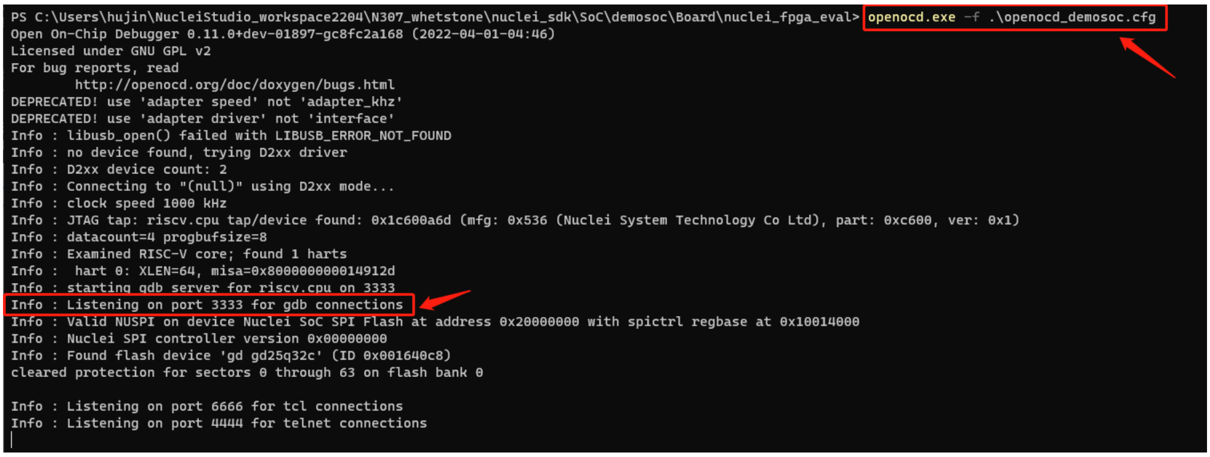
检查 OpenOCD 配置文件 *openocd.cfg* 是否正确

OpenOCD 是用于与 FPGA 上的 RISC-V 核进行 JTAG 调试的重要工具。要启动调试会话，需准备一个正确的 *.cfg* 配置文件。通过以下步骤以确认 `openocd.cfg`文件是否正确。

- 使用 **Nuclei Studio** 安装目录下的 **OpenOCD** 工具：
  - Windows： *NucleiStudio/toolchain/openocd/bin/openocd.exe*
  - Linux： *NucleiStudio/toolchain/openocd/bin/openocd*
- 将配置文件（如 *openocd_demosoc.cfg*）复制到 OpenOCD 的 bin 目录中。
- 执行命令启动 OpenOCD：

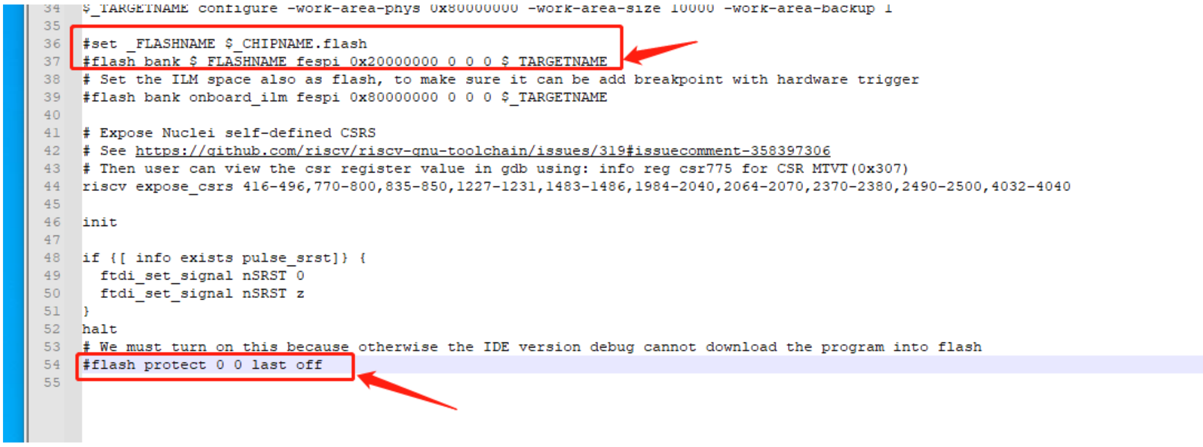
```
openocd -f openocd.cfg
```

如果 OpenOCD 成功启动并显示连接信息，则表示 FPGA 板上的 JTAG 接口和 CPU 正常工作。



注意事项：

- 若 FPGA 上没有 SPI Flash 或无需烧录 Flash，可删除 *.cfg* 文件中关于 Flash 的配置项。
- 确保配置文件中的 JTAG 引脚、CPU 类型、频率等参数与你的硬件一致。



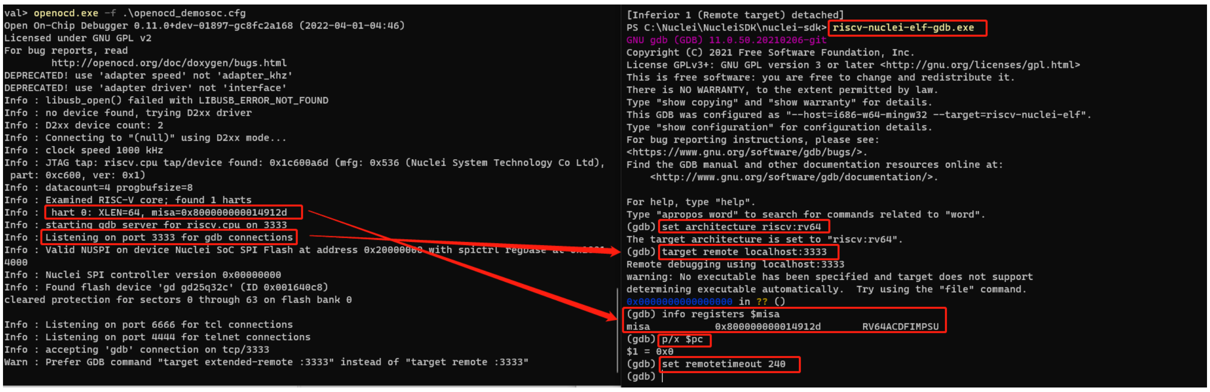
使用命令行确认 GDB 与 OpenOCD 能正常通信

为了进一步验证硬件状态，可以使用 GDB 命令行工具与 OpenOCD 建立连接。

- 启动 OpenOCD（见上一步）
- 新开终端，进入 GDB 工具路径：
  - Windows：*NucleiStudio/toolchain/gcc/bin/riscv-nuclei-elf-gdb.exe*
  - Linux：*NucleiStudio/toolchain/gcc/bin/riscv-nuclei-elf-gdb*
- 启动 GDB，并执行以下命令：

```
(gdb) set arch riscv:rv32      # 或 rv64，根据 CPU 架构选择
(gdb) set remotetimeout 240    # 设置远程连接超时时间
(gdb) target remote :3333      # 连接 OpenOCD 默认端口
```

若能成功连接，则表明 FPGA 上的 JTAG 接口和 RISC-V Core 正常运行。

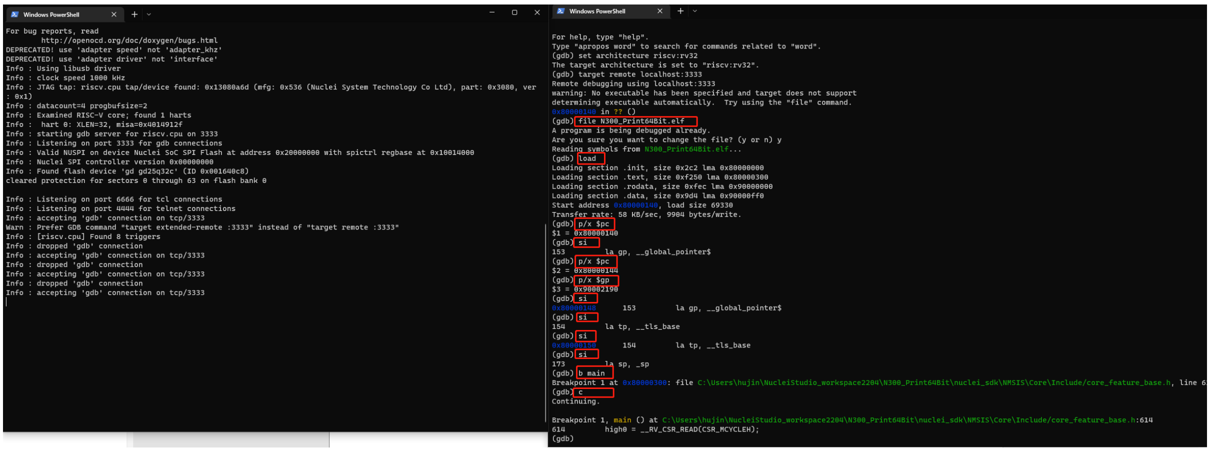


可能问题：

- FPGA 上的 CPU 主频较低 → 首次连接可能会超时，建议提前设置 *set remotetimeout 240* 。具体可以参考*CPU 主频比较低导致 Nuclei Studio 无法调试* (page 294) 。

使用 GDB 命令检查底层硬件功能

一旦 GDB 成功连接，可以通过一系列命令验证 CPU、SRAM 和外设寄存器的功能。



测试项目及命令：

读写通用寄存器（GPR）

```
(gdb) p/x $pc          # 查看当前 PC 值
(gdb) p/x $a0          # 查看 a0 寄存器值
```

读写 CSR 寄存器

```
(gdb) info reg $mstatus     # 查看 mstatus 寄存器
(gdb) info reg $misa        # 查看支持的指令集架构
```

读写 SRAM 内存（验证内存稳定性）

- 准备一个测试 bin 文件，加载到 SRAM：

```
(gdb) restore test.bin binary 0x80000000
```

- 从内存 dump 出来并比对：

```
(gdb) dump binary memory dump.bin 0x80000000 0x80001000
```

- 检查 *test.bin* 和 *dump.bin* 是否完全一致。

如果数据不一致，说明 SRAM 可能存在时序或接口问题。

8bit/16bit 数据访问测试

- 验证是否支持字节级访问：

```
(gdb) x/1bx 0x80000000        # 读取单个字节
(gdb) x/1hx 0x80000000        # 读取半字（16 位）
```

读写 **SOC** 外设寄存器

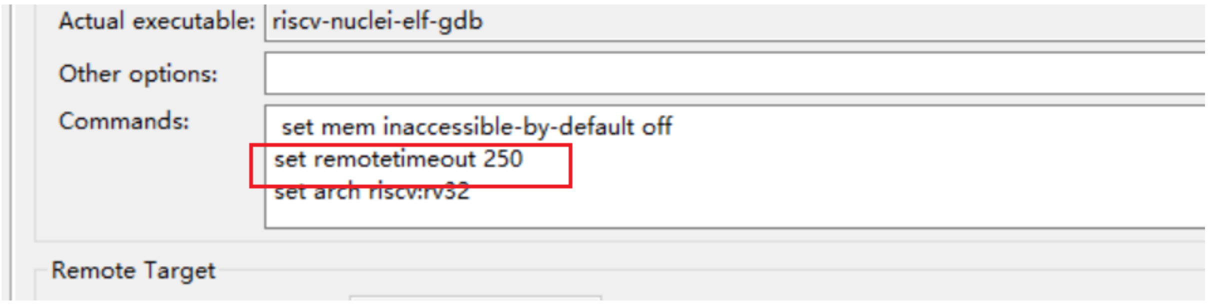
- 对映射在外设地址空间的寄存器进行读写测试：

```
(gdb) p/x *(int *)0x10000000    # 读取某个外设寄存器
(gdb) set *(int *)0x10000000 = 0x1   # 写入值
```

```
(gdb) restore ../ff-4K.bin binary 0x80000000
Restoring binary file ../ff-4K.bin into memory (0x80000000 to 0x80001000)
(gdb) dump binary memory test.bin 0x80000000 0x80000FFF
(gdb) x/50xw 0x80000000
0x80000000:     0xffffffff      0xffffffff      0xffffffff      0xffffffff
0x80000010:     0xffffffff      0xffffffff      0xffffffff      0xffffffff
0x80000020:     0xffffffff      0xffffffff      0xffffffff      0xffffffff
0x80000030:     0xffffffff      0xffffffff      0xffffffff      0xffffffff
0x80000040:     0xffffffff      0xffffffff      0xffffffff      0xffffffff
0x80000050:     0xffffffff      0xffffffff      0xffffffff      0xffffffff
0x80000060:     0xffffffff      0xffffffff      0xffffffff      0xffffffff
0x80000070:     0xffffffff      0xffffffff      0xffffffff      0xffffffff
0x80000080:     0xffffffff      0xffffffff      0xffffffff      0xffffffff
0x80000090:     0xffffffff      0xffffffff      0xffffffff      0xffffffff
0x800000a0:     0xffffffff      0xffffffff      0xffffffff      0xffffffff
0x800000b0:     0xffffffff      0xffffffff      0xffffffff      0xffffffff
0x800000c0:     0xffffffff      0xffffffff
(gdb) set *(short *)0x80000000=0x1122
(gdb) x/8b 0x80000000
0x80000000:     0x22    0x11    0xff    0xff    0xff    0xff    0xff    0xff
(gdb) set *(char *)0x80000000=0x55
(gdb) x/8b 0x80000000
0x80000000:     0x55    0x11    0xff    0xff    0xff    0xff    0xff    0xff
(gdb) p/x $a0
$4 = 0x0
```

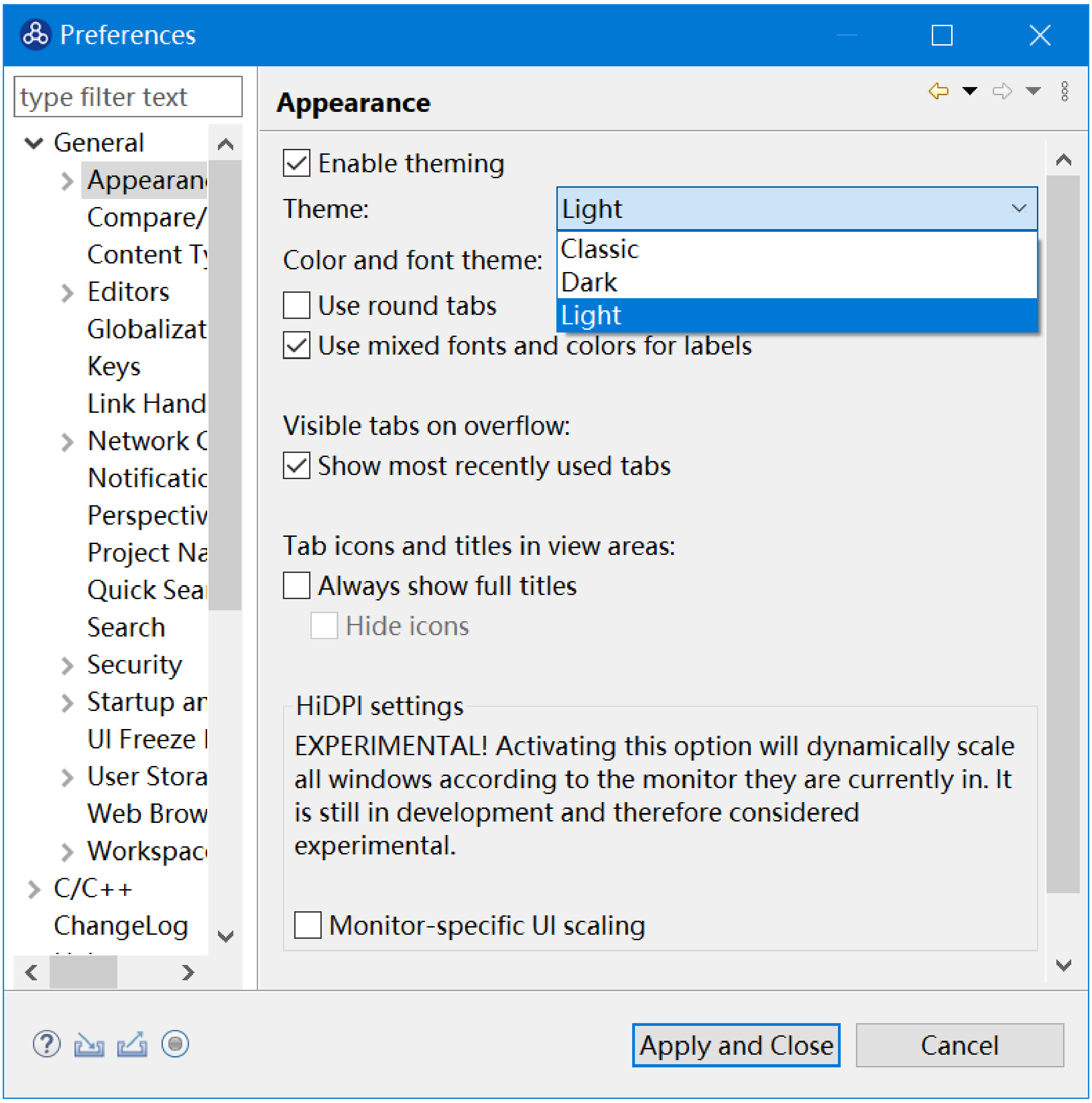
2.12.28 CPU 主频比较低导致 Nuclei Studio 无法调试

如果 CPU 主频较低，可能导致 Nuclei Studio 调试失败或不稳定。此时建议在 OpenOCD 配置文件中降低 JTAG 主频，一般应小于 CPU 主频的一半，推荐设置为 CPU 主频的 1/4 左右。在 Nuclei Studio 中使用时也需要修改 *set remotetimeout 250* 命令，将超时时间设为较大的值（单位为秒），以提升连接成功率。



2.12.29 设置 Nuclei Studio 主题配色

Nuclei Studio 是基于 eclipse 开发，没有针对主题配色进行修改，所以它继承了 eclipse 的主题配色功能。如果想要修改 Nuclei Studio 的主题配色，用户可以自行搜索并参考 eclipse 的主题配色修改方法。也可以通过配置选项快速配置通用主题。



2.13 其他未注明版本问题

如本文档中有疏漏的地方，请关注 https://www.rvmcu.com/NucleiStudio-faq.html[21] 这里将列出不同版本后续遇到的常见问题。

[21] https://www.rvmcu.com/nucleistudio-faq.html

CHAPTER

THREE

NUCLEI TOOLCHAIN

3.1 GNU Toolchain

3.1.1 About GNU Toolchain

The official toolchain repository is located at https://github.com/riscv-collab/riscv-gnu-toolchain.git.

Nuclei maintained toolchain repo is located at https://github.com/riscv-mcu/riscv-gnu-toolchain.

The latest release **2025.02** branch for gcc14 and llvm19 is `nuclei/2025.02`, in which the tools included versions are: gcc 14.2.1, binutils 2.44, gdb 16.2, newlib 4.4.0, llvm 19.1.7, glibc 2.40, and also have merged some important patches from their upstream, as well as additional support for Nuclei custom extensions and pipelines, etc.

For the 2024.06 toolchain branch(gcc13 + llvm17), you can check out the `nuclei/2024-gcc13` branch.

3.1.2 Extensions Support

Standard Extensions

- Basic Extensions

  i, m, a, f, d, c, h, q(Assembly only), zaamo, zalrsc, zicsr, zifencei, zicond, zawrs, zfh, zfhmin, zmmul, svinval, svnapot.

- Z*Inx Extensions

  zfinx, zdinx, zhinx, zhinxmin.

- CMO Extensions

  zicboz, zicbom, zicbop.

- Bitmanip Extensions

  zba, zbb, zbc, zbs

- Crypto Extensions

  zbkb, zbkc, zbkx, zknd, zkne, zknh, zkr, zks, zksed, zksh, zkt.

- Vector Extensions

  zve32x, zve32f, zve64x, zve64f, zve64d, zvfh, zvfhmin, v.

- Zc Extensions

  zca, zcb, zce, zcf, zcd, zcmp, zcmt(Assembly only).

  - zce = zca + zcb + zcmp + zcmt
  - f + zce = zca + zcb + zcf + zcmp + zcmt
  - f + d + zce = zca + zcb + zcf + zcd + zcmp + zcmt

- Zvb Extensions

  zvbb, zvbc

- Zvk Extensions

  zvkg, zvkned, zvknha, zvknhb, zvksed, zvksh, zvkn, zvknc, zvkng, zvks, zvksc, zvksg, zvkt.

- BFloat 16 Extensions
  - Standard: Zfbfmin, Zvfbfmin, Zvfbfwma
  - Nuclei customized: *Xxlfbf, Xxlvfbf* (page 302)

- Zilsd Extensions

  Zilsd, Zclsd

Nuclei Custom Extensions

- Packed SIMD Extension 0.5.4

  xxldsp, xxldspn1x, xxldspn2x, xxldspn3x.

  - xxldsp: P-spec-v0.54 + Nuclei Custom EXPD* instructions
  - xxldspn1x: xxldsp + Nuclei Custom N1
  - xxldspn2x: xxldsp + Nuclei Custom N1 & N2
  - xxldspn3x: xxldsp + Nuclei Custom N1 & N2 & N3

- Xxlcz Extensions

  xxlczpstinc, xxlczbmrk, xxlczbitop, xxlczslet, xxlczabs, xxlczmac, xxlczbri, xxlczbitrev, xxlczgp.

- Nuclei custom VPU Extensions

  xxlvqmacc (page 312)

> **Note**
>
> Extensions starting with **x** are generally reserved for manufacturers to customize, and should be placed after extensions starting with **z** when used.

3.1.3 General Options

-march=ISA-string

Generate code for given RISC-V ISA. ISA strings must be lower-case. Examples include `rv64i`, `rv32g`, `rv32e`, and `rv32imaf`. When `-march=` is not specified, use the setting from `-mcpu`. If both `-march` and `-mcpu=` are not specified, the default for this argument is system dependent, users who want a specific architecture extensions should specify one explicitly.

-mabi=ABI-string

Specify integer and floating-point calling convention. ABI-string contains two parts: the size of integer types and the registers used for floating-point types. For example `-march=rv64ifd -mabi=lp64d` means that **long** and **pointers** are 64-bit (implicitly defining **int** to be 32-bit), and that floating-point values up to 64 bits wide are passed in F registers. Contrast this with `-march=rv64ifd -mabi=lp64f`, which still allows the compiler to generate code that uses the F and D extensions but only allows floating-point values up to 32 bits long to be passed in registers; or `-march=rv64ifd -mabi=lp64`, in which no floating-point arguments will be passed in registers.

The default for this argument is system dependent, users who want a specific calling convention should specify one explicitly. The valid calling conventions are: `ilp32`, `ilp32f`, `ilp32d`, `lp64`, `lp64f`, and `lp64d`. Some calling conventions are impossible to implement on some ISAs: for example, `-march=rv32if -mabi=ilp32d` is invalid because the ABI requires 64-bit values be passed in F registers, but F registers are only 32 bits wide. There is also the `ilp32e` ABI that can only be used with the `rv32e` architecture. This ABI is not well specified at present, and is subject to change.

-mcmodel=medlow

Generate code for the medium-low code model. The program and its statically defined symbols must lie within a single 2 GiB address range and must lie between absolute addresses -2 GiB and +2 GiB. Programs can be statically or dynamically linked. This is the default code model.

-mcmodel=medany

Generate code for the medium-any code model. The program and its statically defined symbols must be within any single 2 GiB address range. Programs can be statically or dynamically linked.

The code generated by the medium-any code model is position-independent, but is not guaranteed to function correctly when linked into position-independent executables or libraries.

-mtune=processor-string

Optimize the output for the specified processor by either microarchitecture or a specific CPU name. The allowable values for this option include: `nuclei-100-series`, `nuclei-200-series`, `nuclei-300-series`, `nuclei-600-series`, `nuclei-900-series`, `nuclei-1000-series`, `nuclei-1000-3w-series`, and `nuclei-1000-4w-series`. Note that `nuclei-1000-series` and `nuclei-1000-4w-series` are considered equivalent. All these options are valid for the -mcpu= flag.

When `-mtune=` is not specified, use the setting from `-mcpu`, the default is `rocket` if both are not specified.

The `size` choice is not intended for use by end-users. This is used when -Os is specified. It overrides the instruction cost info provided by `-mtune=`, but does not override the pipeline info. This helps reduce code size while still giving good performance.

-mautovec-dsp/-mno-autovec-dsp

Controls the generation of automatic vectorization of Nuclei DSP instructions, with the compiler enabling Nuclei DSP instructions instruction auto-vectorization by default.

-fstrict-aliasing

It is recommended to add the optimization option `-fno-strict-aliasing` to the project, In some circumstances, this flag allows the compiler to assume that pointers to different types do not alias.

-ftree-loop-vectorize

If you need to disable the RISC-V RVV automatic vectorization, you can use the options `-fno-tree-loop-vectorize` and `-fno-tree-slp-vectorize`. For GCC 13, you can use `--param=riscv-autovec-preference=none`.

-fno-builtin

The `-fno-builtin` option instructs the compiler to avoid replacing standard library function calls with optimized built-in versions. If your program requires implementing its own system functions, such as memcpy, memset, etc., you need to use this option.

Optimization Options

-O0
: Reduce compilation time and make debugging produce the expected results. This is the default.

-O/-O1
: With -O, the compiler tries to reduce code size and execution time, without performing any optimizations that take a great deal of compilation time.

-O2

Optimize even more. GCC performs nearly all supported optimizations that do not involve a space-speed tradeoff. As compared to -O, this option increases both compilation time and the performance of the generated code.

-O3

This option turns on all options in -O2, as well as several other optimizations to improve the performance of the object code.

-Os

This optimization option is often used to tell the compiler to reduce the size of the object code as much as possible while maintaining performance. It will remove some optimization strategies that increase the object code size from all options enabled by -O2.

-Ofast

Disregard strict standards compliance. -Ofast enables all -O3 optimizations. It also enables optimizations that are not valid for all standard-compliant programs.

For more information about RISC-V Options used in GCC, please check https://gcc.gnu.org/onlinedocs/gcc-14.2.0/gcc/RISC-V-Options.html

For RISC-V ELF psABI Document, please check https://github.com/riscv-non-isa/riscv-elf-psabi-doc

3.1.4 Libraries

Note

- `glibc` is used in Linux GNU Glibc toolchain used to compile linux kernel, opensbi, uboot, and linux applications.
- `newlibc` is used in Baremetal or RTOS toolchain, used to compile baremetal or rtos source code, which contains `newlib`, `newlib-nano` and *libncrt* (page 321)

glibc

glibc stands for GNU C Library which is the standard system C library for all GNU systems. It provides the system API for all programs written in C and C-compatible languages such as C++ and Objective C; the runtime facilities of other programming languages use the C library to access the underlying operating system. This library is only supported on Nuclei linux toolchain, not on Nuclei bare-metal toolchain.

newlib

newlib is written as a glibc replacement for embedded systems. It can be used with no OS (“ bare metal ”) or with a lightweight RTOS. Newlib is the default library for embedded GCC distributions.

newlib-nano

Newlib-nano is a derivative of the newlib C library for embedded systems. It is smaller and faster than newlib by code and data size reduction through optimization and removal of non-MCU features.

libncrt (page 321)

`libncrt` is short of **Nuclei C Runtime Library**, which currently support Nuclei RV32 processor, which is released by Nuclei to reduce c library code size, and improve math library speed, for details, please refer to the user guide located in `gcc\share\pdf\Nuclei C Runtime Library Doc.pdf`

3.1.5 Significant Changes Brought by GCC13 Compared to GCC10

This is the changelog for 2023.10 and 2024.06.

- Instead of using single-letter `bkp` to enable these extensions as we did on gcc10, we split them all into corresponding sub-extensions, for example, `_zba_zkr_zve32f`, please check https://doc.nucleisys.com/nuclei_sdk/develop/buildsystem.html#arch-ext to learn about how to adapt Nuclei SDK to support gcc13 upgraded from gcc10.
- Implement new style of architecture extension test macros: each architecture extension has a corresponding feature test macro, which can be used to test its existence and version information. In addition, we add several custom macros, `__riscv_dsp`, `__riscv_bitmanip`.
- Add new option `-misa-spec=*` to control ISA spec version. This controls the default version of each extensions. The official version is `20191213`, but it is set to `2.2` when configuring nuclei toolchain. The difference between them is that in `20191213` version, `Zicsr` and `Zifencei` are separated from the `i` extension into two independent extensions, and using `-misa-spec=2.2` can avoid incompatible errors when the `Zicsr` and `Zifencei` are not passed to `-march=`. See for details at https://github.com/riscv-collab/riscv-gnu-toolchain/issues/1315
- Support for vector intrinsics as specified in version 0.12 of the RISC-V vector intrinsic specification.
- The toolchain component prefix is `riscv-nuclei-elf-` on gcc10, but is `riscv64-unknown-elf-` on gcc13.
- On gcc10, RISCV intrinsic api heads contain `riscv_vector.h`, `riscv_vector_itr.h`, `rv-intrin.h`, `rvp_intrinsic.h`, but now only `riscv_vector.h`, `rvp_intrinsic.h`, `riscv_nuclei_xlcz.h` are provided in gcc13, if you want to find `b` or `k` intrinsic API, please check https://github.com/riscv/riscv-crypto/blob/main/benchmarks/share/rvintrin.h and https://github.com/riscv/riscv-crypto/blob/main/benchmarks/share/riscv-crypto-intrinsics.h , and for RVV intrinsic API, we support 0.12 in gcc13 now, see https://github.com/riscv-non-isa/rvv-intrinsic-doc/releases/tag/v1.0-rc0
- The version of the libncrt was changed from v2.0.0 to v3.0.0, and libncrt is now split into three parts, ' libncrt ' , ' heapops ' and ' fileops ' , click https://doc.nucleisys.com/nuclei_sdk/develop/buildsystem.html#stdclib to learn about how the newlib/libncrt are used in Nuclei SDK with gcc13.

3.1.6 Significant Changes Brought by GCC14 Compared to GCC13

This is the changelog for 2025.02.

- Support for the zilsd and zclsd extensions.
- Some Nuclei custom CSR naming has been re-revised and corrected.
- Implement custom VPU intrinsics for Nuclei RISC-V CPU.
- Nuclei introduces the `bf16` type and supports the `xxlvfbf` and `xxlfbf` extensions.
- Added support for 3-issue(`nuclei-1000-3w-series`) and 4-issue(`nuclei-1000-4w-series`) in the Nuclei 1000 series CPUs.
- GCC14 introduces additional function attribute checks compared to GCC13. For more details, you can refer to https://gcc.gnu.org/gcc-14/porting_to.html.
- Add the option to automatically generate control for xldsp with `-mautovec-dsp/-mno-autovec-dsp` for gcc, which is enabled by default.
- The `riscv_vector.h` must be included when leverage intrinisc type(s) and API(s). And the scope of this attribute should not excced the function body. Meanwhile, to make rvv types and API(s) available for this attribute, include `riscv_vector.h` will not report error for now if v is not present in march.

3.1.7 Install and Setup

Build Toolchain

For more information about how to build a toolchain, see https://github.com/riscv-mcu/riscv-gnu-toolchain/tree/nuclei/2025.02/scripts/toolchain. (Only for Nuclei internal use, no technical support is provided)

Development

The process of user compilation and development can see from https://github.com/riscv-mcu/riscv-gnu-toolchain/blob/nuclei/2025.02/README.md. To get other technical support, please send issues directly to the upstream repository https://github.com/riscv-collab/riscv-gnu-toolchain.

Examples

1. If you choose a core of Nuclei N300FD, then the parameter you pass to ' march ' should be `rv32*fd*`, and ' mabi ' should choose `ilp32d`.
2. If you want to bring the full `B/K/P` extension, then you also need to bring all the subsets of them in the ' march ' . For example, for the `B` extension, the parameter you pass to ' march ' is `_zba_zbb_zbc_zbs`.

3. When using a library, we can tell the linker which library we need to link by using the ' -l ' , for example, `-lc` for newlib-full, `-lc_nano` for newlib-nano. For libncrt, you should pass `--specs=libncrt_xxx.specs` when using gcc. In addition, you need to link extra ' fileops ' and ' heapops ' static libraries during the linking phase by using the ' -l ' , and for the ' fileops ' , you must select one of the three options: ' uart ' , ' semi ' or ' rtt ' , and for the ' heapops ' , you must select one of the three options: ' basic ' , ' realtime ' or ' minimal ' .

3.1.8 Nuclei Custom BFloat16 Extension

Introduction

在众多机器学习与人工智能应用中，矩阵计算往往对精度的要求并不高，但计算量却极为庞大。传统的 32 位浮点数 (FP32) 在存储和计算速度方面存在明显瓶颈，不仅效率低下，还造成了资源的浪费。而 BF16(Brain Floating-Point 16) 作为一种新兴的 16 位数据类型，能够在计算性能与精度之间取得更优的平衡，从而更好地满足这些应用的需求。

然而，目前 RISC-V 上游仅支持了 BF16 的部分转换指令和向量乘加指令，这远远无法满足将 BF16 应用于人工智能、机器学习等场景的完整需求。

为了填补这一空白, Nuclei GCC 工具链与 Nuclei 硬件设计紧密配合，在兼容上游已支持的 BF16 指令的基础上，全面支持了 BF16 相关的运算及转换操作，充分发挥了 BF16 的潜力，使其能够更好地服务于各类高性能计算场景。

寄存器配置

硬件自定义寄存器 mmisc_ctl1 的第 1 位为 fp16mode 来控制 16 位 FPU 寄存器中存储数据的格式:

- **fp16mode = 0**: fp16 mode
- **fp16mode = 1**: bf16 mode

所以在使用 Nuclei bf16 的运算操作时, 需要先将 mmisc_ctl1 的寄存器 fp16mode 位置为 1

Note

Nuclei bf16 的运算操作, 复用的是 fp16 的相关指令, 这里的开关是用来决定复用的指令是做 fp16 运算还是 bf16 运算.

设置代码如下：

```
#include "nuclei_sdk_soc.h" //需要包含该头文件

__RV_CSR_SET(CSR_MFP16MODE, 0x1); //打开 Nuclei bf16

__RV_CSR_CLEAR(CSR_MFP16MODE, 0x1); //关闭 Nuclei bf16
```

BF16 标量的支持

扩展

Nuclei bf16 标量的运算操作, 需要开启 `xxlfbf` 扩展.

示例：*-march=rv64imafdc_xxlfbf*

支持的指令

- `fadd.h/fsub.h/fmul.h/fdiv.h`
- `fmadd.h/fmsub.h/fnmadd.h/fnmsub.h`
- `fsqrt.h`
- `fcvt.bf16.s/fcvt.s.bf16` (兼容 zfbfmin 扩展的指令)
- `fcvt.h.w[u]/fcvt.w[u].h` (int32 <—> bf16)
- `fcvt.h.l[u]/fcvt.l[u].h` (int64 <—> bf16)
- `fmin.h/fmax.h`
- `flt.h/fle.h/feq.h`
- `fclass.h`

Examples

Listing 3.1: 标量 bf16 的加减乘除运算

```
#include <stdio.h>
#include "nuclei_sdk_soc.h"

 __bf16 a = 1.11;
 __bf16 b = 11.8;
 const __bf16 c = 10.3;

 void test_add(){
     __bf16 sum = a + b;
     printf("%f\n",sum);
 }

 void test_sub(){
     __bf16 sum = a-c;
     printf("%f\n",sum);
 }

 void test_mul(){
     __bf16 sum = a * 5;
     printf("%f\n",sum);
```

(continues on next page)

(continued from previous page)

```
}

void test_div(){
    __bf16 sum = a/(__bf16)3.5;
    printf("%f\n",sum);
}

int main(void)
{
    __RV_CSR_SET(CSR_MFP16MODE, 0x1);
    test_add();
    test_sub();
    test_mul();
    test_div();

    return 0;
}
```

BF16 向量的支持 (rvv intrinsic)

扩展

Nuclei bf16 rvv intrinsic 的使用, 需要开启 `V` 扩展以及 `xxlvfbf` 扩展.

示例： *-march=rv64imafdcv_xxlvfbf*

Nuclei bf16 rvv intrinsic nameing scheme

rvv intrinsic 命名规则: rvv-intrinsic-doc[22] `v-intrinsic-spec.pdf`-> **Chapter 6**.

我们的命名规则遵循上述的命名规则, 并在此基础上为了区分我们自定义的 intrinsic, 在前缀处添加了 `_xl` 示例如下：

float16 的 vfadd 运算的 intrinsic 命名：

`vfloat16mf4_t __riscv_vfadd_vv_f16mf4(vfloat16mf4_t vs2, vfloat16mf4_t vs1, size_t vl);`

`vfloat16mf4_t` vfloat16 的数据类型

`__riscv_` intrinsic 名称前缀

`vfadd` intrinsic 所代表的操作的名字（此处为向量的加法运算）

`vv` intrinsic 函数所代表的操作的操作数类型 (v 代表向量类型, 详细信息查看上述 rvv intrinsic 命令规则)

`f16` float16 数据类型的缩写

`mf4` lmul 的值

Nuclei 自定义的 bfloat16 intrinsic 也遵循 rvv intrinsic 命名的基础规则，只是添加了前缀，如下：

`vbfloat16mf4_t __riscv_xl_vfadd_vv_bf16mf4(vbfloat16mf4_t vs2, vbfloat16mf4_t vs1, size_t vl);`

`vbfloat16mf4_t` vbfloat16 的数据类型

`__riscv_xl_` Nuclei 自定义 intrinsic 名称前缀

`vfadd` intrinsic 所代表的操作的名字

[22] https://github.com/riscv-non-isa/rvv-intrinsic-doc/releases/tag/v1.0.0-rc7

`vv` intrinsic 函数所代表的操作的操作数类型

`bf16` bfloat16 数据类型的缩写

`mf4` lmul 的值

> **Note**
>
> bfloat16 的向量数据类型，参考 vector-bfloat16-spec.adoc[23].

Nuclei bf16 支持的 rvv 指令

Nuclei 自定义的指令

- **Vector Single-Width Floating-Point Add/Subtract Instructions**
  - `vfadd.vv/vfadd.vf`
  - `vfsub.vv/vfsub.vf`
  - `vfrsub.vf`
- **Vector Widening Floating-Point Add/Subtract Instructions**
  - `vfwadd.vv/vfwadd.vf`
  - `vfwsub.vv/vfwsub.vf`
  - `vfwadd.wv/vfwadd.vf`
  - `vfwsub.wv/vfwsub.vf`
- **Vector Single-Width Floating-Point Multiply/Divide Instructions**
  - `vfmul.vv/vfmul.vf`
  - `vfdiv.vv/vfdiv.vf`
  - `vfrdiv.vf`
- **Vector Widening Floating-Point Multiply**
  - `vfwmul.vv/vfwmul.vf`
- **Vector Single-Width Floating-Point Fused Multiply-Add Instructions**
  - `vfmacc.vv/vfmacc.vf`
  - `vfnmacc.vv/vfnmacc.vf`
  - `vfmsac.vv/vfmsac.vf`
  - `vfnmsac.vv/vfnmsac.vf`
  - `vfmadd.vv/vfmadd.vf`
  - `vfnmadd.vv/vfnmadd.vf`
  - `vfmsub.vv/vfmsub.vf`
  - `vfnmsub.vv/vfnmsub.vf`
- **Vector Widening Floating-Point Fused Multiply-Add Instructions**
  - `vfwmacc.vv/vfwmacc.vf`
  - `vfwnmacc.vv/vfwnmacc.vf`

[23] https://github.com/riscv-non-isa/rvv-intrinsic-doc/blob/9328aba3fca494717de08502ff32819a7c168daa/doc/vector-bfloat16-spec.adoc

  - vfwmsac.vv/vfwmsac.vf
  - vfwnmsac.vv/vfwnmsac.vf
- **Vector Floating-Point Square-Root Instruction**
  - vfsqrt.v
- **Vector Floating-Point Reciprocal Square-Root Estimate Instruction**
  - vfsqrt7.v
- **Vector Floating-Point Reciprocal Estimate Instruction**
  - vfrec7.c
- **Vector Floating-Point MIN/MAX Instructions**
  - vfmin.vv/vfmin.vf
  - vfmax.vv/vfmax.vf
- **Vector Floating-Point Sign-Injection Instructions**
  - vfsgnj.vv/vfsgnj.vf
  - vfsgnjn.vv/vfsgnjn.vf
  - vfsgnjx.vv/vfsgnjx.vf
- **Vector Floating-Point Compare Instructions**
  - vmfeq.vv/vmfeq.vf
  - vmfne.vv/vmfne.vf
  - vmflt.vv/vmflt.vf
  - vmfle.vv/vmfle.vf
  - vmfgt.vf
  - vmfge.vf
- **Vector Floating-Point Classify Instruction**
  - vfclass.v
- **Vector Floating-Point Merge Instruction**
  - vfmerge.vfm
- **Vector Floating-Point Move Instruction**
  - vfmv.v
- **Single-Width Floating-Point/Integer Type-Convert Instructions**
  - vfcvt.xu.f.v
  - vfcvt.x.f.v
  - vfcvt.rtz.xu.f.v
  - vfcvt.rtz.x.f.v
  - vfcvt.f.xu.v
  - vfcvt.f.x.v
- **Widening Floating-Point/Integer Type-Convert Instructions**
  - vfwcvt.xu.f.v
  - vfwcvt.x.f.v
  - vfwcvt.rtz.xu.f.v

    - `vfwcvt.rtz.x.f.v`
    - `vfwcvt.f.xu.v`
    - `vfwcvt.f.x.v`
    - `vfwcvt.f.f.v`
- **Narrowing Floating-Point/Integer Type-Convert Instructions**
    - `vfncvt.xu.f.w`
    - `vfncvt.x.f.w`
    - `vfncvt.rtz.xu.f.w`
    - `vfncvt.rtz.x.f.w`
    - `vfncvt.f.xu.w`
    - `vfncvt.f.x.w`
    - `vfncvt.f.f.w`
    - `vfncvt.rod.f.f.w`
- **Vector Single-Width Floating-Point Reduction Instructions**
    - `vfredosum.vs`
    - `vfredusum.vs`
    - `vfredmax.vs`
    - `vfredmin.vs`
- **Vector Widening Floating-Point Reduction Instructions**
    - `vfwredosum.vs`
    - `vfwredusum.vs`
- **Vector Slideup Instructions**
    - `vslideup.vx`
- **Vector Slidedown Instructions**
    - `vslidedown.vx`
- **Vector Floating-Point Slide1up Instruction**
    - `vfslide1up.vf`
- **Vector Floating-Point Slide1down Instruction**
    - `vfslide1down.vf`
- **Vector Register Gather Instructions**
    - `vrgather.vv`
    - `vrgatherei16.vv`
- **Vector Compress Instruction**
    - `vcompress.vm`

与上游兼容的指令

- **Vector Unit-Stride Load Intrinsics**
    - `vle16.v`
- **Vector Unit-Stride Store Intrinsics**
    - `vse16.v`
- **Vector Strided Load Intrinsics**
    - `vlse16.v`
- **Vector Strided Store Intrinsics**
    - `vsse16.v`
- **Vector Indexed Load Intrinsics**
    - `vloxei[8/16/32/64].v`
    - `vluxei[8/16/32/64].v`
- **Vector Indexed Store Intrinsics**
    - `vsoxei[8/16/32/64].v`
    - `vsuxei[8/16/32/64].v`
- **Unit-stride Fault-Only-First Loads Intrinsics**
    - `vle16ff.v`
- **Vector Unit-Stride Segment Load Intrinsics**
    - `vlseg[2-8]e16.v`
    - `vlseg[2-8]e16ff.v`
- **Vector Unit-Stride Segment Store Intrinsics**
    - `vsseg[2-8]e16.v`
- **Vector Strided Segment Load Intrinsics**
    - `vlsseg[2-8]e16.v`
- **Vector Strided Segment Store Intrinsics**
    - `vssseg[2-8]e16.v`
- **Vector Indexed Segment Load Intrinsics**
    - `vloxseg[2-8]ei8.v`
    - `vloxseg[2-8]ei16.v`
    - `vloxseg[2-8]ei32.v`
    - `vloxseg[2-8]ei64.v`
    - `vluxseg[2-8]ei8.v`
    - `vluxseg[2-8]ei16.v`
    - `vluxseg[2-8]ei32.v`
    - `vluxseg[2-8]ei64.v`
- **Vector Indexed Segment Store Intrinsics**
    - `vsoxseg[2-8]ei8.v`
    - `vsoxseg[2-8]ei16.v`

  - vsoxseg[2-8]ei32.v
  - vsoxseg[2-8]ei64.v
  - vsuxseg[2-8]ei8.v
  - vsuxseg[2-8]ei16.v
  - vsuxseg[2-8]ei32.v
  - vsuxseg[2-8]ei64.v

Note

Zvfbfmin、Zvfbfwma 扩展支持的指令 riscv-bfloat16 release v1.0[24] `riscv-bfloat16.pdf` 和我们自定义的 Xxlvfbf 扩展不兼容.

其功能已在 vfncvt.f.f.w、vfncvt.f.f.f、vfwmacc.vv/vfwmacc.vf 中实现.

Nuclei bf16 支持的 rvv intrinsic

Nuclei 自定义的 intrinsic

参考 rvv-intrinsic-doc[25] `v-intrinsic-spec.pdf` -> **Appendix A**.

Nuclei 自定义支持的指令所对应的 rvv intrinsic, 与上述文档中 float16 对应的 rvv intrinsic 只有名字的区别, 区别请参考*Nuclei bf16 rvv intrinsic nameing scheme* (page 304)

Note

目前只全面支持 Appendix A, Appendix B、C、D 中的 intrinsic 暂时未全面覆盖测试.

示例：

- Vector Single-Width Floating-Point Add/Subtract Instructions
  - Float16

    ```
    vfloat16mf4_t __riscv_vfadd_vv_f16mf4(vfloat16mf4_t vs2,
    vfloat16mf4_t vs1, size_t vl);
    ```

    ```
    vfloat16mf4_t __riscv_vfadd_vv_f16mf4_m(vbool64_t vm,
    vfloat16mf4_t vs2, vfloat16mf4_t vs1, size_t vl);
    ```
  - Nuclei Bfloat16

    ```
    vbfloat16mf4_t __riscv_xl_vfadd_vv_bf16mf4(vbfloat16mf4_t vs2,
    vbfloat16mf4_t vs1, size_t vl);
    ```

    ```
    vbfloat16mf4_t __riscv_xl_vfadd_vv_bf16mf4_m(vbool64_t vm,
    vbfloat16mf4_t vs2, vbfloat16mf4_t vs1, size_t vl);
    ```
- Vector Widening Floating-Point Add/Subtract Instructions
  - Float16

    ```
    vfloat32mf2_t __riscv_vfwadd_vf_f32mf2(vfloat16mf4_t vs2,
    _Float16 rs1, size_t vl);
    ```
  - Nuclei Bfloat16

[24] https://github.com/riscv/riscv-bfloat16/releases
[25] https://github.com/riscv-non-isa/rvv-intrinsic-doc/releases/tag/v1.0.0-rc7

```
vfloat32mf2_t __riscv_xl_vfwadd_vf_f32mf2(vbfloat16mf4_t vs2,
__bf16 rs1, size_t vl);
```

与上游兼容的 intrinsic

这部分 intrinsic 所对应的命名与 rvv-intrinsic-doc[26] `v-intrinsic-spec.pdf` -> **Appendix A** 文档中 float16 对应的 rvv intrinsic 也只有名字的区别.

区别只需要将 float16 相关数据类型替换为 bfloat16 数据类型

示例:

- Vector Unit-Stride Load Intrinsics
  - Float16

    ```
    vfloat16mf4_t __riscv_vle16_v_f16mf4(const _Float16 *rs1,
    size_t vl);
    ```

  - Bfloat16

    ```
    vbfloat16mf4_t __riscv_vle16_v_bf16mf4(const __bf16 *rs1,
    size_t vl);
    ```

详细的 intrinsic api 可参考 bfloat16 intrinsic_funcs[27].

> **Note**
>
> 1、部分上游支持的指令对应的 intrinsic bfloat16 intrinsic_funcs[28] 中并未全部列出, 但依然可以使用, 具体如下:
>
> - **Vector Indexed Load Intrinsics**
>   - `vloxei[8/32/64].v`
>   - `vluxei[8/32/64].v`
> - **Vector Indexed Store Intrinsics**
>   - `vsoxei[8/32/64].v`
>   - `vsuxei[8/32/64].v`
> - **Vector Indexed Segment Load Intrinsics**
>   - `vloxseg[2-8]ei8.v`
>   - `vloxseg[2-8]ei32.v`
>   - `vloxseg[2-8]ei64.v`
>   - `vluxseg[2-8]ei8.v`
>   - `vluxseg[2-8]ei32.v`
>   - `vluxseg[2-8]ei64.v`
> - **Vector Indexed Segment Store Intrinsics**
>   - `vsoxseg[2-8]ei8.v`
>   - `vsoxseg[2-8]ei32.v`
>   - `vsoxseg[2-8]ei64.v`
>   - `vsuxseg[2-8]ei8.v`

[26] https://github.com/riscv-non-isa/rvv-intrinsic-doc/releases/tag/v1.0.0-rc7

[27] https://github.com/riscv-non-isa/rvv-intrinsic-doc/tree/9328aba3fca494717de08502ff32819a7c168daa/auto-generated/bfloat16/intrinsic_funcs

– vsuxseg[2-8]ei32.v

– vsuxseg[2-8]ei64.v

对应的 intrisic 名字请参考与上游兼容的 *intrinsic* (page 310)

2、上述 bfloat16 intrinsic_funcs[29] 中 03_bfloat16_arithmetic_intrinsics.adoc 文件里的

- Vector BFloat16 Move Intrinsics
- Vector BFloat16 Merge Intrinsics

已被 Nuclei BFloat16 重新定义，使用时请参考*Nuclei 自定义的 intrinsic* (page 309)

Examples

Attention

使用 Nuclei bf16 rvv intrinsic 需要在配置寄存器的基础上添加以下头文件:

`#include<riscv_vector.h>`

Listing 3.2: An implementation of the add function Nuclei Bfloat16 RVV intrinsics.

```
#include <stdio.h>
#include "nuclei_sdk_soc.h"
#include <riscv_vector.h>

void csr_set_bf16_mode(void)
{
    __RV_CSR_SET(CSR_MFP16MODE, 0x1);
} //打开 nuclei bf16


void csr_clr_bf16_mode(void)
{
    __RV_CSR_CLEAR(CSR_MFP16MODE, 0x1);
} //关闭 nuclei bf16

typedef __bf16 bfloat16_t;

void Add_bfloat16_rvv(bfloat16_t *pSrcA, bfloat16_t *pSrcB, bfloat16_t *pDst,␣
↪uint32_t blockSize)
{
    size_t blkCnt = blockSize; /* Loop counter */
    size_t vl;
    vbfloat16m1_t vx, vy;
    for (; (vl = __riscv_vsetvl_e16m1(blkCnt)) > 0; blkCnt -= vl) {
        vx = __riscv_vle16_v_bf16m1(pSrcA, vl);
        vy = __riscv_vle16_v_bf16m1(pSrcB, vl);
        pSrcA += vl;
        pSrcB += vl;
        __riscv_vse16_v_bf16m1(pDst, __riscv_xl_vfadd_vv_bf16m1(vx, vy, vl), vl);
        pDst += vl;
```

(continues on next page)

[28] https://github.com/riscv-non-isa/rvv-intrinsic-doc/tree/9328aba3fca494717de08502ff32819a7c168daa/auto-generated/bfloat16/intrinsic_funcs

[29] https://github.com/riscv-non-isa/rvv-intrinsic-doc/tree/9328aba3fca494717de08502ff32819a7c168daa/auto-generated/bfloat16/intrinsic_funcs

(continued from previous page)

```
    }
}


int main()
{
    csr_set_bf16_mode();
    bfloat16_t SrcA[8] = {5.312, 5.312, 5.312, 5.312, 5.312, 5.312, 5.312, 5.312};
    bfloat16_t SrcB[8] = {5.312, 5.312, 5.312, 5.312, 5.312, 5.312, 5.312, 5.312};
    bfloat16_t Dst[8];

    Add_bfloat16_rvv(SrcA, SrcB, Dst, 8);

    printf("Dst[3] = %f\n", (float)Dst[3]);

    csr_clr_bf16_mode();

    return 0;
}
```

3.1.9 Nuclei Custom VPU Extension

Introduction

为了加快矩阵运算,Nuclei GCC 工具链添加了自定义的 VPU 扩展 `Xxlvqmacc`, 引入了整数矩阵乘加指令, 可以将 8 位整数输入值扩展为 32 位。

扩展

使用该扩展时, 需要打开 `v` 扩展和 `xxlvqmacc` 扩展

示例：*-march=rv64imafdcv_xxlvqmacc*

Attention

使用该扩展的相关 intrinsic 时，需要添加以下头文件:

#include<riscv_vector.h>

支持的指令

- `xl.vqmacc.vv`
- `xl.vqmaccu.vv`
- `xl.vqmaccus.vv`
- `xl.vqmaccsu.vv`

Note

这些指令是有符号或无符号整数矩阵乘加指令，可将 8 位整数输入值扩展至 32 位。

为了更好的理解这些指令的功能, 示例如下:

Dest (C) += Input (A) * Input (B)

int32 += int8 * int8 //xl.vqmacc.vv vd, vs2, vs1

int32 += uint8 * uint8 //xl.vqmaccu.vv vd, vs2, vs1

int32 += uint8 * int8 //xl.vqmaccus.vv vd, vs2, vs1

int32 += int8 * uint8 //xl.vqmaccsu.vv vd, vs2, vs1

其中 (u)int8 矩阵 A 和 B 和 int32 矩阵 C 均为 4x4。

这些指令的设计旨在高效地处理 8 位整数的矩阵运算，同时通过扩展到 32 位来避免中间结果的溢出。

intrinsic 命名规则

rvv intrinsic 命名规则: https://github.com/riscv-non-isa/rvv-intrinsic-doc/releases/tag/v1.0.0-rc7 v-intrinsic-spec.pdf-> **Chapter 6**.

我们的命名规则遵循上述的命名规则，并在此基础上为了区分我们自定义的 intrinsic，对名字进行了一些更改。示例如下:

`vint32m1_t __riscv_xl_vqmacc_4x4x4_i32m1(vint32m1_t vd, vint8m1_t vs1, vint8mf4_t vs2, size_t vl);`

`vint32m1_t` int32_t 的数据类型

`__riscv_xl` Nuclei 自定义 intrinsic 名称前缀

`vqmacc` intrinsic 所代表的操作的名字

`4x4x4` 输入输出矩阵的维度，输入矩阵 A 是 `4x4`，输入矩阵 B 是 `4x4`，输出矩阵 C 是 `4x4`

`i32` int32 数据类型的缩写

`m1` lmul 的值

Nuclei 自定义的 intrinsic

- `xl.vqmacc.vv`

```
vint32m1_t __riscv_xl_vqmacc_4x4x4_i32m1(vint32m1_t vd, vint8m1_t vs1, vint8mf4_t␣
↪vs2, size_t vl);
vint32m2_t __riscv_xl_vqmacc_4x4x4_i32m2(vint32m2_t vd, vint8m1_t vs1, vint8mf2_t␣
↪vs2, size_t vl);
vint32m4_t __riscv_xl_vqmacc_4x4x4_i32m4(vint32m4_t vd, vint8m1_t vs1, vint8m1_t␣
↪vs2, size_t vl);
vint32m8_t __riscv_xl_vqmacc_4x4x4_i32m8(vint32m8_t vd, vint8m1_t vs1, vint8m2_t␣
↪vs2, size_t vl);
vint32m1_t __riscv_xl_vqmacc_4x4x4(vint32m1_t vd, vint8m1_t vs1, vint8mf4_t vs2,␣
↪size_t vl);
vint32m2_t __riscv_xl_vqmacc_4x4x4(vint32m2_t vd, vint8m1_t vs1, vint8mf2_t vs2,␣
↪size_t vl);
vint32m4_t __riscv_xl_vqmacc_4x4x4(vint32m4_t vd, vint8m1_t vs1, vint8m1_t vs2,␣
↪size_t vl);
vint32m8_t __riscv_xl_vqmacc_4x4x4(vint32m8_t vd, vint8m1_t vs1, vint8m2_t vs2,␣
↪size_t vl);
vint32m1_t __riscv_xl_vqmacc_4x4x4_i32m1_tu(vint32m1_t vd, vint8m1_t vs1, vint8mf4_
↪t vs2, size_t vl);
vint32m2_t __riscv_xl_vqmacc_4x4x4_i32m2_tu(vint32m2_t vd, vint8m1_t vs1, vint8mf2_
↪t vs2, size_t vl);
vint32m4_t __riscv_xl_vqmacc_4x4x4_i32m4_tu(vint32m4_t vd, vint8m1_t vs1, vint8m1_
↪t vs2, size_t vl);
vint32m8_t __riscv_xl_vqmacc_4x4x4_i32m8_tu(vint32m8_t vd, vint8m1_t vs1, vint8m2_
```

(continues on next page)

(continued from previous page)

```
↪t vs2, size_t vl);
vint32m1_t __riscv_xl_vqmacc_4x4x4_tu(vint32m1_t vd, vint8m1_t vs1, vint8mf4_t vs2,
↪ size_t vl);
vint32m2_t __riscv_xl_vqmacc_4x4x4_tu(vint32m2_t vd, vint8m1_t vs1, vint8mf2_t vs2,
↪ size_t vl);
vint32m4_t __riscv_xl_vqmacc_4x4x4_tu(vint32m4_t vd, vint8m1_t vs1, vint8m1_t vs2,␣
↪size_t vl);
vint32m8_t __riscv_xl_vqmacc_4x4x4_tu(vint32m8_t vd, vint8m1_t vs1, vint8m2_t vs2,␣
↪size_t vl);
```

- xl.vqmaccu.vv

```
vint32m1_t __riscv_xl_vqmaccu_4x4x4_i32m1(vint32m1_t vd, vuint8m1_t vs1, vuint8mf4_
↪t vs2, size_t vl);
vint32m2_t __riscv_xl_vqmaccu_4x4x4_i32m2(vint32m2_t vd, vuint8m1_t vs1, vuint8mf2_
↪t vs2, size_t vl);
vint32m4_t __riscv_xl_vqmaccu_4x4x4_i32m4(vint32m4_t vd, vuint8m1_t vs1, vuint8m1_
↪t vs2, size_t vl);
vint32m8_t __riscv_xl_vqmaccu_4x4x4_i32m8(vint32m8_t vd, vuint8m1_t vs1, vuint8m2_
↪t vs2, size_t vl);
vint32m1_t __riscv_xl_vqmaccu_4x4x4(vint32m1_t vd, vuint8m1_t vs1, vuint8mf4_t vs2,
↪ size_t vl);
vint32m2_t __riscv_xl_vqmaccu_4x4x4(vint32m2_t vd, vuint8m1_t vs1, vuint8mf2_t vs2,
↪ size_t vl);
vint32m4_t __riscv_xl_vqmaccu_4x4x4(vint32m4_t vd, vuint8m1_t vs1, vuint8m1_t vs2,␣
↪size_t vl);
vint32m8_t __riscv_xl_vqmaccu_4x4x4(vint32m8_t vd, vuint8m1_t vs1, vuint8m2_t vs2,␣
↪size_t vl);
vint32m1_t __riscv_xl_vqmaccu_4x4x4_i32m1_tu(vint32m1_t vd, vuint8m1_t vs1,␣
↪vuint8mf4_t vs2, size_t vl);
vint32m2_t __riscv_xl_vqmaccu_4x4x4_i32m2_tu(vint32m2_t vd, vuint8m1_t vs1,␣
↪vuint8mf2_t vs2, size_t vl);
vint32m4_t __riscv_xl_vqmaccu_4x4x4_i32m4_tu(vint32m4_t vd, vuint8m1_t vs1,␣
↪vuint8m1_t vs2, size_t vl);
vint32m8_t __riscv_xl_vqmaccu_4x4x4_i32m8_tu(vint32m8_t vd, vuint8m1_t vs1,␣
↪vuint8m2_t vs2, size_t vl);
vint32m1_t __riscv_xl_vqmaccu_4x4x4_tu(vint32m1_t vd, vuint8m1_t vs1, vuint8mf4_t␣
↪vs2, size_t vl);
vint32m2_t __riscv_xl_vqmaccu_4x4x4_tu(vint32m2_t vd, vuint8m1_t vs1, vuint8mf2_t␣
↪vs2, size_t vl);
vint32m4_t __riscv_xl_vqmaccu_4x4x4_tu(vint32m4_t vd, vuint8m1_t vs1, vuint8m1_t␣
↪vs2, size_t vl);
vint32m8_t __riscv_xl_vqmaccu_4x4x4_tu(vint32m8_t vd, vuint8m1_t vs1, vuint8m2_t␣
↪vs2, size_t vl);
```

- xl.vqmaccus.vv

```
vint32m1_t __riscv_xl_vqmaccus_4x4x4_i32m1(vint32m1_t vd, vuint8m1_t vs1, vint8mf4_
↪t vs2, size_t vl);
vint32m2_t __riscv_xl_vqmaccus_4x4x4_i32m2(vint32m2_t vd, vuint8m1_t vs1, vint8mf2_
↪t vs2, size_t vl);
vint32m4_t __riscv_xl_vqmaccus_4x4x4_i32m4(vint32m4_t vd, vuint8m1_t vs1, vint8m1_
↪t vs2, size_t vl);
vint32m8_t __riscv_xl_vqmaccus_4x4x4_i32m8(vint32m8_t vd, vuint8m1_t vs1, vint8m2_
↪t vs2, size_t vl);
vint32m1_t __riscv_xl_vqmaccus_4x4x4(vint32m1_t vd, vuint8m1_t vs1, vint8mf4_t vs2,
↪ size_t vl);
vint32m2_t __riscv_xl_vqmaccus_4x4x4(vint32m2_t vd, vuint8m1_t vs1, vint8mf2_t vs2,
↪ size_t vl);
vint32m4_t __riscv_xl_vqmaccus_4x4x4(vint32m4_t vd, vuint8m1_t vs1, vint8m1_t vs2,␣
↪size_t vl);
```

(continues on next page)

(continued from previous page)

```
vint32m8_t __riscv_xl_vqmaccus_4x4x4(vint32m8_t vd, vuint8m1_t vs1, vint8m2_t vs2,␣
↪size_t vl);
vint32m1_t __riscv_xl_vqmaccus_4x4x4_i32m1_tu(vint32m1_t vd, vuint8m1_t vs1,␣
↪vint8mf4_t vs2, size_t vl);
vint32m2_t __riscv_xl_vqmaccus_4x4x4_i32m2_tu(vint32m2_t vd, vuint8m1_t vs1,␣
↪vint8mf2_t vs2, size_t vl);
vint32m4_t __riscv_xl_vqmaccus_4x4x4_i32m4_tu(vint32m4_t vd, vuint8m1_t vs1,␣
↪vint8m1_t vs2, size_t vl);
vint32m8_t __riscv_xl_vqmaccus_4x4x4_i32m8_tu(vint32m8_t vd, vuint8m1_t vs1,␣
↪vint8m2_t vs2, size_t vl);
vint32m1_t __riscv_xl_vqmaccus_4x4x4_tu(vint32m1_t vd, vuint8m1_t vs1, vint8mf4_t␣
↪vs2, size_t vl);
vint32m2_t __riscv_xl_vqmaccus_4x4x4_tu(vint32m2_t vd, vuint8m1_t vs1, vint8mf2_t␣
↪vs2, size_t vl);
vint32m4_t __riscv_xl_vqmaccus_4x4x4_tu(vint32m4_t vd, vuint8m1_t vs1, vint8m1_t␣
↪vs2, size_t vl);
vint32m8_t __riscv_xl_vqmaccus_4x4x4_tu(vint32m8_t vd, vuint8m1_t vs1, vint8m2_t␣
↪vs2, size_t vl);
```

- `xl.vqmaccsu.vv`

```
vint32m1_t __riscv_xl_vqmaccsu_4x4x4_i32m1(vint32m1_t vd, vint8m1_t vs1, vuint8mf4_
↪t vs2, size_t vl);
vint32m2_t __riscv_xl_vqmaccsu_4x4x4_i32m2(vint32m2_t vd, vint8m1_t vs1, vuint8mf2_
↪t vs2, size_t vl);
vint32m4_t __riscv_xl_vqmaccsu_4x4x4_i32m4(vint32m4_t vd, vint8m1_t vs1, vuint8m1_
↪t vs2, size_t vl);
vint32m8_t __riscv_xl_vqmaccsu_4x4x4_i32m8(vint32m8_t vd, vint8m1_t vs1, vuint8m2_
↪t vs2, size_t vl);
vint32m1_t __riscv_xl_vqmaccsu_4x4x4(vint32m1_t vd, vint8m1_t vs1, vuint8mf4_t vs2,
↪ size_t vl);
vint32m2_t __riscv_xl_vqmaccsu_4x4x4(vint32m2_t vd, vint8m1_t vs1, vuint8mf2_t vs2,
↪ size_t vl);
vint32m4_t __riscv_xl_vqmaccsu_4x4x4(vint32m4_t vd, vint8m1_t vs1, vuint8m1_t vs2,␣
↪size_t vl);
vint32m8_t __riscv_xl_vqmaccsu_4x4x4(vint32m8_t vd, vint8m1_t vs1, vuint8m2_t vs2,␣
↪size_t vl);
vint32m1_t __riscv_xl_vqmaccsu_4x4x4_i32m1_tu(vint32m1_t vd, vint8m1_t vs1,␣
↪vuint8mf4_t vs2, size_t vl);
vint32m2_t __riscv_xl_vqmaccsu_4x4x4_i32m2_tu(vint32m2_t vd, vint8m1_t vs1,␣
↪vuint8mf2_t vs2, size_t vl);
vint32m4_t __riscv_xl_vqmaccsu_4x4x4_i32m4_tu(vint32m4_t vd, vint8m1_t vs1,␣
↪vuint8m1_t vs2, size_t vl);
vint32m8_t __riscv_xl_vqmaccsu_4x4x4_i32m8_tu(vint32m8_t vd, vint8m1_t vs1,␣
↪vuint8m2_t vs2, size_t vl);
vint32m1_t __riscv_xl_vqmaccsu_4x4x4_tu(vint32m1_t vd, vint8m1_t vs1, vuint8mf4_t␣
↪vs2, size_t vl);
vint32m2_t __riscv_xl_vqmaccsu_4x4x4_tu(vint32m2_t vd, vint8m1_t vs1, vuint8mf2_t␣
↪vs2, size_t vl);
vint32m4_t __riscv_xl_vqmaccsu_4x4x4_tu(vint32m4_t vd, vint8m1_t vs1, vuint8m1_t␣
↪vs2, size_t vl);
vint32m8_t __riscv_xl_vqmaccsu_4x4x4_tu(vint32m8_t vd, vint8m1_t vs1, vuint8m2_t␣
↪vs2, size_t vl);
```

Examples

Listing 3.3: xl.vqmacc.vv 所对应的 intrinsic 函数使用样例

```
#include <stdio.h>
#include <riscv_vector.h>


#define MATRIX_SIZE (4 * 4)
#define ARRAY_CNT   (4)
#define DATA_CNT    (ARRAY_CNT * MATRIX_SIZE)

/* Case dose matrix multiply-add like below:
*   C[j] += A * B[j], for j in [0, vl/16)
*/

void normal_case(int8_t *addr_in1, int8_t *addr_in2, int32_t *addr_out, int32_t␣
↪data_cnt)
{
    int8_t *pin1 = addr_in1;
    int8_t *pin2 = addr_in2;
    int32_t *pout = addr_out;
    int sum;
    int array_cnt = 0;

    while (data_cnt)
    {
        for (int32_t ii = 0; ii < 4; ii++)
        {
            for (int32_t jj = 0; jj < 4; jj++)
            {
                sum = 0;
                for (int32_t kk = 0; kk < 4; kk++)
                {
                    sum += pin1[ii * 4 + kk] * pin2[kk * 4 + jj];
                }
                pout[ii * 4 + jj] += sum;
            }
        }
        pin2 += 16;
        pout += 16;
        data_cnt -= 16;
    }
}

void vpu_case(int8_t *addr_in1, int8_t *addr_in2, int32_t *addr_out, int32_t data_
↪cnt)
{
    int8_t *pin1 = addr_in1;
    int8_t *pin2 = addr_in2;
    int32_t *pout = addr_out;
    size_t vl;

    vint8m1_t vin1;
    vint8m2_t vin2;
    vint32m8_t vout;
    for (; (vl = __riscv_vsetvl_e8m1(data_cnt)) > 0; data_cnt -= vl) {
        vin1 = __riscv_vle8_v_i8m1(pin1, vl);
        vin2 = __riscv_vle8_v_i8m2(pin2, vl);
        vout = __riscv_vle32_v_i32m8(pout, vl);
        vout = __riscv_xl_vqmacc_4x4x4_i32m8(vout, vin1, vin2, vl);
        __riscv_vse32_v_i32m8(pout, vout, vl);
```

(continues on next page)

(continued from previous page)

```
        pin2 += vl;
        pout += vl;
    }
}

int compare_result(int32_t* normal_out, int32_t* vpu_out, int32_t data_cnt)
{
    int i, ret = 0;

    for (i = 0; i < data_cnt; i++) {
        if (normal_out[i] != vpu_out[i]) {
            printf("num %d not match: %d vs %d\r\n", i, normal_out[i], vpu_out[i]);
            ret = -1;
        }
    }
    return ret;
}


int main(void)
{
    int ret = 0;

    int8_t Input_A[MATRIX_SIZE] = {0};
    int8_t Input_B[DATA_CNT] = {0};
    int32_t Dest_C1[DATA_CNT] = {0};
    int32_t Dest_C2[DATA_CNT] = {0};


    printf("1. Arrays Init\r\n");
    // Input_A
    for (int i = 0; i < MATRIX_SIZE; i++) {
        Input_A[i] = i;
    }

    // Input_B, Dest_C1, Dest_C2
    for (int i = 0; i < ARRAY_CNT; i++) {
        for (int j = 0; j < MATRIX_SIZE; j++) {
            Input_B[i * 16 + j] = j * 2;
        }
    }

    printf("TEST xl.vqmacc\r\n");

    printf("2. Do normal matrix multiply-add\r\n");
    normal_case(Input_A, Input_B, Dest_C1, DATA_CNT);

    printf("3. Do vpu matrix multiply-add\r\n");
    vpu_case(Input_A, Input_B, Dest_C2, DATA_CNT);

    printf("4. Compare results: ");
    if (compare_result(Dest_C1, Dest_C2, DATA_CNT) == 0) {
        printf("PASS\r\n");
    } else {
        printf("FAIL\r\n");
        ret = 1;
    }


    return ret;
}
```

3.2 LLVM Toolchain

3.2.1 About LLVM Toolchain

The current llvm toolchain is developed based on the upstream V19.1.7, for extensions we supported, check below:

You can find our llvm source code in https://github.com/riscv-mcu/llvm-project/tree/nuclei/19.x

3.2.2 Extensions Support

Ratified Extensions

- Basic Extensions

  i, m, a, f, d, c, h(Assembly only), zaamo, zalrsc, zicsr, zifencei, zihintpause(Assembly only), zicond, zawrs(Assembly only), zfh, zfhmin, zmmul, svinval(Assembly only), svnapot(Assembly only), svpbmt.

- Z*inx Extensions

  zfinx, zdinx, zhinx, zhinxmin.

- CMO Extensions(Assembly only)

  zicboz, zicbom, zicbop.

- Bitmanip Extensions

  zba, zbb, zbc, zbs

- Crypto Extensions

  zbkb, zbkc, zbkx, zknd, zkne, zknh, zkr, zks, zksed, zksh, zkt.

- Vector Extensions

  zve32x, zve32f, zve64x, zve64f, zve64d, zvfh, zvfhmin, v.

- Zc Extensions

  zca, zcb, zce, zcf, zcd, zcmp(Assembly only), zcmt(Assembly only).

- Zvb Extensions

  zvbb, zvbc

- Zvk Extensions

  zvkg, zvkned, zvknha, zvknhb, zvksed, zvksh, zvkn, zvknc, zvkng, zvks, zvksc, zvksg, zvkt.

- Zilsd Extensions

  Zilsd, Zclsd

- Xxlcz Extensions(Assembly only)

  xxlczpstinc, xxlczbmrk, xxlczbitop, xxlczslet, xxlczabs, xxlczmac, xxlczbri, xxlczbitrev, xxlczgp.

- Packed SIMD Extension 0.5.4(Assembly only)

  xxldsp, xxldspn1x, xxldspn2x, xxldspn3x.

  - xxldsp: P-spec-v0.54 + Nuclei Custom EXPD* instructions
  - xxldspn1x: xxldsp + Nuclei Custom N1
  - xxldspn2x: xxldsp + Nuclei Custom N1 & N2
  - xxldspn3x: xxldsp + Nuclei Custom N1 & N2 & N3

- Nuclei custom VPU Extensions

  xxlvqmacc

Experimental Extensions

LLVM supports (to various degrees) a number of experimental extensions. All experimental extensions have `experimental-` as a prefix. Listed below:

> smaia, ssaia, zacas, zfa, zfbfmin, zvfbfmin, zvfbfwma, zicond, zihintntl, ztso, zvbb, zvbc, zvkg, zvkn, zvknc, zvkned, zvkng, zvknha, zvknhb, zvks, zvksc, zvksed, zvksg, zvksh, zvkt, zilsd, zclsd.

Note

To use an experimental extension from *clang*, you must add `-menable-experimental-extensions` to the command line, and specify the exact version of the experimental extension you are using. To use an experimental extension with LLVM ' s internal developer tools (e.g. *llc*, *llvm-objdump*, *llvm-mc*), you must prefix the extension name with `experimental-`.

3.2.3 General Options

-march=<cpu>

Specify that Clang should generate code for a specific processor family member and later. For example, if you specify -march=i486, the compiler is allowed to generate instructions that are valid on i486 and later processors, but which may not exist on earlier ones.

-mcmodel=<arg>

-mcmodel=medany (equivalent to -mcmodel=medium), -mcmodel=medlow (equivalent to -mcmodel=small)

-mcpu=help, -mtune=help

Print out a list of supported processors for the given target (specified through `--target=<architecture>` or `-arch <architecture>`). If no target is specified, the system default target will be used.

-fno-vectorize

Use `-fno-vectorize` and `-fno-slp-vectorize` to disable LLVM19 automatic vectorization for RVV; by default, automatic vectorization is enabled.

Optimization Option

-O0
: Means " no optimization " : this level compiles the fastest and generates the most debuggable code.

-O/-O1
: Somewhere between -O0 and -O2.

-O2
: Moderate level of optimization which enables most optimizations.

-O3
: Like -O2, except that it enables optimizations that take longer to perform or that may generate larger code (in an attempt to make the program run faster).

-Ofast
: Enables all the optimizations from -O3 along with other aggressive optimizations that may violate strict compliance with language standards.

-Os
Like -O2 with extra optimizations to reduce code size.

For more about RISC-V options used by LLVM toolchain, please check https://releases.llvm.org/19.1.0/docs/RISCVUsage.html

3.2.4 Install and Setup

More information on building and running LLVM, see https://llvm.org/docs/GettingStarted.html#getting-the-source-code-and-building-llvm

3.2.5 Changelog

Version 2025.02

- llvm was upgraded to the upstream version 19.1.7
- Add Zilsd & Zclsd V1.0 assembly support
- Add Nuclei Xxldsp/Xxldspn1x/Xxldspn2x/Xxldspn3x extensions assembly support
- Add Nuclei Xxlcz extension assembly support
- Add Nuclei Xxlvamacc extension intrinsic support
- Update Nuclei custom csrs
- Update the multilib list

CHAPTER

FOUR

NUCLEI C RUNTIME LIBRARY

4.1 About Nuclei C Runtime Library

`Nuclei C Runtime Library` is written in standard ANSI C and RISC-V assembly language and can run on RISC-V CPU. Here`s a list summarising the main features of `Nuclei C Runtime Library`:

> **libncrt** is short of `Nuclei C Runtime Library`.

- Clean ISO/ANSI C source code.
- **Fast assembly language floating point support**.
- Conforms to standard runtime ABIs for the RISC-V architectures.
- `Nuclei C Runtime Library` supports RV32 and optimizations for the **B extension and P extension**.
- GCC 10.2 *riscv-nuclei-elf-gcc* to GCC 13 **riscv64-unknown-elf-gcc** for GNU toolchain.
- We also bring `Nuclei C Runtime Library` support for **LLVM toolchain**.
- `Nuclei C Runtime Library` now separated into three parts, *libncrt*, ‘*heapops*’ and ‘*fileops*’ library
- Previously, if you want to select a variant of `Nuclei C Runtime Library`, you can pass `--specs=libncrt_xxx.specs` command when using gcc, but now it is not enough, you need to link extra *fileops* and *heapops* static libraries during the linking phase.
- For the ‘*fileops*’ static library, you must select one of the three options: *uart*, *semi* or *rtt*, and for the ‘*heapops*’ static library, you must select one of the three options: *basic*, *realtime* or *minimal*.

4.2 Nuclei C Runtime Library for Nuclei Toolchain

4.2.1 Usage

You can use prebuilt libncrt library for both **Nuclei GNU** and **LLVM** toolchain to replace **libgcc(-lgcc)**, **newlib c(-lc/-lc_nano)** and **math(-lm)** library.

If you want to select a small variant of libncrt library, with basic ‘*heapops*’ and uart ‘*fileops*’, you can use it like below:

> For **Nuclei GNU** Toolchain(*riscv64-unknown-elf-gcc*) case:

```
riscv64-unknown-elf-gcc --specs=libncrt_small.specs -Wl,--start-group -lheapops_
↪basic -lfileops_uart -Wl,--end-group
```

For **Nuclei LLVM** Toolchain(*riscv64-unknown-elf-clang*) case:

You can ' t use *specs=libncrt_small.specs* option, since it is not supported, gcc can also use following command just like clang.

```
riscv64-unknown-elf-clang -nodefaultlibs -isystem=/include/libncrt -Wl,--start-
↪group -lncrt_small -lheapops_basic -lfileops_uart -Wl,--end-group
```

For details about *libncrt*, ' *heapops* ' and ' *fileops* ' variant, please read following documentation.

When using this method, please don ' t link with **-lm/lc_nano/-lgcc/-lc** library to avoid potential linker error and code size increasement.

4.2.2 Varieties

Libncrt Library choice

	GCC Specs	Library
Fast	libncrt_fast.specs	libncrt_fast.a
Balanced	libncrt_balanced.specs	libncrt_balanced.a
Small	libncrt_small.specs	libncrt_small.a
Nano	libncrt_nano.specs	libncrt_nano.a
Pico	libncrt_pico.specs	libncrt_pico.a

If you are not using gcc, and want to use libncrt small variant, you can pass **-isystem=/include/libncrt** during compile phase, and **-nodefaultlibs -lncrt_small** during link phase.

Optimization

	Optimization
Fast	Favor speed at the expense of size
Balanced	Balanced
Small	Favor size at the expense of speed
Nano	Favor size at the expense of speed
Pico	Favor size at the expense of speed

Formatted I/O

	int	long	long long	float	double	wide character	input character class	Width and precision specification	stdout formatting buffer (byte)
Fast	yes	yes	yes	yes	yes	yes	yes	yes	64
Balanced	yes	yes	yes	yes	yes	yes	yes	yes	64
Small	yes	yes	yes	yes	yes	yes	yes	yes	64
Nano	yes	yes	yes	no	no	no	no	yes	64
Pico	yes	no	no	no	no	no	no	no	32

Algorithm

	Scaled-integer Algorithm
Fast	Algorithms use C-language floating-point arithmetic.
Balanced	Algorithms use C-language floating-point arithmetic.
Small	Algorithms use C-language floating-point arithmetic.
Nano	Algorithms use C-language floating-point arithmetic.
Pico	IEEE single-precision functions use scaled integer arithmetic if there is a scaled-integer implementation of the function.

Extra fileops and heapops libraries used together with libncrt

	Library Type	Library description
-lfileops_uart (page 323)	fileops	Using a UART for I/O, need to implement at least metal_tty_putc and metal_tty_getc
-lfileops_semi (page 324)	fileops	Using semihosting for I/O
-lfileops_rtt (page 324)	fileops	Using SEGGER RTT for I/O
-lheapops_basic (page 324)	heapops	Using a basic heap implementation
-lheapops_realtime (page 324)	heapops	Using a real-time heap implementation
-lheapops_minimal (page 325)	heapops	Using a minimal heap implementation

4.3 Quick start

There are four fundamental parts you need to implement for a minimal `Nuclei C Runtime Library` evaluation. And you can find the reference implementation in the latest **Nuclei SDK** release which support **riscv64-unknown-elf-gcc (>=gcc13)**. The documentation for advanced features is in *External function interface* (page 635).

4.3.1 exit()

```
void exit(int fd);
```

4.3.2 Basic UART I/O functions (-lfileops_uart)

To simplify this tutorial, here you only need to implement 2 basic I/O functions:

```
int metal_tty_putc(int c); // UART output function
int metal_tty_getc(void);  // UART input function
```

4.3.3 Using Semihosting (-lfileops_semi)

If you want to use the semihosting function, simply link the ' *fileops_semi* ' static library; there is no need to implement any additional stub functions.

4.3.4 Using RTT (-lfileops_rtt)

To use SEGGER ' s RTT, you need to use J-LINK(**>=v7.92f**).

You can download the J-LINK installation package from the J-LINK page on the SEGGER official website, then unzip it. Next, use the SEGGER J-LINK GDB Server included in it to ensure that your J-LINK is connected.

Then, you need to compile and link the program with the ' *fileops_rtt* ' static library, use the *riscv64-unknown-elf-gdb* to debug the program, and please note that you need to use *monitor reset* and *monitor halt* to reset the J-Link before loading the program.

Finally, open the J-LINK RTT Viewer to observe bi-directional communication with the target program. Please note that if nothing is displayed on the terminal, you may need to disconnect and then reconnect the terminal.

4.3.5 Thread-local storage

In the linker script, setup *tdata*, *tbss*, *__tls_base* and *__tls_end*:

```
.tdata           : ALIGN(8)
{
    PROVIDE( __tls_base = . );
    *(.tdata .tdata.* .gnu.linkonce.td.*)
} >RAM AT>DATA_LMA

.tbss (NOLOAD)   : ALIGN(8)
{
    *(.tbss .tbss.* .gnu.linkonce.tb.*)
    *(.tcommon)
    PROVIDE( __tls_end = . );
} >RAM AT>RAM
```

In *startup.S*, load *__tls_base* to tp register:

```
/* Initialize GP, TP and Stack Pointer SP */
.option push
.option norelax
la gp, __global_pointer$
la tp, __tls_base
.option pop
la sp, _sp
```

4.3.6 Heap

Heap implementation classification

libncrt provides three heap implementations which you may choose from:

- Basic Heap (**-lheapops_basic**)

  Basic heap implement a low-overhead best-fit heap where allocation and deallocation have very little internal fragmentation.

- Real-Time Heap (**-lheapops_realtime**)

Real-Time heap implement a real-time heap where allocation and deallocation have O(1) performance.

- Minimal Heap (**-lheapops_minimal**)

  Minimal heap implement an allocate-only heap where deallocation and reallocation are not implemented.

SEGGER heap API support

Specific ' *heapops* ' static libraries can be chosen based on the API support. Here are the **SEGGER heap API** support conditions for the three ' *heapops* ' static libraries:

	heapops_realtime	heapops_basic	heapops_minimal
__SEGGER_RTL_alloc	yes	yes	yes
__SEGGER_RTL_aligned_alloc	yes	no	no
__SEGGER_RTL_realloc	yes	yes	no
__SEGGER_RTL_free	yes	yes	no

Before calling heap-related APIs such as malloc, you need to implement the initialization of the heap as shown in nuclei-sdk init_libncrt_heap()[30] . You can call the function during startup such as in the *startup.S* file.

```
extern void __SEGGER_RTL_init_heap(void *ptr, size_t size);
extern char __heap_start[];
extern char __heap_end[];

void init_libncrt_heap(void)
{
    size_t heapsz = (size_t)__heap_end - (size_t)__heap_start;
    __SEGGER_RTL_init_heap((void *)__heap_start, heapsz);
}
```

In the linker script, setup *__heap_start* and *__heap_end*. You need to align the heap to a 16-byte boundary and reserve *__HEAP_SIZE* bytes for it.

```
.heap (NOLOAD)    : ALIGN(16)
{
    . = ALIGN(16);
    PROVIDE( __heap_start = . );
    . += __HEAP_SIZE;
    . = ALIGN(16);
    PROVIDE( __heap_limit = . );
} >RAM AT>RAM

.stack ORIGIN(RAM) + LENGTH(RAM) - __TOT_STACK_SIZE (NOLOAD) :
{
    . = ALIGN(16);
    PROVIDE( _heap_end = . );
    PROVIDE( __heap_end = . );
    PROVIDE( __StackLimit = . );
    PROVIDE( __StackBottom = . );
    . += __TOT_STACK_SIZE;
    . = ALIGN(16);
    PROVIDE( __StackTop = . );
    PROVIDE( _sp = . );
} >RAM AT>RAM
```

[30] https://github.com/Nuclei-Software/nuclei-sdk/blob/master/SoC/evalsoc/Common/Source/Stubs/libncrt/stubs.c

4.4 Runtime support

This section describes how to set up the execution environment for the C library.

4.4.1 Getting to main() and then exit()

Before entering *main()* the execution environment must be set up such that the C standard library will function correctly.

Note

This section does not describe the compiler or linker support for placing code and data into memory, how to configure any RAM, or how to zero memory required for zero-initialized data. For this, please refer to your toolset compiler and linker documentation.

Nor does this section document how to call constructors and destructors in the correct order. Again, refer to your toolset manuals.

At-exit function support

After returning from *main()* or by calling *exit()*, any registered At-exit functions must be called to close down. To do this, call *__SEGGER_RTL_execute_at_exit_fns()* from the runtime startup immediately after the call to *main()*.

4.4.2 Multithreaded protection for the heap

Heap functions (allocation, reallocation, deallocation) can be protected from reentrancy in a multithreaded environment by implementing **lock and unlock** functions. By default, these functions do nothing and memory allocation functions are not protected.

See *__SEGGER_RTL_X_heap_lock()* (page 637) and *__SEGGER_RTL_X_heap_unlock()* (page 637).

4.5 Using customized fileops

If you prefer not to use our provided *fileops_uart* (page 323), *fileops_semi* (page 324) or *fileops_rtt* (page 324) static libraries and would like to implement your own customized ' *fileops* ' , do not link the ' *fileops* ' static library. Instead, you should implement all of the following ' *fileops* ' functions.

```
// Open file. Return NULL if file not opened, !=NULL if opened
__SEGGER_RTL_FILE *__SEGGER_RTL_X_file_open(const char *filename, const char
↪*mode) {

}

// Test for file-error condition. Return <0 if stream is closed, ==0 if stream is
↪not in error, >0 if stream is in error
int __SEGGER_RTL_X_file_error(__SEGGER_RTL_FILE *stream) {

}

// Test for end-of-file condition. Return <0 if stream is closed, ==0 if stream is
↪not at end of file, >0 if stream is at end of file
int __SEGGER_RTL_X_file_end(__SEGGER_RTL_FILE *stream) {

}
```

(continues on next page)

(continued from previous page)

```
// Get file status. Return <0 if stream is not a valid file, >=0 if stream is a␣
↪valid file
int __SEGGER_RTL_X_file_stat(__SEGGER_RTL_FILE *stream) {

}

// Get stream buffer size. Return 1 for unbuffered I/O, nonzero number of␣
↪characters to use for
// buffered I/O
int __SEGGER_RTL_X_file_bufsize(__SEGGER_RTL_FILE *stream) {

}

// Flush unwritten data to file. Return <0 if failure, ==0 if success
int __SEGGER_RTL_X_file_flush(__SEGGER_RTL_FILE *stream) {

}

// Get file position. Return <0 if position not retrieved successfully, ==0 if␣
↪retrieved successfully
int __SEGGER_RTL_X_file_getpos(__SEGGER_RTL_FILE *stream, fpos_t *pos) {

}

// Set file position. Return !=0 if position is not set, ==0 if position is set
int __SEGGER_RTL_X_file_seek(__SEGGER_RTL_FILE *stream, long offset, int whence) {

}

// Clear file-error status.
void __SEGGER_RTL_X_file_clrerr(__SEGGER_RTL_FILE *stream)- {

}

// Close file. Return <0 if stream if already closed, >=0 if stream is closed
int __SEGGER_RTL_X_file_close(__SEGGER_RTL_FILE *stream) {

}

// Read data from file. Return <0 if failure, >=0 if success
int __SEGGER_RTL_X_file_read(__SEGGER_RTL_FILE *stream, char *s, unsigned len) {

}

// Write data to file. Return <0 if failure, >=0 if success
int __SEGGER_RTL_X_file_write(__SEGGER_RTL_FILE *stream, const char *s, unsigned␣
↪len) {

}

// Rename file. Return !=0 if rename failure, ==0 if rename success
int __SEGGER_RTL_X_file_rename(const char *old, const char *new) {

}

// Remove file, Return !=0 if remove failed, ==0 if remove success
int __SEGGER_RTL_X_file_remove(const char *filename) {

}
```

(continues on next page)

(continued from previous page)

```
// Generate name for temporary file. Return !=NULL if pointer to temporary name␣
↪generated, ==NULL if cannot generate a unique temporary name
char *__SEGGER_RTL_X_file_tmpnam(char *s, unsigned max) {

}

// Generate temporary file. Return != NULL if pointer to temporary file, ==NULL if␣
↪cannot generate a unique temporary file
__SEGGER_RTL_FILE *__SEGGER_RTL_X_file_tmpfile(void) {

}

// Push character back to stream. Return <0 if failure, >=0 if success
int __SEGGER_RTL_X_file_unget(__SEGGER_RTL_FILE *stream, int c) {

}
```

You can refer to the example of UART using the complete ' *fileops* ' API, which includes the functions mentioned above, in the file **libncrt_fileops_reference.c** located in the same directory as this document.

Of course, you still need to additionally implement these two basic I/O functions is essential for a complete UART ' *fileops* ' implementation:

```
int metal_tty_putc(int c); // UART output function

int metal_tty_getc(void);  // UART input function
```

4.6 C library API

4.6.1 <assert.h>

Assertion functions

Function	Description
assert	Place assertion.

assert

Description

Place assertion.

Definition

```
#define assert(e)    ...
```

Additional information

If NDEBUG is defined as a macro name at the point in the source file where <assert.h> is included, the assert() macro is defined as:

```
#define assert(ignore) ((void)0)    ...
```

If NDEBUG is not defined as a macro name at the point in the source file where <assert.h> is included, the assert() macro expands to a void expression that calls __SEGGER_RTL_X_assert().

When such an assert is executed and e is false, assert() calls the function __SEGGER_RTL_X_assert() with information about the particular call that failed: the text of the argument, the name of the source file, and the source line number. These are the stringized expression and the values of the preprocessing macros __FILE__ and __LINE__.

Notes

The assert() macro is redefined according to the current state of NDEBUG each time that <assert.h> is included.

4.6.2 <complex.h>

`Nuclei C Runtime Library` provides complex math library functions, including all of those required by ISO C99. These functions are implemented to balance performance with correctness. Because producing the correctly rounded result may be prohibitively expensive, these functions are designed to efficiently produce a close approximation to the correctly rounded result. In most cases, the result produced is within +/-1 ulp of the correctly rounded result, though there may be cases where there is greater inaccuracy.

Manipulation functions

Function	Description
cabs() (page 329)	Compute magnitude, double complex.
cabsf() (page 330)	Compute magnitude, float complex.
cabsl() (page 330)	Compute magnitude, long double complex.
carg() (page 330)	Compute phase, double complex.
cargf() (page 331)	Compute phase, float complex.
cargl() (page 331)	Compute phase, long double complex.
cimag() (page 331)	Imaginary part, double complex.
cimagf() (page 332)	Imaginary part, float complex.
cimagl() (page 332)	Imaginary part, long double complex.
creal() (page 332)	Real part, double complex.
crealf() (page 333)	Real part, float complex.
creall() (page 333)	Real part, long double complex.
cproj() (page 333)	Project, double complex.
cprojf() (page 334)	Project, float complex.
cprojl() (page 334)	Project, long double complex.
conj() (page 335)	Conjugate, double complex.
conjf() (page 335)	Conjugate, float complex.
conjl() (page 335)	Conjugate, long double complex.

cabs()

Description

Compute magnitude, double complex.

Prototype

```
double cabs(__SEGGER_RTL_FLOAT64_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute magnitude of.

Return value

The magnitude of x, |x|.

cabsf()

Description

Compute magnitude, float complex.

Prototype

```
float cabsf(__SEGGER_RTL_FLOAT32_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute magnitude of.

Return value

The magnitude of x, |x|.

cabsl()

Description

Compute magnitude, long double complex.

Prototype

```
long double cabsl(__SEGGER_RTL_LDOUBLE_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute magnitude of.

Return value

The magnitude of x, |x|.

carg()

Description

Compute phase, double complex.

Prototype

```
double carg(__SEGGER_RTL_FLOAT64_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute phase of.

Return value

The phase of x.

cargf()

Description

Compute phase, float complex.

Prototype

```
float cargf(__SEGGER_RTL_FLOAT32_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute phase of.

Return value

The phase of x.

cargl()

Description

Compute phase, long double complex.

Prototype

```
long double cargl(__SEGGER_RTL_LDOUBLE_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute phase of.

Return value

The phase of x.

cimag()

Description

Imaginary part, double complex.

Prototype

```
double cimag(__SEGGER_RTL_FLOAT64_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Argument.

Return value

The imaginary part of the complex value.

cimagf()

Description

Imaginary part, float complex.

Prototype

```
float cimagf(__SEGGER_RTL_FLOAT32_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Argument.

Return value

The imaginary part of the complex value.

cimagl()

Description

Imaginary part, long double complex.

Prototype

```
long double cimagl(__SEGGER_RTL_LDOUBLE_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Argument.

Return value

The imaginary part of the complex value.

creal()

Description

Real part, double complex.

Prototype

```
double creal(__SEGGER_RTL_FLOAT64_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Argument.

Return value

The real part of the complex value.

crealf()

Description

Real part, float complex.

Prototype

```
float crealf(__SEGGER_RTL_FLOAT32_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Argument.

Return value

The real part of the complex value.

creall()

Description

Real part, long double complex.

Prototype

```
long double creall(__SEGGER_RTL_LDOUBLE_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Argument.

Return value

The real part of the complex value.

cproj()

Description

Project, double complex.

Prototype

```
__SEGGER_RTL_FLOAT64_C_COMPLEX cproj(__SEGGER_RTL_FLOAT64_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to project.

Return value

The projection of x to the Reimann sphere.

Additional information

x projects to x, except that all complex infinities (even those with one infinite part and one NaN part) project to positive infinity on the real axis. If x has an infinite part, then cproj(x) is be equivalent to:

- INFINITY + I * copysign(0.0, cimag(x))

cprojf()

Description

Project, float complex.

Prototype

```
__SEGGER_RTL_FLOAT32_C_COMPLEX cprojf(__SEGGER_RTL_FLOAT32_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to project.

Return value

The projection of x to the Reimann sphere.

Additional information

x projects to x, except that all complex infinities (even those with one infinite part and one NaN part) project to positive infinity on the real axis. If x has an infinite part, then cproj(x) is be equivalent to:

- INFINITY + I * copysign(0.0, cimag(x))

cprojl()

Description

Project, long double complex.

Prototype

```
__SEGGER_RTL_LDOUBLE_C_COMPLEX cprojl(__SEGGER_RTL_LDOUBLE_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to project.

Return value

The projection of x to the Reimann sphere.

Additional information

x projects to x, except that all complex infinities (even those with one infinite part and one NaN part) project to positive infinity on the real axis. If x has an infinite part, then cproj(x) is be equivalent to:

- INFINITY + I * copysignl(0.0, cimagl(x))

conj()

Description

Conjugate, double complex.

Prototype

```
__SEGGER_RTL_FLOAT64_C_COMPLEX conj(__SEGGER_RTL_FLOAT64_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to conjugate.

Return value

The complex conjugate of x.

conjf()

Description

Conjugate, float complex.

Prototype

```
__SEGGER_RTL_FLOAT32_C_COMPLEX conjf(__SEGGER_RTL_FLOAT32_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to conjugate.

Return value

The complex conjugate of x.

conjl()

Description

Conjugate, long double complex.

Prototype

```
__SEGGER_RTL_LDOUBLE_C_COMPLEX conjl(__SEGGER_RTL_LDOUBLE_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to conjugate.

Return value

The complex conjugate of x.

Trigonometric functions

Function	Description
csin() (page 336)	Compute sine, double complex.
csinf() (page 336)	Compute sine, float complex.
csinl() (page 337)	Compute sine, long double complex.
ccos() (page 337)	Compute cosine, double complex.
ccosf() (page 337)	Compute cosine, float complex.
ccosl() (page 338)	Compute cosine, long double complex.
ctan() (page 338)	Compute tangent, double complex.
ctanf() (page 338)	Compute tangent, float complex.
ctanl() (page 339)	Compute tangent, long double complex.
casin() (page 339)	Compute inverse sine, double complex.
casinf() (page 339)	Compute inverse sine, float complex.
casinl() (page 340)	Compute inverse sine, long double complex.
cacos() (page 340)	Compute inverse cosine, double complex.
cacosf() (page 341)	Compute inverse cosine, float complex.
cacosl() (page 341)	Compute inverse cosine, long double complex.
catan() (page 341)	Compute inverse tangent, double complex.
catanf() (page 342)	Compute inverse tangent, float complex.
catanl() (page 342)	Compute inverse tangent, long double complex.

csin()

Description

Compute sine, double complex.

Prototype

```
__SEGGER_RTL_FLOAT64_C_COMPLEX csin(__SEGGER_RTL_FLOAT64_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute sine of.

Return value

The sine of x.

csinf()

Description

Compute sine, float complex.

Prototype

```
__SEGGER_RTL_FLOAT32_C_COMPLEX csinf(__SEGGER_RTL_FLOAT32_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute sine of.

Return value

The sine of x.

csinl()

Description

Compute sine, long double complex.

Prototype

```
__SEGGER_RTL_LDOUBLE_C_COMPLEX csinl(__SEGGER_RTL_LDOUBLE_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute sine of.

Return value

The sine of x.

ccos()

Description

Compute cosine, double complex.

Prototype

```
__SEGGER_RTL_FLOAT64_C_COMPLEX ccos(__SEGGER_RTL_FLOAT64_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute cosine of.

Return value

The cosine of x.

ccosf()

Description

Compute cosine, float complex.

Prototype

```
__SEGGER_RTL_FLOAT32_C_COMPLEX ccosf(__SEGGER_RTL_FLOAT32_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute cosine of.

Return value

The cosine of x.

ccosl()

Description

Compute cosine, long double complex.

Prototype

```
__SEGGER_RTL_LDOUBLE_C_COMPLEX ccosl(__SEGGER_RTL_LDOUBLE_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute cosine of.

Return value

The cosine of x.

ctan()

Description

Compute tangent, double complex.

Prototype

```
__SEGGER_RTL_FLOAT64_C_COMPLEX ctan(__SEGGER_RTL_FLOAT64_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute tangent of.

Return value

The tangent of x.

ctanf()

Description

Compute tangent, float complex.

Prototype

```
__SEGGER_RTL_FLOAT32_C_COMPLEX ctanf(__SEGGER_RTL_FLOAT32_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute tangent of.

Return value

The tangent of x.

ctanl()

Description

Compute tangent, long double complex.

Prototype

```
__SEGGER_RTL_LDOUBLE_C_COMPLEX ctanl(__SEGGER_RTL_LDOUBLE_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute tangent of.

Return value

The tangent of x.

casin()

Description

Compute inverse sine, double complex.

Prototype

```
__SEGGER_RTL_FLOAT64_C_COMPLEX casin(__SEGGER_RTL_FLOAT64_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Argument.

Return value

Inverse sine of x.

Notes

casin(z) = -i casinh(i.z)

casinf()

Description

Compute inverse sine, float complex.

Prototype

```
__SEGGER_RTL_FLOAT32_C_COMPLEX casinf(__SEGGER_RTL_FLOAT32_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Argument.

Return value

Inverse sine of x.

Notes

casin(z) = -i casinh(i.z)

casinl()

Description

Compute inverse sine, long double complex.

Prototype

```
__SEGGER_RTL_LDOUBLE_C_COMPLEX casinl(__SEGGER_RTL_LDOUBLE_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Argument.

Return value

Inverse sine of x.

Notes

casinl(z) = -i casinhl(i.z)

cacos()

Description

Compute inverse cosine, double complex.

Prototype

```
__SEGGER_RTL_FLOAT64_C_COMPLEX cacos(__SEGGER_RTL_FLOAT64_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute inverse cosine of.

Return value

The inverse cosine of x.

cacosf()

Description

Compute inverse cosine, float complex.

Prototype

```
__SEGGER_RTL_FLOAT32_C_COMPLEX cacosf(__SEGGER_RTL_FLOAT32_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute inverse cosine of.

Return value

The inverse cosine of x.

cacosl()

Description

Compute inverse cosine, long double complex.

Prototype

```
__SEGGER_RTL_LDOUBLE_C_COMPLEX cacosl(__SEGGER_RTL_LDOUBLE_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute inverse cosine of.

Return value

The inverse cosine of x.

catan()

Description

Compute inverse tangent, double complex.

Prototype

```
__SEGGER_RTL_FLOAT64_C_COMPLEX catan(__SEGGER_RTL_FLOAT64_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Argument.

Return value

Inverse tangent of x.

Notes

catan(z) = -i catanh(i.z)

catanf()

Description

Compute inverse tangent, float complex.

Prototype

```
__SEGGER_RTL_FLOAT32_C_COMPLEX catanf(__SEGGER_RTL_FLOAT32_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Argument.

Return value

Inverse tangent of x.

Notes

catan(z) = -i catanh(i.z)

catanl()

Description

Compute inverse tangent, long double complex.

Prototype

```
__SEGGER_RTL_LDOUBLE_C_COMPLEX catanl(__SEGGER_RTL_LDOUBLE_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Argument.

Return value

Inverse tangent of x.

Notes

catanl(z) = -i catanhl(i.z)

Hyperbolic functions

csinh()

Description

Compute hyperbolic sine, double complex.

Prototype

```
__SEGGER_RTL_FLOAT64_C_COMPLEX csinh(__SEGGER_RTL_FLOAT64_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute hyperbolic sine of.

Return value

The hyperbolic sine of x according to the following table:

Argument	csinh(Argument)
+0 + 0i	+0 + 0i
+0 + ∞i	±0 + NaNi, sign of real part unspecified
+0 + NaNi	±0 + NaNi, sign of real part unspecified
a + ∞i	NaN + NaNi, for positive finite a
a + NaNi	NaN + NaNi, for finite nonzero a
+∞ + 0i	+∞ + 0i
+∞ + bi	+∞×cos(b) + +∞×sin(b).i for positive finite b
+∞ + ∞i	±∞ + NaNi, sign of real part unspecified
+∞ + NaNi	±∞ + NaNi, sign of real part unspecified
NaN + 0i	NaN + 0i
NaN + bi	NaN + NaNi, for all nonzero b
NaN + NaNi	NaN + NaNi

For arguments with a negative imaginary component, use the equality:

- csinh(conj(z)) = conj(csinh(z)).

For arguments with a negative real component, use the equality:

- csinh(-z) = -csinh(z).

csinhf()

Description

Compute hyperbolic sine, float complex.

Prototype

```
__SEGGER_RTL_FLOAT32_C_COMPLEX csinhf(__SEGGER_RTL_FLOAT32_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute hyperbolic sine of.

Return value

The hyperbolic sine of x according to the following table:

Argument	csinh(Argument)
+0 + 0i	+0 + 0i
+0 + ∞i	±0 + NaNi, sign of real part unspecified
+0 + NaNi	±0 + NaNi, sign of real part unspecified
a + ∞i	NaN + NaNi, for positive finite a
a + NaNi	NaN + NaNi, for finite nonzero a
+∞ + 0i	+∞ + 0i
+∞ + bi	+∞×cos(b) + +∞×sin(b).i for positive finite b
+∞ + ∞i	±∞ + NaNi, sign of real part unspecified
+∞ + NaNi	±∞ + NaNi, sign of real part unspecified
NaN + 0i	NaN + 0i
NaN + bi	NaN + NaNi, for all nonzero b
NaN + NaNi	NaN + NaNi

For arguments with a negative imaginary component, use the equality:

- csinh(conj(z)) = conj(csinh(z)).

For arguments with a negative real component, use the equality:

- csinh(-z) = -csinh(z).

csinhl()

Description

Compute hyperbolic sine, long double complex.

Prototype

```
__SEGGER_RTL_LDOUBLE_C_COMPLEX csinhl(__SEGGER_RTL_LDOUBLE_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute hyperbolic sine of.

Return value

The hyperbolic sine of x according to the following table:

Argument	csinh(Argument)
+0 + 0i	+0 + 0i
+0 + ∞i	±0 + NaNi, sign of real part unspecified
+0 + NaNi	±0 + NaNi, sign of real part unspecified
a + ∞i	NaN + NaNi, for positive finite a
a + NaNi	NaN + NaNi, for finite nonzero a
+∞ + 0i	+∞ + 0i
+∞ + bi	+∞×cos(b) + +∞×sin(b).i for positive finite b
+∞ + ∞i	±∞ + NaNi, sign of real part unspecified
+∞ + NaNi	±∞ + NaNi, sign of real part unspecified
NaN + 0i	NaN + 0i
NaN + bi	NaN + NaNi, for all nonzero b
NaN + NaNi	NaN + NaNi

For arguments with a negative imaginary component, use the equality:

- csinh(conj(z)) = conj(csinh(z)).

For arguments with a negative real component, use the equality:

- csinh(-z) = -csinh(z).

ccosh()

Description

Compute hyperbolic cosine, double complex.

Prototype

```
__SEGGER_RTL_FLOAT64_C_COMPLEX ccosh(__SEGGER_RTL_FLOAT64_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute hyperbolic cosine of.

Return value

The hyperbolic cosine of x according to the following table:

Argument	ccosh(Argument)
+0 + 0i	+1 + 0i
+0 + ∞i	NaN + ±0i, sign of imaginary part unspecified
+0 + NaNi	NaN + ±0i, sign of imaginary part unspecified
a + ∞i	NaN + NaNi, for finite nonzero a
a + NaNi	NaN + NaNi, for finite nonzero a
+∞ + 0i	+∞ + 0i
+∞ + bi	+∞×cos(b) + Inf×sin(b).i for finite nonzero b
+∞ + ∞i	+∞ + NaNi
+∞ + NaNi	+∞ + NaNi
NaN + 0i	NaN + ±0i, sign of imaginary part unspecified
NaN + bi	NaN + NaNi, for all nonzero b
NaN + NaNi	NaN + NaNi

For arguments with a negative imaginary component, use the equality:

- ccosh(conj(z)) = conj(ccosh(z)).

ccoshf()

Description

Compute hyperbolic cosine, float complex.

Prototype

```
__SEGGER_RTL_FLOAT32_C_COMPLEX ccoshf(__SEGGER_RTL_FLOAT32_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute hyperbolic cosine of.

Return value

The hyperbolic cosine of x according to the following table:

Argument	ccosh(Argument)
+0 + 0i	+1 + 0i
+0 + ∞i	NaN + ±0i, sign of imaginary part unspecified
+0 + NaNi	NaN + ±0i, sign of imaginary part unspecified
a + ∞i	NaN + NaNi, for finite nonzero a
a + NaNi	NaN + NaNi, for finite nonzero a
+∞ + 0i	+∞ + 0i
+∞ + bi	+∞×cos(b) + Inf×sin(b).i for finite nonzero b
+∞ + ∞i	+∞ + NaNi
+∞ + NaNi	+∞ + NaNi
NaN + 0i	NaN + ±0i, sign of imaginary part unspecified
NaN + bi	NaN + NaNi, for all nonzero b
NaN + NaNi	NaN + NaNi

For arguments with a negative imaginary component, use the equality:

- ccosh(conj(z)) = conj(ccosh(z)).

ccoshl()

Description

Compute hyperbolic cosine, long double complex.

Prototype

```
__SEGGER_RTL_LDOUBLE_C_COMPLEX ccoshl(__SEGGER_RTL_LDOUBLE_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute hyperbolic cosine of.

Return value

The hyperbolic cosine of x according to the following table:

Argument	ccosh(Argument)
+0 + 0i	+1 + 0i
+0 + ∞i	NaN + ±0i, sign of imaginary part unspecified
+0 + NaNi	NaN + ±0i, sign of imaginary part unspecified
a + ∞i	NaN + NaNi, for finite nonzero a
a + NaNi	NaN + NaNi, for finite nonzero a
+∞ + 0i	+∞ + 0i
+∞ + bi	+∞×cos(b) + Inf×sin(b).i for finite nonzero b
+∞ + ∞i	+∞ + NaNi
+∞ + NaNi	+∞ + NaNi
NaN + 0i	NaN + ±0i, sign of imaginary part unspecified
NaN + bi	NaN + NaNi, for all nonzero b
NaN + NaNi	NaN + NaNi

For arguments with a negative imaginary component, use the equality:

- ccosh(conj(z)) = conj(ccosh(z)).

ctanh()

Description

Compute hyperbolic tangent, double complex.

Prototype

```
__SEGGER_RTL_FLOAT64_C_COMPLEX ctanh(__SEGGER_RTL_FLOAT64_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute hyperbolic tangent of.

Return value

The hyperbolic tangent of x according to the following table:

Argument	ctanh(Argument)
+0 + 0i	+0 + 0i
a + ∞i	NaN + NaNi, for finite a
a + NaNi	NaN + NaNi, for finite a
+∞ + bi	+1 + sin(2b)×0i for positive-signed finite b
+∞ + ∞i	+1 + ±0i, sign of imaginary part unspecified
+∞ + NaNi	+1 + ±0i, sign of imaginary part unspecified
NaN + 0i	NaN + 0i
NaN + bi	NaN + NaNi, for all nonzero b
NaN + NaNi	NaN + NaNi

For arguments with a negative imaginary component, use the equality:

- ctanh(conj(z)) = conj(ctanh(z)).

For arguments with a negative real component, use the equality:

- ctanh(-z) = -ctanh(z).

ctanhf()

Description

Compute hyperbolic tangent, float complex.

Prototype

```
__SEGGER_RTL_FLOAT32_C_COMPLEX ctanhf(__SEGGER_RTL_FLOAT32_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute hyperbolic tangent of.

Return value

The hyperbolic tangent of x according to the following table:

Argument	ctanh(Argument)
+0 + 0i	+0 + 0i
a + ∞i	NaN + NaNi, for finite a
a + NaNi	NaN + NaNi, for finite a
+∞ + bi	+1 + sin(2b)×0i for positive-signed finite b
+∞ + ∞i	+1 + ±0i, sign of imaginary part unspecified
+∞ + NaNi	+1 + ±0i, sign of imaginary part unspecified
NaN + 0i	NaN + 0i
NaN + bi	NaN + NaNi, for all nonzero b
NaN + NaNi	NaN + NaNi

For arguments with a negative imaginary component, use the equality:

- ctanhf(conj(z)) = conj(ctanhf(z)).

For arguments with a negative real component, use the equality:

- ctanhf(-z) = -ctanhf(z).

ctanhl()

Description

Compute hyperbolic tangent, long double complex.

Prototype

```
__SEGGER_RTL_LDOUBLE_C_COMPLEX ctanhl(__SEGGER_RTL_LDOUBLE_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute hyperbolic tangent of.

Return value

The hyperbolic tangent of x according to the following table:

Argument	ctanh(Argument)
+0 + 0i	+0 + 0i
a + ∞i	NaN + NaNi, for finite a
a + NaNi	NaN + NaNi, for finite a
+∞ + bi	+1 + sin(2b)×0i for positive-signed finite b
+∞ + ∞i	+1 + ±0i, sign of imaginary part unspecified
+∞ + NaNi	+1 + ±0i, sign of imaginary part unspecified
NaN + 0i	NaN + 0i
NaN + bi	NaN + NaNi, for all nonzero b
NaN + NaNi	NaN + NaNi

For arguments with a negative imaginary component, use the equality:

- ctanh(conj(z)) = conj(ctanh(z)).

For arguments with a negative real component, use the equality:

- ctanh(-z) = -ctanh(z).

casinh()

Description

Compute inverse hyperbolic sine, double complex.

Prototype

```
__SEGGER_RTL_FLOAT64_C_COMPLEX casinh(__SEGGER_RTL_FLOAT64_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute inverse hyperbolic sineof.

Return value

The inverse hyperbolic sine of x according to the following table:

Argument	casinh(Argument)
+0 + 0i	+0 + 0i
+0 + ∞i	+∞ + ½πi
a + NaNi	NaN + NaNi
+∞ + bi	+∞ + 0i, for positive-signed b
+∞ + ∞i	+Pi + 0i
+∞ + NaNi	+∞ + NaNi
NaN + 0i	NaN + 0i
NaN + bi	NaN + NaNi, for finite nonzero b
NaN + ∞i	±∞ + NaNi, sign of real part unspecified
NaN + NaNi	NaN + NaNi

For arguments with a negative imaginary component, use the equality:

- casinh(conj(z)) = conj(casinh(z)).

For arguments with a negative real component, use the equality:

- casinh(-z) = -casinh(z).

casinhf()

Description

Compute inverse hyperbolic sine, float complex.

Prototype

```
__SEGGER_RTL_FLOAT32_C_COMPLEX casinhf(__SEGGER_RTL_FLOAT32_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute inverse hyperbolic sineof.

Return value

The inverse hyperbolic sine of x according to the following table:

Argument	casinh(Argument)
+0 + 0i	+0 + 0i
+0 + ∞i	+∞ + ½πi
a + NaNi	NaN + NaNi
+∞ + bi	+∞ + 0i, for positive-signed b
+∞ + ∞i	+Pi + 0i
+∞ + NaNi	+∞ + NaNi
NaN + 0i	NaN + 0i
NaN + bi	NaN + NaNi, for finite nonzero b
NaN + ∞i	±∞ + NaNi, sign of real part unspecified
NaN + NaNi	NaN + NaNi

For arguments with a negative imaginary component, use the equality:

- casinh(conj(z)) = conj(casinh(z)).

For arguments with a negative real component, use the equality:

- casinh(-z) = -casinh(z).

casinhl()

Description

Compute inverse hyperbolic sine, long double complex.

Prototype

```
__SEGGER_RTL_LDOUBLE_C_COMPLEX casinhl(__SEGGER_RTL_LDOUBLE_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute inverse hyperbolic sineof.

Return value

The inverse hyperbolic sine of x according to the following table:

Argument	casinh(Argument)
+0 + 0i	+0 + 0i
+0 + ∞i	+∞ + ½πi
a + NaNi	NaN + NaNi
+∞ + bi	+∞ + 0i, for positive-signed b
+∞ + ∞i	+Pi + 0i
+∞ + NaNi	+∞ + NaNi
NaN + 0i	NaN + 0i
NaN + bi	NaN + NaNi, for finite nonzero b
NaN + ∞i	±∞ + NaNi, sign of real part unspecified
NaN + NaNi	NaN + NaNi

For arguments with a negative imaginary component, use the equality:

- casinh(conj(z)) = conj(casinh(z)).

For arguments with a negative real component, use the equality:

- casinh(-z) = -casinh(z).

cacosh()

Description

Compute inverse hyperbolic cosine, double complex.

Prototype

```
__SEGGER_RTL_FLOAT64_C_COMPLEX cacosh(__SEGGER_RTL_FLOAT64_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute inverse hyperbolic cosine of.

Return value

The inverse hyperbolic cosine of x according to the following table:

Argument	cacosh(Argument)
±0 + 0i	+0 + 0i
a + ∞i	+∞ + ½πi, for finite a
a + NaNi	NaN + NaNi, for finite a
-∞ + bi	+∞ + πi, for positive-signed finite b
+∞ + bi	+∞ + 0i, for positive-signed finite b
-∞ + ∞i	±∞ + ¾πi
+∞ + ∞i	±∞ + ¼πi
±∞ + NaNi	+∞ + NaNi
NaN + bi	NaN + NaNi, for finite b
NaN + ∞i	+∞ + NaNi
NaN + NaNi	NaN + NaNi

For arguments with a negative imaginary component, use the equality:

- cacosh(conj(z)) = conj(cacosh(z)).

cacoshf()

Description

Compute inverse hyperbolic cosine, float complex.

Prototype

```
__SEGGER_RTL_FLOAT32_C_COMPLEX cacoshf(__SEGGER_RTL_FLOAT32_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute inverse hyperbolic cosine of.

Return value

The inverse hyperbolic cosine of x according to the following table:

Argument	cacosh(Argument)
±0 + 0i	+0 + 0i
a + ∞i	+∞ + ½πi, for finite a
a + NaNi	NaN + NaNi, for finite a
-∞ + bi	+∞ + πi, for positive-signed finite b
+∞ + bi	+∞ + 0i, for positive-signed finite b
-∞ + ∞i	±∞ + ¾πi
+∞ + ∞i	±∞ + ¼πi
±∞ + NaNi	+∞ + NaNi
NaN + bi	NaN + NaNi, for finite b
NaN + ∞i	+∞ + NaNi
NaN + NaNi	NaN + NaNi

For arguments with a negative imaginary component, use the equality:

- cacosh(conj(z)) = conj(cacosh(z)).

cacoshl()

Description

Compute inverse hyperbolic cosine, long double complex.

Prototype

```
__SEGGER_RTL_LDOUBLE_C_COMPLEX cacoshl(__SEGGER_RTL_LDOUBLE_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute inverse hyperbolic cosine of.

Return value

The inverse hyperbolic cosine of x according to the following table:

Argument	cacosh(Argument)
±0 + 0i	+0 + 0i
a + ∞i	+∞ + ½πi, for finite a
a + NaNi	NaN + NaNi, for finite a
-∞ + bi	+∞ + πi, for positive-signed finite b
+∞ + bi	+∞ + 0i, for positive-signed finite b
-∞ + ∞i	±∞ + ¾πi
+∞ + ∞i	±∞ + ¼πi
±∞ + NaNi	+∞ + NaNi
NaN + bi	NaN + NaNi, for finite b
NaN + ∞i	+∞ + NaNi
NaN + NaNi	NaN + NaNi

For arguments with a negative imaginary component, use the equality:

- cacosh(conj(z)) = conj(cacosh(z)).

catanh()

Description

Compute inverse hyperbolic tangent, double complex.

Prototype

```
__SEGGER_RTL_FLOAT64_C_COMPLEX catanh(__SEGGER_RTL_FLOAT64_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute inverse hyperbolic tangent of.

Return value

The inverse hyperbolic tangent of x according to the following table:

Argument	catanh(Argument)
+0 + 0i	+0 + 0i
+0 + NaNi	+0 + NaNi
+1 + 0i	+∞ + 0i
a + ∞i	+0 + ½πi for positive-signed a
a + NaNi	NaN + NaNi, for nonzero finite a
+∞ + bi	+0 + ½πi for positive-signed b
+∞ + ∞i	+0 + ½πi
+∞ + NaNi	+0 + NaNi
NaN + bi	NaN + NaNi, for finite b
NaN + NaNi	NaN + NaNi

For arguments with a negative imaginary component, use the equality:

- catanh(conj(z)) = conj(catanh(z)).

For arguments with a negative real component, use the equality:

- catanh(-z) = -catanh(z).

catanhf()

Description

Compute inverse hyperbolic tangent, float complex.

Prototype

```
__SEGGER_RTL_FLOAT32_C_COMPLEX catanhf(__SEGGER_RTL_FLOAT32_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute inverse hyperbolic tangent of.

Return value

The inverse hyperbolic tangent of x according to the following table:

Argument	catanh(Argument)
+0 + 0i	+0 + 0i
+0 + NaNi	+0 + NaNi
+1 + 0i	+∞ + 0i
a + ∞i	+0 + ½πi for positive-signed a
a + NaNi	NaN + NaNi, for nonzero finite a
+∞ + bi	+0 + ½πi for positive-signed b
+∞ + ∞i	+0 + ½πi
+∞ + NaNi	+0 + NaNi
NaN + bi	NaN + NaNi, for finite b
NaN + NaNi	NaN + NaNi

For arguments with a negative imaginary component, use the equality:

- catanh(conj(z)) = conj(catanh(z)).

For arguments with a negative real component, use the equality:

- catanh(-z) = -catanh(z).

catanhl()

Description

Compute inverse hyperbolic tangent, long double complex.

Prototype

```
__SEGGER_RTL_LDOUBLE_C_COMPLEX catanhl(__SEGGER_RTL_LDOUBLE_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute inverse hyperbolic tangent of.

Return value

The inverse hyperbolic tangent of x according to the following table:

Argument	catanh(Argument)
+0 + 0i	+0 + 0i
+0 + NaNi	+0 + NaNi
+1 + 0i	+∞ + 0i
a + ∞i	+0 + ½πi for positive-signed a
a + NaNi	NaN + NaNi, for nonzero finite a
+∞ + bi	+0 + ½πi for positive-signed b
+∞ + ∞i	+0 + ½πi
+∞ + NaNi	+0 + NaNi
NaN + bi	NaN + NaNi, for finite b
NaN + NaNi	NaN + NaNi

For arguments with a negative imaginary component, use the equality:

- catanh(conj(z)) = conj(catanh(z)).

For arguments with a negative real component, use the equality:

- catanh(-z) = -catanh(z).

Power and absolute value

Function	Description
cabs() (page 329)	Compute magnitude, double complex.
cabsf() (page 330)	Compute magnitude, float complex.
cabsl() (page 330)	Compute magnitude, long double complex.
cpow() (page 357)	Power, double complex.
cpowf() (page 357)	Power, float complex.
cpowl() (page 357)	Power, long double complex.
csqrt() (page 358)	Square root, double complex.
csqrtf() (page 358)	Square root, float complex.
csqrtl() (page 359)	Square root, long double complex.

cabs()

Description

Compute magnitude, double complex.

Prototype

```
double cabs(__SEGGER_RTL_FLOAT64_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute magnitude of.

Return value

The magnitude of x, |x|.

cabsf()

Description

Compute magnitude, float complex.

Prototype

```
float cabsf(__SEGGER_RTL_FLOAT32_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute magnitude of.

Return value

The magnitude of x, |x|.

cabsl()

Description

Compute magnitude, long double complex.

Prototype

```
long double cabsl(__SEGGER_RTL_LDOUBLE_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute magnitude of.

Return value

The magnitude of x, |x|.

cpow()

Description

Power, double complex.

Prototype

```
__SEGGER_RTL_FLOAT64_C_COMPLEX cpow(__SEGGER_RTL_FLOAT64_C_COMPLEX x,
                                    __SEGGER_RTL_FLOAT64_C_COMPLEX y);
```

Parameters

Parameter	Description
x	Base.
y	Power.

Return value

Return x raised to the power of y.

cpowf()

Description

Power, float complex.

Prototype

```
__SEGGER_RTL_FLOAT32_C_COMPLEX cpowf(__SEGGER_RTL_FLOAT32_C_COMPLEX x,
                                     __SEGGER_RTL_FLOAT32_C_COMPLEX y);
```

Parameters

Parameter	Description
x	Base.
y	Power.

Return value

Return x raised to the power of y.

cpowl()

Description

Power, long double complex.

Prototype

```
__SEGGER_RTL_LDOUBLE_C_COMPLEX cpowl(__SEGGER_RTL_LDOUBLE_C_COMPLEX x,
                                     __SEGGER_RTL_LDOUBLE_C_COMPLEX y);
```

Parameters

Parameter	Description
x	Base.
y	Power.

Return value

Return x raised to the power of y.

csqrt()

Description

Square root, double complex.

Prototype

```
__SEGGER_RTL_FLOAT64_C_COMPLEX csqrt(__SEGGER_RTL_FLOAT64_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute squate root of.

Return value

The square root of x according to the following table:

Argument	csqrt(Argument)
±0 + 0i	+0 + 0i
a + ∞i	+∞ + ∞i, for all a
a + NaNi	+NaN + NaNi, for finite a
-∞ + bi	+0 + ∞i for finite positive-signed b
+∞ + bi	+∞ + 0i, for finite positive-signed b
+∞ + ∞i	+∞ + ¼πi
-∞ + NaNi	+NaN + +/∞i, sign of imaginary part unspecified
+∞ + NaNi	+∞ + NaNi
NaN + bi	NaN + NaNi, for finite b
NaN + ∞i	+∞ + ∞i
NaN + NaNi	NaN + NaNi

For arguments with a negative imaginary component, use the equality:

- csqrt(conj(z)) = conj(csqrt(z)).

csqrtf()

Description

Square root, float complex.

Prototype

```
__SEGGER_RTL_FLOAT32_C_COMPLEX csqrtf(__SEGGER_RTL_FLOAT32_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute squate root of.

Return value

The square root of x according to the following table:

Argument	csqrt(Argument)
±0 + 0i	+0 + 0i
a + ∞i	+∞ + ∞i, for all a
a + NaNi	+NaN + NaNi, for finite a
-∞ + bi	+0 + ∞i for finite positive-signed b
+∞ + bi	+∞ + 0i, for finite positive-signed b
+∞ + ∞i	+∞ + ¼πi
-∞ + NaNi	+NaN + +/∞i, sign of imaginary part unspecified
+∞ + NaNi	+∞ + NaNi
NaN + bi	NaN + NaNi, for finite b
NaN + ∞i	+∞ + ∞i
NaN + NaNi	NaN + NaNi

For arguments with a negative imaginary component, use the equality:

- csqrt(conj(z)) = conj(csqrt(z)).

csqrtl()

Description

Square root, long double complex.

Prototype

```
__SEGGER_RTL_LDOUBLE_C_COMPLEX csqrtl(__SEGGER_RTL_LDOUBLE_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute squate root of.

Return value

The square root of x according to the following table:

Argument	csqrt(Argument)
±0 + 0i	+0 + 0i
a + ∞i	+∞ + ∞i, for all a
a + NaNi	+NaN + NaNi, for finite a
-∞ + bi	+0 + ∞i for finite positive-signed b
+∞ + bi	+∞ + 0i, for finite positive-signed b
+∞ + ∞i	+∞ + ¼πi
-∞ + NaNi	+NaN + +/∞i, sign of imaginary part unspecified
+∞ + NaNi	+∞ + NaNi
NaN + bi	NaN + NaNi, for finite b
NaN + ∞i	+∞ + ∞i
NaN + NaNi	NaN + NaNi

For arguments with a negative imaginary component, use the equality:

- csqrt(conj(z)) = conj(csqrt(z)).

Exponential and logarithm functions

Function	Description
clog() (page 360)	Compute natural logarithm, double complex.
clogf() (page 361)	Compute natural logarithm, float complex.
clogl() (page 362)	Compute natural logarithm, long double complex.
cexp() (page 362)	Compute base-e exponential, double complex.
cexpf() (page 363)	Compute base-e exponential, float complex.
cexpl() (page 364)	Compute base-e exponential, long double complex.

clog()

Description

Compute natural logarithm, double complex.

Prototype

```
__SEGGER_RTL_FLOAT64_C_COMPLEX clog(__SEGGER_RTL_FLOAT64_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute logarithm of.

Return value

The natural logarithm of x according to the following table:

Argument	clog(Argument)
-0 + 0i	-∞ + πi
+0 + 0i	-∞ + 0i
a + ∞i	+∞ + ½πi, for finite a
a + NaNi	NaN + NaNi, for finite a
-∞ + bi	+∞ + πi, for finite positive b
+∞ + bi	+∞ + 0i, for finite positive b
-∞ + ∞i	+∞ + ¾πi
+∞ + ∞i	+∞ + ¼πi
±∞ + NaNi	+∞ + NaNi
NaN + bi	NaN + NaNi, for finite b
NaN + ∞i	+∞ + NaNi
NaN + NaNi	NaN + NaNi

For arguments with a negative imaginary component, use the equality:

- clog(conj(z)) = conj(clog(z)).

clogf()

Description

Compute natural logarithm, float complex.

Prototype

```
__SEGGER_RTL_FLOAT32_C_COMPLEX clogf(__SEGGER_RTL_FLOAT32_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute logarithm of.

Return value

The natural logarithm of x according to the following table:

Argument	clog(Argument)
-0 + 0i	-∞ + πi
+0 + 0i	-∞ + 0i
a + ∞i	+∞ + ½πi, for finite a
a + NaNi	NaN + NaNi, for finite a
-∞ + bi	+∞ + πi, for finite positive b
+∞ + bi	+∞ + 0i, for finite positive b
-∞ + ∞i	+∞ + ¾πi
+∞ + ∞i	+∞ + ¼πi
±∞ + NaNi	+∞ + NaNi
NaN + bi	NaN + NaNi, for finite b
NaN + ∞i	+∞ + NaNi
NaN + NaNi	NaN + NaNi

For arguments with a negative imaginary component, use the equality:

- clog(conj(z)) = conj(clog(z)).

clogl()

Description

Compute natural logarithm, long double complex.

Prototype

```
__SEGGER_RTL_LDOUBLE_C_COMPLEX clogl(__SEGGER_RTL_LDOUBLE_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute logarithm of.

Return value

The natural logarithm of x according to the following table:

Argument	clog(Argument)
-0 + 0i	-∞ + πi
+0 + 0i	-∞ + 0i
a + ∞i	+∞ + ½πi, for finite a
a + NaNi	NaN + NaNi, for finite a
-∞ + bi	+∞ + πi, for finite positive b
+∞ + bi	+∞ + 0i, for finite positive b
-∞ + ∞i	+∞ + ¾πi
+∞ + ∞i	+∞ + ¼πi
±∞ + NaNi	+∞ + NaNi
NaN + bi	NaN + NaNi, for finite b
NaN + ∞i	+∞ + NaNi
NaN + NaNi	NaN + NaNi

For arguments with a negative imaginary component, use the equality:

- clog(conj(z)) = conj(clog(z)).

cexp()

Description

Compute base-e exponential, double complex.

Prototype

```
__SEGGER_RTL_FLOAT64_C_COMPLEX cexp(__SEGGER_RTL_FLOAT64_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute exponential of.

Return value

The base-e exponential of x=a+bi according to the following table:

Argument	cexp(Argument)
-/-0 + 0i	+1 + 0i
a + ∞i	NaN + NaNi, for finite a
a + NaNi	NaN + NaNi, for finite a
+∞ + 0i	+∞ + 0i, for finite positive b
-∞ + bi	+0 cis(b) for finite b
+∞ + bi	+∞ cis(b) for finite nonzero b
-∞ + ∞i	±∞ + ±0i, signs unspecified
+∞ + ∞i	±∞ + i.NaN, sign of real part unspecified
-∞ + NaNi	±0 + ±0i, signs unspecified
+∞ + NaNi	±∞ + NaNi, sign of real part unspecified
NaN + 0i	NaN + 0i
NaN + bi	NaN + NaNi, for nonzero b
NaN + NaNi	NaN + NaNi

For arguments with a negative imaginary component, use the equality

- cexp(conj(x)) = conj(cexp(x)).

cexpf()

Description

Compute base-e exponential, float complex.

Prototype

```
__SEGGER_RTL_FLOAT32_C_COMPLEX cexpf(__SEGGER_RTL_FLOAT32_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute exponential of.

Return value

The base-e exponential of x=a+bi according to the following table:

Argument	cexp(Argument)
-/-0 + 0i	+1 + 0i
a + ∞i	NaN + NaNi, for finite a
a + NaNi	NaN + NaNi, for finite a
+∞ + 0i	+∞ + 0i, for finite positive b
-∞ + bi	+0 cis(b) for finite b
+∞ + bi	+∞ cis(b) for finite nonzero b
-∞ + ∞i	±∞ + ±0i, signs unspecified
+∞ + ∞i	±∞ + i.NaN, sign of real part unspecified
-∞ + NaNi	±0 + ±0i, signs unspecified
+∞ + NaNi	±∞ + NaNi, sign of real part unspecified
NaN + 0i	NaN + 0i
NaN + bi	NaN + NaNi, for nonzero b
NaN + NaNi	NaN + NaNi

For arguments with a negative imaginary component, use the equality

- cexp(conj(x)) = conj(cexp(x)).

cexpl()

Description

Compute base-e exponential, long double complex.

Prototype

```
__SEGGER_RTL_LDOUBLE_C_COMPLEX cexpl(__SEGGER_RTL_LDOUBLE_C_COMPLEX x);
```

Parameters

Parameter	Description
x	Value to compute exponential of.

Return value

The base-e exponential of x=a+bi according to the following table:

Argument	cexp(Argument)
-/-0 + 0i	+1 + 0i
a + ∞i	NaN + NaNi, for finite a
a + NaNi	NaN + NaNi, for finite a
+∞ + 0i	+∞ + 0i, for finite positive b
-∞ + bi	+0 cis(b) for finite b
+∞ + bi	+∞ cis(b) for finite nonzero b
-∞ + ∞i	±∞ + ±0i, signs unspecified
+∞ + ∞i	±∞ + i.NaN, sign of real part unspecified
-∞ + NaNi	±0 + ±0i, signs unspecified
+∞ + NaNi	±∞ + NaNi, sign of real part unspecified
NaN + 0i	NaN + 0i
NaN + bi	NaN + NaNi, for nonzero b
NaN + NaNi	NaN + NaNi

For arguments with a negative imaginary component, use the equality

- cexp(conj(x)) = conj(cexp(x)).

4.6.3 <ctype.h>

Classification functions

Function	Description
iscntrl() (page 365)	Is character a control?
iscntrl_l() (page **??**)	Is character a control, per locale? (POSIX.1).
isblank() (page 366)	Is character a blank?
isblank_l() (page **??**)	Is character a blank, per locale? (POSIX.1).
isspace() (page 367)	Is character a whitespace character?
isspace_l() (page **??**)	Is character a whitespace character, per locale? (POSIX.1).
ispunct() (page 368)	Is character a punctuation mark?
ispunct_l() (page **??**)	Is character a punctuation mark, per locale? (POSIX.1).
isdigit() (page 369)	Is character a decimal digit?
isdigit_l() (page **??**)	Is character a decimal digit, per locale? (POSIX.
isxdigit() (page 369)	Is character a hexadecimal digit?
isxdigit_l() (page **??**)	Is character a hexadecimal digit, per locale? (POSIX.1).
isalpha() (page 370)	Is character alphabetic?
isalpha_l() (page **??**)	Is character alphabetic, per locale? (POSIX.1).
isalnum() (page 371)	Is character alphanumeric?
isalnum_l() (page **??**)	Is character alphanumeric, per locale? (POSIX.1).
isupper() (page 372)	Is character an uppercase letter?
isupper_l() (page **??**)	Is character an uppercase letter, per locale? (POSIX.1).
islower() (page 373)	Is character a lowercase letter?
islower_l() (page **??**)	Is character a lowercase letter, per locale? (POSIX.1).
isprint() (page 374)	Is character printable?
isprint_l() (page **??**)	Is character printable, per locale? (POSIX.1).
isgraph() (page 374)	Is character any printing character?
isgraph_l() (page **??**)	Is character any printing character, per locale? (POSIX.1).

iscntrl()

Description

Is character a control?

Prototype

```
int iscntrl(int c);
```

Parameters

Parameter	Description
c	Character to test.

Return value

Returns nonzero (true) if and only if the value of the argument c is a control character in the current locale.

iscntrl_l()

Description

Is character a control, per locale? (POSIX.1).

Prototype

```
int iscntrl_l(int c,
              locale_t loc);
```

Parameters

Parameter	Description
c	Character to test.
loc	Locale used to test c.

Return value

Returns nonzero (true) if and only if the value of the argument c is a control character in the locale loc.

Notes

Conforms to POSIX.1-2017.

isblank()

Description

Is character a blank?

Prototype

```
int isblank(int c);
```

Parameters

Parameter	Description
c	Character to test.

Return value

Returns nonzero (true) if and only if the value of the argument c is either a space character or tab character in the current locale.

isblank_l()

Description

Is character a blank, per locale? (POSIX.1).

Prototype

```
int isblank_l(int c,
              locale_t loc);
```

Parameters

Parameter	Description
c	Character to test.
loc	Locale used to test c.

Return value

Returns nonzero (true) if and only if the value of the argument c is either a space character or the tab character in locale loc.

Notes

Conforms to POSIX.1-2017.

isspace()

Description

Is character a whitespace character?

Prototype

```
int isspace(int c);
```

Parameters

Parameter	Description
c	Character to test.

Return value

Returns nonzero (true) if and only if the value of the argument c is a standard white-space character in the current locale. The standard white-space characters are space, form feed, new-line, carriage return, horizontal tab, and vertical tab.

isspace_l()

Description

Is character a whitespace character, per locale? (POSIX.1).

Prototype

```
int isspace_l(int c, locale_t loc);
```

Parameters

Parameter	Description
c	Character to test.
loc	Locale used to test c.

Return value

Returns nonzero (true) if and only if the value of the argument c is a standard white-space character in the locale loc.

Notes

Conforms to POSIX.1-2017.

ispunct()

Description

Is character a punctuation mark?

Prototype

```
int ispunct(int c);
```

Parameters

Parameter	Description
c	Character to test.

Return value

Returns nonzero (true) for every printing character for which neither *isspace()* (page 367) nor *isalnum()* (page 371) is true in the current locale.

ispunct_l()

Description

Is character a punctuation mark, per locale? (POSIX.1).

Prototype

```
int ispunct_l(int c, locale_t loc);
```

Parameters

Parameter	Description
c	Character to test.
loc	Locale used to test c.

Return value

Returns nonzero (true) for every printing character for which neither *isspace()* (page 367) nor *isalnum()* (page 371) is true in the locale loc.

Notes

Conforms to POSIX.1-2017.

isdigit()

Description

Is character a decimal digit?

Prototype

```
int isdigit(int c);
```

Parameters

Parameter	Description
c	Character to test.

Return value

Returns nonzero (true) if and only if the value of the argument c is a digit in the current locale.

isdigit_l()

Description

Is character a decimal digit, per locale? (POSIX.1)

Prototype

```
int isdigit_l(int c, locale_t loc);
```

Parameters

Parameter	Description
c	Character to test.
loc	Locale used to test c.

Return value

Returns nonzero (true) if and only if the value of the argument c is a digit in the locale loc.

Notes

Conforms to POSIX.1-2017.

isxdigit()

Description

Is character a hexadecimal digit?

Prototype

```
int isxdigit(int c);
```

Parameters

Parameter	Description
c	Character to test.

Return value

Returns nonzero (true) if and only if the value of the argument c is a hexadecimal digit in the current locale.

isxdigit_l()

Description

Is character a hexadecimal digit, per locale? (POSIX.1).

Prototype

```
int isxdigit_l(int c, locale_t loc);
```

Parameters

Parameter	Description
c	Character to test.
loc	Locale used to test c.

Return value

Returns nonzero (true) if and only if the value of the argument c is a hexadecimal digit in the current locale.

Notes

Conforms to POSIX.1-2017.

isalpha()

Description

Is character alphabetic?

Prototype

```
int isalpha(int c);
```

Parameters

Parameter	Description
c	Character to test.

Return value

Returns true if the character c is alphabetic in the current locale. That is, any character for which *isupper()* (page 372) or *islower()* (page 373) returns true is considered alphabetic in addition to any of the locale-specific set of alphabetic characters for which none of *iscntrl()* (page 365), *isdigit()* (page 369), *ispunct()* (page 368), or *isspace()* (page 367) is true.

In the C locale, *isalpha()* (page 370) returns nonzero (true) if and only if *isupper()* (page 372) or *islower()* (page 373) return true for value of the argument c.

isalpha_l()

Description

Is character alphabetic, per locale? (POSIX.1).

Prototype

```c
int isalpha_l(int c, locale_t loc);
```

Parameters

Parameter	Description
c	Character to test.
loc	Locale used to test c.

Return value

Returns true if the character c is alphabetic in the locale loc. That is, any character for which *isupper()* (page 372) or *islower()* (page 373) returns true is considered alphabetic in addition to any of the locale-specific set of alphabetic characters for which none of *iscntrl_l()* (page **??**), *isdigit_l()* (page **??**), *ispunct_l()* (page **??**), or *isspace_l()* (page **??**) is true in the locale loc.

In the C locale, *isalpha_l()* (page **??**) returns nonzero (true) if and only if *isupper_l()* (page **??**) or *islower_l()* (page **??**) return true for value of the argument c.

Notes

Conforms to POSIX.1-2017.

isalnum()

Description

Is character alphanumeric?

Prototype

```c
int isalnum(int c);
```

Parameters

Parameter	Description
c	Character to test.

Return value

Returns nonzero (true) if and only if the value of the argument c is an alphabetic or numeric character in the current locale.

isalnum_l()

Description

Is character alphanumeric, per locale? (POSIX.1).

Prototype

```
int isalnum_l(int c, locale_t loc);
```

Parameters

Parameter	Description
c	Character to test.
loc	Locale used to test c.

Return value

Returns nonzero (true) if and only if the value of the argument c is an alphabetic or numeric character in the locale loc.

Notes

Conforms to POSIX.1-2017.

isupper()

Description

Is character an uppercase letter?

Prototype

```
int isupper(int c);
```

Parameters

Parameter	Description
c	Character to test.

Return value

Returns nonzero (true) if and only if the value of the argument c is an uppercase letter in the current locale.

isupper_l()

Description

Is character an uppercase letter, per locale? (POSIX.1).

Prototype

```
int isupper_l(int c, locale_t loc);
```

Parameters

Parameter	Description
c	Character to test.
loc	Locale used to test c.

Return value

Returns nonzero (true) if and only if the value of the argument c is an uppercase letter in the locale loc.

Notes

Conforms to POSIX.1-2017.

islower()

Description

Is character a lowercase letter?

Prototype

```
int islower(int c);
```

Parameters

Parameter	Description
c	Character to test.

Return value

Returns nonzero (true) if and only if the value of the argument c is a lowercase letter in the current locale.

islower_l()

Description

Is character a lowercase letter, per locale? (POSIX.1).

Prototype

```
int islower_l(int c, locale_t loc);
```

Parameters

Parameter	Description
c	Character to test.
loc	Locale used to test c.

Return value

Returns nonzero (true) if and only if the value of the argument c is a lowercase letter in the locale loc.

Notes

Conforms to POSIX.1-2017.

isprint()

Description

Is character printable?

Prototype

```
int isprint(int c);
```

Parameters

Parameter	Description
c	Character to test.

Return value

Returns nonzero (true) if and only if the value of the argument c is any printing character including space in the current locale.

isprint_l()

Description

Is character printable, per locale? (POSIX.1).

Prototype

```
int isprint_l(int c, locale_t loc);
```

Parameters

Parameter	Description
c	Character to test.
loc	Locale used to test c.

Return value

Returns nonzero (true) if and only if the value of the argument c is any printing character including space in the locale loc.

Notes

Conforms to POSIX.1-2017.

isgraph()

Description

Is character any printing character?

Prototype

```
int isgraph(int c);
```

Parameters

Parameter	Description
c	Character to test.

Return value

Returns nonzero (true) if and only if the value of the argument c is any printing character except space in the current locale.

isgraph_l()

Description

Is character any printing character, per locale? (POSIX.1).

Prototype

```
int isgraph_l(int c, locale_t loc);
```

Parameters

Parameter	Description
c	Character to test.
loc	Locale used to test c.

Return value

Returns nonzero (true) if and only if the value of the argument c is any printing character except space in the locale loc.

Notes

Conforms to POSIX.1-2017.

Conversion functions

Function	Description
toupper() (page 375)	Convert lowercase character to uppercase.
toupper_l() (page **??**)	Convert lowercase character to uppercase, per locale (POSIX.1).
tolower() (page 377)	Convert uppercase character to lowercase.
tolower_l() (page **??**)	Convert uppercase character to lowercase, per locale (POSIX.1).

toupper()

Description

Convert lowercase character to uppercase.

Prototype

```
int toupper(int c);
```

Parameters

Parameter	Description
c	Character to convert.

Return value

Converted character.

Additional information

Converts a lowercase letter to a corresponding uppercase letter.

If the argument c is a character for which *islower()* (page 373) is true and there are one or more corresponding characters, as specified by the current locale, for which *isupper()* (page 372) is true, *toupper()* (page 375) returns one of the corresponding characters (always the same one for any given locale); otherwise, the argument is returned unchanged.

Notes

Even though *islower()* (page 373) can return true for some characters, *toupper()* (page 375) may return that lowercase character unchanged as there are no corresponding uppercase characters in the locale.

toupper_l()

Description

Convert lowercase character to uppercase, per locale (POSIX.1).

Prototype

```
int toupper_l(int c, locale_t loc);
```

Parameters

Parameter	Description
c	Character to convert.
loc	Locale used to convert c.

Return value

Converted character.

Additional information

Converts a lowercase letter to a corresponding uppercase letter in locale loc. If the argument c is a character for which *islower_l()* (page **??**) is true in locale loc, *tolower_l()* (page **??**) returns the corresponding uppercase letter; otherwise, the argument is returned unchanged.

Notes

Conforms to POSIX.1-2017.

tolower()

Description

Convert uppercase character to lowercase.

Prototype

```
int tolower(int c);
```

Parameters

Parameter	Description
c	Character to convert.

Return value

Converted character.

Additional information

Converts an uppercase letter to a corresponding lowercase letter.

If the argument c is a character for which *isupper()* (page 372) is true and there are one or more corresponding characters, as specified by the current locale, for which *islower()* (page 373) is true, the *tolower()* (page 377) function returns one of the corresponding characters (always the same one for any given locale); otherwise, the argument is returned unchanged.

Notes

Even though *isupper()* (page 372) can return true for some characters, *tolower()* (page 377) may return that uppercase character unchanged as there are no corresponding lowercase characters in the locale.

tolower_l()

Description

Convert uppercase character to lowercase, per locale (POSIX.1).

Prototype

```
int tolower_l(int c, locale_t loc);
```

Parameters

Parameter	Description
c	Character to convert.
loc	Locale used to convert c.

Return value

Converted character.

Additional information

Converts an uppercase letter to a corresponding lowercase letter in locale loc. If the argument is a character for which *isupper_l()* (page **??**) is true in locale loc, *tolower_l()* (page **??**) returns the corresponding lowercase letter; otherwise, the argument is returned unchanged.

Notes

Conforms to POSIX.1-2017.

4.6.4 <errno.h>

Errors

Error names

Description

Symbolic error names for raised errors.

Definition

```
#define EDOM      0x01
#define EILSEQ    0x02
#define ERANGE    0x03
#define EHEAP     0x04
#define ENOMEM    0x05
#define EINVAL    0x06
```

Symbols

Definition	Description
EDOM	Domain error
EILSEQ	Illegal multibyte sequence in conversion
ERANGE	Range error
EHEAP	Heap is corrupt
ENOMEM	Out of memory
EINVAL	Invalid parameter

errno

Description

Macro returning the current error.

Definition

```
#define errno    (*__SEGGER_RTL_X_errno_addr())
```

Additional information

The value in errno is significant only when the return value of the call indicated an error. A function that succeeds is allowed to change errno. The value of errno is never set to zero by a library function.

4.6.5 <fenv.h>

Floating-point exceptions

Function	Description
feclearexcept() (page 379)	Clear floating-point exceptions.
feraiseexcept() (page 379)	Raise floating-point exceptions.
fegetexceptflag() (page 380)	Get floating-point exceptions.
fesetexceptflag() (page 380)	Set floating-point exceptions.
fetestexcept() (page 381)	Test floating-point exceptions.

feclearexcept()

Description

Clear floating-point exceptions.

Prototype

```
int feclearexcept(int excepts);
```

Parameters

Parameter	Description
excepts	Bitmask of floating-point exceptions to clear.

Return value

$= 0$	Floating-point exceptions successfully cleared.
$\neq 0$	Floating-point exceptions not cleared or not supported.

Additional information

This function attempts to clear the floating-point exceptions indicated by excepts.

Notes

This function has no return value in ISO C (1999) and an integer return value in ISO C (2008).

feraiseexcept()

Description

Raise floating-point exceptions.

Prototype

```
int feraiseexcept(int excepts);
```

Parameters

Parameter	Description
excepts	Bitmask of floating-point exceptions to raise.

Return value

$= 0$	All floating-point exceptions successfully raised.
$\neq 0$	Floating-point exceptions not successuly raised or not supported.

Additional information

This function attempts to raise the floating-point exceptions indicated by excepts.

Notes

This function has no return value in ISO C (1999) and an integer return value in ISO C (2008).

fegetexceptflag()

Description

Get floating-point exceptions.

Prototype

```
int fegetexceptflag(fexcept_t * flagp,
                    int         excepts);
```

Parameters

Parameter	Description
flagp	Pointer to object that receives the floating-point exception state.
excepts	Bitmask of floating-point exceptions to store.

Return value

= 0	Floating-point exceptions correctly stored.
≠ 0	Floating-point exceptions not correctly stored.

Additional information

This function attempts to save the floating-point exceptions indicated by excepts to the object pointed to by flagp.

See also

fesetexceptflag() (page 380).

fesetexceptflag()

Description

Set floating-point exceptions.

Prototype

```
int fesetexceptflag(const fexcept_t * flagp,
                          int         excepts);
```

Parameters

Parameter	Description
flagp	Pointer to object containing a previously-stored floating-point exception state.
excepts	Bitmask of floating-point exceptions to restore.

Return value

= 0	Floating-point exceptions correctly restored.
≠ 0	Floating-point exceptions not correctly restored.

Additional information

This function attempts to restore the floating-point exceptions indicated by excepts from the object pointed to by flagp. The exceptions to restore as indicated by excepts must have at least been specified when storing the exceptions using *fegetexceptflag()* (page 380).

See also

fegetexceptflag() (page 380).

fetestexcept()

Description

Test floating-point exceptions.

Prototype

```
int fetestexcept(int excepts);
```

Parameters

Parameter	Description
excepts	Bitmask of floating-point exceptions to test.

Return value

The bitmask of all floating-point exceptions that are currently set and are specified in excepts.

Additional information

This function determines which of the floating-point exceptions indicated by excepts are currently set.

Floating-point rounding mode

Function	Description
fegetround() (page 381)	Get floating-point rounding mode.
fesetround() (page 382)	Set floating-point rounding mode.

fegetround()

Description

Get floating-point rounding mode.

Prototype

```
int fegetround(void);
```

Return value

≥ 0	Current floating-point rounding mode.
< 0	Floating-point rounding mode cannot be determined.

Additional information

This function attempts to read the current floating-point rounding mode.

See also

fesetround() (page 382).

fesetround()

Description

Set floating-point rounding mode.

Prototype

```
int fesetround(int round);
```

Parameters

Parameter	Description
round	Rounding mode to set.

Return value

= 0	Current floating-point rounding mode is set to round.
≠ 0	Requested floating-point rounding mode cannot be set.

Additional information

This function attempts to set the current floating-point rounding mode to round.

See also

fegetround() (page 381).

Floating-point environment

Function	Description
fegetenv() (page 382)	Get floating-point environment.
fesetenv() (page 383)	Set floating-point environment.
feupdateenv() (page 383)	Restore floating-point environment and reraise exceptions.
feholdexcept() (page 384)	Save floating-point environment and set non-stop mode.

fegetenv()

Description

Get floating-point environment.

Prototype

```
int fegetenv(fenv_t * envp);
```

Parameters

Parameter	Description
envp	Pointer to object that receives the floating-point environment.

Return value

= 0	Current floating-point environment successfully stored.
≠ 0	Floating-point environment cannot be stored.

Additional information

This function attempts to store the current floating-point environment to the object pointed to by envp.

Notes

This function has no return value in ISO C (1999) and an integer return value in ISO C (2008).

See also

fesetenv() (page 383).

fesetenv()

Description

Set floating-point environment.

Prototype

```
int fesetenv(const fenv_t * envp);
```

Parameters

Parameter	Description
envp	Pointer to object containing previously-stored floating-point environment.

Return value

= 0	Current floating-point environment successfully restored.
≠ 0	Floating-point environment cannot be restored.

Additional information

This function attempts to restore the floating-point environment from the object pointed to by envp.

Notes

This function has no return value in ISO C (1999) and an integer return value in ISO C (2008).

See also

fegetenv() (page 382).

feupdateenv()

Description

Restore floating-point environment and reraise exceptions.

Prototype

```
int feupdateenv(const fenv_t * envp);
```

Parameters

Parameter	Description
envp	Pointer to object containing previously-stored floating-point environment.

Return value

$= 0$	Environment restored and exceptions raised successfully.
$\neq 0$	Failed to restore environment and raise exceptions.

Additional information

This function attempts to save the currently raised floating-point exceptions, restore the floating-point environment from the object pointed to by envp, and raise the saved exceptions.

Notes

This function has no return value in ISO C (1999) and an integer return value in ISO C (2008).

feholdexcept()

Description

Save floating-point environment and set non-stop mode.

Prototype

```
int feholdexcept(fenv_t * envp);
```

Parameters

Parameter	Description
envp	Pointer to object that receives the floating-point environment.

Return value

$= 0$	Environment stored and non-stop mode set successfully.
$\neq 0$	Failed to store environment or set non-stop mode.

Additional information

This function function saves the current floating-point environment to the object pointed to by envp, clears the floating-point status flags, and then installs a non-stop mode for all floating-point exceptions

4.6.6 <float.h>

Floating-point constants

Common parameters

Description

Applies to single-precision and double-precision formats.

Definition

```
#define FLT_ROUNDS          1
#define FLT_EVAL_METHOD     0
#define FLT_RADIX           2
#define DECIMAL_DIG         17
```

Symbols

Definition	Description
FLT_ROUNDS	Rounding mode of floating-point addition is round to nearest.
FLT_EVAL_METHO	All operations and constants are evaluated to the range and precision of the type.
FLT_RADIX	Radix of the exponent representation.
DECIMAL_DIG	Number of decimal digits that can be rounded to a floating-point number without change to the value.

Float parameters

Description

IEEE 32-bit single-precision floating format parameters.

Definition

```
#define FLT_MANT_DIG       24
#define FLT_EPSILON        1.19209290E-07f
#define FLT_DIG            6
#define FLT_MIN_EXP        -125
#define FLT_MIN            1.17549435E-38f
#define FLT_MIN_10_EXP     -37
#define FLT_MAX_EXP        +128
#define FLT_MAX            3.40282347E+38f
#define FLT_MAX_10_EXP     +38
```

Symbols

Definition	Description
FLT_MANT_DIG	Number of base FLT_RADIX digits in the mantissa part of a float.
FLT_EPSILON	Minimum positive number such that 1.0f + FLT_EPSILON ≠ 1.0f.
FLT_DIG	Number of decimal digits of precision of a float.
FLT_MIN_EXP	Minimum value of base FLT_RADIX in the exponent part of a float.
FLT_MIN	Minimum value of a float.
FLT_MIN_10_EXP	Minimum value in base 10 of the exponent part of a float.
FLT_MAX_EXP	Maximum value of base FLT_RADIX in the exponent part of a float.
FLT_MAX	Maximum value of a float.
FLT_MAX_10_EXP	Maximum value in base 10 of the exponent part of a float.

Double parameters

Description

IEEE 64-bit double-precision floating format parameters.

Definition

```
#define DBL_MANT_DIG      53
#define DBL_EPSILON       2.2204460492503131E-16
#define DBL_DIG           15
#define DBL_MIN_EXP       -1021
#define DBL_MIN           2.2250738585072014E-308
#define DBL_MIN_10_EXP    -307
#define DBL_MAX_EXP       +1024
#define DBL_MAX           1.7976931348623157E+308
#define DBL_MAX_10_EXP    +308
```

Symbols

Definition	Description
DBL_MANT_DIG	Number of base DBL_RADIX digits in the mantissa part of a double.
DBL_EPSILON	Minimum positive number such that 1.0 + DBL_EPSILON ≠ 1.0.
DBL_DIG	Number of decimal digits of precision of a double.
DBL_MIN_EXP	Minimum value of base DBL_RADIX in the exponent part of a double.
DBL_MIN	Minimum value of a double.
DBL_MIN_10_EXP	Minimum value in base 10 of the exponent part of a double.
DBL_MAX_EXP	Maximum value of base DBL_RADIX in the exponent part of a double.
DBL_MAX	Maximum value of a double.
DBL_MAX_10_EXP	Maximum value in base 10 of the exponent part of a double.

4.6.7 <iso646.h>

The header <iso646.h> defines macros that expand to the corresponding tokens to ease writing C programs with keyboards that do not have keys for frequently-used operators.

Macros

Replacement macros

Description

Standard replacement macros.

Definition

```
#define and       &&
#define and_eq    &=
#define bitand    &
#define bitor     |
#define compl     ~
#define not       !
#define not_eq    !=
#define or        ||
#define or_eq     |=
#define xor       ^
#define xor_eq    ^=
```

4.6.8 <limits.h>

Minima and maxima

Character minima and maxima

Description

Minimum and maximum values for character types.

Definition

```
#define CHAR_BIT      8
#define CHAR_MIN      0
#define CHAR_MAX      255
#define SCHAR_MAX     127
#define SCHAR_MIN     (-128)
#define UCHAR_MAX     255
```

Symbols

Definition	Description
CHAR_BIT	Number of bits for smallest object that is not a bit-field (byte).
CHAR_MIN	Minimum value of a plain character.
CHAR_MAX	Maximum value of a plain character.
SCHAR_MAX	Maximum value of a signed character.
SCHAR_MIN	Minimum value of a signed character.
UCHAR_MAX	Maximum value of an unsigned character.

Short integer minima and maxima

Description

Minimum and maximum values for short integer types.

Definition

```
#define SHRT_MIN      (-32767 - 1)
#define SHRT_MAX      32767
#define USHRT_MAX     65535
```

Symbols

Definition	Description
SHRT_MIN	Minimum value of a short integer.
SHRT_MAX	Maximum value of a short integer.
USHRT_MAX	Maximum value of an unsigned short integer.

Integer minima and maxima

Description

Minimum and maximum values for integer types.

Definition

```
#define INT_MIN      (-2147483647 - 1)
#define INT_MAX      2147483647
#define UINT_MAX     4294967295u
```

Symbols

Definition	Description
INT_MIN	Minimum value of an integer.
INT_MAX	Maximum value of an integer.
UINT_MAX	Maximum value of an unsigned integer.

Long integer minima and maxima (32-bit)

Description

Minimum and maximum values for long integer types.

Definition

```
#define LONG_MIN      (-2147483647L - 1)
#define LONG_MAX      2147483647L
#define ULONG_MAX     4294967295uL
```

Symbols

Definition	Description
LONG_MIN	Maximum value of a long integer.
LONG_MAX	Minimum value of a long integer.
ULONG_MAX	Maximum value of an unsigned long integer.

Long integer minima and maxima (64-bit)

Description

Minimum and maximum values for long integer types.

Definition

```
#define LONG_MIN      (-9223372036854775807L - 1)
#define LONG_MAX      9223372036854775807L
#define ULONG_MAX     18446744073709551615uL
```

Symbols

Definition	Description
LONG_MIN	Minimum value of a long integer.
LONG_MAX	Maximum value of a long integer.
ULONG_MAX	Maximum value of an unsigned long integer.

Long long integer minima and maxima

Description

Minimum and maximum values for long integer types.

Definition

```
#define LLONG_MIN      (-9223372036854775807LL - 1)
#define LLONG_MAX      9223372036854775807LL
#define ULLONG_MAX     18446744073709551615uLL
```

Symbols

Definition	Description
LLONG_MIN	Minimum value of a long long integer.
LLONG_MAX	Maximum value of a long long integer.
ULLONG_MAX	Maximum value of an unsigned long long integer.

Multibyte characters

Description

Maximum number of bytes in a multi-byte character.

Definition

```
#define MB_LEN_MAX     4
```

Symbols

Definition	Description
MB_LEN_MAX	Maximum

Additional information

The maximum number of bytes in a multi-byte character for any supported locale. Unicode (ISO 10646) characters between 0x000000 and 0x10FFFF inclusive are supported which convert to a maximum of four bytes in the UTF-8 encoding.

4.6.9 <locale.h>

Data types

__SEGGER_RTL_lconv

Type definition

```
typedef struct {
  char * decimal_point;
  char * thousands_sep;
  char * grouping;
  char * int_curr_symbol;
  char * currency_symbol;
  char * mon_decimal_point;
  char * mon_thousands_sep;
```

(continues on next page)

(continued from previous page)

```
  char * mon_grouping;
  char * positive_sign;
  char * negative_sign;
  char   int_frac_digits;
  char   frac_digits;
  char   p_cs_precedes;
  char   p_sep_by_space;
  char   n_cs_precedes;
  char   n_sep_by_space;
  char   p_sign_posn;
  char   n_sign_posn;
  char   int_p_cs_precedes;
  char   int_n_cs_precedes;
  char   int_p_sep_by_space;
  char   int_n_sep_by_space;
  char   int_p_sign_posn;
  char   int_n_sign_posn;
} __SEGGER_RTL_lconv;
```

Structure members

Member	Description
decimal_point	Decimal point separator.
thousands_sep	Separators used to delimit groups of digits to the left of the decimal point for non-monetary quantities.
grouping	Specifies the amount of digits that form each of the groups to be separated by thousands_sep separator for non-monetary quantities.
int_curr_symb	International currency symbol.
currency_symbol	Local currency symbol.
mon_decimal_	Decimal-point separator used for monetary quantities.
mon_thousand	Separators used to delimit groups of digits to the left of the decimal point for monetary quantities.
mon_grouping	Specifies the amount of digits that form each of the groups to be separated by mon_thousands_sep separator for monetary quantities.
positive_sign	Sign to be used for nonnegative (positive or zero) monetary quantities.
negative_sign	Sign to be used for negative monetary quantities.
int_frac_digits	Amount of fractional digits to the right of the decimal point for monetary quantities in the international format.
frac_digits	Amount of fractional digits to the right of the decimal point for monetary quantities in the local format.
p_cs_precedes	Whether the currency symbol should precede nonnegative (positive or zero) monetary quantities.
p_sep_by_spac	Whether a space should appear between the currency symbol and nonnegative (positive or zero) monetary quantities.
n_cs_precedes	Whether the currency symbol should precede negative monetary quantities.
n_sep_by_spac	Whether a space should appear between the currency symbol and negative monetary quantities.
p_sign_posn	Position of the sign for nonnegative (positive or zero) monetary quantities.
n_sign_posn	Position of the sign for negative monetary quantities.
int_p_cs_prece	Whether int_curr_symbol precedes or succeeds the value for a nonnegative internationally formatted monetary quantity.
int_n_cs_prece	Whether int_curr_symbol precedes or succeeds the value for a negative internationally formatted monetary quantity.
int_p_sep_by_	Value indicating the separation of the int_curr_symbol, the sign string, and the value for a nonnegative internationally formatted monetary quantity.
int_n_sep_by_	Value indicating the separation of the int_curr_symbol, the sign string, and the value for a negative internationally formatted monetary quantity.
int_p_sign_pos	Value indicating the positioning of the positive_sign for a nonnegative internationally formatted monetary quantity.
int_n_sign_pos	Value indicating the positioning of the positive_sign for a negative internationally formatted monetary quantity.

Locale management

Function	Description
setlocale() (page 392)	Set locale.
localeconv() (page 392)	Get current locale data.

setlocale()

Description

Set locale.

Prototype

```
char *setlocale(      int    category,
                const char * loc);
```

Parameters

Parameter	Description
category	Category of locale to set, see below.
loc	Pointer to name of locale to set or, if NULL, the current locale.

Return value

Returns the name of the current locale.

Additional information

The category parameter can have the following values:

Value	Description
LC_ALL	Entire locale.
LC_COLLATE	Affects strcoll() and strxfrm().
LC_CTYPE	Affects character handling.
LC_MONETARY	Affects monetary formatting information.
LC_NUMERIC	Affects decimal-point character in I/O and string formatting operations.
LC_TIME	Affects *strftime()* (page 555).

localeconv()

Description

Get current locale data.

Prototype

```
localeconv(void);
```

Return value

Returns a pointer to a structure of type lconv with the corresponding values for the current locale filled in.

4.6.10 <math.h>

Exponential and logarithm functions

Function	Description
sqrt() (page 394)	Compute square root, double.
sqrtf() (page 394)	Compute square root, float.
sqrtl() (page 395)	Compute square root, long double.

continues on next page

Table 4.1 – continued from previous page

Function	Description
cbrt() (page 395)	Compute cube root, double.
cbrtf() (page 396)	Compute cube root, float.
cbrtl() (page 396)	Compute cube root, long double.
rsqrt() (page 397)	Compute reciprocal square root, double.
rsqrtf() (page 397)	Compute reciprocal square root, float.
rsqrtl() (page 398)	Compute reciprocal square root, long double.
exp() (page 398)	Compute base-e exponential, double.
expf() (page 399)	Compute base-e exponential, float.
expl() (page 399)	Compute base-e exponential, long double.
expm1() (page 400)	Compute base-e exponential, modified, double.
expm1f() (page 400)	Compute base-e exponential, modified, float.
expm1l() (page 400)	Compute base-e exponential, modified, long double.
exp2() (page 401)	Compute base-2 exponential, double.
exp2f() (page 401)	Compute base-2 exponential, float.
exp2l() (page 402)	Compute base-2 exponential, long double.
exp10() (page 402)	Compute base-10 exponential, double.
exp10f() (page 403)	Compute base-10 exponential, float.
exp10l() (page 403)	Compute base-10 exponential, long double.
frexp() (page 404)	Split to significand and exponent, double.
frexpf() (page 404)	Split to significand and exponent, float.
frexpl() (page 405)	Split to significand and exponent, long double.
hypot() (page 405)	Compute magnitude of complex, double.
hypotf() (page 406)	Compute magnitude of complex, float.
hypotl() (page 406)	Compute magnitude of complex, long double.
log() (page 407)	Compute natural logarithm, double.
logf() (page 407)	Compute natural logarithm, float.
logl() (page 408)	Compute natural logarithm, long double.
log2() (page 408)	Compute base-2 logarithm, double.
log2f() (page 409)	Compute base-2 logarithm, float.
log2l() (page 409)	Compute base-2 logarithm, long double.
log10() (page 410)	Compute common logarithm, double.
log10f() (page 410)	Compute common logarithm, float.
log10l() (page 411)	Compute common logarithm, long double.
logb() (page 411)	Radix-indpendent exponent, double.
logbf() (page 412)	Radix-indpendent exponent, float.
logbl() (page 412)	Radix-indpendent exponent, long double.
ilogb() (page 413)	Radix-independent exponent, double.
ilogbf() (page 413)	Radix-independent exponent, float.
ilogbl() (page 414)	Radix-independent exponent, long double.
log1p() (page 414)	Compute natural logarithm plus one, double.
log1pf() (page 415)	Compute natural logarithm plus one, float.
log1pl() (page 415)	Compute natural logarithm plus one, long double.
ldexp() (page 416)	Scale by power of two, double.
ldexpf() (page 416)	Scale by power of two, float.
ldexpl() (page 417)	Scale by power of two, long double.
pow() (page 417)	Raise to power, double.
powf() (page 418)	Raise to power, float.
powl() (page 418)	Raise to power, long double.
scalbn() (page 418)	Scale, double.
scalbnf() (page 419)	Scale, float.
scalbnl() (page 420)	Scale, long double.
scalbln() (page 420)	Scale, double.
scalblnf() (page 421)	Scale, float.
scalblnl() (page 421)	Scale, long double.

sqrt()

Description

Compute square root, double.

Prototype

```
double sqrt(double x);
```

Parameters

Parameter	Description
x	Value to compute square root of.

Return value

- If x is zero, return x.
- If x is infinite, return x.
- If x is NaN, return x.
- If x < 0, return NaN.
- Else, return square root of x.

Additional information

sqrt() (page 394) computes the nonnegative square root of x. C90 and C99 require that a domain error occurs if the argument is less than zero, *sqrt()* (page 394) deviates and always uses IEC 60559 semantics.

sqrtf()

Description

Compute square root, float.

Prototype

```
float sqrtf(float x);
```

Parameters

Parameter	Description
x	Value to compute square root of.

Return value

- If x is zero, return x.
- If x is infinite, return x.
- If x is NaN, return x.
- If x < 0, return NaN.
- Else, return square root of x.

Additional information

sqrt() (page 394) computes the nonnegative square root of x. C90 and C99 require that a domain error occurs if the argument is less than zero, *sqrt()* (page 394) deviates and always uses IEC 60559 semantics.

sqrtl()

Description

Compute square root, long double.

Prototype

```
long double sqrtl(long double x);
```

Parameters

Parameter	Description
x	Value to compute square root of.

Return value

- If x is zero, return x.
- If x is infinite, return x.
- If x is NaN, return x.
- If x < 0, return NaN.
- Else, return square root of x.

Additional information

sqrtl() (page 395) computes the nonnegative square root of x. C90 and C99 require that a domain error occurs if the argument is less than zero, *sqrtl()* (page 395) deviates and always uses IEC 60559 semantics.

cbrt()

Description

Compute cube root, double.

Prototype

```
double cbrt(double x);
```

Parameters

Parameter	Description
x	Value to compute cube root of.

Return value

- If x is zero, return x.
- If x is infinite, return x.
- If x is NaN, return x.
- Else, return cube root of x.

cbrtf()

Description

Compute cube root, float.

Prototype

```
float cbrtf(float x);
```

Parameters

Parameter	Description
x	Value to compute cube root of.

Return value

- If x is zero, return x.
- If x is infinite, return x.
- If x is NaN, return x.
- Else, return cube root of x.

cbrtl()

Description

Compute cube root, long double.

Prototype

```
long double cbrtl(long double x);
```

Parameters

Parameter	Description
x	Value to compute cube root of.

Return value

- If x is zero, return x.
- If x is infinite, return x.
- If x is NaN, return x.
- Else, return cube root of x.

rsqrt()

Description

Compute reciprocal square root, double.

Prototype

```
double rsqrt(double x);
```

Parameters

Parameter	Description
x	Value to compute reciprocal square root of.

Return value

- If x is +/-zero, return +/-infinity.
- If x is positively infinite, return 0.
- If x is NaN, return x.
- If x < 0, return NaN.
- Else, return reciprocal square root of x.

rsqrtf()

Description

Compute reciprocal square root, float.

Prototype

```
float rsqrtf(float x);
```

Parameters

Parameter	Description
x	Value to compute reciprocal square root of.

Return value

- If x is +/-zero, return +/-infinity.
- If x is positively infinite, return 0.
- If x is NaN, return x.
- If x < 0, return NaN.
- Else, return reciprocal square root of x.

rsqrtl()

Description

Compute reciprocal square root, long double.

Prototype

```
long double rsqrtl(long double x);
```

Parameters

Parameter	Description
x	Value to compute reciprocal square root of.

Return value

- If x is +/-zero, return +/-infinity.
- If x is positively infinite, return 0.
- If x is NaN, return x.
- If x < 0, return NaN.
- Else, return reciprocal square root of x.

exp()

Description

Compute base-e exponential, double.

Prototype

```
double exp(double x);
```

Parameters

Parameter	Description
x	Value to compute base-e exponential of.

Return value

- If x is NaN, return x.
- If x is positively infinite, return x.
- If x is negatively infinite, return 0.
- Else, return base-e exponential of x.

expf()

Description

Compute base-e exponential, float.

Prototype

```
float expf(float x);
```

Parameters

Parameter	Description
x	Value to compute base-e exponential of.

Return value

- If x is NaN, return x.
- If x is positively infinite, return x.
- If x is negatively infinite, return 0.
- Else, return base-e exponential of x.

expl()

Description

Compute base-e exponential, long double.

Prototype

```
long double expl(long double x);
```

Parameters

Parameter	Description
x	Value to compute base-e exponential of.

Return value

- If x is NaN, return x.
- If x is positively infinite, return x.
- If x is negatively infinite, return 0.
- Else, return base-e exponential of x.

expm1()

Description

Compute base-e exponential, modified, double.

Prototype

```c
double expm1(double x);
```

Parameters

Parameter	Description
x	Value to compute exponential of.

Return value

- If x is NaN, return x.
- Else, return base-e exponential of x minus 1 (e**x - 1).

expm1f()

Description

Compute base-e exponential, modified, float.

Prototype

```c
float expm1f(float x);
```

Parameters

Parameter	Description
x	Value to compute exponential of.

Return value

- If x is NaN, return x.
- Else, return base-e exponential of x minus 1 (e**x - 1).

expm1l()

Description

Compute base-e exponential, modified, long double.

Prototype

```c
long double expm1l(long double x);
```

Parameters

Parameter	Description
x	Value to compute exponential of.

Return value

- If x is NaN, return x.
- Else, return base-e exponential of x minus 1 (e**x - 1).

exp2()

Description

Compute base-2 exponential, double.

Prototype

```
double exp2(double x);
```

Parameters

Parameter	Description
x	Value to compute base-2 exponential of.

Return value

- If x is NaN, return x.
- If x is positively infinite, return x.
- If x is negatively infinite, return 0.
- Else, return base-e exponential of x.

exp2f()

Description

Compute base-2 exponential, float.

Prototype

```
float exp2f(float x);
```

Parameters

Parameter	Description
x	Value to compute base-e exponential of.

Return value

- If x is NaN, return x.
- If x is positively infinite, return x.
- If x is negatively infinite, return 0.
- Else, return base-e exponential of x.

exp2l()

Description

Compute base-2 exponential, long double.

Prototype

```
long double exp2l(long double x);
```

Parameters

Parameter	Description
x	Value to compute base-2 exponential of.

Return value

- If x is NaN, return x.
- If x is positively infinite, return x.
- If x is negatively infinite, return 0.
- Else, return base-e exponential of x.

exp10()

Description

Compute base-10 exponential, double.

Prototype

```
double exp10(double x);
```

Parameters

Parameter	Description
x	Value to compute base-e exponential of.

Return value

- If x is NaN, return x.
- If x is positively infinite, return x.
- If x is negatively infinite, return 0.
- Else, return base-e exponential of x.

exp10f()

Description

Compute base-10 exponential, float.

Prototype

```
float exp10f(float x);
```

Parameters

Parameter	Description
x	Value to compute base-e exponential of.

Return value

- If x is NaN, return x.
- If x is positively infinite, return x.
- If x is negatively infinite, return 0.
- Else, return base-e exponential of x.

exp10l()

Description

Compute base-10 exponential, long double.

Prototype

```
long double exp10l(long double x);
```

Parameters

Parameter	Description
x	Value to compute base-e exponential of.

Return value

- If x is NaN, return x.
- If x is positively infinite, return x.
- If x is negatively infinite, return 0.
- Else, return base-e exponential of x.

frexp()

Description

Split to significand and exponent, double.

Prototype

```
double frexp(double   x,
             int     * exp);
```

Parameters

Parameter	Description
x	Floating value to operate on.
exp	Pointer to integer receiving the power-of-two exponent of x.

Return value

- If x is zero, infinite or NaN, return x and store zero into the integer pointed to by exp.
- Else, return the value f, such that f has a magnitude in the interval [0.5, 1) and x equals f * pow(2, *exp)

Additional information

Breaks a floating-point number into a normalized fraction and an integral power of two.

frexpf()

Description

Split to significand and exponent, float.

Prototype

```
float frexpf(float   x,
             int   * exp);
```

Parameters

Parameter	Description
x	Floating value to operate on.
exp	Pointer to integer receiving the power-of-two exponent of x.

Return value

- If x is zero, infinite or NaN, return x and store zero into the integer pointed to by exp.
- Else, return the value f, such that f has a magnitude in the interval [0.5, 1) and x equals f * pow(2, *exp)

Additional information

Breaks a floating-point number into a normalized fraction and an integral power of two.

frexpl()

Description

Split to significand and exponent, long double.

Prototype

```
long double frexpl(long double   x,
                   int          * exp);
```

Parameters

Parameter	Description
x	Floating value to operate on.
exp	Pointer to integer receiving the power-of-two exponent of x.

Return value

- If x is zero, infinite or NaN, return x and store zero into the integer pointed to by exp.
- Else, return the value f, such that f has a magnitude in the interval [0.5, 1) and x equals f * pow(2, *exp)

Additional information

Breaks a floating-point number into a normalized fraction and an integral power of two.

hypot()

Description

Compute magnitude of complex, double.

Prototype

```
double hypot(double x,
             double y);
```

Parameters

Parameter	Description
x	Value #1.
y	Value #2.

Return value

- If x or y are infinite, return infinity.
- If x or y is NaN, return NaN.
- Else, return sqrt(x*x + y*y).

Additional information

Computes the square root of the sum of the squares of x and y without undue overflow or underflow. If x and y are the lengths of the sides of a right-angled triangle, then this computes the length of the hypotenuse.

hypotf()

Description

Compute magnitude of complex, float.

Prototype

```
float hypotf(float x,
             float y);
```

Parameters

Parameter	Description
x	Value #1.
y	Value #2.

Return value

- If x or y are infinite, return infinity.
- If x or y is NaN, return NaN.
- Else, return sqrt(x*x + y*y).

Additional information

Computes the square root of the sum of the squares of x and y without undue overflow or underflow. If x and y are the lengths of the sides of a right-angled triangle, then this computes the length of the hypotenuse.

hypotl()

Description

Compute magnitude of complex, long double.

Prototype

```
long double hypotl(long double x,
                   long double y);
```

Parameters

Parameter	Description
x	Value #1.
y	Value #2.

Return value

- If x or y are infinite, return infinity.
- If x or y is NaN, return NaN.
- Else, return sqrtl(x*x + y*y).

Additional information

Computes the square root of the sum of the squares of x and y without undue overflow or underflow. If x and y are the lengths of the sides of a right-angled triangle, then this computes the length of the hypotenuse.

log()

Description

Compute natural logarithm, double.

Prototype

```
double log(double x);
```

Parameters

Parameter	Description
x	Value to compute logarithm of.

Return value

- If x = NaN, return x.
- If x < 0, return NaN.
- If x = 0, return -∞.
- If x is +∞, return +∞.
- ELse, return base-e logarithm of x.

logf()

Description

Compute natural logarithm, float.

Prototype

```
float logf(float x);
```

Parameters

Parameter	Description
x	Value to compute logarithm of.

Return value

- If x = NaN, return x.
- If x < 0, return NaN.
- If x = 0, return negative infinity.
- If x is positively infinite, return infinity.
- ELse, return base-e logarithm of x.

logl()

Description

Compute natural logarithm, long double.

Prototype

```
long double logl(long double x);
```

Parameters

Parameter	Description
x	Value to compute logarithm of.

Return value

- If x = NaN, return x.
- If x < 0, return NaN.
- If x = 0, return -∞.
- If x is +∞, return +∞.
- ELse, return base-e logarithm of x.

log2()

Description

Compute base-2 logarithm, double.

Prototype

```
double log2(double x);
```

Parameters

Parameter	Description
x	Value to compute logarithm of.

Return value

- If x = NaN, return x.
- If x < 0, return NaN.
- If x = 0, return negative infinity.
- If x is positively infinite, return infinity.
- ELse, return base-10 logarithm of x.

log2f()

Description

Compute base-2 logarithm, float.

Prototype

```
float log2f(float x);
```

Parameters

Parameter	Description
x	Value to compute logarithm of.

Return value

- If x = NaN, return x.
- If x < 0, return NaN.
- If x = 0, return negative infinity.
- If x is positively infinite, return infinity.
- ELse, return base-10 logarithm of x.

log2l()

Description

Compute base-2 logarithm, long double.

Prototype

```
long double log2l(long double x);
```

Parameters

Parameter	Description
x	Value to compute logarithm of.

Return value

- If x = NaN, return x.
- If x < 0, return NaN.
- If x = 0, return negative infinity.
- If x is positively infinite, return infinity.
- ELse, return base-10 logarithm of x.

log10()

Description

Compute common logarithm, double.

Prototype

```
double log10(double x);
```

Parameters

Parameter	Description
x	Value to compute logarithm of.

Return value

- If x = NaN, return x.
- If x < 0, return NaN.
- If x = 0, return negative infinity.
- If x is positively infinite, return infinity.
- ELse, return base-10 logarithm of x.

log10f()

Description

Compute common logarithm, float.

Prototype

```
float log10f(float x);
```

Parameters

Parameter	Description
x	Value to compute logarithm of.

Return value

- If x = NaN, return x.
- If x < 0, return NaN.
- If x = 0, return negative infinity.
- If x is positively infinite, return infinity.
- ELse, return base-10 logarithm of x.

log10l()

Description

Compute common logarithm, long double.

Prototype

```
long double log10l(long double x);
```

Parameters

Parameter	Description
x	Value to compute logarithm of.

Return value

- If x = NaN, return x.
- If x < 0, return NaN.
- If x = 0, return negative infinity.
- If x is positively infinite, return infinity.
- ELse, return base-10 logarithm of x.

logb()

Description

Radix-indpendent exponent, double.

Prototype

```
double logb(double x);
```

Parameters

Parameter	Description
x	Floating value to operate on.

Return value

- If x is zero, return $-\infty$.
- If x is infinite, return $+\infty$.
- If x is NaN, return NaN.
- Else, return integer part of $\log_{\mathrm{FLTRADIX}}(x)$.

Additional information

Calculates the exponent of x, which is the integral part of the FLTRADIX-logarithm of x.

logbf()

Description

Radix-indpendent exponent, float.

Prototype

```
float logbf(float x);
```

Parameters

Parameter	Description
x	Floating value to operate on.

Return value

- If x is zero, return -∞.
- If x is infinite, return +∞.
- If x is NaN, return NaN.
- Else, return integer part of $\log_{\mathrm{FLTRADIX}}(x)$.

Additional information

Calculates the exponent of x, which is the integral part of the FLTRADIX-logarithm of x.

logbl()

Description

Radix-indpendent exponent, long double.

Prototype

```
long double logbl(long double x);
```

Parameters

Parameter	Description
x	Floating value to operate on.

Return value

- If x is zero, return -∞.
- If x is infinite, return +∞.
- If x is NaN, return NaN.
- Else, return integer part of $\log_{\mathrm{FLTRADIX}}(x)$.

Additional information

Calculates the exponent of x, which is the integral part of the FLTRADIX-logarithm of x.

ilogb()

Description

Radix-independent exponent, double.

Prototype

```
int ilogb(double x);
```

Parameters

Parameter	Description
x	Floating value to operate on.

Return value

- If x is zero, return FP_ILOGB0.
- If x is NaN, return FP_ILOGBNAN.
- If x is infinite, return MAX_INT.
- Else, return integer part of $\log_{\mathrm{FLTRADIX}}(x)$.

ilogbf()

Description

Radix-independent exponent, float.

Prototype

```
int ilogbf(float x);
```

Parameters

Parameter	Description
x	Floating value to operate on.

Return value

- If x is zero, return FP_ILOGB0.
- If x is NaN, return FP_ILOGBNAN.
- If x is infinite, return MAX_INT.
- Else, return integer part of $\log_{\mathrm{FLTRADIX}}(x)$.

ilogbl()

Description

Radix-independent exponent, long double.

Prototype

```
int ilogbl(long double x);
```

Parameters

Parameter	Description
x	Floating value to operate on.

Return value

- If x is zero, return FP_ILOGB0.
- If x is NaN, return FP_ILOGBNAN.
- If x is infinite, return MAX_INT.
- Else, return integer part of $\log_{FLTRADIX}(x)$.

log1p()

Description

Compute natural logarithm plus one, double.

Prototype

```
double log1p(double x);
```

Parameters

Parameter	Description
x	Value to compute logarithm of.

Return value

- If x = NaN, return x.
- If x < 0, return NaN.
- If x = 0, return negative infinity.
- If x is positively infinite, return infinity.
- ELse, return base-e logarithm of x+1.

log1pf()

Description

Compute natural logarithm plus one, float.

Prototype

```
float log1pf(float x);
```

Parameters

Parameter	Description
x	Value to compute logarithm of.

Return value

- If x = NaN, return x.
- If x < 0, return NaN.
- If x = 0, return negative infinity.
- If x is positively infinite, return infinity.
- ELse, return base-e logarithm of x+1.

log1pl()

Description

Compute natural logarithm plus one, long double.

Prototype

```
long double log1pl(long double x);
```

Parameters

Parameter	Description
x	Value to compute logarithm of.

Return value

- If x = NaN, return x.
- If x < 0, return NaN.
- If x = 0, return negative infinity.
- If x is positively infinite, return infinity.
- ELse, return base-e logarithm of x+1.

ldexp()

Description

Scale by power of two, double.

Prototype

```
double ldexp(double x,
             int    n);
```

Parameters

Parameter	Description
x	Value to scale.
n	Power of two to scale by.

Return value

- If x is ±0, return x;
- If x is ±∞, return x.
- If x is NaN, return x.
- Else, return x * 2 ^ n.

Additional information

Multiplies a floating-point number by an integral power of two.

See also

scalbn() (page 418)

ldexpf()

Description

Scale by power of two, float.

Prototype

```
float ldexpf(float x,
             int   n);
```

Parameters

Parameter	Description
x	Value to scale.
n	Power of two to scale by.

Return value

- If x is zero, return x;
- If x is infinite, return x.
- If x is NaN, return x.
- Else, return x * 2^n.

Additional information

Multiplies a floating-point number by an integral power of two.

See also

scalbnf() (page 419)

ldexpl()

Description

Scale by power of two, long double.

Prototype

```
long double ldexpl(long double x,
                   int         n);
```

Parameters

Parameter	Description
x	Value to scale.
n	Power of two to scale by.

Return value

- If x is ±0, return x;
- If x is ±∞, return x.
- If x is NaN, return x.
- Else, return x * 2 ^ n.

Additional information

Multiplies a floating-point number by an integral power of two.

See also

scalbn() (page 418)

pow()

Description

Raise to power, double.

Prototype

```
double pow(double x,
           double y);
```

Parameters

Parameter	Description
x	Base.
y	Power.

Return value

Return x raised to the power y.

powf()

Description

Raise to power, float.

Prototype

```
float powf(float x,
           float y);
```

Parameters

Parameter	Description
x	Base.
y	Power.

Return value

Return x raised to the power y.

powl()

Description

Raise to power, long double.

Prototype

```
long double powl(long double x,
                 long double y);
```

Parameters

Parameter	Description
x	Base.
y	Power.

Return value

Return x raised to the power y.

scalbn()

Description

Scale, double.

Prototype

```
double scalbn(double x,
              int    n);
```

Parameters

Parameter	Description
x	Value to scale.
n	Power of DBL_RADIX to scale by.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return x * DBL_RADIX ^ n.

Additional information

Multiplies a floating-point number by an integral power of DBL_RADIX.

As floating-point arithmetic conforms to IEC 60559, DBL_RADIX is 2 and *scalbn()* (page 418) is (in this implementation) identical to *ldexp()* (page 416).

See also

ldexp() (page 416)

scalbnf()

Description

Scale, float.

Prototype

```
float scalbnf(float x,
              int   n);
```

Parameters

Parameter	Description
x	Value to scale.
n	Power of FLT_RADIX to scale by.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return x * FLT_RADIX ^ n.

Additional information

Multiplies a floating-point number by an integral power of FLT_RADIX.

As floating-point arithmetic conforms to IEC 60559, FLT_RADIX is 2 and *scalbnf()* (page 419) is (in this implementation) identical to *ldexpf()* (page 416).

See also

ldexpf() (page 416)

scalbnl()

Description

Scale, long double.

Prototype

```
long double scalbnl(long double x,
                    int         n);
```

Parameters

Parameter	Description
x	Value to scale.
n	Power of LDBL_RADIX to scale by.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return x * LDBL_RADIX ^ n.

Additional information

Multiplies a floating-point number by an integral power of LDBL_RADIX.

As floating-point arithmetic conforms to IEC 60559, LDBL_RADIX is 2 and *scalbnl()* (page 420) is (in this implementation) identical to *ldexpl()* (page 417).

See also

ldexpl() (page 417)

scalbln()

Description

Scale, double.

Prototype

```
double scalbln(double x,
               long   n);
```

Parameters

Parameter	Description
x	Value to scale.
n	Power of DBL_RADIX to scale by.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return x * DBL_RADIX ^ n.

Additional information

Multiplies a floating-point number by an integral power of DBL_RADIX.

As floating-point arithmetic conforms to IEC 60559, DBL_RADIX is 2 and *scalbln()* (page 420) is (in this implementation) identical to *ldexp()* (page 416).

See also

ldexp() (page 416)

scalblnf()

Description

Scale, float.

Prototype

```
float scalblnf(float x,
               long  n);
```

Parameters

Parameter	Description
x	Value to scale.
n	Power of FLT_RADIX to scale by.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return x * FLT_RADIX ^ n.

Additional information

Multiplies a floating-point number by an integral power of FLT_RADIX.

As floating-point arithmetic conforms to IEC 60559, FLT_RADIX is 2 and *scalbnf()* (page 419) is (in this implementation) identical to *ldexpf()* (page 416).

scalblnl()

Description

Scale, long double.

Prototype

```
long double scalblnl(long double x,
                     long        n);
```

Parameters

Parameter	Description
x	Value to scale.
n	Power of LDBL_RADIX to scale by.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return x * LDBL_RADIX ^ n.

Additional information

Multiplies a floating-point number by an integral power of LDBL_RADIX.

As floating-point arithmetic conforms to IEC 60559, LDBL_RADIX is 2 and *scalblnl()* (page 421) is (in this implementation) identical to *ldexpl()* (page 417).

See also

ldexpl() (page 417)

Trigonometric functions

Function	Description
sin() (page 422)	Calculate sine, double.
sinf() (page 423)	Calculate sine, float.
sinl() (page 423)	Calculate sine, long double.
cos() (page 424)	Calculate cosine, double.
cosf() (page 424)	Calculate cosine, float.
cosl() (page 424)	Calculate cosine, long double.
tan() (page 425)	Compute tangent, double.
tanf() (page 425)	Compute tangent, float.
tanl() (page 426)	Compute tangent, long double.
sinh() (page 426)	Compute hyperbolic sine, double.
sinhf() (page 427)	Compute hyperbolic sine, float.
sinhl() (page 427)	Compute hyperbolic sine, long double.
cosh() (page 427)	Compute hyperbolic cosine, double.
coshf() (page 428)	Compute hyperbolic cosine, float.
coshl() (page 428)	Compute hyperbolic cosine, long double.
tanh() (page 429)	Compute hyperbolic tangent, double.
tanhf() (page 429)	Compute hyperbolic tangent, float.
tanhl() (page 429)	Compute hyperbolic tangent, long double.

sin()

Description

Calculate sine, double.

Prototype

```
double sin(double x);
```

Parameters

Parameter	Description
x	Angle to compute sine of, radians.

Return value

- If x is NaN, return x.
- If x is infinite, return NaN.

- Else, return circular sine of x.

sinf()

Description

Calculate sine, float.

Prototype

```
float sinf(float x);
```

Parameters

Parameter	Description
x	Angle to compute sine of, radians.

Return value

- If x is NaN, return x.
- If x is infinite, return NaN.
- Else, return circular sine of x.

sinl()

Description

Calculate sine, long double.

Prototype

```
long double sinl(long double x);
```

Parameters

Parameter	Description
x	Angle to compute sine of, radians.

Return value

- If x is NaN, return x.
- If x is infinite, return NaN.
- Else, return circular sine of x.

cos()

Description

Calculate cosine, double.

Prototype

```
double cos(double x);
```

Parameters

Parameter	Description
x	Angle to compute cosine of, radians.

Return value

- If x is NaN, return x.
- If x is infinite, return NaN.
- Else, return circular cosine of x.

cosf()

Description

Calculate cosine, float.

Prototype

```
float cosf(float x);
```

Parameters

Parameter	Description
x	Angle to compute cosine of, radians.

Return value

- If x is NaN, return x.
- If x is infinite, return NaN.
- Else, return circular cosine of x.

cosl()

Description

Calculate cosine, long double.

Prototype

```
long double cosl(long double x);
```

Parameters

Parameter	Description
x	Angle to compute cosine of, radians.

Return value

- If x is NaN, return x.
- If x is infinite, return NaN.
- Else, return circular cosine of x.

tan()

Description

Compute tangent, double.

Prototype

```
double tan(double x);
```

Parameters

Parameter	Description
x	Angle to compute tangent of, radians.

Return value

- If x is zero, return x.
- If x is infinite, return NaN.
- If x is NaN, return x.
- Else, return tangent of x.

tanf()

Description

Compute tangent, float.

Prototype

```
float tanf(float x);
```

Parameters

Parameter	Description
x	Angle to compute tangent of, radians.

Return value

- If x is zero, return x.
- If x is infinite, return NaN.

- If x is NaN, return x.
- Else, return tangent of x.

tanl()

Description

Compute tangent, long double.

Prototype

```
long double tanl(long double x);
```

Parameters

Parameter	Description
x	Angle to compute tangent of, radians.

Return value

- If x is zero, return x.
- If x is infinite, return NaN.
- If x is NaN, return x.
- Else, return tangent of x.

sinh()

Description

Compute hyperbolic sine, double.

Prototype

```
double sinh(double x);
```

Parameters

Parameter	Description
x	Value to compute hyperbolic sine of.

Return value

- If x is NaN, return x.
- If x is infinite, return x.
- Else, return hyperbolic sine of x.

sinhf()

Description

Compute hyperbolic sine, float.

Prototype

```
float sinhf(float x);
```

Parameters

Parameter	Description
x	Value to compute hyperbolic sine of.

Return value

- If x is NaN, return x.
- If x is infinite, return x.
- Else, return hyperbolic sine of x.

sinhl()

Description

Compute hyperbolic sine, long double.

Prototype

```
long double sinhl(long double x);
```

Parameters

Parameter	Description
x	Value to compute hyperbolic sine of.

Return value

- If x is NaN, return x.
- If x is infinite, return x.
- Else, return hyperbolic sine of x.

cosh()

Description

Compute hyperbolic cosine, double.

Prototype

```
double cosh(double x);
```

Parameters

Parameter	Description
x	Value to compute hyperbolic cosine of.

Return value

- If x is NaN, return x.
- If x is infinite, return +∞.
- Else, return hyperbolic cosine of x.

coshf()

Description

Compute hyperbolic cosine, float.

Prototype

```
float coshf(float x);
```

Parameters

Parameter	Description
x	Value to compute hyperbolic cosine of.

Return value

- If x is NaN, return x.
- If x is infinite, return +∞.
- Else, return hyperbolic cosine of x.

coshl()

Description

Compute hyperbolic cosine, long double.

Prototype

```
long double coshl(long double x);
```

Parameters

Parameter	Description
x	Value to compute hyperbolic cosine of.

Return value

- If x is NaN, return x.
- If x is infinite, return +∞.
- Else, return hyperbolic cosine of x.

tanh()

Description

Compute hyperbolic tangent, double.

Prototype

```
double tanh(double x);
```

Parameters

Parameter	Description
x	Value to compute hyperbolic tangent of.

Return value

- If x is NaN, return x.
- Else, return hyperbolic tangent of x.

tanhf()

Description

Compute hyperbolic tangent, float.

Prototype

```
float tanhf(float x);
```

Parameters

Parameter	Description
x	Value to compute hyperbolic tangent of.

Return value

- If x is NaN, return x.
- Else, return hyperbolic tangent of x.

tanhl()

Description

Compute hyperbolic tangent, long double.

Prototype

```
long double tanhl(long double x);
```

Parameters

Parameter	Description
x	Value to compute hyperbolic tangent of.

Return value

- If x is NaN, return x.
- Else, return hyperbolic tangent of x.

Inverse trigonometric functions

asin()

Description

Compute inverse sine, double.

Prototype

```
double asin(double x);
```

Parameters

Parameter	Description
x	Value to compute inverse sine of.

Return value

- If x is NaN, return x.
- If |x| > 1, return NaN.
- Else, return inverse circular sine of x.

Additional information

Calculates the principal value, in radians, of the inverse circular sine of x. The principal value lies in the interval [-Pi/2, Pi/2] radians.

asinf()

Description

Compute inverse sine, float.

Prototype

```
float asinf(float x);
```

Parameters

Parameter	Description
x	Value to compute inverse sine of.

Return value

- If x is NaN, return x.
- If |x| > 1, return NaN.
- Else, return inverse circular sine of x.

Additional information

Calculates the principal value, in radians, of the inverse circular sine of x. The principal value lies in the interval [-Pi/2, Pi/2] radians.

asinl()

Description

Compute inverse sine, long double.

Prototype

```
long double asinl(long double x);
```

Parameters

Parameter	Description
x	Value to compute inverse sine of.

Return value

- If x is NaN, return x.
- If |x| > 1, return NaN.
- Else, return inverse circular sine of x.

Additional information

Calculates the principal value, in radians, of the inverse circular sine of x. The principal value lies in the interval [-Pi/2, Pi/2] radians.

acos()

Description

Compute inverse cosine, double.

Prototype

```
double acos(double x);
```

Parameters

Parameter	Description
x	Value to compute inverse cosine of.

Return value

- If x is NaN, return x.
- If |x| > 1, return NaN.
- Else, return inverse circular cosine of x.

Additional information

Calculates the principal value, in radians, of the inverse circular cosine of x. The principal value lies in the interval [0, Pi] radians.

acosf()

Description

Compute inverse cosine, float.

Prototype

```
float acosf(float x);
```

Parameters

Parameter	Description
x	Value to compute inverse cosine of.

Return value

- If x is NaN, return x.
- If |x| > 1, return NaN.
- Else, return inverse circular cosine of x.

Additional information

Calculates the principal value, in radians, of the inverse circular cosine of x. The principal value lies in the interval [0, Pi] radians.

acosl()

Description

Compute inverse cosine, long double.

Prototype

```
long double acosl(long double x);
```

Parameters

Parameter	Description
x	Value to compute inverse cosine of.

Return value

- If x is NaN, return x.
- If |x| > 1, return NaN.
- Else, return inverse circular cosine of x.

Additional information

Calculates the principal value, in radians, of the inverse circular cosine of x. The principal value lies in the interval [0, Pi] radians.

atan()

Description

Compute inverse tangent, double.

Prototype

```
double atan(double x);
```

Parameters

Parameter	Description
x	Value to compute inverse tangent of.

Return value

- If x is NaN, return x.
- Else, return inverse tangent of x.

Additional information

Calculates the principal value, in radians, of the inverse tangent of x. The principal value lies in the interval [-Pi/2, Pi/2] radians.

atanf()

Description

Compute inverse tangent, float.

Prototype

```
float atanf(float x);
```

Parameters

Parameter	Description
x	Value to compute inverse tangent of.

Return value

- If x is NaN, return x.
- Else, return inverse tangent of x.

Additional information

Calculates the principal value, in radians, of the inverse tangent of x. The principal value lies in the interval [-Pi/2, Pi/2] radians.

atanl()

Description

Compute inverse tangent, long double.

Prototype

```
long double atanl(long double x);
```

Parameters

Parameter	Description
x	Value to compute inverse tangent of.

Return value

- If x is NaN, return x.
- Else, return inverse tangent of x.

Additional information

Calculates the principal value, in radians, of the inverse tangent of x. The principal value lies in the interval [-Pi/2, Pi/2] radians.

atan2()

Description

Compute inverse tangent, with quadrant, double.

Prototype

```
double atan2(double y,
             double x);
```

Parameters

Parameter	Description
y	Rise value of angle.
x	Run value of angle.

Return value

Inverse tangent of y/x.

Additional information

This calculates the value, in radians, of the inverse tangent of y divided by x using the signs of x and y to compute the quadrant of the return value. The principal value lies in the interval [-Pi, +Pi] radians.

atan2f()

Description

Compute inverse tangent, with quadrant, float.

Prototype

```
float atan2f(float y,
             float x);
```

Parameters

Parameter	Description
y	Rise value of angle.
x	Run value of angle.

Return value

Inverse tangent of y/x.

Additional information

This calculates the value, in radians, of the inverse tangent of y divided by x using the signs of x and y to compute the quadrant of the return value. The principal value lies in the interval [-Pi, +Pi] radians.

atan2l()

Description

Compute inverse tangent, with quadrant, long double.

Prototype

```
long double atan2l(long double y,
                   long double x);
```

Parameters

Parameter	Description
y	Rise value of angle.
x	Run value of angle.

Return value

Inverse tangent of y/x.

Additional information

This calculates the value, in radians, of the inverse tangent of y divided by x using the signs of x and y to compute the quadrant of the return value. The principal value lies in the interval [-Pi, +Pi] radians.

asinh()

Description

Compute inverse hyperbolic sine, double.

Prototype

```
double asinh(double x);
```

Parameters

Parameter	Description
x	Value to compute inverse hyperbolic sine of.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return inverse hyperbolic sine of x.

asinhf()

Description

Compute inverse hyperbolic sine, float.

Prototype

```
float asinhf(float x);
```

Parameters

Parameter	Description
x	Value to compute inverse hyperbolic sine of.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return inverse hyperbolic sine of x.

Additional information

Calculates the inverse hyperbolic sine of x.

asinhl()

Description

Compute inverse hyperbolic sine, long double.

Prototype

```
long double asinhl(long double x);
```

Parameters

Parameter	Description
x	Value to compute inverse hyperbolic sine of.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return inverse hyperbolic sine of x.

acosh()

Description

Compute inverse hyperbolic cosine, double.

Prototype

```
double acosh(double x);
```

Parameters

Parameter	Description
x	Value to compute inverse hyperbolic cosine of.

Return value

- If x < 1, return NaN.
- If x is NaN, return x.
- Else, return non-negative inverse hyperbolic cosine of x.

acoshf()

Description

Compute inverse hyperbolic cosine, float.

Prototype

```
float acoshf(float x);
```

Parameters

Parameter	Description
x	Value to compute inverse hyperbolic cosine of.

Return value

- If x < 1, return NaN.
- If x is NaN, return x.
- Else, return non-negative inverse hyperbolic cosine of x.

acoshl()

Description

Compute inverse hyperbolic cosine, long double.

Prototype

```
long double acoshl(long double x);
```

Parameters

Parameter	Description
x	Value to compute inverse hyperbolic cosine of.

Return value

- If x < 1, return NaN.
- If x is NaN, return x.
- Else, return non-negative inverse hyperbolic cosine of x.

atanh()

Description

Compute inverse hyperbolic tangent, double.

Prototype

```
double atanh(double x);
```

Parameters

Parameter	Description
x	Value to compute inverse hyperbolic tangent of.

Return value

- If x is NaN, return x.
- If |x| > 1, return NaN.
- If x = +/-1, return +/-infinity.
- Else, return non-negative inverse hyperbolic tangent of x.

atanhf()

Description

Compute inverse hyperbolic tangent, float.

Prototype

```
float atanhf(float x);
```

Parameters

Parameter	Description
x	Value to compute inverse hyperbolic tangent of.

Return value

- If x is NaN, return x.
- If |x| > 1, return NaN.

- If x = +/-1, return +/-infinity.
- Else, return non-negative inverse hyperbolic tangent of x.

atanhl()

Description

Compute inverse hyperbolic tangent, long double.

Prototype

```
long double atanhl(long double x);
```

Parameters

Parameter	Description
x	Value to compute inverse hyperbolic tangent of.

Return value

- If x is NaN, return x.
- If |x| > 1, return NaN.
- If x = +/-1, return +/-infinity.
- Else, return non-negative inverse hyperbolic tangent of x.

Special functions

Function	Description
erf() (page 440)	Error function, double.
erff() (page 441)	Error function, float.
erfl() (page 441)	Error function, long double.
erfc() (page 442)	Complementary error function, double.
erfcf() (page 442)	Complementary error function, float.
erfcl() (page 442)	Complementary error function, long double.
lgamma() (page 443)	Log-Gamma function, double.
lgammaf() (page 443)	Log-Gamma function, float.
lgammal() (page 443)	Log-Gamma function, long double.
tgamma() (page 444)	Gamma function, double.
tgammaf() (page 444)	Gamma function, float.
tgammal() (page 444)	Gamma function, long double.

erf()

Description

Error function, double.

Prototype

```
double erf(double x);
```

Parameters

Parameter	Description
x	Argument.

Return value

erf(x).

erff()

Description

Error function, float.

Prototype

```
float erff(float x);
```

Parameters

Parameter	Description
x	Argument.

Return value

erf(x).

erfl()

Description

Error function, long double.

Prototype

```
long double erfl(long double x);
```

Parameters

Parameter	Description
x	Argument.

Return value

erf(x).

erfc()

Description

Complementary error function, double.

Prototype

```
double erfc(double x);
```

Parameters

Parameter	Description
x	Argument.

Return value

erfc(x).

erfcf()

Description

Complementary error function, float.

Prototype

```
float erfcf(float x);
```

Parameters

Parameter	Description
x	Argument.

Return value

erfc(x).

erfcl()

Description

Complementary error function, long double.

Prototype

```
long double erfcl(long double x);
```

Parameters

Parameter	Description
x	Argument.

Return value

erfc(x).

lgamma()

Description

Log-Gamma function, double.

Prototype

```
double lgamma(double x);
```

Parameters

Parameter	Description
x	Argument.

Return value

log(gamma(x)).

lgammaf()

Description

Log-Gamma function, float.

Prototype

```
float lgammaf(float x);
```

Parameters

Parameter	Description
x	Argument.

Return value

log(gamma(x)).

lgammal()

Description

Log-Gamma function, long double.

Prototype

```
long double lgammal(long double x);
```

Parameters

Parameter	Description
x	Argument.

Return value

log(gamma(x)).

tgamma()

Description

Gamma function, double.

Prototype

```
double tgamma(double x);
```

Parameters

Parameter	Description
x	Argument.

Return value

gamma(x).

tgammaf()

Description

Gamma function, float.

Prototype

```
float tgammaf(float x);
```

Parameters

Parameter	Description
x	Argument.

Return value

gamma(x).

tgammal()

Description

Gamma function, long double.

Prototype

```
long double tgammal(long double x);
```

Parameters

Parameter	Description
x	Argument.

Return value

gamma(x).

Rounding and remainder functions

Function	Description
ceil() (page 446)	Compute smallest integer not less than, double.
ceilf() (page 446)	Compute smallest integer not less than, float.
ceill() (page 447)	Compute smallest integer not less than, long double.
floor() (page 447)	Compute largest integer not greater than, double.
floorf() (page 448)	Compute largest integer not greater than, float.
floorl() (page 448)	Compute largest integer not greater than, long double.
trunc() (page 449)	Truncate to integer, double.
truncf() (page 449)	Truncate to integer, float.
truncl() (page 449)	Truncate to integer, long double.
rint() (page 450)	Round to nearest integer, double.
rintf() (page 450)	Round to nearest integer, float.
rintl() (page 451)	Round to nearest integer, long double.
lrint() (page 451)	Round to nearest integer, double.
lrintf() (page 451)	Round to nearest integer, float.
lrintl() (page 452)	Round to nearest integer, long double.
llrint() (page 452)	Round to nearest integer, double.
llrintf() (page 453)	Round to nearest integer, float.
llrintl() (page 453)	Round to nearest integer, long double.
round() (page 453)	Round to nearest integer, double.
roundf() (page 454)	Round to nearest integer, float.
roundl() (page 454)	Round to nearest integer, long double.
lround() (page 455)	Round to nearest integer, double.
lroundf() (page 455)	Round to nearest integer, float.
lroundl() (page 455)	Round to nearest integer, long double.
llround() (page 456)	Round to nearest integer, double.
llroundf() (page 456)	Round to nearest integer, float.
llroundl() (page 457)	Round to nearest integer, long double.
nearbyint() (page 457)	Round to nearest integer, double.
nearbyintf() (page 457)	Round to nearest integer, float.
nearbyintl() (page 458)	Round to nearest integer, long double.
fmod() (page 458)	Compute remainder after division, double.
fmodf() (page 459)	Compute remainder after division, float.
fmodl() (page 459)	Compute remainder after division, long double.
modf() (page 460)	Separate integer and fractional parts, double.
modff() (page 461)	Separate integer and fractional parts, float.
modfl() (page 461)	Separate integer and fractional parts, long double.
remainder() (page 462)	Compute remainder after division, double.
remainderf() (page 462)	Compute remainder after division, float.
remainderl() (page 463)	Compute remainder after division, long double.
remquo() (page 463)	Compute remainder after division, double.
remquof() (page 464)	Compute remainder after division, float.
remquol() (page 465)	Compute remainder after division, long double.

ceil()

Description

Compute smallest integer not less than, double.

Prototype

```
double ceil(double x);
```

Parameters

Parameter	Description
x	Value to compute ceiling of.

Return value

- If x is zero, return x.
- If x is infinite, return x.
- If x is NaN, return x.
- Else, return the smallest integer value not greater than x.

ceilf()

Description

Compute smallest integer not less than, float.

Prototype

```
float ceilf(float x);
```

Parameters

Parameter	Description
x	Value to compute ceiling of.

Return value

- If x is zero, return x.
- If x is infinite, return x.
- If x is NaN, return x.
- Else, return the smallest integer value not greater than x.

ceill()

Description

Compute smallest integer not less than, long double.

Prototype

```
long double ceill(long double x);
```

Parameters

Parameter	Description
x	Value to compute ceiling of.

Return value

- If x is zero, return x.
- If x is infinite, return x.
- If x is NaN, return x.
- Else, return the smallest integer value not greater than x.

floor()

Description

Compute largest integer not greater than, double.

Prototype

```
double floor(double x);
```

Parameters

Parameter	Description
x	Value to floor.

Return value

- If x is zero, return x.
- If x is infinite, return x.
- If x is NaN, return x.
- Else, return the largest integer value not greater than x.

floorf()

Description

Compute largest integer not greater than, float.

Prototype

```
float floorf(float x);
```

Parameters

Parameter	Description
x	Value to floor.

Return value

- If x is zero, return x.
- If x is infinite, return x.
- If x is NaN, return x.
- Else, return the largest integer value not greater than x.

floorl()

Description

Compute largest integer not greater than, long double.

Prototype

```
long double floorl(long double x);
```

Parameters

Parameter	Description
x	Value to floor.

Return value

- If x is zero, return x.
- If x is infinite, return x.
- If x is NaN, return x.
- Else, return the largest integer value not greater than x.

trunc()

Description

Truncate to integer, double.

Prototype

```
double trunc(double x);
```

Parameters

Parameter	Description
x	Value to truncate.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return x with fractional part removed.

truncf()

Description

Truncate to integer, float.

Prototype

```
float truncf(float x);
```

Parameters

Parameter	Description
x	Value to truncate.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return x with fractional part removed.

truncl()

Description

Truncate to integer, long double.

Prototype

```
long double truncl(long double x);
```

Parameters

Parameter	Description
x	Value to truncate.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return x with fractional part removed.

rint()

Description

Round to nearest integer, double.

Prototype

```
double rint(double x);
```

Parameters

Parameter	Description
x	Value to compute nearest integer of.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return the nearest integer value to x.

rintf()

Description

Round to nearest integer, float.

Prototype

```
float rintf(float x);
```

Parameters

Parameter	Description
x	Value to compute nearest integer of.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return the nearest integer value to x.

rintl()

Description

Round to nearest integer, long double.

Prototype

```
long double rintl(long double x);
```

Parameters

Parameter	Description
x	Value to compute nearest integer of.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return the nearest integer value to x.

lrint()

Description

Round to nearest integer, double.

Prototype

```
long lrint(double x);
```

Parameters

Parameter	Description
x	Value to compute nearest integer of.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return the nearest integer value to x.

lrintf()

Description

Round to nearest integer, float.

Prototype

```
long lrintf(float x);
```

Parameters

Parameter	Description
x	Value to compute nearest integer of.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return the nearest integer value to x.

lrintl()

Description

Round to nearest integer, long double.

Prototype

```
long lrintl(long double x);
```

Parameters

Parameter	Description
x	Value to compute nearest integer of.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return the nearest integer value to x.

llrint()

Description

Round to nearest integer, double.

Prototype

```
long long llrint(double x);
```

Parameters

Parameter	Description
x	Value to compute nearest integer of.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return the nearest integer value to x.

llrintf()

Description

Round to nearest integer, float.

Prototype

```
long long llrintf(float x);
```

Parameters

Parameter	Description
x	Value to compute nearest integer of.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return the nearest integer value to x.

llrintl()

Description

Round to nearest integer, long double.

Prototype

```
long long llrintl(long double x);
```

Parameters

Parameter	Description
x	Value to compute nearest integer of.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return the nearest integer value to x.

round()

Description

Round to nearest integer, double.

Prototype

```
double round(double x);
```

Parameters

Parameter	Description
x	Value to compute nearest integer of.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return the nearest integer value to x, ties away from zero.

roundf()

Description

Round to nearest integer, float.

Prototype

```
float roundf(float x);
```

Parameters

Parameter	Description
x	Value to compute nearest integer of.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return the nearest integer value to x, ties away from zero.

roundl()

Description

Round to nearest integer, long double.

Prototype

```
long double roundl(long double x);
```

Parameters

Parameter	Description
x	Value to compute nearest integer of.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return the nearest integer value to x, ties away from zero.

lround()

Description

Round to nearest integer, double.

Prototype

```
long lround(double x);
```

Parameters

Parameter	Description
x	Value to compute nearest integer of.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return the nearest integer value to x.

lroundf()

Description

Round to nearest integer, float.

Prototype

```
long lroundf(float x);
```

Parameters

Parameter	Description
x	Value to compute nearest integer of.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return the nearest integer value to x.

lroundl()

Description

Round to nearest integer, long double.

Prototype

```
long lroundl(long double x);
```

Parameters

Parameter	Description
x	Value to compute nearest integer of.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return the nearest integer value to x.

llround()

Description

Round to nearest integer, double.

Prototype

```
long long llround(double x);
```

Parameters

Parameter	Description
x	Value to compute nearest integer of.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return the nearest integer value to x.

llroundf()

Description

Round to nearest integer, float.

Prototype

```
long long llroundf(float x);
```

Parameters

Parameter	Description
x	Value to compute nearest integer of.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return the nearest integer value to x.

llroundl()

Description

Round to nearest integer, long double.

Prototype

```
long long llroundl(long double x);
```

Parameters

Parameter	Description
x	Value to compute nearest integer of.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return the nearest integer value to x.

nearbyint()

Description

Round to nearest integer, double.

Prototype

```
double nearbyint(double x);
```

Parameters

Parameter	Description
x	Value to compute nearest integer of.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return the nearest integer value to x.

nearbyintf()

Description

Round to nearest integer, float.

Prototype

```
float nearbyintf(float x);
```

Parameters

Parameter	Description
x	Value to compute nearest integer of.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return the nearest integer value to x.

nearbyintl()

Description

Round to nearest integer, long double.

Prototype

```
long double nearbyintl(long double x);
```

Parameters

Parameter	Description
x	Value to compute nearest integer of.

Return value

- If x is infinite, return x.
- If x is NaN, return x.
- Else, return the nearest integer value to x.

fmod()

Description

Compute remainder after division, double.

Prototype

```
double fmod(double x,
            double y);
```

Parameters

Parameter	Description
x	Value #1.
y	Value #2.

Return value

- If x is NaN, return NaN.
- If x is zero and y is nonzero, return x.

- If x is infinite, return NaN.
- If x is finite and y is infinite, return x.
- If y is NaN, return NaN.
- If y is zero, return NaN.
- Else, return remainder of x divided by y.

Additional information

Computes the floating-point remainder of x divided by y, i.e. the value x - i*y for some integer i such that, if y is nonzero, the result has the same sign as x and magnitude less than the magnitude of y.

fmodf()

Description

Compute remainder after division, float.

Prototype

```
float fmodf(float x,
            float y);
```

Parameters

Parameter	Description
x	Value #1.
y	Value #2.

Return value

- If x is NaN, return NaN.
- If x is zero and y is nonzero, return x.
- If x is infinite, return NaN.
- If x is finite and y is infinite, return x.
- If y is NaN, return NaN.
- If y is zero, return NaN.
- Else, return remainder of x divided by y.

Additional information

Computes the floating-point remainder of x divided by y, i.e. the value x - i*y for some integer i such that, if y is nonzero, the result has the same sign as x and magnitude less than the magnitude of y.

fmodl()

Description

Compute remainder after division, long double.

Prototype

```
long double fmodl(long double x,
                  long double y);
```

Parameters

Parameter	Description
x	Value #1.
y	Value #2.

Return value

- If x is NaN, return NaN.
- If x is zero and y is nonzero, return x.
- If x is infinite, return NaN.
- If x is finite and y is infinite, return x.
- If y is NaN, return NaN.
- If y is zero, return NaN.
- Else, return remainder of x divided by y.

Additional information

Computes the floating-point remainder of x divided by y, i.e. the value x - i*y for some integer i such that, if y is nonzero, the result has the same sign as x and magnitude less than the magnitude of y.

modf()

Description

Separate integer and fractional parts, double.

Prototype

```
double modf(double   x,
            double * iptr);
```

Parameters

Parameter	Description
x	Value to separate.
iptr	Pointer to object that receives the integral part of x.

Return value

The signed fractional part of x.

Additional information

Breaks x into integral and fractional parts, each of which has the same type and sign as x.

The integral part (in floating-point format) is stored in the object pointed to by iptr and *modf()* (page 460) returns the signed fractional part of x.

modff()

Description

Separate integer and fractional parts, float.

Prototype

```
float modff(float   x,
            float * iptr);
```

Parameters

Parameter	Description
x	Value to separate.
iptr	Pointer to object that receives the integral part of x.

Return value

The signed fractional part of x.

Additional information

Breaks x into integral and fractional parts, each of which has the same type and sign as x.

The integral part (in floating-point format) is stored in the object pointed to by iptr and *modff()* (page 461) returns the signed fractional part of x.

modfl()

Description

Separate integer and fractional parts, long double.

Prototype

```
long double modfl(long double   x,
                  long double * iptr);
```

Parameters

Parameter	Description
x	Value to separate.
iptr	Pointer to object that receives the integral part of x.

Return value

The signed fractional part of x.

Additional information

Breaks x into integral and fractional parts, each of which has the same type and sign as x.

The integral part (in floating-point format) is stored in the object pointed to by iptr and *modf()* (page 460) returns the signed fractional part of x.

remainder()

Description

Compute remainder after division, double.

Prototype

```
double remainder(double x,
                 double y);
```

Parameters

Parameter	Description
x	Value #1.
y	Value #2.

Return value

- If x is NaN, return NaN.
- If x is zero and y is nonzero, return x.
- If x is infinite, return NaN.
- If x is finite and y is infinite, return x.
- If y is NaN, return NaN.
- If y is zero, return NaN.
- Else, return remainder of x divided by y.

Additional information

Computes the floating-point remainder of x divided by y, i.e. the value x - i*y for some integer i such that, if y is nonzero, the result has the same sign as x and magnitude less than the magnitude of y.

remainderf()

Description

Compute remainder after division, float.

Prototype

```
float remainderf(float x,
                 float y);
```

Parameters

Parameter	Description
x	Value #1.
y	Value #2.

Return value

- If x is NaN, return NaN.
- If x is zero and y is nonzero, return x.
- If x is infinite, return NaN.

- If x is finite and y is infinite, return x.
- If y is NaN, return NaN.
- If y is zero, return NaN.
- Else, return remainder of x divided by y.

Additional information

Computes the floating-point remainder of x divided by y, i.e. the value x - i*y for some integer i such that, if y is nonzero, the result has the same sign as x and magnitude less than the magnitude of y.

remainderl()

Description

Compute remainder after division, long double.

Prototype

```
long double remainderl(long double x,
                       long double y);
```

Parameters

Parameter	Description
x	Value #1.
y	Value #2.

Return value

- If x is NaN, return NaN.
- If x is zero and y is nonzero, return x.
- If x is infinite, return NaN.
- If x is finite and y is infinite, return x.
- If y is NaN, return NaN.
- If y is zero, return NaN.
- Else, return remainder of x divided by y.

Additional information

Computes the floating-point remainder of x divided by y, i.e. the value x - i*y for some integer i such that, if y is nonzero, the result has the same sign as x and magnitude less than the magnitude of y.

remquo()

Description

Compute remainder after division, double.

Prototype

```
double remquo(double    x,
              double    y,
              int     * quo);
```

Parameters

Parameter	Description
x	Value #1.
y	Value #2.
quo	Pointer to object that receives the integer part of x divided by y.

Return value

- If x is NaN, return NaN.
- If x is zero and y is nonzero, return x.
- If x is infinite, return NaN.
- If x is finite and y is infinite, return x.
- If y is NaN, return NaN.
- If y is zero, return NaN.
- Else, return remainder of x divided by y.

Additional information

Computes the floating-point remainder of x divided by y, i.e. the value x - i*y for some integer i such that, if y is nonzero, the result has the same sign as x and magnitude less than the magnitude of y.

remquof()

Description

Compute remainder after division, float.

Prototype

```
float remquof(float   x,
              float   y,
              int   * quo);
```

Parameters

Parameter	Description
x	Value #1.
y	Value #2.
quo	Pointer to object that receives the integer part of x divided by y.

Return value

- If x is NaN, return NaN.
- If x is zero and y is nonzero, return x.
- If x is infinite, return NaN.
- If x is finite and y is infinite, return x.
- If y is NaN, return NaN.
- If y is zero, return NaN.
- Else, return remainder of x divided by y.

Additional information

Computes the floating-point remainder of x divided by y, i.e. the value x - i*y for some integer i such that, if y is nonzero, the result has the same sign as x and magnitude less than the magnitude of y.

remquol()

Description

Compute remainder after division, long double.

Prototype

```
long double remquol(long double   x,
                    long double   y,
                    int         * quo);
```

Parameters

Parameter	Description
x	Value #1.
y	Value #2.
quo	Pointer to object that receives the integer part of x divided by y.

Return value

- If x is NaN, return NaN.
- If x is zero and y is nonzero, return x.
- If x is infinite, return NaN.
- If x is finite and y is infinite, return x.
- If y is NaN, return NaN.
- If y is zero, return NaN.
- Else, return remainder of x divided by y.

Additional information

Computes the floating-point remainder of x divided by y, i.e. the value x - i*y for some integer i such that, if y is nonzero, the result has the same sign as x and magnitude less than the magnitude of y.

Absolute value functions

Function	Description
fabs() (page 466)	Compute absolute value, double.
fabsf() (page 466)	Compute absolute value, float.
fabsl() (page 466)	Compute absolute value, long double.

fabs()

Description

Compute absolute value, double.

Prototype

```
double fabs(double x);
```

Parameters

Parameter	Description
x	Value to compute magnitude of.

Return value

- If x is NaN, return x.
- Else, absolute value of x.

fabsf()

Description

Compute absolute value, float.

Prototype

```
float fabsf(float x);
```

Parameters

Parameter	Description
x	Value to compute magnitude of.

Return value

- If x is NaN, return x.
- Else, absolute value of x.

fabsl()

Description

Compute absolute value, long double.

Prototype

```
long double fabsl(long double x);
```

Parameters

Parameter	Description
x	Value to compute magnitude of.

Return value

- If x is NaN, return x.
- Else, absolute value of x.

Fused multiply functions

Function	Description
fma() (page 467)	Compute fused multiply-add, double.
fmaf() (page 467)	Compute fused multiply-add, float.
fmal() (page 468)	Compute fused multiply-add, long double.

fma()

Description

Compute fused multiply-add, double.

Prototype

```
double fma(double x,
           double y,
           double z);
```

Parameters

Parameter	Description
x	Multiplicand.
y	Multiplier.
z	Summand.

Return value

Return (x * y) + z.

fmaf()

Description

Compute fused multiply-add, float.

Prototype

```
float fmaf(float x,
           float y,
           float z);
```

Parameters

Parameter	Description
x	Multiplier.
y	Multiplicand.
z	Summand.

Return value

Return (x * y) + z.

fmal()

Description

Compute fused multiply-add, long double.

Prototype

```
long double fmal(long double x,
                 long double y,
                 long double z);
```

Parameters

Parameter	Description
x	Multiplicand.
y	Multiplier.
z	Summand.

Return value

Return (x * y) + z.

Maximum, minimum, and positive difference functions

Function	Description
fmin() (page 468)	Compute minimum, double.
fminf() (page 469)	Compute minimum, float.
fminl() (page 469)	Compute minimum, long double.
fmax() (page 470)	Compute maximum, double.
fmaxf() (page 470)	Compute maximum, float.
fmaxl() (page 471)	Compute maximum, long double.
fdim() (page 471)	Positive difference, double.
fdimf() (page 471)	Positive difference, float.
fdiml() (page 472)	Positive difference, long double.

fmin()

Description

Compute minimum, double.

Prototype

```
double fmin(double x,
            double y);
```

Parameters

Parameter	Description
x	Value #1.
y	Value #2.

Return value

- If x is NaN, return y.
- If y is NaN, return x.
- Else, return minimum of x and y.

fminf()

Description

Compute minimum, float.

Prototype

```
float fminf(float x,
            float y);
```

Parameters

Parameter	Description
x	Value #1.
y	Value #2.

Return value

- If x is NaN, return y.
- If y is NaN, return x.
- Else, return minimum of x and y.

fminl()

Description

Compute minimum, long double.

Prototype

```
long double fminl(long double x,
                  long double y);
```

Parameters

Parameter	Description
x	Value #1.
y	Value #2.

Return value

- If x is NaN, return y.
- If y is NaN, return x.
- Else, return minimum of x and y.

fmax()

Description

Compute maximum, double.

Prototype

```
double fmax(double x,
            double y);
```

Parameters

Parameter	Description
x	Value #1.
y	Value #2.

Return value

- If x is NaN, return y.
- If y is NaN, return x.
- Else, return maximum of x and y.

fmaxf()

Description

Compute maximum, float.

Prototype

```
float fmaxf(float x,
            float y);
```

Parameters

Parameter	Description
x	Value #1.
y	Value #2.

Return value

- If x is NaN, return y.
- If y is NaN, return x.
- Else, return maximum of x and y.

fmaxl()

Description

Compute maximum, long double.

Prototype

```
long double fmaxl(long double x,
                  long double y);
```

Parameters

Parameter	Description
x	Value #1.
y	Value #2.

Return value

- If x is NaN, return y.
- If y is NaN, return x.
- Else, return maximum of x and y.

fdim()

Description

Positive difference, double.

Prototype

```
double fdim(double x,
            double y);
```

Parameters

Parameter	Description
x	Value #1.
y	Value #2.

Return value

- If x > y, x-y.
- Else, +0.

fdimf()

Description

Positive difference, float.

Prototype

```
float fdimf(float x,
            float y);
```

Parameters

Parameter	Description
x	Value #1.
y	Value #2.

Return value

- If x > y, x-y.
- Else, +0.

fdiml()

Description

Positive difference, long double.

Prototype

```
long double fdiml(long double x,
                  long double y);
```

Parameters

Parameter	Description
x	Value #1.
y	Value #2.

Return value

- If x > y, x-y.
- Else, +0.

Miscellaneous functions

Function	Description
nextafter() (page 473)	Next machine-floating value, double.
nextafterf() (page 473)	Next machine-floating value, float.
nextafterl() (page 473)	Next machine-floating value, long double.
nexttoward() (page 474)	Next machine-floating value, double.
nexttowardf() (page 474)	Next machine-floating value, float.
nexttowardl() (page 475)	Next machine-floating value, long double.
nan() (page 475)	Parse NaN, double.
nanf() (page 475)	Parse NaN, float.
nanl() (page 476)	Parse NaN, long double.
copysign() (page 476)	Copy sign, double.
copysignf() (page 476)	Copy sign, float.
copysignl() (page 477)	Copy sign, long double.

nextafter()

Description

Next machine-floating value, double.

Prototype

```
double nextafter(double x,
                 double y);
```

Parameters

Parameter	Description
x	Value to step from.
y	Director to step in.

Return value

Next machine-floating value after x in direction of y.

nextafterf()

Description

Next machine-floating value, float.

Prototype

```
float nextafterf(float x,
                 float y);
```

Parameters

Parameter	Description
x	Value to step from.
y	Director to step in.

Return value

Next machine-floating value after x in direction of y.

nextafterl()

Description

Next machine-floating value, long double.

Prototype

```
long double nextafterl(long double x,
                       long double y);
```

Parameters

Parameter	Description
x	Value to step from.
y	Director to step in.

Return value

Next machine-floating value after x in direction of y.

nexttoward()

Description

Next machine-floating value, double.

Prototype

```
double nexttoward(double      x,
                  long double y);
```

Parameters

Parameter	Description
x	Value to step from.
y	Direction to step in.

Return value

Next machine-floating value after x in direction of y.

nexttowardf()

Description

Next machine-floating value, float.

Prototype

```
float nexttowardf(float       x,
                  long double y);
```

Parameters

Parameter	Description
x	Value to step from.
y	Direction to step in.

Return value

Next machine-floating value after x in direction of y.

nexttowardl()

Description

Next machine-floating value, long double.

Prototype

```
long double nexttowardl(long double x,
                        long double y);
```

Parameters

Parameter	Description
x	Value to step from.
y	Direction to step in.

Return value

Next machine-floating value after x in direction of y.

nan()

Description

Parse NaN, double.

Prototype

```
double nan(const char * tag);
```

Parameters

Parameter	Description
tag	NaN tag.

Return value

Quiet NaN formed from tag.

nanf()

Description

Parse NaN, float.

Prototype

```
float nanf(const char * tag);
```

Parameters

Parameter	Description
tag	NaN tag.

Return value

Quiet NaN formed from tag.

nanl()

Description

Parse NaN, long double.

Prototype

```
long double nanl(const char * tag);
```

Parameters

Parameter	Description
tag	NaN tag.

Return value

Quiet NaN formed from tag.

copysign()

Description

Copy sign, double.

Prototype

```
double copysign(double x,
                double y);
```

Parameters

Parameter	Description
x	Floating value to inject sign into.
y	Floating value carrying the sign to inject.

Return value

x with the sign of y.

copysignf()

Description

Copy sign, float.

Prototype

```
float copysignf(float x,
                float y);
```

Parameters

Parameter	Description
x	Floating value to inject sign into.
y	Floating value carrying the sign to inject.

Return value

x with the sign of y.

copysignl()

Description

Copy sign, long double.

Prototype

```
long double copysignl(long double x,
                      long double y);
```

Parameters

Parameter	Description
x	Floating value to inject sign into.
y	Floating value carrying the sign to inject.

Return value

x with the sign of y.

4.6.11 <setjmp.h>

Non-local flow control

setjmp()

Description

Save calling environment for non-local jump.

Prototype

```
int setjmp(jmp_buf buf);
```

Parameters

Parameter	Description
buf	Buffer to save context into.

Return value

On return from a direct invocation, returns the value zero. On return from a call to the *longjmp()* (page 478) function, returns a nonzero value determined by the call to *longjmp()* (page 478).

Additional information

Saves its calling environment in env for later use by the *longjmp()* (page 478) function.

The environment saved by a call to setjmp () consists of information sufficient for a call to the *longjmp()* (page 478) function to return execution to the correct block and invocation of that block, were it called recursively.

longjmp()

Description

Restores the saved environment.

Prototype

```
void longjmp(jmp_buf buf,
             int      val);
```

Parameters

Parameter	Description
buf	Buffer to restore context from.
val	Value to return to *setjmp()* (page 477) call.

Additional information

Restores the environment saved by *setjmp()* (page 477) in the corresponding env argument. If there has been no such invocation, or if the function containing the invocation of *setjmp()* (page 477) has terminated execution in the interim, the behavior of *longjmp()* (page 478) is undefined.

After *longjmp()* (page 478) is completed, program execution continues as if the corresponding invocation of *setjmp()* (page 477) had just returned the value specified by val.

Objects of automatic storage allocation that are local to the function containing the invocation of the corresponding *setjmp()* (page 477) that do not have volatile-qualified type and have been changed between the *setjmp()* (page 477) invocation and *longjmp()* (page 478) call are indeterminate.

Notes

longjmp() (page 478) cannot cause *setjmp()* (page 477) to return the value 0; if val is 0, *setjmp()* (page 477) returns the value 1.

4.6.12 <stdbool.h>

Macros

bool

Description

Macros expanding to support the Boolean type.

Definition

```
#define bool     _Bool
#define true     1
#define false    0
```

Symbols

Definition	Description
bool	Underlying boolean type
true	Boolean true value
false	Boolean false value

4.6.13 <stddef.h>

Macros

NULL

Description

Null-pointer constant.

Definition

```
#define NULL     0
```

Symbols

Definition	Description
NULL	Null pointer

offsetof

Description

Calculate offset of member from start of structure.

Definition

```
#define offsetof(s,m)     ((size_t)&(((s *)0)->m))
```

Symbols

Definition	Description
offsetof(s,m)	Offset of m within s

Types

size_t

Description

Unsigned integral type returned by the sizeof operator.

Type definition

```
typedef __SEGGER_RTL_SIZE_T size_t;
```

ptrdiff_t

Description

Signed integral type of the result of subtracting two pointers.

Type definition

```
typedef __SEGGER_RTL_PTRDIFF_T ptrdiff_t;
```

wchar_t

Description

Integral type that can hold one wide character.

Type definition

```
typedef __SEGGER_RTL_WCHAR_T wchar_t;
```

4.6.14 <stdint.h>

Minima and maxima

Signed integer minima and maxima

Description

Minimum and maximum values for signed integer types.

Definition

```
#define INT8_MIN      (-128)
#define INT8_MAX      127
#define INT16_MIN     (-32767-1)
#define INT16_MAX     32767
#define INT32_MIN     (-2147483647L-1)
#define INT32_MAX     2147483647L
#define INT64_MIN     (-9223372036854775807LL-1)
#define INT64_MAX     9223372036854775807LL
```

Symbols

Definition	Description
INT8_MIN	Minimum value of *int8_t* (page **??**)
INT8_MAX	Maximum value of *int8_t* (page **??**)
INT16_MIN	Minimum value of *int16_t* (page **??**)
INT16_MAX	Maximum value of *int16_t* (page **??**)
INT32_MIN	Minimum value of *int32_t* (page **??**)
INT32_MAX	Maximum value of *int32_t* (page **??**)
INT64_MIN	Minimum value of *int64_t* (page **??**)
INT64_MAX	Maximum value of *int64_t* (page **??**)

Unsigned integer minima and maxima

Description

Minimum and maximum values for unsigned integer types.

Definition

```
#define UINT8_MAX      255
#define UINT16_MAX     65535
#define UINT32_MAX     4294967295UL
#define UINT64_MAX     18446744073709551615ULL
```

Symbols

Definition	Description
UINT8_MAX	Maximum value of *uint8_t* (page **??**)
UINT16_MAX	Maximum value of *uint16_t* (page **??**)
UINT32_MAX	Maximum value of *uint32_t* (page **??**)
UINT64_MAX	Maximum value of *uint64_t* (page **??**)

Maximal integer minima and maxima

Description

Minimum and maximum values for signed and unsigned maximal-integer types.

Definition

```
#define INTMAX_MIN      INT64_MIN
#define INTMAX_MAX      INT64_MAX
#define UINTMAX_MAX     UINT64_MAX
```

Symbols

Definition	Description
INTMAX_MIN	Minimum value of *intmax_t* (page **??**)
INTMAX_MAX	Maximum value of *intmax_t* (page **??**)
UINTMAX_MAX	Maximum value of *uintmax_t* (page **??**)

Least integer minima and maxima

Description

Minimum and maximum values for signed and unsigned least-integer types.

Definition

```
#define INT_LEAST8_MIN       INT8_MIN
#define INT_LEAST8_MAX       INT8_MAX
#define INT_LEAST16_MIN      INT16_MIN
#define INT_LEAST16_MAX      INT16_MAX
#define INT_LEAST32_MIN      INT32_MIN
#define INT_LEAST32_MAX      INT32_MAX
#define INT_LEAST64_MIN      INT64_MIN
#define INT_LEAST64_MAX      INT64_MAX
#define UINT_LEAST8_MAX      UINT8_MAX
```

(continues on next page)

(continued from previous page)

```
#define UINT_LEAST16_MAX    UINT16_MAX
#define UINT_LEAST32_MAX    UINT32_MAX
#define UINT_LEAST64_MAX    UINT64_MAX
```

Symbols

Definition	Description
INT_LEAST8_MIN	Minimum value of *int_least8_t* (page **??**)
INT_LEAST8_MAX	Maximum value of *int_least8_t* (page **??**)
INT_LEAST16_MIN	Minimum value of *int_least16_t* (page **??**)
INT_LEAST16_MAX	Maximum value of *int_least16_t* (page **??**)
INT_LEAST32_MIN	Minimum value of *int_least32_t* (page **??**)
INT_LEAST32_MAX	Maximum value of *int_least32_t* (page **??**)
INT_LEAST64_MIN	Minimum value of *int_least64_t* (page **??**)
INT_LEAST64_MAX	Maximum value of *int_least64_t* (page **??**)
UINT_LEAST8_MAX	Maximum value of *uint_least8_t* (page **??**)
UINT_LEAST16_MAX	Maximum value of *uint_least16_t* (page **??**)
UINT_LEAST32_MAX	Maximum value of *uint_least32_t* (page **??**)
UINT_LEAST64_MAX	Maximum value of *uint_least64_t* (page **??**)

Fast integer minima and maxima

Description

Minimum and maximum values for signed and unsigned fast-integer types.

Definition

```
#define INT_FAST8_MIN      INT8_MIN
#define INT_FAST8_MAX      INT8_MAX
#define INT_FAST16_MIN     INT32_MIN
#define INT_FAST16_MAX     INT32_MAX
#define INT_FAST32_MIN     INT32_MIN
#define INT_FAST32_MAX     INT32_MAX
#define INT_FAST64_MIN     INT64_MIN
#define INT_FAST64_MAX     INT64_MAX
#define UINT_FAST8_MAX     UINT8_MAX
#define UINT_FAST16_MAX    UINT32_MAX
#define UINT_FAST32_MAX    UINT32_MAX
#define UINT_FAST64_MAX    UINT64_MAX
```

Symbols

Definition	Description
INT_FAST8_MIN	Minimum value of *int_fast8_t* (page **??**)
INT_FAST8_MAX	Maximum value of *int_fast8_t* (page **??**)
INT_FAST16_MIN	Minimum value of *int_fast16_t* (page **??**)
INT_FAST16_MAX	Maximum value of *int_fast16_t* (page **??**)
INT_FAST32_MIN	Minimum value of *int_fast32_t* (page **??**)
INT_FAST32_MAX	Maximum value of *int_fast32_t* (page **??**)
INT_FAST64_MIN	Minimum value of *int_fast64_t* (page **??**)
INT_FAST64_MAX	Maximum value of *int_fast64_t* (page **??**)
UINT_FAST8_MAX	Maximum value of *uint_fast8_t* (page **??**)
UINT_FAST16_MAX	Maximum value of *uint_fast16_t* (page **??**)
UINT_FAST32_MAX	Maximum value of *uint_fast32_t* (page **??**)
UINT_FAST64_MAX	Maximum value of *uint_fast64_t* (page **??**)

Pointer types minima and maxima

Description

Minimum and maximum values for pointer-related types.

Definition

```
#define PTRDIFF_MIN     INT64_MIN
#define PTRDIFF_MAX     INT64_MAX
#define SIZE_MAX        INT64_MAX
#define INTPTR_MIN      INT64_MIN
#define INTPTR_MAX      INT64_MAX
#define UINTPTR_MAX     UINT64_MAX
```

Symbols

Definition	Description
PTRDIFF_MIN	Minimum value of *ptrdiff_t* (page **??**)
PTRDIFF_MAX	Maximum value of *ptrdiff_t* (page **??**)
SIZE_MAX	Maximum value of *size_t* (page **??**)
INTPTR_MIN	Minimum value of *intptr_t* (page **??**)
INTPTR_MAX	Maximum value of *intptr_t* (page **??**)
UINTPTR_MAX	Maximum value of *uintptr_t* (page **??**)
PTRDIFF_MIN	Minimum value of *ptrdiff_t* (page **??**)
PTRDIFF_MAX	Maximum value of *ptrdiff_t* (page **??**)
SIZE_MAX	Maximum value of *size_t* (page **??**)
INTPTR_MIN	Minimum value of *intptr_t* (page **??**)
INTPTR_MAX	Maximum value of *intptr_t* (page **??**)
UINTPTR_MAX	Maximum value of *uintptr_t* (page **??**)

Wide integer minima and maxima

Description

Minimum and maximum values for the wint_t type.

Definition

```
#define WINT_MIN     (-2147483647L-1)
#define WINT_MAX     2147483647L
```

Symbols

Definition	Description
WINT_MIN	Minimum value of wint_t
WINT_MAX	Maximum value of wint_t

Constant construction macros

Signed integer construction macros

Description

Macros that create constants of type intx_t.

Definition

```
#define INT8_C(x)      (x)
#define INT16_C(x)     (x)
#define INT32_C(x)     (x)
#define INT64_C(x)     (x##LL)
```

Symbols

Definition	Description
INT8_C(x)	Create constant of type *int8_t* (page **??**)
INT16_C(x)	Create constant of type *int16_t* (page **??**)
INT32_C(x)	Create constant of type *int32_t* (page **??**)
INT64_C(x)	Create constant of type *int64_t* (page **??**)

Unsigned integer construction macros

Description

Macros that create constants of type uintx_t.

Definition

```
#define UINT8_C(x)      (x##u)
#define UINT16_C(x)     (x##u)
#define UINT32_C(x)     (x##u)
#define UINT64_C(x)     (x##uLL)
```

Symbols

Definition	Description
UINT8_C(x)	Create constant of type *uint8_t* (page **??**)
UINT16_C(x)	Create constant of type *uint16_t* (page **??**)
UINT32_C(x)	Create constant of type *uint32_t* (page **??**)
UINT64_C(x)	Create constant of type *uint64_t* (page **??**)

Maximal integer construction macros

Description

Macros that create constants of type *intmax_t* (page **??**) and uintmax_t.

Definition

```
#define INTMAX_C(x)      (x##LL)
#define UINTMAX_C(x)     (x##uLL)
```

Symbols

Definition	Description
INTMAX_C(x)	Create constant of type *intmax_t* (page **??**)
UINTMAX_C(x)	Create constant of type *uintmax_t* (page **??**)

4.6.15 <stdio.h>

Formatted output control strings

The functions in this section that accept a formatted output control string do so according to the specification that follows.

Composition

The format is composed of zero or more directives: ordinary characters (not %, which are copied unchanged to the output stream; and conversion specifications, each of which results in fetching zero or more subsequent arguments, converting them, if applicable, according to the corresponding conversion specifier, and then writing the result to the output stream.

Each conversion specification is introduced by the character %. After the % the following appear in sequence:

- Zero or more flags (in any order) that modify the meaning of the conversion specification.
- An optional minimum field width. If the converted value has fewer characters than the field width, it is padded with spaces (by default) on the left (or right, if the left adjustment flag has been given) to the field width. The field width takes the form of an asterisk * or a decimal integer.
- An optional precision that gives the minimum number of digits to appear for the d, i, o, u, x, and X conversions, the number of digits to appear after the decimal-point character for e, E, f, and F conversions, the maximum number of significant digits for the g and G conversions, or the maximum number of bytes to be written for s conversions. The precision takes the form of a period . followed either by an asterisk * or by an optional decimal integer; if only the period is specified, the precision is taken as zero. If a precision appears with any other conversion specifier, the behavior is undefined.
- An optional length modifier that specifies the size of the argument.
- A conversion specifier character that specifies the type of conversion to be applied.

As noted above, a field width, or precision, or both, may be indicated by an asterisk. In this case, an int argument supplies the field width or precision. The arguments specifying field width, or precision, or both, must appear (in that order) before the argument (if any) to be converted. A negative field width argument is taken as a - flag followed by a positive field width. A negative precision argument is taken as if the precision were omitted.

Flag characters

The flag characters and their meanings are:

Flag	Description
•	The result of the conversion is left-justified within the field. The default, if this flag is not specified, is that the result of the conversion is left-justified within the field.
•	The result of a signed conversion always begins with a plus or minus sign. The default, if this flag is not specified, is that it begins with a sign only when a negative value is converted.
space	If the first character of a signed conversion is not a sign, or if a signed conversion results in no characters, a space is prefixed to the result. If the space and + flags both appear, the space flag is ignored.
#	The result is converted to an alternative form. For o conversion, it increases the precision, if and only if necessary, to force the first digit of the result to be a zero (if the value and precision are both zero, a single 0 is printed). For x or X conversion, a nonzero result has 0x or 0X prefixed to it. For e, E, f, F, g, and G conversions, the result of converting a floating-point number always contains a decimal-point character, even if no digits follow it. (Normally, a decimal-point character appears in the result of these conversions only if a digit follows it.) For g and F conversions, trailing zeros are not removed from the result. As an extension, when used in p conversion, the results has # prefixed to it. For other conversions, the behavior is undefined.
0	For d, i, o, u, x, X, e, E, f, F, g, and G conversions, leading zeros (following any indication of sign or base) are used to pad to the field width rather than performing space padding, except when converting an infinity or NaN. If the 0 and - flags both appear, the 0 flag is ignored. For d, i, o, u, x, and X conversions, if a precision is specified, the 0 flag is ignored. For other conversions, the behavior is undefined.

Length modifiers

The length modifiers and their meanings are:

Flag	Description
hh	Specifies that a following d, i, o, u, x, or X conversion specifier applies to a signed char or unsigned char argument (the argument will have been promoted according to the integer promotions, but its value will be converted to signed char or unsigned char before printing); or that a following n conversion specifier applies to a pointer to a signed char argument.
h	Specifies that a following d, i, o, u, x, or X conversion specifier applies to a short int or unsigned short int argument (the argument will have been promoted according to the integer promotions, but its value is converted to short int or unsigned short int before printing); or that a following n conversion specifier applies to a pointer to a short int argument.
l	Specifies that a following d, i, o, u, x, or X conversion specifier applies to a long int or unsigned long int argument; that a following n conversion specifier applies to a pointer to a long int argument; or has no effect on a following e, E, f, F, g, or G conversion specifier.
ll	Specifies that a following d, i, o, u, x, or X conversion specifier applies to a long long int or unsigned long long int argument; that a following n conversion specifier applies to a pointer to a long long int argument.

If a length modifier appears with any conversion specifier other than as specified above, the behavior is undefined.

Conversion specifiers

The conversion specifiers and their meanings are:

Flag	Description
d, i	The argument is converted to signed decimal in the style [-]dddd. The precision specifies the minimum number of digits to appear; if the value being converted can be represented in fewer digits, it is expanded with leading spaces. The default precision is one. The result of converting a zero value with a precision of zero is no characters.
o, u, x, X	The unsigned argument is converted to unsigned octal for o, unsigned decimal for u, or unsigned hexadecimal notation for x or X in the style dddd the letters abcdef are used for x conversion and the letters ABCDEF for X conversion. The precision specifies the minimum number of digits to appear; if the value being converted can be represented in fewer digits, it is expanded with leading spaces. The default precision is one. The result of converting a zero value with a precision of zero is no characters.
f, F	A double argument representing a floating-point number is converted to decimal notation in the style [-]ddd.ddd, where the number of digits after the decimal-point character is equal to the precision specification. If the precision is missing, it is taken as 6; if the precision is zero and the # flag is not specified, no decimal-point character appears. If a decimal-point character appears, at least one digit appears before it. The value is rounded to the appropriate number of digits. A double argument representing an infinity is converted to inf. A double argument representing a NaN is converted to nan. The F conversion specifier produces INF or NAN instead of inf or nan, respectively.
e, E	A double argument representing a floating-point number is converted in the style [-]d.ddde±dd, where there is one digit (which is nonzero if the argument is nonzero) before the decimal-point character and the number of digits after it is equal to the precision; if the precision is missing, it is taken as 6; if the precision is zero and the # flag is not specified, no decimal-point character appears. The value is rounded to the appropriate number of digits. The E conversion specifier produces a number with E instead of e introducing the exponent. The exponent always contains at least two digits, and only as many more digits as necessary to represent the exponent. If the value is zero, the exponent is zero. A double argument representing an infinity is converted to inf. A double argument representing a NaN is converted to nan. The E conversion specifier produces INF or NAN instead of inf or nan, respectively.
g, G	A double argument representing a floating-point number is converted in style f or e (or in style F or e in the case of a G conversion specifier), with the precision specifying the number of significant digits. If the precision is zero, it is taken as one. The style used depends on the value converted; style e (or E) is used only if the exponent resulting from such a conversion is less than -4 or greater than or equal to the precision. Trailing zeros are removed from the fractional portion of the result unless the # flag is specified; a decimal-point character appears only if it is followed by a digit. A double argument representing an infinity is converted to inf. A double argument representing a NaN is converted to nan. The G conversion specifier produces INF or NAN instead of inf or nan, respectively.
c	The argument is converted to an unsigned char, and the resulting character is written.
s	The argument is be a pointer to the initial element of an array of character type. Characters from the array are written up to (but not including) the terminating null character. If the precision is specified, no more than that many characters are written. If the precision is not specified or is greater than the size of the array, the array must contain a null character.
p	The argument is a pointer to void. The value of the pointer is converted in the same format as the x conversion specifier with a fixed precision of 2*sizeof(void *).
n	The argument is a pointer to a signed integer into which is written the number of characters written to the output stream so far by the call to the formatting function. No argument is converted, but one is consumed. If the conversion specification includes any flags, a field width, or a precision, the behavior is undefined.
%	A % character is written. No argument is converted.

Note that the C99 width modifier l used in conjunction with the c and s conversion specifiers is not supported and nor are the conversion specifiers a and A.

Formatted input control strings

The format is composed of zero or more directives: one or more white-space characters, an ordinary character (neither % nor a white-space character), or a conversion specification.

Each conversion specification is introduced by the character %. After the %, the following appear in sequence:

- An optional assignment-suppressing character *.
- An optional nonzero decimal integer that specifies the maximum field width (in characters).
- An optional length modifier that specifies the size of the receiving object.
- A conversion specifier character that specifies the type of conversion to be applied.

The formatted input function executes each directive of the format in turn. If a directive fails, the function returns. Failures are described as input failures (because of the occurrence of an encoding error or the unavailability of input characters), or matching failures (because of inappropriate input).

A directive composed of white-space character(s) is executed by reading input up to the first non-white-space character (which remains unread), or until no more characters can be read.

A directive that is an ordinary character is executed by reading the next characters of the stream. If any of those characters differ from the ones composing the directive, the directive fails and the differing and subsequent characters remain unread. Similarly, if end-of-file, an encoding error, or a read error prevents a character from being read, the directive fails.

A directive that is a conversion specification defines a set of matching input sequences, as described below for each specifier. A conversion specification is executed in the following steps:

- Input white-space characters (as specified by the *isspace()* (page 367) function) are skipped, unless the specification includes a [, c, or n specifier.
- An input item is read from the stream, unless the specification includes an n specifier. An input item is defined as the longest sequence of input characters which does not exceed any specified field width and which is, or is a prefix of, a matching input sequence. The first character, if any, after the input item remains unread. If the length of the input item is zero, the execution of the directive fails; this condition is a matching failure unless end-of-file, an encoding error, or a read error prevented input from the stream, in which case it is an input failure.
- Except in the case of a % specifier, the input item (or, in the case of a %n directive, the count of input characters) is converted to a type appropriate to the conversion specifier. If the input item is not a matching sequence, the execution of the directive fails: this condition is a matching failure. Unless assignment suppression was indicated by a *, the result of the conversion is placed in the object pointed to by the first argument following the format argument that has not already received a conversion result. If this object does not have an appropriate type, or if the result of the conversion cannot be represented in the object, the behavior is undefined.

Length modifiers

The length modifiers and their meanings are:

Flag	Description
hh	Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to signed char or pointer to unsigned char.
h	Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to short int or unsigned short int.
l	Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to long int or unsigned long int; that a following e, E, f, F, g, or G conversion specifier applies to an argument with type pointer to double.
ll	Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to long long int or unsigned long long int.

If a length modifier appears with any conversion specifier other than as specified above, the behavior is undefined. Note that the C99 length modifiers j, z, and t are not supported.

Conversion specifiers

Flag	Description
d	Matches an optionally signed decimal integer, whose format is the same as expected for the subject sequence of the *strtol()* (page 513) function with the value 10 for the base argument. The corresponding argument must be a pointer to signed integer.
i	Matches an optionally signed integer, whose format is the same as expected for the subject sequence of the *strtol()* (page 513) function with the value zero for the base argument. The corresponding argument must be a pointer to signed integer.
o	Matches an optionally signed octal integer, whose format is the same as expected for the subject sequence of the *strtol()* (page 513) function with the value 18 for the base argument. The corresponding argument must be a pointer to signed integer.
u	Matches an optionally signed decimal integer, whose format is the same as expected for the subject sequence of the *strtoul()* (page 515) function with the value 10 for the base argument. The corresponding argument must be a pointer to unsigned integer.
x	Matches an optionally signed hexadecimal integer, whose format is the same as expected for the subject sequence of the *strtoul()* (page 515) function with the value 16 for the base argument. The corresponding argument must be a pointer to unsigned integer.
e, f, g	Matches an optionally signed floating-point number whose format is the same as expected for the subject sequence of the *strtod()* (page 517) function. The corresponding argument shall be a pointer to floating.
c	Matches a sequence of characters of exactly the number specified by the field width (one if no field width is present in the directive). The corresponding argument must be a pointer to the initial element of a character array large enough to accept the sequence. No null character is added.
s	Matches a sequence of non-white-space characters The corresponding argument must be a pointer to the initial element of a character array large enough to accept the sequence and a terminating null character, which will be added automatically.
[	Matches a nonempty sequence of characters from a set of expected characters (the scanset). The corresponding argument must be a pointer to the initial element of a character array large enough to accept the sequence and a terminating null character, which will be added automatically. The conversion specifier includes all subsequent characters in the format string, up to and including the matching right bracket]. The characters between the brackets (the scanlist) compose the scanset, unless the character after the left bracket is a circumflex ^, in which case the scanset contains all characters that do not appear in the scanlist between the circumflex and the right bracket. If the conversion specifier begins with [] or[^], the right bracket character is in the scanlist and the next following right bracket character is the matching right bracket that ends the specification; otherwise the first following right bracket character is the one that ends the specification. If a - character is in the scanlist and is not the first, nor the second where the first character is a ^, nor the last character, it is treated as a member of the scanset.
p	Reads a sequence output by the corresponding %p formatted output conversion. The corresponding argument must be a pointer to a pointer to void.
n	No input is consumed. The corresponding argument shall be a pointer to signed integer into which is to be written the number of characters read from the input stream so far by this call to the formatted input function. Execution of a %n directive does not increment the assignment count returned at the completion of execution of the fscanf function. No argument is converted, but one is consumed. If the conversion specification includes an assignment-suppressing character or a field width, the behavior is undefined.
%	Matches a single % character; no conversion or assignment occurs.

Note that the C99 width modifier l used in conjunction with the c, s, and [conversion specifiers is not supported and nor are the conversion specifiers a and A.

Character and string I/O functions

Function	Description
getchar() (page 490)	Read character from standard input.
gets() (page 490)	Read string from standard input.
putc() (page 491)	Write character to file.
putchar() (page 491)	Write character to standard output.
puts() (page 491)	Write string to standard output.

getchar()

Description

Read character from standard input.

Prototype

```
int getchar(void);
```

Return value

If the stream is at end-of-file or a read error occurs, returns EOF, otherwise a nonnegative value.

Additional information

Reads a single character from the standard input stream.

gets()

Description

Read string from standard input.

Prototype

```
char *gets(char * s);
```

Parameters

Parameter	Description
s	Pointer to object that receives the string.

Return value

Returns s if successful. If end-of-file is encountered and no characters have been read into the array, the contents of the array remain unchanged and a null pointer is returned. If a read error occurs during the operation, the array contents are indeterminate and a null null pointer is return.

Additional information

This function reads characters from standard input into the array pointed to by s until end-of-file is encountered or a new-line character is read. Any new-line character is discarded, and a null character is written immediately after the last character read into the array.

putc()

Description

Write character to file.

Prototype

```
int putc(int     c,
         FILE * stream);
```

Parameters

Parameter	Description
c	Character to write.
stream	Pointer to stream to write to.

Return value

If no error, the character written. If a write error occurs, returns EOF.

Additional information

Writes the character c to stream.

putchar()

Description

Write character to standard output.

Prototype

```
int putchar(int c);
```

Parameters

Parameter	Description
c	Character to write.

Return value

If no error, the character written. If a write error occurs, returns EOF.

Additional information

Writes the character c to the standard output stream.

puts()

Description

Write string to standard output.

Prototype

```
int puts(const char * s);
```

Parameters

Parameter	Description
s	Pointer to zero-terminated string.

Return value

Returns EOF if a write error occurs; otherwise it returns a nonnegative value.

Additional information

Writes the string pointed to by s to the standard output stream using *putchar()* (page 491) and appends a new-line character to the output. The terminating null character is not written.

Formatted input functions

Function	Description
scanf() (page 492)	Formatted read from standard input.
sscanf() (page 493)	Formatted read from string.
vscanf() (page 493)	Formatted read from standard input, variadic.
vsscanf() (page 494)	Formatted read from string, variadic.

scanf()

Description

Formatted read from standard input.

Prototype

```
int scanf(const char * format,
                       ...);
```

Parameters

Parameter	Description
format	Pointer to zero-terminated format control string.

Return value

Returns the value of the macro EOF if an input failure occurs before any conversion. Otherwise, returns the number of input items assigned, which can be fewer than provided for, or even zero, in the event of an early matching failure.

Additional information

Reads input from the standard input stream under control of the string pointed to by format that specifies the admissible input sequences and how they are to be converted for assignment, using subsequent arguments as pointers to the objects to receive the converted input.

If there are insufficient arguments for the format, the behavior is undefined. If the format is exhausted while arguments remain, the excess arguments are evaluated but are otherwise ignored.

sscanf()

Description

Formatted read from string.

Prototype

```
int sscanf(const char * s,
           const char * format,
                        ...);
```

Parameters

Parameter	Description
s	Pointer to string to read from.
format	Pointer to zero-terminated format control string.

Return value

Returns the value of the macro EOF if an input failure occurs before any conversion. Otherwise, returns the number of input items assigned, which can be fewer than provided for, or even zero, in the event of an early matching failure.

Additional information

Reads input from the string s under control of the string pointed to by format that specifies the admissible input sequences and how they are to be converted for assignment, using subsequent arguments as pointers to the objects to receive the converted input.

If there are insufficient arguments for the format, the behavior is undefined. If the format is exhausted while arguments remain, the excess arguments are evaluated but are otherwise ignored.

vscanf()

Description

Formatted read from standard input, variadic.

Prototype

```
int vscanf(const char    * format,
                 va_list   arg);
```

Parameters

Parameter	Description
format	Pointer to zero-terminated format control string.
arg	Variable parameter list.

Return value

Returns the value of the macro EOF if an input failure occurs before any conversion. Otherwise, returns the number of input items assigned, which can be fewer than provided for, or even zero, in the event of an early matching failure.

Additional information

Reads input from the standard input stream under control of the string pointed to by format that specifies the admissible input sequences and how they are to be converted for assignment, using subsequent arguments as pointers to the objects to receive the converted input. Before calling *vscanf()* (page 493), arg must be initialized by the va_start() macro (and possibly subsequent va_arg() calls). *vscanf()* (page 493) does not invoke the va_end() macro.

If there are insufficient arguments for the format, the behavior is undefined.

vsscanf()

Description

Formatted read from string, variadic.

Prototype

```
int vsscanf(const char     * s,
            const char     * format,
                  va_list    arg);
```

Parameters

Parameter	Description
s	Pointer to string to read from.
format	Pointer to zero-terminated format control string.
arg	Variable parameter list.

Return value

Returns the value of the macro EOF if an input failure occurs before any conversion. Otherwise, returns the number of input items assigned, which can be fewer than provided for, or even zero, in the event of an early matching failure.

Additional information

Reads input from the standard input stream under control of the string pointed to by format that specifies the admissible input sequences and how they are to be converted for assignment, using subsequent arguments as pointers to the objects to receive the converted input. Before calling *vsscanf()* (page 494), arg must be initialized by the va_start() macro (and possibly subsequent va_arg() calls). *vsscanf()* (page 494) does not invoke the va_end() macro.

If there are insufficient arguments for the format, the behavior is undefined.

Formatted output functions

Function	Description
printf() (page 495)	Formatted write to standard output.
sprintf() (page 495)	Formatted write to string.
snprintf() (page 496)	Formatted write to string, limit length.
vprintf() (page 496)	Formatted write to standard output, variadic.
vsprintf() (page 497)	Formatted write to string, variadic.
vsnprintf() (page 497)	Formatted write to string, limit length, variadic.

printf()

Description

Formatted write to standard output.

Prototype

```
int printf(const char * format,
                        ...);
```

Parameters

Parameter	Description
format	Pointer to zero-terminated format control string.

Return value

Returns the number of characters written, or a negative value if an output or encoding error occurred.

Additional information

Writes to the standard output stream under control of the string pointed to by format that specifies how subsequent arguments are converted for output.

If there are insufficient arguments for the format, the behavior is undefined. If the format is exhausted while arguments remain, the excess arguments are evaluated but are otherwise ignored.

sprintf()

Description

Formatted write to string.

Prototype

```
int sprintf(      char * s,
            const char * format,
                         ...);
```

Parameters

Parameter	Description
s	Pointer to array that receives the formatted output.
format	Pointer to zero-terminated format control string.

Return value

Returns number of characters written to s (not counting the terminating null), or a negative value if an output or encoding error occurred.

Additional information

Writes to the string pointed to by s under control of the string pointed to by format that specifies how subsequent arguments are converted for output. A null character is written at the end of the characters written; it is not counted as part of the returned value.

If there are insufficient arguments for the format, the behavior is undefined. If the format is exhausted while arguments remain, the excess arguments are evaluated but are otherwise ignored.

If copying takes place between objects that overlap, the behavior is undefined.

snprintf()

Description

Formatted write to string, limit length.

Prototype

```
int snprintf(       char    * s,
                    size_t    n,
              const char    * format,
                              ...);
```

Parameters

Parameter	Description
s	Pointer to array that receives the formatted output.
n	Maximum number of characters to write to the array pointed to by s.
format	Pointer to zero-terminated format control string.

Return value

Returns the number of characters that would have been written had n been sufficiently large, not counting the terminating null character, or a negative value if an encoding error occurred. Thus, the null-terminated output has been completely written if and only if the returned value is nonnegative and less than n.

Additional information

Writes to the string pointed to by s under control of the string pointed to by format that specifies how subsequent arguments are converted for output.

If n is zero, nothing is written, and s can be a null pointer. Otherwise, output characters beyond count n-1 are discarded rather than being written to the array, and a null character is written at the end of the characters actually written into the array. A null character is written at the end of the conversion; it is not counted as part of the returned value.

If there are insufficient arguments for the format, the behavior is undefined. If the format is exhausted while arguments remain, the excess arguments are evaluated but are otherwise ignored.

If copying takes place between objects that overlap, the behavior is undefined.

vprintf()

Description

Formatted write to standard output, variadic.

Prototype

```
int vprintf(const char    * format,
                  va_list   arg);
```

Parameters

Parameter	Description
format	Pointer to zero-terminated format control string.
arg	Variable parameter list.

Return value

Returns the number of characters written, or a negative value if an output or encoding error occurred.

Additional information

Writes to the standard output stream using under control of the string pointed to by format that specifies how subsequent arguments are converted for output. Before calling *vprintf()* (page 496), arg must be initialized by the va_start macro (and possibly subsequent va_arg calls). *vprintf()* (page 496) does not invoke the va_end macro.

vsprintf()

Description

Formatted write to string, variadic.

Prototype

```
int vsprintf(       char    * s,
              const char    * format,
                    va_list   arg);
```

Parameters

Parameter	Description
s	Pointer to array that receives the formatted output.
format	Pointer to zero-terminated format control string.
arg	Variable parameter list.

Return value

Returns number of characters written to s (not counting the terminating null), or a negative value if an output or encoding error occurred.

Additional information

Writes to the string pointed to by s under control of the string pointed to by format that specifies how subsequent arguments are converted for output. A null character is written at the end of the characters written; it is not counted as part of the returned value.

Before calling *vsprintf()* (page 497), arg must be initialized by the va_start macro (and possibly subsequent va_arg calls). *vsprintf()* (page 497) does not invoke the va_end macro.

If there are insufficient arguments for the format, the behavior is undefined. If the format is exhausted while arguments remain, the excess arguments are evaluated but are otherwise ignored.

If copying takes place between objects that overlap, the behavior is undefined.

Notes

This is equivalent to *sprintf()* (page 495) with the variable argument list replaced by arg.

vsnprintf()

Description

Formatted write to string, limit length, variadic.

Prototype

```
int vsnprintf(       char    * s,
                     size_t    n,
               const char    * format,
                     va_list   arg);
```

Parameters

Parameter	Description
s	Pointer to array that receives the formatted output.
n	Maximum number of characters to write to the array pointed to by s.
format	Pointer to zero-terminated format control string.
arg	Variable parameter list.

Return value

Returns the number of characters that would have been written had n been sufficiently large, not counting the terminating null character, or a negative value if an encoding error occurred. Thus, the null-terminated output has been completely written if and only if the returned value is nonnegative and less than n.

Additional information

Writes to the string pointed to by s under control of the string pointed to by format that specifies how subsequent arguments are converted for output. Before calling *vsnprintf()* (page 497), arg must be initialized by the va_start macro (and possibly subsequent va_arg() calls). *vsnprintf()* (page 497) does not invoke the va_end macro.

If n is zero, nothing is written, and s can be a null pointer. Otherwise, output characters beyond count n-1 are discarded rather than being written to the array, and a null character is written at the end of the characters actually written into the array. A null character is written at the end of the conversion; it is not counted as part of the returned value.

If there are insufficient arguments for the format, the behavior is undefined. If the format is exhausted while arguments remain, the excess arguments are evaluated but are otherwise ignored.

If copying takes place between objects that overlap, the behavior is undefined.

Notes

This is equivalent to *snprintf()* (page 496) with the variable argument list replaced by arg.

4.6.16 <stdlib.h>

Process control functions

Function	Description
atexit() (page 498)	Set function to be called on exit.
abort() (page 499)	Abort execution.

atexit()

Description

Set function to be called on exit.

Prototype

```
int atexit(__SEGGER_RTL_exit_func fn);
```

Parameters

Parameter	Description
fn	Function to register.

Return value

$= 0$	Success registering function.
$\neq 0$	Did not register function.

Additional information

Registers function fn to be called when the application has exited. The functions registered with *atexit()* (page 498) are executed in reverse order of their registration.

abort()

Description

Abort execution.

Prototype

```
void abort(void);
```

Additional information

Calls exit() with the exit status -1.

Integer arithmetic functions

Function	Description
abs() (page 499)	Calculate absolute value, int.
labs() (page 500)	Calculate absolute value, long.
llabs() (page 500)	Calculate absolute value, long long.
div() (page 500)	Divide returning quotient and remainder, int.
ldiv() (page 501)	Divide returning quotient and remainder, long.
lldiv() (page 501)	Divide returning quotient and remainder, long long.

abs()

Description

Calculate absolute value, int.

Prototype

```
int abs(int Value);
```

Parameters

Parameter	Description
Value	Integer value.

Return value

The absolute value of the integer argument Value.

labs()

Description

Calculate absolute value, long.

Prototype

```
long int labs(long int Value);
```

Parameters

Parameter	Description
Value	Long integer value.

Return value

The absolute value of the long integer argument Value.

llabs()

Description

Calculate absolute value, long long.

Prototype

```
long long int llabs(long long int Value);
```

Parameters

Parameter	Description
Value	Long long integer value.

Return value

The absolute value of the long long integer argument Value.

div()

Description

Divide returning quotient and remainder, int.

Prototype

```
div_t div(int Numer,
          int Denom);
```

Parameters

Parameter	Description
Numer	Numerator.
Denom	Demoninator.

Return value

Returns a structure of type div_t comprising both the quotient and the remainder. The structures contain the members quot (the quotient) and rem (the remainder), each of which has the same type as the arguments Numer and Denom. If either part of the result cannot be represented, the behavior is undefined.

Additional information

This computes Numer divided by Denom and Numer modulo Denom in a single operation.

See also

div_t

ldiv()

Description

Divide returning quotient and remainder, long.

Prototype

```
ldiv_t ldiv(long Numer,
            long Denom);
```

Parameters

Parameter	Description
Numer	Numerator.
Denom	Demoninator.

Return value

Returns a structure of type ldiv_t comprising both the quotient and the remainder. The structures contain the members quot (the quotient) and rem (the remainder), each of which has the same type as the arguments Numer and Denom. If either part of the result cannot be represented, the behavior is undefined.

Additional information

This computes Numer divided by Denom and Numer modulo Denom in a single operation.

See also

ldiv_t

lldiv()

Description

Divide returning quotient and remainder, long long.

Prototype

```
lldiv_t lldiv(long long Numer,
              long long Denom);
```

Parameters

Parameter	Description
Numer	Numerator.
Denom	Demoninator.

Return value

Returns a structure of type lldiv_t comprising both the quotient and the remainder. The structures contain the members quot (the quotient) and rem (the remainder), each of which has the same type as the arguments Numer and Denom. If either part of the result cannot be represented, the behavior is undefined.

Additional information

This computes Numer divided by Denom and Numer modulo Denom in a single operation.

See also

lldiv_t

Pseudo-random sequence generation functions

Function	Description
rand() (page 502)	Return next random number in sequence.
srand() (page 502)	Set seed of random number sequence.

rand()

Description

Return next random number in sequence.

Prototype

```
int rand(void);
```

Return value

Returns the computed pseudo-random integer.

Additional information

This computes a sequence of pseudo-random integers in the range 0 to RAND_MAX.

See also

srand() (page 502)

srand()

Description

Set seed of random number sequence.

Prototype

```
void srand(unsigned s);
```

Parameters

Parameter	Description
s	New seed value for pseudo-random sequence.

Additional information

This uses the argument Seed as a seed for a new sequence of pseudo-random numbers to be returned by subsequent calls to *rand()* (page 502). If *srand()* (page 502) is called with the same seed value, the same sequence of pseudo-random numbers is generated.

If *rand()* (page 502) is called before any calls to *srand()* (page 502) have been made, a sequence is generated as if *srand()* (page 502) is first called with a seed value of 1.

See also

rand() (page 502)

Memory allocation functions

Function	Description
malloc() (page 503)	Allocate space for single object.
calloc() (page 503)	Allocate space for multiple objects and zero them.
realloc() (page 504)	Resize or allocate memory space.
free() (page 504)	Free allocated memory for reuse.

malloc()

Description

Allocate space for single object.

Prototype

```
void *malloc(size_t sz);
```

Parameters

Parameter	Description
sz	Number of characters to allocate for the object.

Return value

Returns a null pointer if the space for the object cannot be allocated from free memory; if space for the object can be allocated, *malloc()* (page 503) returns a pointer to the start of the allocated space.

Additional information

Allocates space for an object whose size is specified by sz and whose value is indeterminate.

calloc()

Description

Allocate space for multiple objects and zero them.

Prototype

```
void *calloc(size_t nobj,
             size_t sz);
```

Parameters

Parameter	Description
nobj	Number of objects to allocate.
sz	Number of characters to allocate per object.

Return value

Returns a null pointer if the space for the object cannot be allocated from free memory; if space for the object can be allocated, *malloc()* (page 503) returns a pointer to the start of the allocated space.

Additional information

Allocates space for an array of nobj objects, each of whose size is sz. The space is initialized to all zero bits.

realloc()

Description

Resize or allocate memory space.

Prototype

```
void *realloc(void   * ptr,
              size_t   sz);
```

Parameters

Parameter	Description
ptr	Pointer to resize, or NULL to allocate.
sz	New size of object.

Return value

Returns a pointer to the new object (which may have the same value as a pointer to the old object), or a null pointer if the new object could not be allocated.

Additional information

Deallocates the old object pointed to by ptr and returns a pointer to a new object that has the size specified by sz. The contents of the new object is identical to that of the old object prior to deallocation, up to the lesser of the new and old sizes. Any bytes in the new object beyond the size of the old object have indeterminate values.

If ptr is a null pointer, *realloc()* (page 504) behaves like *malloc()* (page 503) for the specified size. If memory for the new object cannot be allocated, the old object is not deallocated and its value is unchanged.

If ptr does not match a pointer earlier returned by *calloc()* (page 503), *malloc()* (page 503), or *realloc()* (page 504), or if the space has been deallocated by a call to *free()* (page 504) or *realloc()* (page 504), the behavior is undefined.

free()

Description

Free allocated memory for reuse.

Prototype

```
void free(void * ptr);
```

Parameters

Parameter	Description
ptr	Pointer to object to free.

Additional information

Causes the space pointed to by ptr to be deallocated, that is, made available for further allocation. If ptr is a null pointer, no action occurs.

If ptr does not match a pointer earlier returned by *calloc()* (page 503), *malloc()* (page 503), or *realloc()* (page 504), or if the space has been deallocated by a call to *free()* (page 504) or *realloc()* (page 504), the behavior is undefined.

Search and sort functions

Function	Description
qsort() (page 505)	Sort array.
bsearch() (page 505)	Search sorted array.

qsort()

Description

Sort array.

Prototype

```
void qsort(void   * base,
           size_t   nmemb,
           size_t   sz,
           int      ( *compare)(const void * elem1 , const void * elem2 ));
```

Parameters

Parameter	Description
base	Pointer to the start of the array.
nmemb	Number of array elements.
sz	Number of characters per array element.
compare	Pointer to element comparison function.

Additional information

Sorts the array pointed to by base using the compare function. The array should have nmemb elements of sz bytes. The compare function should return a negative value if the first parameter is less than the second parameter, zero if the parameters are equal, and a positive value if the first parameter is greater than the second parameter.

bsearch()

Description

Search sorted array.

Prototype

```
void *bsearch
            (const void   * key,
             const void   * base,
                   size_t   nmemb,
                   size_t   sz,
                   int      ( *compare)(const void * elem1 , const void * elem2 ));
```

Parameters

Parameter	Description
key	Pointer to object to search for.
base	Pointer to the start of the array.
nmemb	Number of array elements.
sz	Number of characters per array element.
compare	Pointer to element comparison function.

Return value

= NULL	Key not found.
≠ NULL	Pointer to found object.

Additional information

Searches the array pointed to by base for the specified key and returns a pointer to the first entry that matches, or null if no match. The array should have nmemb elements of sz bytes and be sorted by the same algorithm as the compare function.

The compare function should return a negative value if the first parameter is less than second parameter, zero if the parameters are equal, and a positive value if the first parameter is greater than the second parameter.

Number to string conversions

Function	Description
itoa() (page 506)	Convert to string, int.
ltoa() (page 507)	Convert to string, long.
lltoa() (page 508)	Convert to string, long long.
utoa() (page 508)	Convert to string, unsigned.
ultoa() (page 509)	Convert to string, unsigned long.
ulltoa() (page 510)	Convert to string, unsigned long long.

itoa()

Description

Convert to string, int.

Prototype

```
char *itoa(int     val,
           char *  buf,
           int     radix);
```

Parameters

Parameter	Description
val	Value to convert.
buf	Pointer to array of characters that receives the string.
radix	Number base to use for conversion, 2 to 36.

Return value

Returns buf.

Additional information

Converts val to a string in base radix and places the result in buf which must be large enough to hold the output. If radix is greater than 36, the result is undefined.

If val is negative and radix is 10, the string has a leading minus sign (-); for all other values of radix, value is considered unsigned and never has a leading minus sign.

Notes

This is a non-standard function. Even though this function is commonly used by compilers on other platforms, there is no guarantee that this function will behave the same on all platforms, in all cases.

See also

ltoa() (page 507), *lltoa()* (page 508), *utoa()* (page 508), *ultoa()* (page 509), *ulltoa()* (page 510)

ltoa()

Description

Convert to string, long.

Prototype

```
char *ltoa(long   val,
           char * buf,
           int    radix);
```

Parameters

Parameter	Description
val	Value to convert.
buf	Pointer to array of characters that receives the string.
radix	Number base to use for conversion, 2 to 36.

Return value

Returns buf.

Additional information

Converts val to a string in base radix and places the result in buf which must be large enough to hold the output. If radix is greater than 36, the result is undefined.

If val is negative and radix is 10, the string has a leading minus sign (-); for all other values of radix, value is considered unsigned and never has a leading minus sign.

Notes

This is a non-standard function. Even though this function is commonly used by compilers on other platforms, there is no guarantee that this function will behave the same on all platforms, in all cases.

See also

itoa() (page 506), *lltoa()* (page 508), *utoa()* (page 508), *ultoa()* (page 509), *ulltoa()* (page 510)

lltoa()

Description

Convert to string, long long.

Prototype

```
char *lltoa(long long   val,
            char      * buf,
            int         radix);
```

Parameters

Parameter	Description
val	Value to convert.
buf	Pointer to array of characters that receives the string.
radix	Number base to use for conversion, 2 to 36.

Return value

Returns buf.

Additional information

Converts val to a string in base radix and places the result in buf which must be large enough to hold the output. If radix is greater than 36, the result is undefined.

If val is negative and radix is 10, the string has a leading minus sign (-); for all other values of radix, value is considered unsigned and never has a leading minus sign.

Notes

This is a non-standard function. Even though this function is commonly used by compilers on other platforms, there is no guarantee that this function will behave the same on all platforms, in all cases.

See also

itoa() (page 506), *ltoa()* (page 507), *utoa()* (page 508), *ultoa()* (page 509), *ulltoa()* (page 510)

utoa()

Description

Convert to string, unsigned.

Prototype

```
char *utoa(unsigned   val,
           char     * buf,
           int        radix);
```

Parameters

Parameter	Description
val	Value to convert.
buf	Pointer to array of characters that receives the string.
radix	Number base to use for conversion, 2 to 36.

Return value

Returns buf.

Additional information

Converts val to a string in base radix and places the result in buf which must be large enough to hold the output. If radix is greater than 36, the result is undefined.

Notes

This is a non-standard function. Even though this function is commonly used by compilers on other platforms, there is no guarantee that this function will behave the same on all platforms, in all cases.

See also

itoa() (page 506), *ltoa()* (page 507), *lltoa()* (page 508), *ultoa()* (page 509), *ulltoa()* (page 510)

ultoa()

Description

Convert to string, unsigned long.

Prototype

```
char *ultoa(unsigned long   val,
            char          * buf,
            int             radix);
```

Parameters

Parameter	Description
val	Value to convert.
buf	Pointer to array of characters that receives the string.
radix	Number base to use for conversion, 2 to 36.

Return value

Returns buf.

Additional information

Converts val to a string in base radix and places the result in buf which must be large enough to hold the output. If radix is greater than 36, the result is undefined.

Notes

This is a non-standard function. Even though this function is commonly used by compilers on other platforms, there is no guarantee that this function will behave the same on all platforms, in all cases.

See also

itoa() (page 506), *ltoa()* (page 507), *lltoa()* (page 508), *ulltoa()* (page 510), *utoa()* (page 508)

ulltoa()

Description

Convert to string, unsigned long long.

Prototype

```
char *ulltoa(unsigned long long   val,
             char               * buf,
             int                  radix);
```

Parameters

Parameter	Description
val	Value to convert.
buf	Pointer to array of characters that receives the string.
radix	Number base to use for conversion, 2 to 36.

Return value

Returns buf.

Additional information

Converts val to a string in base radix and places the result in buf which must be large enough to hold the output. If radix is greater than 36, the result is undefined.

Notes

This is a non-standard function. Even though this function is commonly used by compilers on other platforms, there is no guarantee that this function will behave the same on all platforms, in all cases.

See also

itoa() (page 506), *ltoa()* (page 507), *lltoa()* (page 508), *ultoa()* (page 509), *utoa()* (page 508)

String to number conversions

Function	Description
atoi() (page 511)	Convert to number, int.
atol() (page 511)	Convert to number, long.
atoll() (page 512)	Convert to number, long long.
atof() (page 512)	Convert to number, double.
strtol() (page 513)	Convert to number, long.
strtoll() (page 514)	Convert to number, long long.
strtoul() (page 515)	Convert to number, unsigned long.
strtoull() (page 516)	Convert to number, unsigned long long.
strtof() (page 517)	Convert to number, float.
strtod() (page 517)	Convert to number, double.
strtold() (page 518)	Convert to number, long double.

atoi()

Description

Convert to number, int.

Prototype

```
int atoi(const char * nptr);
```

Parameters

Parameter	Description
nptr	Pointer to string to convert from.

Return value

Returns the converted value, if any. If the value of the result cannot be represented, the behavior is undefined.

Additional information

Converts the initial portion of the string pointed to by nptr to an int representation.

atoi() (page 511) does not affect the value of errno on an error.

Notes

Except for the behavior on error, *atoi()* (page 511) is equivalent to (int)strtol(nptr, NULL, 10).

See also

strtol() (page 513)

atol()

Description

Convert to number, long.

Prototype

```
long int atol(const char * nptr);
```

Parameters

Parameter	Description
nptr	Pointer to string to convert from.

Return value

Returns the converted value, if any. If the value of the result cannot be represented, the behavior is undefined.

Additional information

Converts the initial portion of the string pointed to by nptr to a long representation.

atol() (page 511) does not affect the value of errno on an error.

Notes

Except for the behavior on error, *atol()* (page 511) is equivalent to strtol(nptr, NULL, 10).

See also

strtol() (page 513)

atoll()

Description

Convert to number, long long.

Prototype

```
long long int atoll(const char * nptr);
```

Parameters

Parameter	Description
nptr	Pointer to string to convert from.

Return value

Returns the converted value, if any. If the value of the result cannot be represented, the behavior is undefined.

Additional information

Converts the initial portion of the string pointed to by nptr to a long-long representation.

atoll() (page 512) does not affect the value of errno on an error.

Notes

Except for the behavior on error, *atoll()* (page 512) is equivalent to strtoll(nptr, NULL, 10).

See also

strtoll() (page 514)

atof()

Description

Convert to number, double.

Prototype

```
double atof(const char * nptr);
```

Parameters

Parameter	Description
nptr	Pointer to string to convert from.

Return value

Returns the converted value, if any. If the value of the result cannot be represented, the behavior is undefined.

Additional information

Converts the initial portion of the string pointed to by nptr to an double representation.

atof() (page 512) does not affect the value of errno on an error.

Notes

Except for the behavior on error, *atof()* (page 512) is equivalent to (int)strtod(nptr, NULL).

See also

strtod() (page 517)

strtol()

Description

Convert to number, long.

Prototype

```
long strtol(const char  * nptr,
                  char ** endptr,
                  int     base);
```

Parameters

Parameter	Description
nptr	Pointer to string to convert from.
endptr	If nonnull, a pointer to object that receives the pointer to the first unconverted character.
base	Radix to use for conversion, 2 to 36.

Return value

Returns the converted value, if any. If no conversion could be performed, zero is returned. If the correct value is outside the range of representable values, LONG_MIN or LONG_MAX is returned according to the sign of the value, if any, and the value of the macro ERANGE is stored in errno.

Additional information

Converts the initial portion of the string pointed to by nptr to a long representation.

First, *strtol()* (page 513) decomposes the input string into three parts: an initial, possibly empty, sequence of white-space characters, as specified by *isspace()* (page 367), a subject sequence resembling an integer represented in some radix determined by the value of base, and a final string of one or more unrecognized characters, including the terminating null character of the input string. *strtol()* (page 513) then attempts to convert the subject sequence to an integer, and return the result.

When converting, no integer suffix (such as U, L, UL, LL, ULL) is allowed.

If the value of base is zero, the expected form of the subject sequence is an optional plus or minus sign followed by an integer constant.

If the value of base is between 2 and 36 (inclusive), the expected form of the subject sequence is an optional plus or minus sign followed by a sequence of letters and digits representing an integer with the radix specified by base. The letters from a (or A) through z (or Z) represent the values 10 through 35; only letters and digits whose ascribed values are less than that of base are permitted.

If the value of base is 16, the characters "0x" or "0X" may optionally precede the sequence of letters and digits, following the optional sign.

The subject sequence is defined as the longest initial subsequence of the input string, starting with the first non-white-space character, that is of the expected form. The subject sequence contains no characters if the input string is empty or consists entirely of white space, or if the first non-white-space character is other than a sign or a permissible letter or digit.

If the subject sequence has the expected form and the value of base is zero, the sequence of characters starting with the first digit is interpreted as an integer constant. If the subject sequence has the expected form and the value of base is between 2 and 36, it is used as the base for conversion.

If the subject sequence begins with a minus sign, the value resulting from the conversion is negated.

A pointer to the final string is stored in the object pointed to by endptr, provided that endptr is not a null pointer.

If the subject sequence is empty or does not have the expected form, no conversion is performed, the value of nptr is stored in the object pointed to by endptr, provided that endptr is not a null pointer.

strtoll()

Description

Convert to number, long long.

Prototype

```
long long strtoll(const char  * nptr,
                  char ** endptr,
                  int     base);
```

Parameters

Parameter	Description
nptr	Pointer to string to convert from.
endptr	If nonnull, a pointer to object that receives the pointer to the first unconverted character.
base	Radix to use for conversion, 2 to 36.

Return value

Returns the converted value, if any. If no conversion could be performed, zero is returned. If the correct value is outside the range of representable values, LLONG_MIN or LLONG_MAX is returned according to the sign of the value, if any, and the value of the macro ERANGE is stored in errno.

Additional information

Converts the initial portion of the string pointed to by nptr to a long representation.

First, *strtoll()* (page 514) decomposes the input string into three parts: an initial, possibly empty, sequence of white-space characters, as specified by *isspace()* (page 367), a subject sequence resembling an integer represented in some radix determined by the value of base, and a final string of one or more unrecognized characters, including the terminating null character of the input string. *strtoll()* (page 514) then attempts to convert the subject sequence to an integer, and return the result.

When converting, no integer suffix (such as U, L, UL, LL, ULL) is allowed.

If the value of base is zero, the expected form of the subject sequence is an optional plus or minus sign followed by an integer constant.

If the value of base is between 2 and 36 (inclusive), the expected form of the subject sequence is an optional plus or minus sign followed by a sequence of letters and digits representing an integer with the radix specified by base. The letters from a (or A) through z (or Z) represent the values 10 through 35; only letters and digits whose ascribed values are less than that of base are permitted.

If the value of base is 16, the characters "0x" or "0X" may optionally precede the sequence of letters and digits, following the optional sign.

The subject sequence is defined as the longest initial subsequence of the input string, starting with the first non-white-space character, that is of the expected form. The subject sequence contains no characters if the input string is empty or consists entirely of white space, or if the first non-white-space character is other than a sign or a permissible letter or digit.

If the subject sequence has the expected form and the value of base is zero, the sequence of characters starting with the first digit is interpreted as an integer constant. If the subject sequence has the expected form and the value of base is between 2 and 36, it is used as the base for conversion.

If the subject sequence begins with a minus sign, the value resulting from the conversion is negated.

A pointer to the final string is stored in the object pointed to by endptr, provided that endptr is not a null pointer.

If the subject sequence is empty or does not have the expected form, no conversion is performed, the value of nptr is stored in the object pointed to by endptr, provided that endptr is not a null pointer.

strtoul()

Description

Convert to number, unsigned long.

Prototype

```
unsigned long strtoul(const char  * nptr,
                            char ** endptr,
                            int     base);
```

Parameters

Parameter	Description
nptr	Pointer to string to convert from.
endptr	If nonnull, a pointer to object that receives the pointer to the first unconverted character.
base	Radix to use for conversion, 2 to 36.

Return value

strtoul() (page 515) returns the converted value, if any. If no conversion could be performed, zero is returned. If the correct value is outside the range of representable values, ULONG_MAX is and the value of the macro ERANGE is stored in errno.

Additional information

Converts the initial portion of the string pointed to by nptr to a long int representation.

First, *strtoul()* (page 515) decomposes the input string into three parts: an initial, possibly empty, sequence of white-space characters, as specified by *isspace()* (page 367), a subject sequence resembling an integer represented in some radix determined by the value of base, and a final string of one or more unrecognized characters, including the terminating null character of the input string. *strtoul()* (page 515) then attempts to convert the subject sequence to an integer, and return the result.

When converting, no integer suffix (such as U, L, UL, LL, ULL) is allowed.

If the value of base is zero, the expected form of the subject sequence is an optional plus or minus sign followed by an integer constant.

If the value of base is between 2 and 36 (inclusive), the expected form of the subject sequence is an optional plus or minus sign followed by a sequence of letters and digits representing an integer with the radix specified by base. The letters from a (or A) through z (or Z) represent the values 10 through 35; only letters and digits whose ascribed values are less than that of base are permitted.

If the value of base is 16, the characters "0x" or "0X" may optionally precede the sequence of letters and digits, following the optional sign.

The subject sequence is defined as the longest initial subsequence of the input string, starting with the first non-white-space character, that is of the expected form. The subject sequence contains no characters if the input string is empty or consists entirely of white space, or if the first non-white-space character is other than a sign or a permissible letter or digit.

If the subject sequence has the expected form and the value of base is zero, the sequence of characters starting with the first digit is interpreted as an integer constant. If the subject sequence has the expected form and the value of base is between 2 and 36, it is used as the base for conversion.

If the subject sequence begins with a minus sign, the value resulting from the conversion is negated.

A pointer to the final string is stored in the object pointed to by endptr, provided that endptr is not a null pointer.

If the subject sequence is empty or does not have the expected form, no conversion is performed, the value of nptr is stored in the object pointed to by endptr, provided that endptr is not a null pointer.

strtoull()

Description

Convert to number, unsigned long long.

Prototype

```
unsigned long long strtoull(const char  * nptr,
                            char ** endptr,
                            int      base);
```

Parameters

Parameter	Description
nptr	Pointer to string to convert from.
endptr	If nonnull, a pointer to object that receives the pointer to the first unconverted character.
base	Radix to use for conversion, 2 to 36.

Return value

strtoull() (page 516) returns the converted value, if any. If no conversion could be performed, zero is returned. If the correct value is outside the range of representable values, ULLONG_MAX is and the value of the macro ERANGE is stored in errno.

Additional information

Converts the initial portion of the string pointed to by nptr to a long int representation.

First, *strtoull()* (page 516) decomposes the input string into three parts: an initial, possibly empty, sequence of white-space characters, as specified by *isspace()* (page 367), a subject sequence resembling an integer represented in some radix determined by the value of base, and a final string of one or more unrecognized characters, including the terminating null character of the input string. *strtoull()* (page 516) then attempts to convert the subject sequence to an integer, and return the result.

When converting, no integer suffix (such as U, L, UL, LL, ULL) is allowed.

If the value of base is zero, the expected form of the subject sequence is an optional plus or minus sign followed by an integer constant.

If the value of base is between 2 and 36 (inclusive), the expected form of the subject sequence is an optional plus or minus sign followed by a sequence of letters and digits representing an integer with the radix specified by base. The letters from a (or A) through z (or Z) represent the values 10 through 35; only letters and digits whose ascribed values are less than that of base are permitted.

If the value of base is 16, the characters "0x" or "0X" may optionally precede the sequence of letters and digits, following the optional sign.

The subject sequence is defined as the longest initial subsequence of the input string, starting with the first non-white-space character, that is of the expected form. The subject sequence contains no characters if the input string is empty or consists entirely of white space, or if the first non-white-space character is other than a sign or a permissible letter or digit.

If the subject sequence has the expected form and the value of base is zero, the sequence of characters starting with the first digit is interpreted as an integer constant. If the subject sequence has the expected form and the value of base is between 2 and 36, it is used as the base for conversion.

If the subject sequence begins with a minus sign, the value resulting from the conversion is negated.

A pointer to the final string is stored in the object pointed to by endptr, provided that endptr is not a null pointer.

If the subject sequence is empty or does not have the expected form, no conversion is performed, the value of nptr is stored in the object pointed to by endptr, provided that endptr is not a null pointer.

strtof()

Description

Convert to number, float.

Prototype

```
float strtof(const char  * nptr,
                   char ** endptr);
```

Parameters

Parameter	Description
nptr	Pointer to string to convert from.
endptr	If nonnull, a pointer to object that receives the pointer to the first unconverted character.

Return value

Returns the converted value, if any. If no conversion could be performed, zero is returned. If the correct value is outside the range of representable values, HUGE_VALF is returned according to the sign of the value, if any, and the value of the macro ERANGE is stored in errno.

Additional information

Converts the initial portion of the string pointed to by nptr to float representation.

First, *strtof()* (page 517) decomposes the input string into three parts: an initial, possibly empty, sequence of white-space characters, as specified by *isspace()* (page 367), a subject sequence resembling a floating-point constant, and a final string of one or more unrecognized characters, including the terminating null character of the input string. *strtof()* (page 517) then attempts to convert the subject sequence to a floating-point number, and return the result.

The subject sequence is defined as the longest initial subsequence of the input string, starting with the first non-white-space character, that is of the expected form. The subject sequence contains no characters if the input string is empty or consists entirely of white space, or if the first non-white-space character is other than a sign or a permissible letter or digit.

The expected form of the subject sequence is an optional plus or minus sign followed by a nonempty sequence of decimal digits optionally containing a decimal-point character, then an optional exponent part.

If the subject sequence begins with a minus sign, the value resulting from the conversion is negated. A pointer to the final string is stored in the object pointed to by endptr, provided that endptr is not a null pointer.

If the subject sequence is empty or does not have the expected form, no conversion is performed, the value of nptr is stored in the object pointed to by endptr, provided that endptr is not a null pointer.

See also

strtod() (page 517)

strtod()

Description

Convert to number, double.

Prototype

```
double strtod(const char  * nptr,
                    char ** endptr);
```

Parameters

Parameter	Description
nptr	Pointer to string to convert from.
endptr	If nonnull, a pointer to object that receives the pointer to the first unconverted character.

Return value

Returns the converted value, if any. If no conversion could be performed, zero is returned. If the correct value is outside the range of representable values, HUGE_VAL is returned according to the sign of the value, if any, and the value of the macro ERANGE is stored in errno.

Additional information

Converts the initial portion of the string pointed to by nptr to double representation.

First, *strtod()* (page 517) decomposes the input string into three parts: an initial, possibly empty, sequence of white-space characters, as specified by *isspace()* (page 367), a subject sequence resembling a floating-point constant, and a final string of one or more unrecognized characters, including the terminating null character of the input string. *strtod()* (page 517) then attempts to convert the subject sequence to a floating-point number, and return the result.

The subject sequence is defined as the longest initial subsequence of the input string, starting with the first non-white-space character, that is of the expected form. The subject sequence contains no characters if the input string is empty or consists entirely of white space, or if the first non-white-space character is other than a sign or a permissible letter or digit.

The expected form of the subject sequence is an optional plus or minus sign followed by a nonempty sequence of decimal digits optionally containing a decimal-point character, then an optional exponent part.

If the subject sequence begins with a minus sign, the value resulting from the conversion is negated. A pointer to the final string is stored in the object pointed to by endptr, provided that endptr is not a null pointer.

If the subject sequence is empty or does not have the expected form, no conversion is performed, the value of nptr is stored in the object pointed to by endptr, provided that endptr is not a null pointer.

See also

strtof() (page 517)

strtold()

Description

Convert to number, long double.

Prototype

```
long double strtold(const char  * nptr,
                          char ** endptr);
```

Parameters

Parameter	Description
nptr	Pointer to string to convert from.
endptr	If nonnull, a pointer to object that receives the pointer to the first unconverted character.

Return value

Returns the converted value, if any. If no conversion could be performed, zero is returned. If the correct value is outside the range of representable values, HUGE_VAL is returned according to the sign of the value, if any, and the value of the macro ERANGE is stored in errno.

Additional information

Converts the initial portion of the string pointed to by nptr to long double representation.

First, *strtold()* (page 518) decomposes the input string into three parts: an initial, possibly empty, sequence of white-space characters, as specified by *isspace()* (page 367), a subject sequence resembling a floating-point constant, and a final string of one or more unrecognized characters, including the terminating null character of the input string. *strtod()* (page 517) then attempts to convert the subject sequence to a floating-point number, and return the result.

The subject sequence is defined as the longest initial subsequence of the input string, starting with the first non-white-space character, that is of the expected form. The subject sequence contains no characters if the input string is empty or consists entirely of white space, or if the first non-white-space character is other than a sign or a permissible letter or digit.

The expected form of the subject sequence is an optional plus or minus sign followed by a nonempty sequence of decimal digits optionally containing a decimal-point character, then an optional exponent part.

If the subject sequence begins with a minus sign, the value resulting from the conversion is negated. A pointer to the final string is stored in the object pointed to by endptr, provided that endptr is not a null pointer.

If the subject sequence is empty or does not have the expected form, no conversion is performed, the value of nptr is stored in the object pointed to by endptr, provided that endptr is not a null pointer.

See also

strtod() (page 517)

Multi-byte/wide character functions

Function	Description
btowc() (page 519)	Convert single-byte character to wide character.
btowc_l() (page **??**)	Convert single-byte character to wide character, per locale, (POSIX.1).
mblen() (page 520)	Count number of bytes in multi-byte character.
mblen_l() (page **??**)	Count number of bytes in multi-byte character, per locale (POSIX.1).
mbtowc() (page 522)	Convert multi-byte character to wide character.
mbtowc_l() (page **??**)	Convert multi-byte character to wide character, per locale (POSIX.1).
mbstowcs() (page 523)	Convert multi-byte string to wide string.
mbstowcs_l() (page **??**)	Convert multi-byte string to wide string, per locale (POSIX.1).
mbsrtowcs() (page 524)	Convert multi-byte string to wide character string, restartable.
mbsrtowcs_l() (page **??**)	Convert multi-byte string to wide character string, restartable, per locale (POSIX.1).
wctomb() (page 526)	Convert wide character to multi-byte character.
wctomb_l() (page **??**)	Convert wide character to multi-byte character, per locale (POSIX.1).
wcstombs() (page 527)	Convert wide string to multi-byte string.
wcstombs_l() (page **??**)	Convert wide string to multi-byte string.

btowc()

Description

Convert single-byte character to wide character.

Prototype

```
wint_t btowc(int c);
```

Parameters

Parameter	Description
c	Character to convert.

Return value

Returns WEOF if c has the value EOF or if c, converted to an unsigned char and in the current locale, does not constitute a valid single-byte character in the initial shift state.

Additional information

Determines whether c constitutes a valid single-byte character in the current locale. If c is a valid single-byte character, *btowc()* (page 519) returns the wide character representation of that character.

btowc_l()

Description

Convert single-byte character to wide character, per locale, (POSIX.1).

Prototype

```
wint_t btowc_l(int c,
                     locale_t loc);
```

Parameters

Parameter	Description
c	Character to convert.
loc	Locale used for conversion.

Return value

Returns WEOF if c has the value EOF or if c, converted to an unsigned char and in the locale loc, does not constitute a valid single-byte character in the initial shift state.

Additional information

Determines whether c constitutes a valid single-byte character in the locale loc. If c is a valid single-byte character, *btowc_l()* (page **??**) returns the wide character representation of that character.

Notes

Conforms to POSIX.1-2008.

mblen()

Description

Count number of bytes in multi-byte character.

Prototype

```
int mblen(const char   * s,
                size_t    n);
```

Parameters

Parameter	Description
s	Pointer to multi-byte character.
n	Maximum number of bytes to examine.

Return value

If s is a null pointer, returns a nonzero or zero value, if multi-byte character encodings, respectively, do or do not have state-dependent encodings.

If s is not a null pointer, either returns 0 (if s points to the null character), or returns the number of bytes that are contained in the multi-byte character (if the next n or fewer bytes form a valid multi-byte character), or returns -1 (if they do not form a valid multi-byte character).

Additional information

Determines the number of bytes contained in the multi-byte character pointed to by s in the current locale.

Except that the conversion state of the *mbtowc()* (page 522) function is not affected, it is equivalent to

mbtowc(NULL, s, n);

See also

mblen_l() (page **??**), *mbtowc()* (page 522)

mblen_l()

Description

Count number of bytes in multi-byte character, per locale (POSIX.1).

Prototype

```
int mblen_l(const char   * s,
                  size_t   n,
                           locale_t loc);
```

Parameters

Parameter	Description
s	Pointer to multi-byte character.
n	Maximum number of bytes to examine.
loc	Locale to use for conversion.

Return value

If s is a null pointer, returns a nonzero or zero value, if multi-byte character encodings, respectively, do or do not have state-dependent encodings in locale loc.

If s is not a null pointer, either returns 0 (if s points to the null character), or returns the number of bytes that are contained in the multi-byte character (if the next n or fewer bytes form a valid multi-byte character), or returns -1 (if they do not form a valid multi-byte character).

Additional information

Determines the number of bytes contained in the multi-byte character pointed to by s in the locale loc.

Except that the conversion state of the *mbtowc()* (page 522) function is not affected, it is equivalent to

mbtowc_l(NULL, s, n, loc);

Notes

Conforms to POSIX.1-2008.

See also

mblen() (page 520), *mbtowc()* (page 522)

mbtowc()

Description

Convert multi-byte character to wide character.

Prototype

```
int mbtowc(      wchar_t * pwc,
           const char    * s,
                 size_t    n);
```

Parameters

Parameter	Description
pwc	Pointer to object that receives the wide character.
s	Pointer to multi-byte character string.
n	Maximum number of bytes that will be examined.

Return value

If s is a null pointer, *mbtowc()* (page 522) returns a nonzero value if multi-byte character encodings are state-dependent in the current locale, and zero otherwise.

If s is not null and the object that s points to is a wide character null, *mbtowc()* (page 522) returns 0.

If s is not null and the object that s points to forms a valid multi-byte character, *mbtowc()* (page 522) returns the length in bytes of the multi-byte character.

If the object that *mbtowc()* (page 522) points to does not form a valid multi-byte character within the first n characters, it returns -1.

Additional information

Converts a single multi-byte character to a wide character in the current locale. The wide character, if the multi-byte character string is converted correctly, is stored into the object pointed to by pwc.

See also

mbtowc_l() (page **??**)

mbtowc_l()

Description

Convert multi-byte character to wide character, per locale (POSIX.1).

Prototype

```
int mbtowc_l(      wchar_t * pwc,
             const char    * s,
                   size_t    n,
                             locale_t loc);
```

Parameters

Parameter	Description
pwc	Pointer to object that receives the wide character.
s	Pointer to multi-byte character string.
n	Maximum number of bytes that will be examined.
loc	Locale used to convert the multi-byte character.

Return value

If s is a null pointer, *mbtowc_l()* (page **??**) returns a nonzero value if multi-byte character encodings are state-dependent in locale loc, and zero otherwise.

If s is not null and the object that s points to is a wide null character, *mbtowc_l()* (page **??**) returns 0.

If s is not null and the object that s points to forms a valid multi-byte character, *mbtowc_l()* (page **??**) returns the length in bytes of the multi-byte character.

If the object that *mbtowc_l()* (page **??**) points to does not form a valid multi-byte character within the first n characters, it returns -1.

Additional information

Converts a single multi-byte character to a wide character in the locale loc. The wide character, if the multi-byte character string is converted correctly, is stored into the object pointed to by pwc.

Notes

Conforms to POSIX.1-2008.

See also

mbtowc() (page 522)

mbstowcs()

Description

Convert multi-byte string to wide string.

Prototype

```
size_t mbstowcs(      wchar_t * pwcs,
                const char    * s,
                      size_t    n);
```

Parameters

Parameter	Description
pwcs	Pointer to array that receives the wide character string.
s	Pointer to array that contains the multi-byte string.
n	Maximum number of wide characters to write into pwcs.

Return value

Returns -1 if an invalid multi-byte character is encountered, otherwise returns the number of array elements modified (if any), not including a terminating null wide character.

Additional information

Converts a sequence of multi-byte characters, in the current locale, that begins in the initial shift state from the array pointed to by s into a sequence of corresponding wide characters and stores not more than n wide characters into the array pointed to by pwcs.

No multi-byte characters that follow a null character (which is converted into a null wide character) will be examined or converted. Each multi-byte character is converted as if by a call to the *mbtowc()* (page 522) function, except that the conversion state of the *mbtowc()* (page 522) function is not affected.

No more than n elements will be modified in the array pointed to by pwcs. If copying takes place between objects that overlap, the behavior is undefined.

mbstowcs_l()

Description

Convert multi-byte string to wide string, per locale (POSIX.1).

Prototype

```
size_t mbstowcs_l(      wchar_t * pwcs,
                  const char    * s,
                        size_t    n,
                                  locale_t loc);
```

Parameters

Parameter	Description
pwcs	Pointer to array that receives the wide character string.
s	Pointer to array that contains the multi-byte string.
n	Maximum number of wide characters to write into pwcs.
loc	Locale to use for conversion.

Return value

Returns -1 if an invalid multi-byte character is encountered, otherwise returns the number of array elements modified (if any), not including a terminating null wide character.

Additional information

Converts a sequence of multi-byte characters, in the locale loc, that begins in the initial shift state from the array pointed to by s into a sequence of corresponding wide characters and stores not more than n wide characters into the array pointed to by pwcs.

No multi-byte characters that follow a null character (which is converted into a null wide character) will be examined or converted. Each multi-byte character is converted as if by a call to the *mbtowc()* (page 522) function, except that the conversion state of the *mbtowc()* (page 522) function is not affected.

No more than n elements will be modified in the array pointed to by pwcs. If copying takes place between objects that overlap, the behavior is undefined.

Notes

Conforms to POSIX.1-2017.

mbsrtowcs()

Description

Convert multi-byte string to wide character string, restartable.

Prototype

```
size_t mbsrtowcs(      wchar_t    * dst,
                 const char      ** src,
                       size_t       len,
                       mbstate_t  * ps);
```

Parameters

Parameter	Description
dst	Pointer to object that receives the converted wide characters.
src	Pointer to pointer to multi-byte character string.
len	Maximum number of wide characters that will be written to dst.
ps	Pointer to multi-byte conversion state.

Return value

The number of wide characters written to dst (not including the eventual terminating null character).

Additional information

Converts a sequence of multi-byte characters, in the current locale, that begins in the conversion state described by the object pointed to by ps, from the array indirectly pointed to by src into a sequence of corresponding wide characters.

If dst is not a null pointer, the converted characters are stored into the array pointed to by dst. Conversion continues up to and including a terminating null character, which is also stored.

Conversion stops earlier in two cases: when a sequence of bytes is encountered that does not form a valid multi-byte character, or (if dst is not a null pointer) when len wide characters have been stored into the array pointed to by dst. Each conversion takes place as if by a call to the *mbrtowc()* (page 577) function.

If dst is not a null pointer, the pointer object pointed to by src is assigned either a null pointer (if conversion stopped due to reaching a terminating null character) or the address just past the last multi-byte character converted (if any). If conversion stopped due to reaching a terminating null character and if dst is not a null pointer, the resulting state described is the initial conversion state.

See also

mbsrtowcs_l() (page **??**), *mbrtowc()* (page 577)

mbsrtowcs_l()

Description

Convert multi-byte string to wide character string, restartable, per locale (POSIX.1).

Prototype

```
size_t mbsrtowcs_l(       wchar_t    * dst,
                    const char      ** src,
                          size_t       len,
                          mbstate_t  * ps,
                                       locale_t loc);
```

Parameters

Parameter	Description
dst	Pointer to object that receives the converted wide characters.
src	Pointer to pointer to multi-byte character string.
len	Maximum number of wide characters that will be written to dst.
ps	Pointer to multi-byte conversion state.
loc	Locale used for conversion.

Return value

The number of wide characters written to dst (not including the eventual terminating null character).

Additional information

Converts a sequence of multi-byte characters, in the locale loc, that begins in the conversion state described by the object pointed to by ps, from the array indirectly pointed to by src into a sequence of corresponding wide characters.

If dst is not a null pointer, the converted characters are stored into the array pointed to by dst. Conversion continues up to and including a terminating null character, which is also stored.

Conversion stops earlier in two cases: when a sequence of bytes is encountered that does not form a valid multi-byte character, or (if dst is not a null pointer) when len wide characters have been stored into the array pointed to by dst. Each conversion takes place as if by a call to the *mbrtowc()* (page 577) function.

If dst is not a null pointer, the pointer object pointed to by src is assigned either a null pointer (if conversion stopped due to reaching a terminating null character) or the address just past the last multi-byte character converted (if any). If conversion stopped due to reaching a terminating null character and if dst is not a null pointer, the resulting state described is the initial conversion state.

Notes

Conforms to POSIX.1-2008.

See also

mbsrtowcs() (page 524), *mbrtowc()* (page 577)

wctomb()

Description

Convert wide character to multi-byte character.

Prototype

```
int wctomb(char * s,
                  wchar_t wc);
```

Parameters

Parameter	Description
s	Pointer to array that receives the multi-byte character.
wc	Wide character to convert.

Return value

Returns the number of bytes stored in the array object. When wc is not a valid wide character, an encoding error occurs: *wctomb()* (page 526) stores the value of the macro EILSEQ in errno and returns (*size_t* (page **??**))(-1); the conversion state is unspecified.

Additional information

If s is a null pointer, *wctomb()* (page 526) is equivalent to the call wcrtomb(buf, 0, ps) where buf is an internal buffer.

If s is not a null pointer, *wctomb()* (page 526) determines the number of bytes needed to represent the multi-byte character that corresponds to the wide character given by wc in the current locale, and stores the multi-byte character representation in the array whose first element is pointed to by s. At most MB_CUR_MAX bytes are stored. If wc is a null wide character, a null byte is stored; the resulting state described is the initial conversion state.

wctomb_l()

Description

Convert wide character to multi-byte character, per locale (POSIX.1).

Prototype

```
int wctomb_l(char * s,
             wchar_t wc,
             locale_t loc);
```

Parameters

Parameter	Description
s	Pointer to array that receives the multi-byte character.
wc	Wide character to convert.
loc	Locale used for conversion.

Return value

Returns the number of bytes stored in the array object. When wc is not a valid wide character, an encoding error occurs: *wctomb_l()* (page **??**) stores the value of the macro EILSEQ in errno and returns (*size_t* (page **??**))(-1); the conversion state is unspecified.

Additional information

If s is a null pointer, *wctomb_l()* (page **??**) is equivalent to the call wcrtomb_l(buf, 0, ps, loc) where buf is an internal buffer.

If s is not a null pointer, *wctomb_l()* (page **??**) determines the number of bytes needed to represent the multi-byte character that corresponds to the wide character given by wc in the locale loc, and stores the multi-byte character representation in the array whose first element is pointed to by s. At most MB_CUR_MAX bytes are stored. If wc is a null wide character, a null byte is stored; the resulting state described is the initial conversion state.

Notes

Conforms to POSIX.1-2008.

wcstombs()

Description

Convert wide string to multi-byte string.

Prototype

```
size_t wcstombs(      char    * s,
                const wchar_t * pwcs,
                      size_t    n);
```

Parameters

Parameter	Description
s	Pointer to array that receives the multi-byte string.
pwcs	Pointer to wide character string to convert.
n	Maximum number of bytes to write into s.

Return value

If a wide character is encountered that does not correspond to a valid multibyte character in the current locale, returns (*size_t* (page **??**))(-1). Otherwise, returns the number of bytes written, not including a terminating null character (if any).

Additional information

Converts a sequence of wide characters in the current locale from the array pointed to by pwcs into a sequence of corresponding multi-byte characters that begins in the initial shift state, and stores these multi-byte characters into the array pointed to by s, stopping if a multi-byte character would exceed the limit of n total bytes or if a null character is stored. Each wide character is converted as if by a call to *wctomb()* (page 526), except that the conversion state of *wctomb()* (page 526) is not affected.

wcstombs_l()

Description

Convert wide string to multi-byte string.

Prototype

```
size_t wcstombs_l(       char    * s,
                   const wchar_t * pwcs,
                         size_t    n,
                                   locale_t loc);
```

Parameters

Parameter	Description
s	Pointer to array that receives the multi-byte string.
pwcs	Pointer to wide character string to convert.
n	Maximum number of bytes to write into s.
loc	Locale used for conversion.

Return value

If a wide character is encountered that does not correspond to a valid multibyte character in the locale loc, returns (*size_t* (page **??**))(-1). Otherwise, returns the number of bytes written, not including a terminating null character (if any).

Additional information

Converts a sequence of wide characters in the locale loc from the array pointed to by pwcs into a sequence of corresponding multi-byte characters that begins in the initial shift state, and stores these multi-byte characters into the array pointed to by s, stopping if a multi-byte character would exceed the limit of n total bytes or if a null character is stored. Each wide character is converted as if by a call to *wctomb()* (page 526), except that the conversion state of *wctomb()* (page 526) is not affected.

4.6.17 <string.h>

The header file <string.h> defines functions that operate on arrays that are interpreted as null-terminated strings.

Various methods are used for determining the lengths of the arrays, but in all cases a char * or void * argument points to the initial (lowest addressed) character of the array. If an array is accessed beyond the end of an object, the behavior is undefined.

Where an argument declared as *size_t* (page **??**) n specifies the length of an array for a function, n can have the value zero on a call to that function. Unless explicitly stated otherwise in the description of a particular function, pointer arguments must have valid values on a call with a zero size. On such a call, a function that locates a character finds no occurrence, a function that compares two character sequences returns zero, and a function that copies characters copies zero characters.

Copying functions

Function	Description
memset() (page 529)	Set memory to character.
memcpy() (page 530)	Copy memory.
memccpy() (page 530)	Copy memory, specify terminator (POSIX.1).
mempcpy() (page 531)	Copy memory (GNU).
memmove() (page 531)	Copy memory, tolerate overlaps.
strcpy() (page 532)	Copy string.
strncpy() (page 532)	Copy string, limit length.
strlcpy() (page 533)	Copy string, limit length, always zero terminate (BSD).
stpcpy() (page 533)	Copy string, return end.
stpncpy() (page 534)	Copy string, limit length, return end.
strcat() (page 534)	Concatenate strings.
strncat() (page 535)	Concatenate strings, limit length.
strlcat() (page 535)	Concatenate strings, limit length, always zero terminate (BSD).
strdup() (page 536)	Duplicate string (POSIX.1).
strndup() (page 536)	Duplicate string, limit length (POSIX.1).

memset()

Description

Set memory to character.

Prototype

```
void *memset(void    * s,
             int       c,
             size_t    n);
```

Parameters

Parameter	Description
s	Pointer to destination object.
c	Character to copy.
n	Length of destination object in characters.

Return value

Returns s.

Additional information

Copies the value of c (converted to an unsigned char) into each of the first n characters of the object pointed to by s.

memcpy()

Description

Copy memory.

Prototype

```
void *memcpy(       void   * s1,
              const void   * s2,
                    size_t   n);
```

Parameters

Parameter	Description
s1	Pointer to destination object.
s2	Pointer to source object.
n	Number of characters to copy.

Return value

Returns a pointer to the destination object.

Additional information

Copies n characters from the object pointed to by s2 into the object pointed to by s1. The behavior of *memcpy()* (page 530) is undefined if copying takes place between objects that overlap.

memccpy()

Description

Copy memory, specify terminator (POSIX.1).

Prototype

```
void *memccpy(       void   * s1,
               const void   * s2,
                     int      c,
                     size_t   n);
```

Parameters

Parameter	Description
s1	Pointer to destination object.
s2	Pointer to source object.
c	Character that terminates copy.
n	Maximum number of characters to copy.

Return value

Returns a pointer to the character immediately following c in s1, or NULL if c was not found in the first n characters of s2.

Additional information

Copies at most n characters from the object pointed to by s2 into the object pointed to by s1. The copying stops as soon as n characters are copied or the character c is copied into the destination object pointed to by s1.

The behavior of *memccpy()* (page 530) is undefined if copying takes place between objects that overlap.

Notes

Conforms to POSIX.1-2008.

mempcpy()

Description

Copy memory (GNU).

Prototype

```
void *mempcpy(       void   * s1,
              const void   * s2,
                    size_t   n);
```

Parameters

Parameter	Description
s1	Pointer to destination object.
s2	Pointer to source object.
n	Number of characters to copy.

Return value

Returns a pointer to the character immediately following the final character written into s1.

Additional information

Copies n characters from the object pointed to by s2 into the object pointed to by s1. The behavior of *mempcpy()* (page 531) is undefined if copying takes place between objects that overlap.

Notes

This is an extension found in GNU libc.

memmove()

Description

Copy memory, tolerate overlaps.

Prototype

```
void *memmove(       void   * s1,
              const void   * s2,
                    size_t   n);
```

Parameters

Parameter	Description
s1	Pointer to destination object.
s2	Pointer to source object.
n	Number of characters to copy.

Return value

Returns the value of s1.

Additional information

Copies n characters from the object pointed to by s2 into the object pointed to by s1 ensuring that if s1 and s2 overlap, the copy works correctly. Copying takes place as if the n characters from the object pointed to by s2 are first copied into a temporary array of n characters that does not overlap the objects pointed to by s1 and s2, and then the n characters from the temporary array are copied into the object pointed to by s1.

strcpy()

Description

Copy string.

Prototype

```
char *strcpy(       char * s1,
              const char * s2);
```

Parameters

Parameter	Description
s1	String to copy to.
s2	String to copy.

Return value

Returns the value of s1.

Additional information

Copies the string pointed to by s2 (including the terminating null character) into the array pointed to by s1. The behavior of *strcpy()* (page 532) is undefined if copying takes place between objects that overlap.

strncpy()

Description

Copy string, limit length.

Prototype

```
char *strncpy(       char    * s1,
               const char    * s2,
                     size_t    n);
```

Parameters

Parameter	Description
s1	String to copy to.
s2	String to copy.
n	Maximum number of characters to copy.

Return value

Returns the value of s1.

Additional information

Copies not more than n characters from the array pointed to by s2 to the array pointed to by s1. Characters that follow a null character in s2 are not copied. The behavior of *strncpy()* (page 532) is undefined if copying takes place between objects that overlap. If the array pointed to by s2 is a string that is shorter than n characters, null characters are appended to the copy in the array pointed to by s1, until n characters in all have been written.

Notes

No null character is implicitly appended to the end of s1, so s1 will only be terminated by a null character if the length of the string pointed to by s2 is less than n.

strlcpy()

Description

Copy string, limit length, always zero terminate (BSD).

Prototype

```
size_t strlcpy(       char   * s1,
               const char   * s2,
                     size_t   n);
```

Parameters

Parameter	Description
s1	Pointer to string to copy to.
s2	Pointer to string to copy.
n	Maximum number of characters, including terminating null, in s1.

Return value

Returns the number of characters it tried to copy, which is the length of the string s2 or n, whichever is smaller.

Additional information

Copies up to n-1 characters from the string pointed to by s2 into the array pointed to by s1 and always terminates the result with a null character.

The behavior of *strlcpy()* (page 533) is undefined if copying takes place between objects that overlap.

Notes

Commonly found in BSD libraries and contrasts with *strncpy()* (page 532) in that the resulting string is always terminated with a null character.

stpcpy()

Description

Copy string, return end.

Prototype

```
char *stpcpy(       char * s1,
             const char * s2);
```

Parameters

Parameter	Description
s1	String to copy to.
s2	String to copy.

Return value

A pointer to the end of the string s1, i.e. the terminating null byte of the string s1, after s2 is copied to it.

Additional information

Copies the string pointed to by s2 (including the terminating null character) into the array pointed to by s1. The behavior of *stpcpy()* (page 533) is undefined if copying takes place between objects that overlap.

stpncpy()

Description

Copy string, limit length, return end.

Prototype

```
char *stpncpy(       char    * s1,
               const char    * s2,
                     size_t    n);
```

Parameters

Parameter	Description
s1	String to copy to.
s2	String to copy.
n	Maximum number of characters to copy.

Return value

stpncpy() (page 534) returns a pointer to the terminating null byte in s1 after it is copied to, or, if s1 is not null-terminated, s1+n.

Additional information

Copies not more than n characters from the array pointed to by s2 to the array pointed to by s1. Characters that follow a null character in s2 are not copied. The behavior of *strncpy()* (page 532) is undefined if copying takes place between objects that overlap. If the array pointed to by s2 is a string that is shorter than n characters, null characters are appended to the copy in the array pointed to by s1, until n characters in all have been written.

Notes

No null character is implicitly appended to the end of s1, so s1 will only be terminated by a null character if the length of the string pointed to by s2 is less than n.

strcat()

Description

Concatenate strings.

Prototype

```
char *strcat(       char * s1,
              const char * s2);
```

Parameters

Parameter	Description
s1	Zero-terminated string to append to.
s2	Zero-terminated string to append.

Return value

Returns the value of s1.

Additional information

Appends a copy of the string pointed to by s2 (including the terminating null character) to the end of the string pointed to by s1. The initial character of s2 overwrites the null character at the end of s1. The behavior of *strcat()* (page 534) is undefined if copying takes place between objects that overlap.

strncat()

Description

Concatenate strings, limit length.

Prototype

```
char *strncat(       char    * s1,
               const char    * s2,
                     size_t    n);
```

Parameters

Parameter	Description
s1	String to append to.
s2	String to append.
n	Maximum number of characters in s1.

Return value

Returns the value of s1.

Additional information

Appends not more than n characters from the array pointed to by s2 to the end of the string pointed to by s1. A null character in s1 and characters that follow it are not appended. The initial character of s2 overwrites the null character at the end of s1. A terminating null character is always appended to the result.

The behavior of *strncat()* (page 535) is undefined if copying takes place between objects that overlap.

strlcat()

Description

Concatenate strings, limit length, always zero terminate (BSD).

Prototype

```
size_t strlcat(       char    * s1,
                const char    * s2,
                      size_t    n);
```

Parameters

Parameter	Description
s1	Pointer to string to append to.
s2	Pointer to string to append.
n	Maximum number of characters, including terminating null, in s1.

Return value

Returns the number of characters it tried to copy, which is the sum of the lengths of the strings s1 and s2 or n, whichever is smaller.

Additional information

Appends no more than n-strlen(s1}-1 characters pointed to by s2 into the array pointed to by s1 and always terminates the result with a null character if n is greater than zero. Both the strings s1 and s2 must be terminated with a null character on entry to *strlcat()* (page 535) and a character position for the terminating null should be included in n.

The behavior of *strlcat()* (page 535) is undefined if copying takes place between objects that overlap.

Notes

Commonly found in BSD libraries.

strdup()

Description

Duplicate string (POSIX.1).

Prototype

```
char *strdup(const char * s1);
```

Parameters

Parameter	Description
s1	Pointer to string to duplicate.

Return value

Returns a pointer to the new string or a null pointer if the new string cannot be created. The returned pointer can be passed to *free()* (page 504).

Additional information

Duplicates the string pointed to by s1 by using *malloc()* (page 503) to allocate memory for a copy of s and then copyies s, including the terminating null, to that memory

Notes

Conforms to POSIX.1-2008 and SC22 TR 24731-2.

strndup()

Description

Duplicate string, limit length (POSIX.1).

Prototype

```
char *strndup(const char   * s,
                    size_t   n);
```

Parameters

Parameter	Description
s	Pointer to string to duplicate.
n	Maximum number of characters to duplicate.

Return value

Returns a pointer to the new string or a null pointer if the new string cannot be created. The returned pointer can be passed to *free()* (page 504).

Additional information

Duplicates at most n characters from the the string pointed to by s by using *malloc()* (page 503) to allocate memory for a copy of s.

If the length of string pointed to by s is greater than n characters, only n characters will be duplicated. If n is greater than the length of the string pointed to by s, all characters in the string are copied into the allocated array including the terminating null character.

Notes

Conforms to POSIX.1-2008 and SC22 TR 24731-2.

Comparison functions

Function	Description
memcmp() (page 537)	Compare memory.
strcmp() (page 538)	Compare strings.
strncmp() (page 538)	Compare strings, limit length.
strcasecmp() (page 539)	Compare strings, ignore case (POSIX.1).
strncasecmp() (page 539)	Compare strings, ignore case, limit length (POSIX.1).

memcmp()

Description

Compare memory.

Prototype

```
int memcmp(const void   * s1,
           const void   * s2,
                 size_t   n);
```

Parameters

Parameter	Description
s1	Pointer to object #1.
s2	Pointer to object #2.
n	Number of characters to compare.

Return value

< 0	s1 is less than s2.
= 0	s1 is equal to s2.
> 0	s1 is greater than to s2.

Additional information

Compares the first n characters of the object pointed to by s1 to the first n characters of the object pointed to by s2. *memcmp()* (page 537) returns an integer greater than, equal to, or less than zero as the object pointed to by s1 is greater than, equal to, or less than the object pointed to by s2.

strcmp()

Description

Compare strings.

Prototype

```
int strcmp(const char * s1,
           const char * s2);
```

Parameters

Parameter	Description
s1	Pointer to string #1.
s2	Pointer to string #2.

Return value

Returns an integer greater than, equal to, or less than zero, if the null-terminated array pointed to by s1 is greater than, equal to, or less than the null-terminated array pointed to by s2.

strncmp()

Description

Compare strings, limit length.

Prototype

```
int strncmp(const char   * s1,
            const char   * s2,
                  size_t   n);
```

Parameters

Parameter	Description
s1	Pointer to string #1.
s2	Pointer to string #2.
n	Maximum number of characters to compare.

Return value

Returns an integer greater than, equal to, or less than zero, if the possibly null-terminated array pointed to by s1 is greater than, equal to, or less than the possibly null-terminated array pointed to by s2.

Additional information

Compares not more than n characters from the array pointed to by s1 to the array pointed to by s2. Characters that follow a null character are not compared.

strcasecmp()

Description

Compare strings, ignore case (POSIX.1).

Prototype

```
int strcasecmp(const char * s1,
               const char * s2);
```

Parameters

Parameter	Description
s1	Pointer to string #1.
s2	Pointer to string #2.

Return value

< 0	s1 is less than s2.
= 0	s1 is equal to s2.
> 0	s1 is greater than to s2.

Additional information

Compares the string pointed to by s1 to the string pointed to by s2 ignoring differences in case.

strcasecmp() (page 539) returns an integer greater than, equal to, or less than zero if the string pointed to by s1 is greater than, equal to, or less than the string pointed to by s2.

Notes

Conforms to POSIX.1-2008.

strncasecmp()

Description

Compare strings, ignore case, limit length (POSIX.1).

Prototype

```
int strncasecmp(const char   * s1,
                const char   * s2,
                      size_t   n);
```

Parameters

Parameter	Description
s1	Pointer to string #1.
s2	Pointer to string #2.
n	Maximum number of characters to compare.

Return value

< 0	s1 is less than s2.
= 0	s1 is equal to s2.
> 0	s1 is greater than to s2.

Additional information

Compares not more than n characters from the array pointed to by s1 to the array pointed to by s2 ignoring differences in case. Characters that follow a null character are not compared.

strncasecmp() (page 539) returns an integer greater than, equal to, or less than zero, if the possibly null-terminated array pointed to by s1 is greater than, equal to, or less than the possibly null-terminated array pointed to by s2.

Notes

Conforms to POSIX.1-2008.

Search functions

Function	Description
memchr() (page 540)	Find character in memory, forward.
memrchr() (page 541)	Find character in memory, reverse (BSD).
memmem() (page 541)	Find memory in memory, forward (BSD).
strchr() (page 542)	Find character within string, forward.
strnchr() (page 542)	Find character within string, forward, limit length.
strrchr() (page 543)	Find character within string, reverse.
strlen() (page 543)	Calculate length of string.
strnlen() (page 544)	Calculate length of string, limit length (POSIX.1).
strstr() (page 544)	Find string within string, forward.
strnstr() (page 545)	Find string within string, forward, limit length (BSD).
strcasestr() (page 545)	Find string within string, forward, ignore case (BSD).
strncasestr() (page 546)	Find string within string, forward, ignore case, limit length (BSD).
strpbrk() (page 546)	Find first occurrence of characters within string.
strspn() (page 547)	Compute size of string prefixed by a set of characters.
strcspn() (page 547)	Compute size of string not prefixed by a set of characters.
strtok() (page 548)	Break string into tokens.
strtok_r() (page **??**)	Break string into tokens, reentrant (POSIX.1).
strsep() (page 549)	Break string into tokens (BSD).

memchr()

Description

Find character in memory, forward.

Prototype

```
void *memchr(const void   * s,
                   int       c,
                   size_t    n);
```

Parameters

Parameter	Description
s	Pointer to object to search.
c	Character to search for.
n	Number of characters in object to search.

Return value

= NULL	c does not occur in the object.
≠ NULL	Pointer to the located character.

Additional information

Locates the first occurrence of c (converted to an unsigned char) in the initial n characters (each interpreted as unsigned char) of the object pointed to by s. Unlike *strchr()* (page 542), *memchr()* (page 540) does not terminate a search when a null character is found in the object pointed to by s.

memrchr()

Description

Find character in memory, reverse (BSD).

Prototype

```
void *memrchr(const void   * s,
                    int      c,
                    size_t   n);
```

Parameters

Parameter	Description
s	Pointer to object to search.
c	Character to search for.
n	Number of characters in object to search.

Return value

Returns a pointer to the located character, or a null pointer if c does not occur in the string.

Additional information

Locates the last occurrence of c (converted to a char) in the string pointed to by s.

Notes

Commonly found in Linux and BSD C libraries.

memmem()

Description

Find memory in memory, forward (BSD).

Prototype

```
void *memmem(const void   * s1,
                   size_t   n1,
             const void   * s2,
                   size_t   n2);
```

Parameters

Parameter	Description
s1	Pointer to object to search.
n1	Number of characters to search in s1.
s2	Pointer to object to search for.
n2	Number of characters to search from s2.

Return value

= NULL	(s2, n2) does not occur in (s1, n1).
≠ NULL	Pointer to the first occurrence of (s2, n2) in (s1, n1).

Additional information

Locates the first occurrence of the octet string s2 of length n2 in the octet string s1 of length n1.

Notes

Commonly found in Linux and BSD C libraries.

strchr()

Description

Find character within string, forward.

Prototype

```
char *strchr(const char * s,
                   int    c);
```

Parameters

Parameter	Description
s	String to search.
c	Character to search for.

Return value

Returns a pointer to the located character, or a null pointer if c does not occur in the string.

Additional information

Locates the first occurrence of c (converted to a char) in the string pointed to by s. The terminating null character is considered to be part of the string.

strnchr()

Description

Find character within string, forward, limit length.

Prototype

```
char *strnchr(const char   * s,
                    size_t   n,
                    int      c);
```

Parameters

Parameter	Description
s	String to search.
n	Number of characters to search.
c	Character to search for.

Return value

Returns a pointer to the located character, or a null pointer if c does not occur in the string.

Additional information

Searches not more than n characters to locate the first occurrence of c (converted to a char) in the string pointed to by s. The terminating null character is considered to be part of the string.

strrchr()

Description

Find character within string, reverse.

Prototype

```
char *strrchr(const char * s,
              int    c);
```

Parameters

Parameter	Description
s	String to search.
c	Character to search for.

Return value

Returns a pointer to the located character, or a null pointer if c does not occur in the string.

Additional information

Locates the last occurrence of c (converted to a char) in the string pointed to by s. The terminating null character is considered to be part of the string.

strlen()

Description

Calculate length of string.

Prototype

```
size_t strlen(const char * s);
```

Parameters

Parameter	Description
s	Pointer to zero-terminated string.

Return value

Returns the length of the string pointed to by s, that is the number of characters that precede the terminating null character.

strnlen()

Description

Calculate length of string, limit length (POSIX.1).

Prototype

```
size_t strnlen(const char   * s,
                   size_t   n);
```

Parameters

Parameter	Description
s	Pointer to string.
n	Maximum number of characters to examine.

Return value

Returns the length of the string pointed to by s, up to a maximum of n characters. *strnlen()* (page 544) only examines the first n characters of the string s.

Notes

Conforms to POSIX.1-2008.

strstr()

Description

Find string within string, forward.

Prototype

```
char *strstr(const char * s1,
             const char * s2);
```

Parameters

Parameter	Description
s1	String to search.
s2	String to search for.

Return value

Returns a pointer to the located string, or a null pointer if the string is not found. If s2 points to a string with zero length, *strstr()* (page 544) returns s1.

Additional information

Locates the first occurrence in the string pointed to by s1 of the sequence of characters (excluding the terminating null character) in the string pointed to by s2.

strnstr()

Description

Find string within string, forward, limit length (BSD).

Prototype

```
char *strnstr(const char   * s1,
              const char   * s2,
                    size_t   n);
```

Parameters

Parameter	Description
s1	String to search.
s2	String to search for.
n	Maximum number of characters to search for.

Return value

Returns a pointer to the located string, or a null pointer if the string is not found. If s2 points to a string with zero length, *strnstr()* (page 545) returns s1.

Additional information

Searches at most n characters to locate the first occurrence in the string pointed to by s1 of the sequence of characters (excluding the terminating null character) in the string pointed to by s2.

Notes

Commonly found in Linux and BSD C libraries.

strcasestr()

Description

Find string within string, forward, ignore case (BSD).

Prototype

```
char *strcasestr(const char * s1,
                 const char * s2);
```

Parameters

Parameter	Description
s1	String to search for.
s2	String to search.

Return value

Returns a pointer to the located string, or a null pointer if the string is not found. If s2 points to a string with zero length, returns s1.

Additional information

Locates the first occurrence in the string pointed to by s1 of the sequence of characters (excluding the terminating null character) in the string pointed to by s2 without regard to character case.

Notes

This extension is commonly found in Linux and BSD C libraries.

strncasestr()

Description

Find string within string, forward, ignore case, limit length (BSD).

Prototype

```
char *strncasestr(const char   * s1,
                  const char   * s2,
                        size_t   n);
```

Parameters

Parameter	Description
s1	String to search for.
s2	String to search.
n	Maximum number of characters to compare in s2.

Return value

Returns a pointer to the located string, or a null pointer if the string is not found. If s2 points to a string with zero length, returns s1.

Additional information

Searches at most n characters to locate the first occurrence in the string pointed to by s1 of the sequence of characters (excluding the terminating null character) in the string pointed to by s2 without regard to character case.

Notes

This extension is commonly found in Linux and BSD C libraries.

strpbrk()

Description

Find first occurrence of characters within string.

Prototype

```
char *strpbrk(const char * s1,
              const char * s2);
```

Parameters

Parameter	Description
s1	Pointer to string to search.
s2	Pointer to string to search for.

Return value

Returns a pointer to the first character, or a null pointer if no character from s2 occurs in s1.

Additional information

Locates the first occurrence in the string pointed to by s1 of any character from the string pointed to by s2.

strspn()

Description

Compute size of string prefixed by a set of characters.

Prototype

```
size_t strspn(const char * s1,
              const char * s2);
```

Parameters

Parameter	Description
s1	Pointer to zero-terminated string to search.
s2	Pointer to zero-terminated acceptable-set string.

Return value

Returns the length of the string pointed to by s1 which consists entirely of characters from the string pointed to by s2

Additional information

Computes the length of the maximum initial segment of the string pointed to by s1 which consists entirely of characters from the string pointed to by s2.

strcspn()

Description

Compute size of string not prefixed by a set of characters.

Prototype

```
size_t strcspn(const char * s1,
               const char * s2);
```

Parameters

Parameter	Description
s1	Pointer to string to search.
s2	Pointer to string containing characters to skip.

Return value

Returns the length of the segment of string s1 prefixed by characters from s2.

Additional information

Computes the length of the maximum initial segment of the string pointed to by s1 which consists entirely of characters not from the string pointed to by s2.

strtok()

Description

Break string into tokens.

Prototype

```
char *strtok(       char * s1,
            const char * s2);
```

Parameters

Parameter	Description
s1	Pointer to zero-terminated string to parse.
s2	Pointer to zero-terminated set of separators.

Return value

NULL if no further tokens else a pointer to the next token.

Additional information

A sequence of calls to *strtok()* (page 548) breaks the string pointed to by s1 into a sequence of tokens, each of which is delimited by a character from the string pointed to by s2. The first call in the sequence has a non-null first argument; subsequent calls in the sequence have a null first argument. The separator string pointed to by s2 may be different from call to call.

The first call in the sequence searches the string pointed to by s1 for the first character that is not contained in the current separator string pointed to by s2. If no such character is found, then there are no tokens in the string pointed to by s1 and *strtok()* (page 548) returns a null pointer. If such a character is found, it is the start of the first token.

strtok() (page 548) then searches from there for a character that is contained in the current separator string. If no such character is found, the current token extends to the end of the string pointed to by s1, and subsequent searches for a token will return a null pointer. If such a character is found, it is overwritten by a null character, which terminates the current token. *strtok()* (page 548) saves a pointer to the following character, from which the next search for a token will start.

Each subsequent call, with a null pointer as the value of the first argument, starts searching from the saved pointer and behaves as described above.

Notes

strtok() (page 548) maintains static state and is therefore not reentrant and not thread safe. See *strtok_r()* (page **??**) for a thread-safe and reentrant variant.

See also

strsep() (page 549), *strtok_r()* (page **??**).

strtok_r()

Description

Break string into tokens, reentrant (POSIX.1).

Prototype

```
char *strtok_r(       char  * s1,
                const char  * s2,
                      char ** lasts);
```

Parameters

Parameter	Description
s1	Pointer to zero-terminated string to parse.
s2	Pointer to zero-terminated set of separators.
lasts	Pointer to pointer to char that maintains parse state.

Return value

NULL if no further tokens else a pointer to the next token.

Additional information

strtok_r() (page **??**) is a reentrant version of the function *strtok()* (page 548) where the state is maintained in the object of type char * pointed to by s3.

Notes

Conforms to POSIX.1-2008 and is commonly found in Linux and BSD C libraries.

See also

strtok() (page 548)

strsep()

Description

Break string into tokens (BSD).

Prototype

```
char *strsep(      char ** stringp,
             const char  * delim);
```

Parameters

Parameter	Description
stringp	Pointer to pointer to zero-terminated string.
delim	Pointer to delimiter set string.

Return value

See below.

Additional information

Locates, in the string referenced by *stringp, the first occurrence of any character in the string delim (or the terminating null character) and replaces it with a null character. The location of the next character after the delimiter character (or NULL, if the end of the string was reached) is stored in *stringp. The original value of *stringp is returned.

An empty field (that is, a character in the string delim occurs as the first character of *stringp) can be detected by comparing the location referenced by the returned pointer to the null wide character.

If *stringp is initially null, *strsep()* (page 549) returns null.

Notes

Commonly found in Linux and BSD C libraries.

Miscellaneous functions

Function	Description
strerror() (page 550)	Decode error code.

strerror()

Description

Decode error code.

Prototype

```
char *strerror(int num);
```

Parameters

Parameter	Description
num	Error number.

Return value

Returns a pointer to the message string. The program must not modify the returned message string. The message may be overwritten by a subsequent call to *strerror()* (page 550).

Additional information

Maps the number in num to a message string. Typically, the values for num come from errno, but *strerror()* (page 550) can map any value of type int to a message.

4.6.18 <time.h>

Operations

Function	Description
mktime() (page 550)	Convert a struct tm to time_t.
difftime() (page 551)	Calculate difference between two times.

mktime()

Description

Convert a struct tm to time_t.

Prototype

```
time_t mktime(tm * tp);
```

Parameters

Parameter	Description
tp	Pointer to time object.

Return value

Number of seconds since UTC 1 January 1970 of the validated object.

Additional information

Validates (and updates) the object pointed to by tp to ensure that the tm_sec, tm_min, tm_hour, and tm_mon fields are within the supported integer ranges and the tm_mday, tm_mon and tm_year fields are consistent. The validated object is converted to the number of seconds since UTC 1 January 1970 and returned.

difftime()

Description

Calculate difference between two times.

Prototype

```
double difftime(time_t time2,
                time_t time1);
```

Parameters

Parameter	Description
time2	End time.
time1	Start time.

Return value

returns time2-time1 as a double precision number.

Conversion functions

Function	Description
ctime() (page 551)	Convert time_t to a string.
ctime_r() (page **??**)	Convert time_t to a string, reentrant.
asctime() (page 552)	Convert time_t to a string.
asctime_r() (page **??**)	Convert time_t to a string, reentrant.
gmtime() (page 553)	Convert time_t to struct tm.
gmtime_r() (page **??**)	Convert time_t to struct tm, reentrant.
localtime() (page 554)	Convert time to local time.
localtime_r() (page **??**)	Convert time to local time, reentrant.
strftime() (page 555)	Convert time to a string.
strftime_l() (page **??**)	Convert time to a string.

ctime()

Description

Convert time_t to a string.

Prototype

```
char *ctime(const time_t * tp);
```

Parameters

Parameter	Description
tp	Pointer to time to convert.

Return value

Pointer to zero-terminated converted string.

Additional information

Converts the time pointed to by tp to a null-terminated string.

Notes

The returned string is held in a static buffer: this function is not thread safe.

ctime_r()

Description

Convert time_t to a string, reentrant.

Prototype

```
char *ctime_r(const time_t * tp,
              char    * buf);
```

Parameters

Parameter	Description
tp	Pointer to time to convert.
buf	Pointer to array of characters that receives the zero-terminated string; the array must be at least 26 characters in length.

Return value

Returns the value of buf.

Additional information

Converts the time pointed to by tp to a null-terminated string.

Notes

The returned string is held in a static buffer: this function is not thread safe.

asctime()

Description

Convert time_t to a string.

Prototype

```
char *asctime(const tm * tp);
```

Parameters

Parameter	Description
tp	Pointer to time to convert.

Return value

Pointer to zero-terminated converted string.

Additional information

Converts the time pointed to by tp to a null-terminated string of the Sun Sep 16 01:03:52 1973. The returned string is held in a static buffer.

Notes

The returned string is held in a static buffer: this function is not thread safe.

asctime_r()

Description

Convert time_t to a string, reentrant.

Prototype

```
char *asctime_r(const tm    * tp,
                      char * buf);
```

Parameters

Parameter	Description
tp	Pointer to time to convert.
buf	Pointer to array of characters that receives the zero-terminated string; the array must be at least 26 characters in length.

Return value

Returns the value of buf.

Additional information

Converts the time pointed to by tp to a null-terminated string of the Sun Sep 16 01:03:52 1973. The converted string is written into the array pointed to by buf.

gmtime()

Description

Convert time_t to struct tm.

Prototype

```
gmtime(const time_t * tp);
```

Parameters

Parameter	Description
tp	Pointer to time to convert.

Return value

Pointer to converted time.

Additional information

Converts the time pointed to by tp to a struct tm.

Notes

The returned pointer points to a static buffer: this function is not thread safe.

gmtime_r()

Description

Convert time_t to struct tm, reentrant.

Prototype

```
gmtime_r(const time_t * tp,
                        tm *tm);
```

Parameters

Parameter	Description
tp	Pointer to time to convert.
tm	Pointer to object that receives the converted time.

Return value

Returns tm.

Additional information

Converts the time pointed to by tp to a struct tm.

localtime()

Description

Convert time to local time.

Prototype

```
localtime(const time_t * tp);
```

Parameters

Parameter	Description
tp	Pointer to time to convert.

Return value

Pointer to a statically-allocated object holding the local time.

Additional information

Converts the time pointed to by tp to local time format.

Notes

The returned pointer points to a static object: this function is not thread safe.

localtime_r()

Description

Convert time to local time, reentrant.

Prototype

```
localtime_r(const time_t * tp,
                           tm *tm);
```

Parameters

Parameter	Description
tp	Pointer to time to convert.
tm	Pointer to object that receives the converted local time.

Return value

Returns tm.

Additional information

Converts the time pointed to by tp to local time format and writes it to the object pointed to by tm.

strftime()

Description

Convert time to a string.

Prototype

```
size_t strftime(      char   * s,
                      size_t   smax,
                const char   * fmt,
                const tm     * tp);
```

Parameters

Parameter	Description
s	Pointer to object that receives the converted string.
smax	Maximum number of characters written to the array pointed to by s.
fmt	Pointer to zero-terminated format control string.
tp	Pointer to time to convert.

Return value

Returns the name of the current locale.

Additional information

Formats the time pointed to by tp to a null-terminated string of maximum size smax-1 into the pointed to by *s based on the fmt format string. The format string consists of conversion specifications and ordinary characters. Conversion specifications start with a "%" character followed by an optional "#" character.

The following conversion specifications are supported:

Specification	Description
%a	Abbreviated weekday name
%A	Full weekday name
%b	Abbreviated month name
%B	Full month name
%c	Date and time representation appropriate for locale
%#c	Date and time formatted as "%A, %B %#d, %Y, %H:%M:%S" (Microsoft extension)
%C	Century number
%d	Day of month as a decimal number [01,31]
%#d	Day of month without leading zero [1,31]
%D	Date in the form %m/%d/%y (POSIX.1-2008 extension)
%e	Day of month [1,31], single digit preceded by space
%F	Date in the format %Y-%m-%d
%h	Abbreviated month name as %b
%H	Hour in 24-hour format [00,23]
%#H	Hour in 24-hour format without leading zeros [0,23]
%I	Hour in 12-hour format [01,12]
%#I	Hour in 12-hour format without leading zeros [1,12]
%j	Day of year as a decimal number [001,366]
%#j	Day of year as a decimal number without leading zeros [1,366]
%k	Hour in 24-hour clock format [0,23] (POSIX.1-2008 extension)
%l	Hour in 12-hour clock format [0,12] (POSIX.1-2008 extension)
%m	Month as a decimal number [01,12]
%#m	Month as a decimal number without leading zeros [1,12]
%M	Minute as a decimal number [00,59]
%#M	Minute as a decimal number without leading zeros [0,59]
%n	Insert newline character (POSIX.1-2008 extension)
%p	Locale's a.m or p.m indicator for 12-hour clock
%r	Time as %I:%M:%s %p (POSIX.1-2008 extension)
%R	Time as %H:%M (POSIX.1-2008 extension)
%S	Second as a decimal number [00,59]
%t	Insert tab character (POSIX.1-2008 extension)
%T	Time as %H:%M:%S
%#S	Second as a decimal number without leading zeros [0,59]
%U	Week of year as a decimal number [00,53], Sunday is first day of the week
%#U	Week of year as a decimal number without leading zeros [0,53], Sunday is first day of the week
%w	Weekday as a decimal number [0,6], Sunday is 0
%W	Week number as a decimal number [00,53], Monday is first day of the week
%#W	Week number as a decimal number without leading zeros [0,53], Monday is first day of the week
%x	Locale's date representation
%#x	Locale's long date representation
%X	Locale's time representation
%y	Year without century, as a decimal number [00,99]
%#y	Year without century, as a decimal number without leading zeros [0,99]
%Y	Year with century, as decimal number
%z,%Z	Timezone name or abbreviation
%%	%

strftime_l()

Description

Convert time to a string.

Prototype

```
size_t strftime_l(       char   * s,
                         size_t   smax,
                   const char   * fmt,
                   const tm     * tp,
                                  locale_t loc);
```

Parameters

Parameter	Description
s	Pointer to object that receives the converted string.
smax	Maximum number of characters written to the array pointed to by s.
fmt	Pointer to zero-terminated format control string.
tp	Pointer to time to convert.
loc	Locale to use for conversion.

Return value

Returns the name of the current locale.

Additional information

Formats the time pointed to by tp to a null-terminated string of maximum size smax-1 into the pointed to by *s based on the fmt format string and using the locale loc.

The format string consists of conversion specifications and ordinary characters. Conversion specifications start with a "%" character followed by an optional "#" character.

See *strftime()* (page 555) for a description of the format conversion specifications.

4.6.19 <wchar.h>

Copying functions

Function	Description
wmemset() (page 558)	Set memory to wide character.
wmemcpy() (page 558)	Copy memory.
wmemccpy() (page 559)	Copy memory, specify terminator (POSIX.1).
wmempcpy() (page 559)	Copy memory (GNU).
wmemmove() (page 560)	Copy memory, tolerate overlaps.
wcscpy() (page 560)	Copy string.
wcsncpy() (page 561)	Copy string, limit length.
wcslcpy() (page 561)	Copy string, limit length, always zero terminate (BSD).
wcscat() (page 562)	Concatenate strings.
wcsncat() (page 562)	Concatenate strings, limit length.
wcslcat() (page 563)	Concatenate strings, limit length, always zero terminate (BSD).
wcsdup() (page 564)	Duplicate string (POSIX.1).
wcsndup() (page 564)	Duplicate string, limit length (GNU).

wmemset()

Description

Set memory to wide character.

Prototype

```
wchar_t *wmemset(wchar_t * s,
                 wchar_t   c,
                 size_t    n);
```

Parameters

Parameter	Description
s	Pointer to destination object.
c	Wide character to copy.
n	Length of destination object in wide characters.

Return value

Returns s.

Additional information

Copies the value of c into each of the first n wide characters of the object pointed to by s.

wmemcpy()

Description

Copy memory.

Prototype

```
wchar_t *wmemcpy(      wchar_t * s1,
                 const wchar_t * s2,
                       size_t    n);
```

Parameters

Parameter	Description
s1	Pointer to destination object.
s2	Pointer to source object.
n	Number of wide characters to copy.

Return value

Returns the value of s1.

Additional information

Copies n wide characters from the object pointed to by s2 into the object pointed to by s1. The behavior of *wmemcpy()* (page 558) is undefined if copying takes place between objects that overlap.

wmemccpy()

Description

Copy memory, specify terminator (POSIX.1).

Prototype

```
wchar_t *wmemccpy(       wchar_t * s1,
                   const wchar_t * s2,
                         wchar_t   c,
                         size_t    n);
```

Parameters

Parameter	Description
s1	Pointer to destination object.
s2	Pointer to source object.
c	Character that terminates copy.
n	Maximum number of characters to copy.

Return value

Returns a pointer to the wide character immediately following c in s1, or NULL if c was not found in the first n wide characters of s2.

Additional information

Copies at most n wide characters from the object pointed to by s2 into the object pointed to by s1. The copying stops as soon as n wide characters are copied or the wide character c is copied into the destination object pointed to by s1.

The behavior of *wmemccpy()* (page 559) is undefined if copying takes place between objects that overlap.

Notes

Conforms to POSIX.1-2008.

wmempcpy()

Description

Copy memory (GNU).

Prototype

```
wchar_t *wmempcpy(       wchar_t * s1,
                   const wchar_t * s2,
                         size_t    n);
```

Parameters

Parameter	Description
s1	Pointer to destination object.
s2	Pointer to source object.
n	Number of wide characters to copy.

Return value

Returns a pointer to the wide character immediately following the final wide character written into s1.

Additional information

Copies n wide characters from the object pointed to by s2 into the object pointed to by s1. The behavior of *wmempcpy()* (page 559) is undefined if copying takes place between objects that overlap.

Notes

This is an extension found in GNU libc.

wmemmove()

Description

Copy memory, tolerate overlaps.

Prototype

```
wchar_t *wmemmove(       wchar_t * s1,
                   const wchar_t * s2,
                         size_t    n);
```

Parameters

Parameter	Description
s1	Pointer to destination object.
s2	Pointer to source object.
n	Number of wide characters to copy.

Return value

Returns the value of s1.

Additional information

Copies n wide characters from the object pointed to by s2 into the object pointed to by s1 ensuring that if s1 and s2 overlap, the copy works correctly. Copying takes place as if the n wide characters from the object pointed to by s2 are first copied into a temporary array of n wide characters that does not overlap the objects pointed to by s1 and s2, and then the n wide characters from the temporary array are copied into the object pointed to by s1.

wcscpy()

Description

Copy string.

Prototype

```
wchar_t *wcscpy(       wchar_t * s1,
                 const wchar_t * s2);
```

Parameters

Parameter	Description
s1	Pointer to wide string to copy to.
s2	Pointer to wide string to copy.

Return value

Returns the value of s1.

Additional information

Copies the wide string pointed to by s2 (including the terminating null wide character) into the array pointed to by s1. The behavior of *wcscpy()* (page 560) is undefined if copying takes place between objects that overlap.

wcsncpy()

Description

Copy string, limit length.

Prototype

```
wchar_t *wcsncpy(      wchar_t * s1,
                 const wchar_t * s2,
                       size_t    n);
```

Parameters

Parameter	Description
s1	Pointer to wide string to copy to.
s2	Pointer to wide string to copy.
n	Maximum number of wide characters to copy.

Return value

Returns the value of s1.

Additional information

Copies not more than n wide characters from the array pointed to by s2 to the array pointed to by s1. Wide characters that follow a null wide character in s2 are not copied. The behavior of *wcsncpy()* (page 561) is undefined if copying takes place between objects that overlap. If the array pointed to by s2 is a wide string that is shorter than n wide characters, null wide characters are appended to the copy in the array pointed to by s1, until n characters in all have been written.

Notes

No wide null character is implicitly appended to the end of s1, so s1 will only be terminated by a wide null character if the length of the wide string pointed to by s2 is less than n.

wcslcpy()

Description

Copy string, limit length, always zero terminate (BSD).

Prototype

```
size_t wcslcpy(      wchar_t * s1,
               const wchar_t * s2,
                     size_t    n);
```

Parameters

Parameter	Description
s1	Pointer to wide string to copy to.
s2	Pointer to wide string to copy.
n	Maximum number of wide characters, including terminating null, in s1.

Return value

Returns the number of wide characters it tried to copy, which is the length of the wide string s2 or n, whichever is smaller.

Additional information

Copies up to n-1 wide characters from the wide string pointed to by s2 into the array pointed to by s1 and always terminates the result with a null character.

The behavior of *strlcpy()* (page 533) is undefined if copying takes place between objects that overlap.

Notes

Commonly found in BSD libraries and contrasts with *wcsncpy()* (page 561) in that the resulting string is always terminated with a null wide character.

wcscat()

Description

Concatenate strings.

Prototype

```
wchar_t *wcscat(       wchar_t * s1,
                 const wchar_t * s2);
```

Parameters

Parameter	Description
s1	Zero-terminated wide string to append to.
s2	Zero-terminated wide string to append.

Return value

Returns the value of s1.

Additional information

Appends a copy of the wide string pointed to by s2 (including the terminating null wide character) to the end of the wide string pointed to by s1. The initial character of s2 overwrites the null wide character at the end of s1. The behavior of *wcscat()* (page 562) is undefined if copying takes place between objects that overlap.

wcsncat()

Description

Concatenate strings, limit length.

Prototype

```
wchar_t *wcsncat(       wchar_t * s1,
                  const wchar_t * s2,
                        size_t    n);
```

Parameters

Parameter	Description
s1	Wide string to append to.
s2	Wide string to append.
n	Maximum number of wide characters in s1.

Return value

Returns the value of s1.

Additional information

Appends not more than n wide characters from the array pointed to by s2 to the end of the wide string pointed to by s1. A null wide character in s1 and wide characters that follow it are not appended. The initial wide character of s2 overwrites the null wide character at the end of s1. A terminating wide null character is always appended to the result. The behavior of *wcsncat()* (page 562) is undefined if copying takes place between objects that overlap.

wcslcat()

Description

Concatenate strings, limit length, always zero terminate (BSD).

Prototype

```
size_t wcslcat(       char    * s1,
               const char    * s2,
                     size_t    n);
```

Parameters

Parameter	Description
s1	Pointer to wide string to append to.
s2	Pointer to wide string to append.
n	Maximum number of characters, including terminating wide null, in s1.

Return value

Returns the number of wide characters it tried to copy, which is the sum of the lengths of the wide strings s1 and s2 or n, whichever is smaller.

Additional information

Appends no more than n-strlen(s1}-1 wide characters pointed to by s2 into the array pointed to by s1 and always terminates the result with a wide null character if n is greater than zero. Both the wide strings s1 and s2 must be terminated with a wide null character on entry to *wcslcat()* (page 563) and a character position for the terminating wide null should be included in n.

The behavior of *wcslcat()* (page 563) is undefined if copying takes place between objects that overlap.

Notes

Commonly found in BSD libraries.

wcsdup()

Description

Duplicate string (POSIX.1).

Prototype

```
wchar_t *wcsdup(const wchar_t * s);
```

Parameters

Parameter	Description
s	Pointer to wide string to duplicate.

Return value

Returns a pointer to the new wide string or a null pointer if the new wide string cannot be created. The returned pointer can be passed to *free()* (page 504).

Additional information

Duplicates the wide string pointed to by s by using *malloc()* (page 503) to allocate memory for a copy of s and then copies s, including the terminating null, to that memory

Notes

Conforms to POSIX.1-2008 and SC22 TR 24731-2.

wcsndup()

Description

Duplicate string, limit length (GNU).

Prototype

```
wchar_t *wcsndup(const wchar_t * s,
                 size_t      n);
```

Parameters

Parameter	Description
s	Pointer to wide string to duplicate.
n	Maximum number of wide characters to duplicate.

Return value

Returns a pointer to the new wide string or a null pointer if the new wide string cannot be created. The returned pointer can be passed to *free()* (page 504).

Additional information

Duplicates at most n wide characters from the the string pointed to by s by using *malloc()* (page 503) to allocate memory for a copy of s.

If the length of string pointed to by s is greater than n wide characters, only n wide characters will be duplicated. If n is greater than the length of the wide string pointed to by s, all characters in the string are copied into the allocated array including the terminating null character.

Notes

This is a GNU extension.

Comparison functions

Function	Description
wmemcmp() (page 565)	Compare memory.
wcsncmp() (page 565)	Compare strings, limit length.
wcscasecmp() (page 566)	Compare strings, ignore case (POSIX.1).
wcsncasecmp() (page 567)	Compare strings, ignore case, limit length (POSIX.1).

wmemcmp()

Description

Compare memory.

Prototype

```
int wmemcmp(const wchar_t * s1,
            const wchar_t * s2,
                  size_t    n);
```

Parameters

Parameter	Description
s1	Pointer to object #1.
s2	Pointer to object #2.
n	Number of wide characters to compare.

Return value

< 0	s1 is less than s2.
$= 0$	s1 is equal to s2.
> 0	s1 is greater than to s2.

Additional information

Compares the first n wide characters of the object pointed to by s1 to the first n wide characters of the object pointed to by s2. *wmemcmp()* (page 565) returns an integer greater than, equal to, or less than zero as the object pointed to by s1 is greater than, equal to, or less than the object pointed to by s2.

wcsncmp()

Description

Compare strings, limit length.

Prototype

```
int wcsncmp(const wchar_t * s1,
            const wchar_t * s2,
                  size_t    n);
```

Parameters

Parameter	Description
s1	Pointer to wide string #1.
s2	Pointer to wide string #2.
n	Maximum number of wide characters to compare.

Return value

Returns an integer greater than, equal to, or less than zero, if the possibly null-terminated array pointed to by s1 is greater than, equal to, or less than the possibly null-terminated array pointed to by s2.

Additional information

Compares not more than n wide characters from the array pointed to by s1 to the array pointed to by s2. Wide characters that follow a null wide character are not compared.

wcscasecmp()

Description

Compare strings, ignore case (POSIX.1).

Prototype

```
int wcscasecmp(const char * s1,
               const char * s2);
```

Parameters

Parameter	Description
s1	Pointer to wide string #1.
s2	Pointer to wide string #2.

Return value

< 0	s1 is less than s2.
= 0	s1 is equal to s2.
> 0	s1 is greater than to s2.

Additional information

Compares the wide string pointed to by s1 to the wide string pointed to by s2 ignoring differences in case.

wcscasecmp() (page 566) returns an integer greater than, equal to, or less than zero if the wide string pointed to by s1 is greater than, equal to, or less than the wide string pointed to by s2.

Notes

Conforms to POSIX.1-2017.

wcsncasecmp()

Description

Compare strings, ignore case, limit length (POSIX.1).

Prototype

```
int wcsncasecmp(const char   * s1,
                const char   * s2,
                      size_t   n);
```

Parameters

Parameter	Description
s1	Pointer to wide string #1.
s2	Pointer to wide string #2.
n	Maximum number of wide characters to compare.

Return value

< 0	s1 is less than s2.
= 0	s1 is equal to s2.
> 0	s1 is greater than to s2.

Additional information

Compares not more than n wide characters from the array pointed to by s1 to the array pointed to by s2 ignoring differences in case. Characters that follow a wide null character are not compared.

strncasecmp() (page 539) returns an integer greater than, equal to, or less than zero, if the possibly null-terminated array pointed to by s1 is greater than, equal to, or less than the possibly null-terminated array pointed to by s2.

Notes

Conforms to POSIX.1-2017.

Search functions

Function	Description
wmemchr() (page 568)	Find character in memory, forward.
wcschr() (page 568)	Find character within string, forward.
wcsnchr() (page 569)	Find character within string, forward, limit length.
wcsrchr() (page 569)	Find character within string, reverse.
wcslen() (page 570)	Calculate length of string.
wcsnlen() (page 570)	Calculate length of string, limit length (POSIX.1).
wcsstr() (page 570)	Find string within string, forward.
wcsnstr() (page 571)	Find string within string, forward, limit length (BSD).
wcspbrk() (page 572)	Find first occurrence of characters within string.
wcsspn() (page 572)	Compute size of string prefixed by a set of characters.
wcscspn() (page 573)	Compute size of string not prefixed by a set of characters.
wcstok() (page 573)	Break string into tokens.
wcssep() (page 574)	Break string into tokens (BSD).

wmemchr()

Description

Find character in memory, forward.

Prototype

```
wchar_t *wmemchr(const wchar_t * s,
                       wchar_t   c,
                       size_t    n);
```

Parameters

Parameter	Description
s	Pointer to object to search.
c	Wide character to search for.
n	Number of wide characters in object to search.

Return value

= NULL	c does not occur in the object.
≠ NULL	Pointer to the located wide character.

Additional information

Locates the first occurrence of c in the initial n wide characters of the object pointed to by s. Unlike *wcschr()* (page 568), *wmemchr()* (page 568) does not terminate a search when a null wide character is found in the object pointed to by s.

wcschr()

Description

Find character within string, forward.

Prototype

```
wchar_t *wcschr(const wchar_t * s,
                      wchar_t   c);
```

Parameters

Parameter	Description
s	Wide string to search.
c	Wide character to search for.

Return value

Returns a pointer to the located wide character, or a null pointer if c does not occur in the wide string.

Additional information

Locates the first occurrence of c in the wide string pointed to by s. The terminating wide null character is considered to be part of the string.

wcsnchr()

Description

Find character within string, forward, limit length.

Prototype

```
wchar_t *wcsnchr(const wchar_t * s,
                 size_t    n,
                 wchar_t   c);
```

Parameters

Parameter	Description
s	Pointer to wide string to search.
n	Number of wide characters to search.
c	Wide character to search for.

Return value

Returns a pointer to the located wide character, or a null pointer if c does not occur in the string.

Additional information

Searches not more than n wide characters to locate the first occurrence of c in the wide string pointed to by s. The terminating wide null character is considered to be part of the wide string.

wcsrchr()

Description

Find character within string, reverse.

Prototype

```
wchar_t *wcsrchr(const wchar_t * s,
                 wchar_t   c);
```

Parameters

Parameter	Description
s	Pointer to wide string to search.
c	Wide character to search for.

Return value

Returns a pointer to the located wide character, or a null pointer if c does not occur in the string.

Additional information

Locates the last occurrence of c in the wide string pointed to by s. The terminating wide null character is considered to be part of the string.

wcslen()

Description

Calculate length of string.

Prototype

```
size_t wcslen(const wchar_t * s);
```

Parameters

Parameter	Description
s	Pointer to zero-terminated wide string.

Return value

Returns the length of the wide string pointed to by s, that is the number of wide characters that precede the terminating wide null character.

wcsnlen()

Description

Calculate length of string, limit length (POSIX.1).

Prototype

```
size_t wcsnlen(const wchar_t * s,
                     size_t    n);
```

Parameters

Parameter	Description
s	Pointer to wide string.
n	Maximum number of wide characters to examine.

Return value

Returns the length of the wide string pointed to by s, up to a maximum of n wide characters. *wcsnlen()* (page 570) only examines the first n wide characters of the string s.

Notes

Conforms to POSIX.1-2008.

wcsstr()

Description

Find string within string, forward.

Prototype

```
wchar_t *wcsstr(const wchar_t * s1,
                const wchar_t * s2);
```

Parameters

Parameter	Description
s1	Pointer to wide string to search.
s2	Pointer to wide string to search for.

Return value

Returns a pointer to the located wide string, or a null pointer if the wide string is not found. If s2 points to a wide string with zero length, *wcsstr()* (page 570) returns s1.

Additional information

Locates the first occurrence in the wide string pointed to by s1 of the sequence of wide characters (excluding the terminating null wide character) in the wide string pointed to by s2.

wcsnstr()

Description

Find string within string, forward, limit length (BSD).

Prototype

```
wchar_t *wcsnstr(const wchar_t * s1,
                 const wchar_t * s2,
                       size_t    n);
```

Parameters

Parameter	Description
s1	Pointer to wide string to search.
s2	Pointer to wide string to search for.
n	Maximum number of characters to search for.

Return value

Returns a pointer to the located wide string, or a null pointer if the wide string is not found. If s2 points to a wide string with zero length, *wcsnstr()* (page 571) returns s1.

Additional information

Searches at most n wide characters to locate the first occurrence in the wide string pointed to by s1 of the sequence of wide characters (excluding the terminating wide null character) in the string pointed to by s2.

Notes

Commonly found in Linux and BSD C libraries.

wcspbrk()

Description

Find first occurrence of characters within string.

Prototype

```
wchar_t *wcspbrk(const wchar_t * s1,
                 const wchar_t * s2);
```

Parameters

Parameter	Description
s1	Pointer to wide string to search.
s2	Pointer to wide string to search for.

Return value

Returns a pointer to the first wide character, or a null pointer if no wide character from s2 occurs in s1.

Additional information

Locates the first occurrence in the wide string pointed to by s1 of any wide character from the string pointed to by s2.

wcsspn()

Description

Compute size of string prefixed by a set of characters.

Prototype

```
size_t wcsspn(const wchar_t * s1,
              const wchar_t * s2);
```

Parameters

Parameter	Description
s1	Pointer to zero-terminated wide string to search.
s2	Pointer to zero-terminated acceptable-set wide string.

Return value

Returns the length of the wide string pointed to by s1 which consists entirely of wide characters from the wide string pointed to by s2

Additional information

Computes the length of the maximum initial segment of the wide string pointed to by s1 which consists entirely of wide characters from the string pointed to by s2.

wcscspn()

Description

Compute size of string not prefixed by a set of characters.

Prototype

```
size_t wcscspn(const wchar_t * s1,
               const wchar_t * s2);
```

Parameters

Parameter	Description
s1	Pointer to wide string to search.
s2	Pointer to wide string containing characters to skip.

Return value

Returns the length of the segment of wide string s1 prefixed by wide characters from s2.

Additional information

Computes the length of the maximum initial segment of the wide string pointed to by s1 which consists entirely of wide characters not from the wide string pointed to by s2.

wcstok()

Description

Break string into tokens.

Prototype

```
wchar_t *wcstok(      wchar_t  * s1,
                const wchar_t  * s2,
                      wchar_t ** ptr);
```

Parameters

Parameter	Description
s1	Pointer to zero-terminated wide string to parse.
s2	Pointer to zero-terminated set of separators.
ptr	Pointer to object that maintains parse state.

Return value

NULL if no further tokens else a pointer to the next token.

Additional information

A sequence of calls to *wcstok()* (page 573) breaks the wide string pointed to by s1 into a sequence of tokens, each of which is delimited by a wide character from the wide string pointed to by s2. The first call in the sequence has a non-null first argument; subsequent calls in the sequence have a null first argument. The separator wide string pointed to by s2 may be different from call to call.

The first call in the sequence searches the wide string pointed to by s1 for the wide first character that is not contained in the current separator wide string pointed to by s2. If no such wide character is found, then there are no tokens in the string pointed to by s1 and *wcstok()* (page 573) returns a null pointer. If such a wide character is found, it is the start of the first token.

wcstok() (page 573) then searches from there for a wide character that is contained in the current separator wide string. If no such wide character is found, the current token extends to the end of the wide string pointed to by s1, and subsequent searches for a token will return a null pointer. If such a wide character is found, it is overwritten by a null wide character, which terminates the current token. *wcstok()* (page 573) saves a pointer to the following wide character, from which the next search for a token will start.

Each subsequent call, with a null pointer as the value of the first argument, starts searching from the saved pointer and behaves as described above.

See also

wcssep() (page 574).

wcssep()

Description

Break string into tokens (BSD).

Prototype

```
wchar_t *wcssep(       wchar_t ** stringp,
                 const wchar_t  * delim);
```

Parameters

Parameter	Description
stringp	Pointer to pointer to zero-terminated wide string.
delim	Pointer to delimiter set wide string.

Return value

See below.

Additional information

Locates, in the wide string referenced by *stringp, the first occurrence of any wide character in the wide string delim (or the terminating null character) and replaces it with a null wide character. The location of the next wide character after the delimiter wide character (or NULL, if the end of the wide string was reached) is stored in *stringp. The original value of *stringp is returned.

An empty field (that is, a wide character in the string delim occurs as the first character of *stringp) can be detected by comparing the location referenced by the returned pointer to the null wide character.

If *stringp is initially null, *wcssep()* (page 574) returns null.

Notes

Commonly found in Linux and BSD C libraries.

Multi-byte/wide string conversion functions

Function	Description
mbsinit() (page 575)	Query initial conversion state.
mbrlen() (page 575)	Count number of bytes in multi-byte character, restartable.
mbrlen_l() (page **??**)	Count number of bytes in multi-byte character, restartable, per locale (POSIX.1).
mbrtowc() (page 577)	Convert multi-byte character to wide character, restartable.
mbrtowc_l() (page **??**)	Convert multi-byte character to wide character, restartable, per locale (POSIX.1).
wctob() (page 578)	Convert wide character to single-byte character.
wctob_l() (page **??**)	Convert wide character to single-byte character, per locale (POSIX.1).
wcrtomb() (page 579)	Convert wide character to multi-byte character, restartable.
wcrtomb_l() (page **??**)	Convert wide character to multi-byte character, restartable, per locale (POSIX.1).
wcsrtombs() (page 580)	Convert wide string to multi-byte string, restartable.
wcsrtombs_l() (page **??**)	Convert wide string to multi-byte string, restartable (POSIX.1).

mbsinit()

Description

Query initial conversion state.

Prototype

```
int mbsinit(const mbstate_t * ps);
```

Parameters

Parameter	Description
ps	Pointer to conversion state.

Return value

Returns nonzero (true) if ps is a null pointer or if the pointed-to object describes an initial conversion state; otherwise, returns zero.

mbrlen()

Description

Count number of bytes in multi-byte character, restartable.

Prototype

```
size_t mbrlen(const char      * s,
              size_t            n,
              mbstate_t       * ps);
```

Parameters

Parameter	Description
s	Pointer to multi-byte character.
n	Maximum number of bytes to examine.
ps	Pointer to multi-byte conversion state.

Return value

Number of bytes in multi-byte character.

Additional information

Determines the number of bytes contained in the multi-byte character pointed to by s in the current locale.

Except that except that the expression designated by ps is evaluated only once, this function is equivalent to the call:

```
mbrtowc(NULL, s, n, ps != NULL ? ps : &internal);
```

where internal is the mbstate_t object for the *mbrlen()* (page 575) function.

See also

mbrlen_l() (page **??**), *mbrtowc()* (page 577)

mbrlen_l()

Description

Count number of bytes in multi-byte character, restartable, per locale (POSIX.1).

Prototype

```
size_t mbrlen_l(const char      * s,
                size_t            n,
                mbstate_t       * ps,
                                  locale_t loc);
```

Parameters

Parameter	Description
s	Pointer to multi-byte character.
n	Maximum number of bytes to examine.
ps	Pointer to multi-byte conversion state.
loc	Locale used for conversion.

Return value

Number of bytes in multi-byte character.

Additional information

Determines the number of bytes contained in the multi-byte character pointed to by s in the locale loc.

Except that except that the expression designated by ps is evaluated only once, this function is equivalent to the call:

```
mbrtowc_l(NULL, s, n, ps != NULL ? ps : &internal, loc);
```

where internal is the mbstate_t object for the *mbrlen()* (page 575) function,

Notes

Conforms to POSIX.1-2008.

See also

mbrlen_l() (page **??**), *mbrtowc()* (page 577)

mbrtowc()

Description

Convert multi-byte character to wide character, restartable.

Prototype

```
size_t mbrtowc(       wchar_t   * pwc,
               const char      * s,
                     size_t      n,
                     mbstate_t * ps);
```

Parameters

Parameter	Description
pwc	Pointer to object that receives the wide character.
s	Pointer to multi-byte character string.
n	Maximum number of bytes that will be examined.
ps	Pointer to multi-byte conversion state.

Return value

If s is a null pointer, *mbrtowc()* (page 577) is equivalent to mbrtowc(NULL, “” , 1, ps), ignoring pwc and n.

If s is not null and the object that s points to is a wide null character, *mbrtowc()* (page 577) returns 0.

If s is not null and the object that s points to forms a valid multi-byte character in the current locale with a most n bytes, *mbrtowc()* (page 577) returns the length in bytes of the multi-byte character and stores that wide character to the object pointed to by pwc (if pwc is not null).

If the object that s points to forms an incomplete, but possibly valid, multi-byte character, *mbrtowc()* (page 577) returns -2.

If the object that s points to does not form a partial multi-byte character, *mbrtowc()* (page 577) returns -1.

Additional information

Converts a single multi-byte character to a wide character in the current locale.

See also

mbtowc() (page 522), *mbrtowc_l()* (page **??**)

mbrtowc_l()

Description

Convert multi-byte character to wide character, restartable, per locale (POSIX.1).

Prototype

```
size_t mbrtowc_l(       wchar_t   * pwc,
                 const char      * s,
                       size_t      n,
                       mbstate_t * ps,
                                  locale_t loc);
```

Parameters

Parameter	Description
pwc	Pointer to object that receives the wide character.
s	Pointer to multi-byte character string.
n	Maximum number of bytes that will be examined.
ps	Pointer to multi-byte conversion state.
loc	Locale used for conversion.

Return value

If s is a null pointer, *mbrtowc()* (page 577) is equivalent to mbrtowc(NULL, "", 1, ps), ignoring pwc and n.

If s is not null and the object that s points to is a wide null character, *mbrtowc()* (page 577) returns 0.

If s is not null and the object that s points to forms a valid multi-byte character in the locale loc with a most n bytes, *mbrtowc()* (page 577) returns the length in bytes of the multi-byte character and stores that wide character to the object pointed to by pwc (if pwc is not null).

If the object that s points to forms an incomplete, but possibly valid, multi-byte character, *mbrtowc()* (page 577) returns -2.

If the object that s points to does not form a partial multi-byte character, *mbrtowc()* (page 577) returns -1.

Additional information

Converts a single multi-byte character to a wide character in the locale loc.

Notes

Conforms to POSIX.1-2008.

See also

mbtowc() (page 522), *mbrtowc_l()* (page **??**)

wctob()

Description

Convert wide character to single-byte character.

Prototype

```
int wctob(wint_t c);
```

Parameters

Parameter	Description
c	Character to convert.

Return value

Returns EOF if c does not correspond to a multi-byte character with length one in the initial shift state in the current locale. Otherwise, it returns the single-byte representation of that character as an unsigned char converted to an int.

Additional information

Determines whether c corresponds to a member of the extended character set whose multi-byte character representation is a single byte in the current locale when in the initial shift state.

wctob_l()

Description

Convert wide character to single-byte character, per locale (POSIX.1).

Prototype

```
int wctob_l(wint_t c,
                    locale_t loc);
```

Parameters

Parameter	Description
c	Character to convert.
loc	Locale used for conversion.

Return value

Returns EOF if c does not correspond to a multi-byte character with length one in the initial shift state in the locale loc. Otherwise, it returns the single-byte representation of that character as an unsigned char converted to an int.

Additional information

Determines whether c corresponds to a member of the extended character set whose multi-byte character representation is a single byte in the locale loc when in the initial shift state.

wcrtomb()

Description

Convert wide character to multi-byte character, restartable.

Prototype

```
size_t wcrtomb(char        * s,
                     wchar_t wc,
               mbstate_t * ps);
```

Parameters

Parameter	Description
s	Pointer to array that receives the multi-byte character.
wc	Wide character to convert.
ps	Pointer to multi-byte conversion state.

Return value

Returns the number of bytes stored in the array object. When wc is not a valid wide character, an encoding error occurs: *wcrtomb()* (page 579) stores the value of the macro EILSEQ in errno and returns (*size_t* (page **??**))(-1); the conversion state is unspecified.

Additional information

If s is a null pointer, *wcrtomb()* (page 579) is equivalent to the call wcrtomb(buf, 0, ps) where buf is an internal buffer.

If s is not a null pointer, *wcrtomb()* (page 579) determines the number of bytes needed to represent the multi-byte character that corresponds to the wide character given by wc in the locale loc, and stores the multi-byte character representation in the array whose first element is pointed to by s. At most MB_CUR_MAX bytes are stored. If wc is a null wide character, a null byte is stored; the resulting state described is the initial conversion state.

wcrtomb_l()

Description

Convert wide character to multi-byte character, restartable, per locale (POSIX.1).

Prototype

```
size_t wcrtomb_l(char       * s,
                            wchar_t wc,
                 mbstate_t * ps,
                            locale_t loc);
```

Parameters

Parameter	Description
s	Pointer to array that receives the multi-byte character.
wc	Wide character to convert.
ps	Pointer to multi-byte conversion state.
loc	Locale used for conversion.

Return value

Returns the number of bytes stored in the array object. When wc is not a valid wide character, an encoding error occurs: *wcrtomb_l()* (page **??**) stores the value of the macro EILSEQ in errno and returns (*size_t* (page **??**))(-1); the conversion state is unspecified.

Additional information

If s is a null pointer, *wcrtomb()* (page 579) is equivalent to the call wcrtomb(buf, 0, ps) where buf is an internal buffer.

If s is not a null pointer, *wcrtomb()* (page 579) determines the number of bytes needed to represent the multi-byte character that corresponds to the wide character given by wc in the current locale, and stores the multi-byte character representation in the array whose first element is pointed to by s. At most MB_CUR_MAX bytes are stored. If wc is a null wide character, a null byte is stored; the resulting state described is the initial conversion state.

wcsrtombs()

Description

Convert wide string to multi-byte string, restartable.

Prototype

```
size_t wcsrtombs(      char        * dst,
                 const wchar_t    ** src,
                       size_t        len,
                       mbstate_t   * ps);
```

Parameters

Parameter	Description
dst	Pointer to array that receives the multi-byte string.
src	Indirect pointer to wide character string being converted.
len	Maximum number of bytes to write into the array pointed to by dst.
ps	Pointer to multi-byte conversion state.

Return value

If conversion stops because a wide character is reached that does not correspond to a valid multi-byte character, an encoding error occurs: *wcsrtombs()* (page 580) stores the value of the macro EILSEQ in errno and returns (*size_t* (page **??**))(-1); the conversion state is unspecified. Otherwise, it returns the number of bytes in the resulting multi-byte character sequence, not including the terminating null character (if any).

Additional information

Converts a sequence of wide characters in the current locale from the array indirectly pointed to by src into a sequence of corresponding multi-byte characters that begins in the conversion state described by the object pointed to by ps. If dst is not a null pointer, the converted characters are then stored into the array pointed to by dst. Conversion continues up to and including a terminating null wide character, which is also stored.

Conversion stops earlier in two cases: when a wide character is reached that does not correspond to a valid multi-byte character, or (if dst is not a null pointer) when the next multi-byte character would exceed the limit of len total bytes to be stored into the array pointed to by dst. Each conversion takes place as if by a call to *wcrtomb()* (page 579).

If dst is not a null pointer, the pointer object pointed to by src is assigned either a null pointer (if conversion stopped due to reaching a terminating null wide character) or the address just past the last wide character converted (if any). If conversion stopped due to reaching a terminating null wide character, the resulting state described is the initial conversion state.

wcsrtombs_l()

Description

Convert wide string to multi-byte string, restartable (POSIX.1).

Prototype

```
size_t wcsrtombs_l(       char       * dst,
                   const wchar_t    ** src,
                         size_t        len,
                         mbstate_t   * ps,
                                       locale_t loc);
```

Parameters

Parameter	Description
dst	Pointer to array that receives the multi-byte string.
src	Indirect pointer to wide character string being converted.
len	Maximum number of bytes to write into the array pointed to by dst.
ps	Pointer to multi-byte conversion state.
loc	Locale used for conversion.

Return value

If conversion stops because a wide character is reached that does not correspond to a valid multi-byte character, an encoding error occurs: *wcsrtombs()* (page 580) stores the value of the macro EILSEQ in errno and returns (*size_t* (page **??**))(-1); the conversion state is unspecified. Otherwise, it returns the number of bytes in the resulting multi-byte character sequence, not including the terminating null character (if any).

Additional information

Converts a sequence of wide characters in the locale loc from the array indirectly pointed to by src into a sequence of corresponding multi-byte characters that begins in the conversion state described by the object pointed to by ps. If dst is not a null pointer, the converted characters are then stored into the array pointed to by dst. Conversion continues up to and including a terminating null wide character, which is also stored.

Conversion stops earlier in two cases: when a wide character is reached that does not correspond to a valid multi-byte character, or (if dst is not a null pointer) when the next multi-byte character would exceed the limit of len total bytes to be stored into the array pointed to by dst. Each conversion takes place as if by a call to *wcrtomb_l()* (page **??**).

If dst is not a null pointer, the pointer object pointed to by src is assigned either a null pointer (if conversion stopped due to reaching a terminating null wide character) or the address just past the last wide character converted (if any). If conversion stopped due to reaching a terminating null wide character, the resulting state described is the initial conversion state.

Notes

Conforms to POSIX.1-2008.

4.6.20 <wctype.h>

Classification functions

Function	Description
iswcntrl() (page 583)	Is character a control?
iswcntrl_l() (page **??**)	Is character a control, per locale? (POSIX.1).
iswblank() (page 583)	Is character a blank?
iswblank_l() (page **??**)	Is character a blank, per locale? (POSIX.1).
iswspace() (page 584)	Is character a whitespace character?
iswspace_l() (page **??**)	Is character a whitespace character, per locale? (POSIX.1).
iswpunct() (page 585)	Is character a punctuation mark?
iswpunct_l() (page **??**)	Is character a punctuation mark, per locale? (POSIX.1).
iswdigit() (page 586)	Is character a decimal digit?
iswdigit_l() (page **??**)	Is character a decimal digit, per locale? (POSIX.
iswxdigit() (page 587)	Is character a hexadecimal digit?
iswxdigit_l() (page **??**)	Is character a hexadecimal digit, per locale? (POSIX.1).
iswalpha() (page 587)	Is character alphabetic?
iswalpha_l() (page **??**)	Is character alphabetic, per locale? (POSIX.1).
iswalnum() (page 588)	Is character alphanumeric?
iswalnum_l() (page **??**)	Is character alphanumeric, per locale? (POSIX.1).
iswupper() (page 589)	Is character an uppercase letter?
iswupper_l() (page **??**)	Is character an uppercase letter, per locale? (POSIX.1).
iswlower() (page 590)	Is character a lowercase letter?
iswlower_l() (page **??**)	Is character a lowercase letter, per locale? (POSIX.1).
iswprint() (page 591)	Is character printable?
iswprint_l() (page **??**)	Is character printable, per locale? (POSIX.1).
iswgraph() (page 591)	Is character any printing character?
iswgraph_l() (page **??**)	Is character any printing character, per locale? (POSIX.1).
iswctype() (page 592)	Construct character mapping.
iswctype_l() (page **??**)	Construct character mapping, per locale (POSIX.1).
wctype() (page 593)	Construct character class.

iswcntrl()

Description

Is character a control?

Prototype

```
int iswcntrl(wint_t c);
```

Parameters

Parameter	Description
c	Wide character to test.

Return value

Returns nonzero (true) if and only if the value of the argument c is a control character in the current locale.

iswcntrl_l()

Description

Is character a control, per locale? (POSIX.1).

Prototype

```
int iswcntrl_l(wint_t c,
               locale_t loc);
```

Parameters

Parameter	Description
c	Wide character to test.
loc	Locale used to test c.

Return value

Returns nonzero (true) if and only if the value of the argument c is a control character in the locale loc.

Notes

Conforms to POSIX.1-2017.

iswblank()

Description

Is character a blank?

Prototype

```
int iswblank(wint_t c);
```

Parameters

Parameter	Description
c	Wide character to test.

Return value

Returns nonzero (true) if and only if the value of the argument c is either a space character or tab character in the current locale.

iswblank_l()

Description

Is character a blank, per locale? (POSIX.1).

Prototype

```
int iswblank_l(wint_t c,
                          locale_t loc);
```

Parameters

Parameter	Description
c	Wide character to test.
loc	Locale used to test c.

Return value

Returns nonzero (true) if and only if the value of the argument c is either a space character or the tab character in locale loc.

Notes

Conforms to POSIX.1-2017.

iswspace()

Description

Is character a whitespace character?

Prototype

```
int iswspace(wint_t c);
```

Parameters

Parameter	Description
c	Wide character to test.

Return value

Returns nonzero (true) if and only if the value of the argument c is a standard white-space character in the current locale. The standard white-space characters are space, form feed, new-line, carriage return, horizontal tab, and vertical tab.

iswspace_l()

Description

Is character a whitespace character, per locale? (POSIX.1).

Prototype

```
int iswspace_l(wint_t c,
               locale_t loc);
```

Parameters

Parameter	Description
c	Wide character to test.
loc	Locale used to test c.

Return value

Returns nonzero (true) if and only if the value of the argument c is a standard white-space character in the locale loc.

Notes

Conforms to POSIX.1-2017.

iswpunct()

Description

Is character a punctuation mark?

Prototype

```
int iswpunct(wint_t c);
```

Parameters

Parameter	Description
c	Wide character to test.

Return value

Returns nonzero (true) for every printing character for which neither *isspace()* (page 367) nor *isalnum()* (page 371) is true in the current locale.

iswpunct_l()

Description

Is character a punctuation mark, per locale? (POSIX.1).

Prototype

```
int iswpunct_l(wint_t c,
               locale_t loc);
```

Parameters

Parameter	Description
c	Wide character to test.
loc	Locale used to test c.

Return value

Returns nonzero (true) for every printing character for which neither *isspace()* (page 367) nor *isalnum()* (page 371) is true in the locale loc.

Notes

Conforms to POSIX.1-2017.

iswdigit()

Description

Is character a decimal digit?

Prototype

```
int iswdigit(wint_t c);
```

Parameters

Parameter	Description
c	Wide character to test.

Return value

Returns nonzero (true) if and only if the value of the argument c is a digit in the current locale.

iswdigit_l()

Description

Is character a decimal digit, per locale? (POSIX.1)

Prototype

```
int iswdigit_l(wint_t c,
               locale_t loc);
```

Parameters

Parameter	Description
c	Wide character to test.
loc	Locale used to test c.

Return value

Returns nonzero (true) if and only if the value of the argument c is a digit in the locale loc.

Notes

Conforms to POSIX.1-2017.

iswxdigit()

Description

Is character a hexadecimal digit?

Prototype

```
int iswxdigit(wint_t c);
```

Parameters

Parameter	Description
c	Wide character to test.

Return value

Returns nonzero (true) if and only if the value of the argument c is a hexadecimal digit in the current locale.

iswxdigit_l()

Description

Is character a hexadecimal digit, per locale? (POSIX.1).

Prototype

```
int iswxdigit_l(wint_t c,
                locale_t loc);
```

Parameters

Parameter	Description
c	Wide character to test.
loc	Locale used to test c.

Return value

Returns nonzero (true) if and only if the value of the argument c is a hexadecimal digit in the current locale.

Notes

Conforms to POSIX.1-2017.

iswalpha()

Description

Is character alphabetic?

Prototype

```
int iswalpha(wint_t c);
```

Parameters

Parameter	Description
c	Wide character to test.

Return value

Returns true if the character c is alphabetic in the current locale. That is, any character for which *iswupper()* (page 589) or *iswlower()* (page 590) returns true is considered alphabetic in addition to any of the locale-specific set of alphabetic characters for which none of *iswcntrl()* (page 583), *iswdigit()* (page 586), *iswpunct()* (page 585), or *isspace()* (page 367) is true.

In the C locale, *isalpha()* (page 370) returns nonzero (true) if and only if *isupper()* (page 372) or *islower()* (page 373) return true for value of the argument c.

iswalpha_l()

Description

Is character alphabetic, per locale? (POSIX.1).

Prototype

```
int iswalpha_l(wint_t c,
               locale_t loc);
```

Parameters

Parameter	Description
c	Wide character to test.
loc	Locale used to test c.

Return value

Returns true if the wide character c is alphabetic in the locale loc. That is, any character for which *iswupper()* (page 589) or *iswlower()* (page 590) returns true is considered alphabetic in addition to any of the locale-specific set of alphabetic characters for which none of *iswcntrl_l()* (page **??**), *iswdigit_l()* (page **??**), *iswpunct_l()* (page **??**), or *iswspace_l()* (page **??**) is true in the locale loc.

In the C locale, *iswalpha_l()* (page **??**) returns nonzero (true) if and only if *iswupper_l()* (page **??**) or *iswlower_l()* (page **??**) return true for value of the argument c.

Notes

Conforms to POSIX.1-2017.

iswalnum()

Description

Is character alphanumeric?

Prototype

```
int iswalnum(wint_t c);
```

Parameters

Parameter	Description
c	Wide character to test.

Return value

Returns nonzero (true) if and only if the value of the argument c is an alphabetic or numeric character in the current locale.

iswalnum_l()

Description

Is character alphanumeric, per locale? (POSIX.1).

Prototype

```
int iswalnum_l(wint_t c,
                       locale_t loc);
```

Parameters

Parameter	Description
c	Wide character to test.
loc	Locale used to test c.

Return value

Returns nonzero (true) if and only if the value of the argument c is an alphabetic or numeric character in the locale loc.

Notes

Conforms to POSIX.1-2017.

iswupper()

Description

Is character an uppercase letter?

Prototype

```
int iswupper(wint_t c);
```

Parameters

Parameter	Description
c	Wide character to test.

Return value

Returns nonzero (true) if and only if the value of the argument c is an uppercase letter in the current locale.

iswupper_l()

Description

Is character an uppercase letter, per locale? (POSIX.1).

Prototype

```
int iswupper_l(wint_t c,
                       locale_t loc);
```

Parameters

Parameter	Description
c	Wide character to test.
loc	Locale used to test c.

Return value

Returns nonzero (true) if and only if the value of the argument c is an uppercase letter in the locale loc.

Notes

Conforms to POSIX.1-2017.

iswlower()

Description

Is character a lowercase letter?

Prototype

```
int iswlower(wint_t c);
```

Parameters

Parameter	Description
c	Wide character to test.

Return value

Returns nonzero (true) if and only if the value of the argument c is a lowercase letter in the current locale.

iswlower_l()

Description

Is character a lowercase letter, per locale? (POSIX.1).

Prototype

```
int iswlower_l(wint_t c,
               locale_t loc);
```

Parameters

Parameter	Description
c	Wide character to test.
loc	Locale used to test c.

Return value

Returns nonzero (true) if and only if the value of the argument c is a lowercase letter in the locale loc.

Notes

Conforms to POSIX.1-2017.

iswprint()

Description

Is character printable?

Prototype

```
int iswprint(wint_t c);
```

Parameters

Parameter	Description
c	Wide character to test.

Return value

Returns nonzero (true) if and only if the value of the argument c is any printing character including space in the current locale.

iswprint_l()

Description

Is character printable, per locale? (POSIX.1).

Prototype

```
int iswprint_l(wint_t c,
               locale_t loc);
```

Parameters

Parameter	Description
c	Wide character to test.
loc	Locale used to test c.

Return value

Returns nonzero (true) if and only if the value of the argument c is any printing character including space in the locale loc.

Notes

Conforms to POSIX.1-2017.

iswgraph()

Description

Is character any printing character?

Prototype

```
int iswgraph(wint_t c);
```

Parameters

Parameter	Description
c	Wide character to test.

Return value

Returns nonzero (true) if and only if the value of the argument c is any printing character except space in the current locale.

iswgraph_l()

Description

Is character any printing character, per locale? (POSIX.1).

Prototype

```
int iswgraph_l(wint_t c,
               locale_t loc);
```

Parameters

Parameter	Description
c	Wide character to test.
loc	Locale used to test c.

Return value

Returns nonzero (true) if and only if the value of the argument c is any printing character except space in the locale loc.

Notes

Conforms to POSIX.1-2017.

iswctype()

Description

Construct character mapping.

Prototype

```
int iswctype(wint_t   c,
             wctype_t t);
```

Parameters

Parameter	Description
c	Wide character to test.
t	Property to test, typically delivered by calling *wctype()* (page 593).

Return value

Returns nonzero (true) if and only if the wide character c has the property t in the current locale.

Additional information

Determines whether the wide character c has the property described by t in the current locale.

iswctype_l()

Description

Construct character mapping, per locale (POSIX.1).

Prototype

```
int iswctype_l(wint_t   c,
               wctype_t t,
                        locale_t loc);
```

Parameters

Parameter	Description
c	Wide character to test.
t	Property to test, typically delivered by calling wctype_l().
loc	Locale used for mapping.

Return value

Returns nonzero (true) if and only if the wide character c has the property t in the locale loc.

Additional information

Determines whether the wide character c has the property described by t in the locale loc.

Notes

Conforms to POSIX.1-2017.

wctype()

Description

Construct character class.

Prototype

```
wctype_t wctype(char const * name);
```

Parameters

Parameter	Description
name	Name of mapping.

Return value

Character class; zero if class unrecognized.

Additional information

Constructs a value of type wctype_t that describes a class of wide characters identified by the string argument property.

If property identifies a valid class of wide characters in the current locale, returns a nonzero value that is valid as the second argument to *iswctype()* (page 592); otherwise, it returns zero.

Notes

The only mappings supported are:

- “alnum”
- “alpha”,
- “blank”
- “cntrl”
- “digit”
- “graph”
- “lower”
- “print”
- “punct”
- “space”
- “upper”
- “xdigit”

Conversion functions

Function	Description
towupper() (page 594)	Convert uppercase character to lowercase.
towupper_l() (page **??**)	Convert uppercase character to lowercase, per locale (POSIX.1).
towlower() (page 595)	Convert uppercase character to lowercase.
towlower_l() (page **??**)	Convert uppercase character to lowercase, per locale (POSIX.1).
towctrans() (page 596)	Translate character.
towctrans_l() (page **??**)	Translate character, per locale (POSIX.1).
wctrans() (page 597)	Construct character mapping.
wctrans_l() (page **??**)	Construct character mapping, per locale (POSIX.1).

towupper()

Description

Convert uppercase character to lowercase.

Prototype

```
wint_t towupper(wint_t c);
```

Parameters

Parameter	Description
c	Wide character to convert.

Return value

Converted wide character.

Additional information

Converts a lowercase letter to a corresponding uppercase letter.

If the argument c is a wide character for which *iswlower()* (page 590) is true and there are one or more corresponding wide characters, in the current current locale, for which *iswupper()* (page 589) is true, *towupper()* (page 594) returns one (and always the same one for any given locale) of the corresponding wide characters; otherwise, c is returned unchanged.

towupper_l()

Description

Convert uppercase character to lowercase, per locale (POSIX.1).

Prototype

```
wint_t towupper_l(wint_t c,
                  locale_t loc);
```

Parameters

Parameter	Description
c	Wide character to convert.
loc	Locale used to convert c.

Return value

Converted wide character.

Additional information

Converts a lowercase letter to a corresponding uppercase letter.

If the argument c is a wide character for which *iswlower_l()* (page **??**) is true and there are one or more corresponding wide characters, in the current locale loc, for which *iswupper_l()* (page **??**) is true, *towupper_l()* (page **??**) returns one (and always the same one for any given locale) of the corresponding wide characters; otherwise, c is returned unchanged.

Notes

Conforms to POSIX.1-2017.

towlower()

Description

Convert uppercase character to lowercase.

Prototype

```
wint_t towlower(wint_t c);
```

Parameters

Parameter	Description
c	Wide character to convert.

Return value

Converted wide character.

Additional information

Converts an uppercase letter to a corresponding lowercase letter.

If the argument c is a wide character for which *iswupper()* (page 589) is true and there are one or more corresponding wide characters, in the current locale, for which *iswlower()* (page 590) is true, *towlower()* (page 595) returns one (and always the same one for any given locale) of the corresponding wide characters; otherwise, c is returned unchanged.

towlower_l()

Description

Convert uppercase character to lowercase, per locale (POSIX.1).

Prototype

```
wint_t towlower_l(wint_t c,
                          locale_t loc);
```

Parameters

Parameter	Description
c	Wide character to convert.
loc	Locale used to convert c.

Return value

Converted wide character.

Additional information

Converts an uppercase letter to a corresponding lowercase letter.

If the argument c is a wide character for which *iswupper_l()* (page **??**) is true and there are one or more corresponding wide characters, in the locale loc, for which *iswlower_l()* (page **??**) is true, *towlower_l()* (page **??**) returns one (and always the same one for any given locale) of the corresponding wide characters; otherwise, c is returned unchanged.

Notes

Conforms to POSIX.1-2017.

towctrans()

Description

Translate character.

Prototype

```
wint_t towctrans(wint_t    c,
                 wctrans_t t);
```

Parameters

Parameter	Description
c	Wide character to convert.
t	Mapping to use for conversion.

Return value

Converted wide character.

Additional information

Maps the wide character c using the mapping described by t in the current locale.

towctrans_l()

Description

Translate character, per locale (POSIX.1).

Prototype

```
wint_t towctrans_l(wint_t    c,
                   wctrans_t t,
                             locale_t loc);
```

Parameters

Parameter	Description
c	Wide character to convert.
t	Mapping to use for conversion.
loc	Locale used for conversion.

Return value

Converted wide character.

Additional information

Maps the wide character c using the mapping described by t in the locale loc.

Notes

Conforms to POSIX.1-2017.

wctrans()

Description

Construct character mapping.

Prototype

```
wctrans_t wctrans(const char * name);
```

Parameters

Parameter	Description
name	Name of mapping.

Return value

Transformation mapping; zero if mapping unrecognized.

Additional information

Constructs a value of type wctrans_t that describes a mapping between wide characters identified by the string argument property.

If property identifies a valid mapping of wide characters in the current locale, *wctrans_l()* (page **??**) returns a nonzero value that is valid as the second argument to *towctrans()* (page 596); otherwise, it returns zero.

The only mappings supported are "tolower" and "toupper".

wctrans_l()

Description

Construct character mapping, per locale (POSIX.1).

Prototype

```
wctrans_t wctrans_l(const char * name,
                    locale_t loc);
```

Parameters

Parameter	Description
name	Name of mapping.
loc	Locale used for mapping.

Return value

Transformation mapping; zero if mapping unrecognized.

Additional information

Constructs a value of type wctrans_t that describes a mapping between wide characters identified by the string argument property.

If property identifies a valid mapping of wide characters in the locale loc, *wctrans_l()* (page **??**) returns a nonzero value that is valid as the second argument to *towctrans()* (page 596); otherwise, it returns zero.

The only mappings supported are "tolower" and "toupper".

Notes

Conforms to POSIX.1-2017.

4.6.21 <xlocale.h>

Locale management

Function	Description
newlocale() (page 599)	Duplicate locale data (POSIX.1).
duplocale() (page 600)	Duplicate locale data (POSIX.1).
freelocale() (page 600)	Free locale (POSIX.1).
localeconv_l() (page **??**)	Get locale data (POSIX.1).

newlocale()

Description

Duplicate locale data (POSIX.1).

Prototype

```
locale_t newlocale(      int        category_mask,
                   const char      * loc,
                         locale_t   base);
```

Parameters

Parameter	Description
category_mask	Locale categories to be set or modified.
loc	Locale name.
base	Base to modify or NULL to create a new locale.

Return value

Pointer to modified locale.

Additional information

Creates a new locale object or modifies an existing one. If the base argument is NULL, a new locale object is created.

category_mask specifies the locale categories to be set or modified. Values for category_mask are constructed by a bitwise-inclusive OR of the symbolic constants LC_CTYPE_MASK, LC_NUMERIC_MASK, LC_TIME_MASK, LC_COLLATE_MASK, LC_MONETARY_MASK, and LC_MESSAGES_MASK.

For each category with the corresponding bit set in category_mask, the data from the locale named by loc is used. In the case of modifying an existing locale object, the data from the locale named by loc replaces the existing data within the locale object. If a completely new locale object is created, the data for all sections not requested by category_mask are taken from the default locale.

The locales "C" and "POSIX" are equivalent and defined for all settings of category_mask:

If loc is NULL, then the "C" locale is used. If loc is an empty string, *newlocale()* (page 599) will use the default locale.

If base is NULL, the current locale is used. If base is LC_GLOBAL_LOCALE, the global locale is used.

If mask is LC_ALL_MASK, base is ignored.

Notes

Conforms to POSIX.1-2008.

POSIX.1-2008 does not specify whether the locale object pointed to by base is modified or whether it is freed and a new locale object created.

The category mask LC_MESSAGES_MASK is not implemented as POSIX messages are not implemented.

duplocale()

Description

Duplicate locale data (POSIX.1).

Prototype

```
locale_t duplocale(locale_t base);
```

Parameters

Parameter	Description
base	Locale to duplicate.

Return value

If there is insufficient memory to duplicate loc, returns a NULL and sets errno to ENOMEM as required by POSIX.1-2008. Otherwise, returns a new, duplicated locale.

Additional information

Duplicates the locale object referenced by loc. Duplicated locales must be freed with *freelocale()* (page 600).

Notes

Conforms to POSIX.1-2008.

freelocale()

Description

Free locale (POSIX.1).

Prototype

```
int freelocale(locale_t loc);
```

Parameters

Parameter	Description
loc	Locale to free.

Return value

Zero on success, -1 on error.

Additional information

Frees the storage associated with loc.

Notes

Conforms to POSIX.1-2008.

localeconv_l()

Description

Get locale data (POSIX.1).

Prototype

```
localeconv_l(locale_t loc);
```

Parameters

Parameter	Description
loc	Locale to inquire.

Return value

Returns a pointer to a structure of type lconv with the corresponding values for the locale loc filled in.

Notes

Conforms to POSIX.1-2008.

4.7 Compiler support API

4.7.1 GNU library API

Integer arithmetic

Function	Description
__mulsi3() (page 602)	Multiply, signed 32-bit integer.
__muldi3() (page 602)	Multiply, signed 64-bit integer.
__divsi3() (page 602)	Divide, signed 32-bit integer.
__divdi3() (page 603)	Divide, signed 64-bit integer.
__udivsi3() (page 603)	Divide, unsigned 32-bit integer.
__udivdi3() (page 604)	Divide, unsigned 64-bit integer.
__modsi3() (page 604)	Remainder after divide, signed 32-bit integer.
__moddi3() (page 604)	Remainder after divide, signed 64-bit integer.
__umodsi3() (page 605)	Remainder after divide, unsigned 32-bit integer.
__umoddi3() (page 605)	Remainder after divide, unsigned 64-bit integer.
__udivmodsi4() (page 606)	Divide with remainder, unsigned 32-bit integer.
__udivmoddi4() (page 606)	Divide with remainder, unsigned 64-bit integer.
__clzsi2() (page 606)	Count leading zeros, 32-bit integer.
__clzdi2() (page 607)	Count leading zeros, 64-bit integer.
__ctzsi2() (page 607)	Count trailing zeros, 32-bit integer.
__ctzdi2() (page 608)	Count trailing zeros, 64-bit integer.
__ffssi2() (page 608)	Find first set, 32-bit integer.
__ffsdi2() (page 608)	Find first set, 64-bit integer.
__bswapsi2() (page 609)	Byte swap, 32-bit integer.
__bswapdi2() (page 609)	Byte swap, 64-bit integer.
__popcountsi2() (page 609)	Population count, 32-bit integer.
__popcountdi2() (page 610)	Population count, 64-bit integer.
__paritysi2() (page 610)	Parity, 32-bit integer.
__paritydi2() (page 610)	Parity, 64-bit integer.

__mulsi3()

Description

Multiply, signed 32-bit integer.

Prototype

```
__SEGGER_RTL_U32 __mulsi3(__SEGGER_RTL_U32 a,
                          __SEGGER_RTL_U32 b);
```

Parameters

Parameter	Description
a	Multiplier.
b	Multiplicand.

Return value

Product.

__muldi3()

Description

Multiply, signed 64-bit integer.

Prototype

```
__SEGGER_RTL_U64 __muldi3(__SEGGER_RTL_U64 a,
                          __SEGGER_RTL_U64 b);
```

Parameters

Parameter	Description
a	Multiplier.
b	Multiplicand.

Return value

Product.

__divsi3()

Description

Divide, signed 32-bit integer.

Prototype

```
int32_t __divsi3(int32_t num,
                 int32_t den);
```

Parameters

Parameter	Description
num	Dividend.
den	Divisor.

Return value

Quotient.

__divdi3()

Description

Divide, signed 64-bit integer.

Prototype

```
int64_t __divdi3(int64_t num,
                 int64_t den);
```

Parameters

Parameter	Description
num	Dividend.
den	Divisor.

Return value

Quotient.

__udivsi3()

Description

Divide, unsigned 32-bit integer.

Prototype

```
__SEGGER_RTL_U32 __udivsi3(__SEGGER_RTL_U32 num,
                           __SEGGER_RTL_U32 den);
```

Parameters

Parameter	Description
num	Dividend.
den	Divisor.

Return value

Quotient.

__udivdi3()

Description

Divide, unsigned 64-bit integer.

Prototype

```
__SEGGER_RTL_U64 __udivdi3(__SEGGER_RTL_U64 num,
                           __SEGGER_RTL_U64 den);
```

Parameters

Parameter	Description
num	Dividend.
den	Divisor.

Return value

Quotient.

__modsi3()

Description

Remainder after divide, signed 32-bit integer.

Prototype

```
int32_t __modsi3(int32_t num,
                 int32_t den);
```

Parameters

Parameter	Description
num	Dividend.
den	Divisor.

Return value

Remainder.

__moddi3()

Description

Remainder after divide, signed 64-bit integer.

Prototype

```
int64_t __moddi3(int64_t num,
                 int64_t den);
```

Parameters

Parameter	Description
num	Dividend.
den	Divisor.

Return value

Remainder.

__umodsi3()

Description

Remainder after divide, unsigned 32-bit integer.

Prototype

```
__SEGGER_RTL_U32 __umodsi3(__SEGGER_RTL_U32 num,
                           __SEGGER_RTL_U32 den);
```

Parameters

Parameter	Description
num	Dividend.
den	Divisor.

Return value

Remainder.

__umoddi3()

Description

Remainder after divide, unsigned 64-bit integer.

Prototype

```
__SEGGER_RTL_U64 __umoddi3(__SEGGER_RTL_U64 num,
                           __SEGGER_RTL_U64 den);
```

Parameters

Parameter	Description
num	Dividend.
den	Divisor.

Return value

Remainder.

__udivmodsi4()

Description

Divide with remainder, unsigned 32-bit integer.

Prototype

```
__SEGGER_RTL_U32 __udivmodsi4(__SEGGER_RTL_U32 num,
                              __SEGGER_RTL_U32 den,
                              __SEGGER_RTL_U32 *rem);
```

Parameters

Parameter	Description
num	Dividend.
den	Divisor.
rem	Pointer to object that receives the remainder.

Return value

Quotient.

__udivmoddi4()

Description

Divide with remainder, unsigned 64-bit integer.

Prototype

```
__SEGGER_RTL_U64 __udivmoddi4(__SEGGER_RTL_U64 num,
                              __SEGGER_RTL_U64 den,
                              __SEGGER_RTL_U64 *rem);
```

Parameters

Parameter	Description
num	Dividend.
den	Divisor.
rem	Pointer to object that receives the remainder.

Return value

Quotient.

__clzsi2()

Description

Count leading zeros, 32-bit integer.

Prototype

```
int __clzsi2(__SEGGER_RTL_U32 x);
```

Parameters

Parameter	Description
x	Argument; x must not be zero.

Return value

Number of leading zeros in x.

__clzdi2()

Description

Count leading zeros, 64-bit integer.

Prototype

```
int __clzdi2(__SEGGER_RTL_U64 x);
```

Parameters

Parameter	Description
x	Argument; x must not be zero.

Return value

Number of leading zeros in x.

__ctzsi2()

Description

Count trailing zeros, 32-bit integer.

Prototype

```
int __ctzsi2(__SEGGER_RTL_U32 x);
```

Parameters

Parameter	Description
x	Argument; x must not be zero.

Return value

Number of trailing zeros in x.

__ctzdi2()

Description

Count trailing zeros, 64-bit integer.

Prototype

```
int __ctzdi2(__SEGGER_RTL_U64 x);
```

Parameters

Parameter	Description
x	Argument; x must not be zero.

Return value

Number of trailing zeros in x.

__ffssi2()

Description

Find first set, 32-bit integer.

Prototype

```
int __ffssi2(__SEGGER_RTL_U32 x);
```

Parameters

Parameter	Description
x	Argument; Return value zero if x is zero.

Return value

The index of the least significant 1-bit in x.

__ffsdi2()

Description

Find first set, 64-bit integer.

Prototype

```
int __ffsdi2(__SEGGER_RTL_U64 x);
```

Parameters

Parameter	Description
x	Argument; Return value zero if x is zero.

Return value

The index of the least significant 1-bit in x.

__bswapsi2()

Description

Byte swap, 32-bit integer.

Prototype

```
int __bswapsi2(__SEGGER_RTL_U32 x);
```

Parameters

Parameter	Description
x	Argument.

Return value

The result of byte swap in x.

__bswapdi2()

Description

Byte swap, 64-bit integer.

Prototype

```
int __bswapdi2(__SEGGER_RTL_U64 x);
```

Parameters

Parameter	Description
x	Argument.

Return value

The result of byte swap in x.

__popcountsi2()

Description

Population count, 32-bit integer.

Prototype

```
int __popcountsi2(__SEGGER_RTL_U32 x);
```

Parameters

Parameter	Description
x	Argument.

Return value

Count of number of one bits in x.

__popcountdi2()

Description

Population count, 64-bit integer.

Prototype

```
int __popcountdi2(__SEGGER_RTL_U64 x);
```

Parameters

Parameter	Description
x	Argument.

Return value

Count of number of one bits in x.

__paritysi2()

Description

Parity, 32-bit integer.

Prototype

```
int __paritysi2(__SEGGER_RTL_U32 x);
```

Parameters

Parameter	Description
x	Argument.

Return value

1	number of one bits in x is odd.
0	number of one bits in x is even.

__paritydi2()

Description

Parity, 64-bit integer.

Prototype

```
int __paritydi2(__SEGGER_RTL_U64 x);
```

Parameters

Parameter	Description
x	Argument.

Return value

1	number of one bits in x is odd.
0	number of one bits in x is even.

Floating arithmetic

Function	Description
__addsf3() (page 611)	Add, float.
__adddf3() (page 611)	Add, double.
__addtf3() (page 612)	Add, long double.
__subsf3() (page 612)	Subtract, float.
__subdf3() (page 613)	Subtract, double.
__subtf3() (page 613)	Subtract, long double.
__mulsf3() (page 613)	Multiply, float.
__muldf3() (page 614)	Multiply, double.
__multf3() (page 614)	Multiply, long double.
__divsf3() (page 615)	Divide, float.
__divdf3() (page 615)	Divide, double.
__divtf3() (page 615)	Divide, long double.

__addsf3()

Description

Add, float.

Prototype

```
float __addsf3(float x,
               float y);
```

Parameters

Parameter	Description
x	Augend.
y	Addend.

Return value

Sum.

__adddf3()

Description

Add, double.

Prototype

```
double __adddf3(double x,
                double y);
```

Parameters

Parameter	Description
x	Augend.
y	Addend.

Return value

Sum.

__addtf3()

Description

Add, long double.

Prototype

```
long double __addtf3(long double x,
                     long double y);
```

Parameters

Parameter	Description
x	Augend.
y	Addend.

Return value

Sum.

__subsf3()

Description

Subtract, float.

Prototype

```
float __subsf3(float x,
               float y);
```

Parameters

Parameter	Description
x	Minuend.
y	Subtrahend.

Return value

Difference.

__subdf3()

Description

Subtract, double.

Prototype

```
double __subdf3(double x,
                double y);
```

Parameters

Parameter	Description
x	Minuend.
y	Subtrahend.

Return value

Difference.

__subtf3()

Description

Subtract, long double.

Prototype

```
long double __subtf3(long double x,
                     long double y);
```

Parameters

Parameter	Description
x	Minuend.
y	Subtrahend.

Return value

Difference.

__mulsf3()

Description

Multiply, float.

Prototype

```
float __mulsf3(float x,
               float y);
```

Parameters

Parameter	Description
x	Multiplicand.
y	Multiplier.

Return value

Product.

__muldf3()

Description

Multiply, double.

Prototype

```
double __muldf3(double x,
                double y);
```

Parameters

Parameter	Description
x	Multiplicand.
y	Multiplier.

Return value

Product.

__multf3()

Description

Multiply, long double.

Prototype

```
long double __multf3(long double x,
                     long double y);
```

Parameters

Parameter	Description
x	Multiplicand.
y	Multiplier.

Return value

Product.

__divsf3()

Description

Divide, float.

Prototype

```
float __divsf3(float x,
               float y);
```

Parameters

Parameter	Description
x	Dividend.
y	Divisor.

Return value

Quotient.

__divdf3()

Description

Divide, double.

Prototype

```
double __divdf3(double x,
                double y);
```

Parameters

Parameter	Description
x	Dividend.
y	Divisor.

Return value

Quotient.

__divtf3()

Description

Divide, long double.

Prototype

```
long double __divtf3(long double x,
                     long double y);
```

Parameters

Parameter	Description
x	Dividend.
y	Divisor.

Return value

Quotient.

Floating conversions

Function	Description
__fixsfsi() (page 616)	Convert float to int.
__fixdfsi() (page 617)	Convert double to int.
__fixtfsi() (page 617)	Convert long double to int.
__fixsfdi() (page 618)	Convert float to long long.
__fixdfdi() (page 618)	Convert double to long long.
__fixtfdi() (page 618)	Convert long double to long long.
__fixunssfsi() (page 619)	Convert float to unsigned.
__fixunsdfsi() (page 619)	Convert double to unsigned.
__fixunstfsi() (page 620)	Convert long double to int.
__fixunssfdi() (page 620)	Convert float to unsigned long long.
__fixunsdfdi() (page 620)	Convert double to unsigned long long.
__fixunstfdi() (page 621)	Convert long double to unsigned long long.
__floatsisf() (page 621)	Convert int to float.
__floatsidf() (page 621)	Convert int to double.
__floatsitf() (page 622)	Convert int to long double.
__floatdisf() (page 622)	Convert long long to float.
__floatdidf() (page 622)	Convert long long to double.
__floatditf() (page 623)	Convert long long to long double.
__floatunsisf() (page 623)	Convert unsigned to float.
__floatunsidf() (page 623)	Convert unsigned to double.
__floatunsitf() (page 624)	Convert unsigned to long double.
__floatundisf() (page 624)	Convert unsigned long long to float.
__floatundidf() (page 624)	Convert unsigned long long to double.
__floatunditf() (page 625)	Convert unsigned long long to long double.
__extendsfdf2() (page 625)	Extend float to double.
__extendsftf2() (page 625)	Extend float to long double.
__extenddftf2() (page 626)	Extend double to long double.
__truncdfsf2() (page 626)	Truncate double to float.
__trunctfsf2() (page 627)	Truncate long double to float.
__trunctfdf2() (page 626)	Truncate long double to double.

__fixsfsi()

Description

Convert float to int.

Prototype

```
__SEGGER_RTL_I32 __fixsfsi(float x);
```

Parameters

Parameter	Description
x	Floating value to convert.

Return value

Integerized value.

__fixdfsi()

Description

Convert double to int.

Prototype

```
__SEGGER_RTL_I32 __fixdfsi(double x);
```

Parameters

Parameter	Description
x	Floating value to convert.

Return value

Integerized value.

__fixtfsi()

Description

Convert long double to int.

Prototype

```
__SEGGER_RTL_I32 __fixtfsi(long double x);
```

Parameters

Parameter	Description
x	Floating value to convert.

Return value

Integerized value.

__fixsfdi()

Description

Convert float to long long.

Prototype

```
__SEGGER_RTL_I64 __fixsfdi(float f);
```

Parameters

Parameter	Description
f	Floating value to convert.

Return value

Integerized value.

Notes

The RV32 compiler converts a float to a 64-bit integer by calling runtime support to handle it.

__fixdfdi()

Description

Convert double to long long.

Prototype

```
__SEGGER_RTL_I64 __fixdfdi(double x);
```

Parameters

Parameter	Description
x	Floating value to convert.

Return value

Integerized value.

Notes

RV32 always calls runtime for double to int64 conversion.

__fixtfdi()

Description

Convert long double to long long.

Prototype

```
__SEGGER_RTL_I64 __fixtfdi(long double x);
```

Parameters

Parameter	Description
x	Floating value to convert.

Return value

Integerized value.

__fixunssfsi()

Description

Convert float to unsigned.

Prototype

```
__SEGGER_RTL_U32 __fixunssfsi(float x);
```

Parameters

Parameter	Description
x	Float value to convert.

Return value

Integerized value.

__fixunsdfsi()

Description

Convert double to unsigned.

Prototype

```
__SEGGER_RTL_U32 __fixunsdfsi(double x);
```

Parameters

Parameter	Description
x	Float value to convert.

Return value

Integerized value.

__fixunstfsi()

Description

Convert long double to int.

Prototype

```
int __fixunstfsi(long double x);
```

Parameters

Parameter	Description
x	Float value to convert.

Return value

Integerized value.

__fixunssfdi()

Description

Convert float to unsigned long long.

Prototype

```
__SEGGER_RTL_U64 __fixunssfdi(float f);
```

Parameters

Parameter	Description
f	Float value to convert.

Return value

Integerized value.

__fixunsdfdi()

Description

Convert double to unsigned long long.

Prototype

```
__SEGGER_RTL_U64 __fixunsdfdi(double x);
```

Parameters

Parameter	Description
x	Float value to convert.

Return value

Integerized value.

__fixunstfdi()

Description

Convert long double to unsigned long long.

Prototype

```
__SEGGER_RTL_U64 __fixunstfdi(long double x);
```

Parameters

Parameter	Description
x	Float value to convert.

Return value

Integerized value.

__floatsisf()

Description

Convert int to float.

Prototype

```
float __floatsisf(__SEGGER_RTL_I32 x);
```

Parameters

Parameter	Description
x	Integer value to convert.

Return value

Floating value.

__floatsidf()

Description

Convert int to double.

Prototype

```
double __floatsidf(__SEGGER_RTL_I32 x);
```

Parameters

Parameter	Description
x	Integer value to convert.

Return value

Floating value.

__floatsitf()

Description

Convert int to long double.

Prototype

```
long double __floatsitf(__SEGGER_RTL_I32 x);
```

Parameters

Parameter	Description
x	Integer value to convert.

Return value

Floating value.

__floatdisf()

Description

Convert long long to float.

Prototype

```
float __floatdisf(__SEGGER_RTL_I64 x);
```

Parameters

Parameter	Description
x	Integer value to convert.

Return value

Floating value.

__floatdidf()

Description

Convert long long to double.

Prototype

```
double __floatdidf(__SEGGER_RTL_I64 x);
```

Parameters

Parameter	Description
x	Integer value to convert.

Return value

Floating value.

__floatditf()

Description

Convert long long to long double.

Prototype

```
long double __floatditf(__SEGGER_RTL_I64 x);
```

Parameters

Parameter	Description
x	Integer value to convert.

Return value

Floating value.

__floatunsisf()

Description

Convert unsigned to float.

Prototype

```
float __floatunsisf(__SEGGER_RTL_U32 x);
```

Parameters

Parameter	Description
x	Integer value to convert.

Return value

Floating value.

__floatunsidf()

Description

Convert unsigned to double.

Prototype

```
double __floatunsidf(__SEGGER_RTL_U32 x);
```

Parameters

Parameter	Description
x	Unsigned value to convert.

Return value

Double value.

__floatunsitf()

Description

Convert unsigned to long double.

Prototype

```
long double __floatunsitf(__SEGGER_RTL_U32 x);
```

Parameters

Parameter	Description
x	Unsigned value to convert.

Return value

Long double value.

__floatundisf()

Description

Convert unsigned long long to float.

Prototype

```
float __floatundisf(__SEGGER_RTL_U64 x);
```

Parameters

Parameter	Description
x	Unsigned long long value to convert.

Return value

Float value.

__floatundidf()

Description

Convert unsigned long long to double.

Prototype

```
double __floatundidf(__SEGGER_RTL_U64 x);
```

Parameters

Parameter	Description
x	Unsigned long long value to convert.

Return value

Double value.

__floatunditf()

Description

Convert unsigned long long to long double.

Prototype

```
long double __floatunditf(__SEGGER_RTL_U64 x);
```

Parameters

Parameter	Description
x	Unsigned long long value to convert.

Return value

Long double value.

__extendsfdf2()

Description

Extend float to double.

Prototype

```
double __extendsfdf2(float x);
```

Parameters

Parameter	Description
x	Float value to extend.

Return value

Double value.

__extendsftf2()

Description

Extend float to long double.

Prototype

```
long double __extendsftf2(float x);
```

Parameters

Parameter	Description
x	Float value to extend.

Return value

Double value.

__extenddftf2()

Description

Extend double to long double.

Prototype

```
long double __extenddftf2(double x);
```

Parameters

Parameter	Description
x	Double value to extend.

Return value

Long double value.

__truncdfsf2()

Description

Truncate double to float.

Prototype

```
float __truncdfsf2(double x);
```

Parameters

Parameter	Description
x	Double value to truncate.

Return value

Float value.

__trunctfdf2()

Description

Truncate long double to double.

Prototype

```
double __trunctfdf2(long double x);
```

Parameters

Parameter	Description
x	Long double value to truncate.

Return value

Double value.

__trunctfsf2()

Description

Truncate long double to float.

Prototype

```
float __trunctfsf2(long double x);
```

Parameters

Parameter	Description
x	Long double value to truncate.

Return value

Float value.

Floating comparisons

Function	Description
__eqsf2() (page 627)	Equal, float.
__eqdf2() (page 628)	Equal, double.
__eqtf2() (page 628)	Equal, long double.
__nesf2() (page 629)	Not equal, float.
__nedf2() (page 629)	Not equal, double.
__netf2() (page 629)	Not equal, long double.
__ltsf2() (page 630)	Less than, float.
__ltdf2() (page 630)	Less than, double.
__lttf2() (page 631)	Less than, long double.
__lesf2() (page 631)	Less than or equal, float.
__ledf2() (page 631)	Less than or equal, double.
__letf2() (page 632)	Less than or equal, long double.
__gtsf2() (page 632)	Greater than, float.
__gtdf2() (page 633)	Greater than, double.
__gttf2() (page 633)	Greater than, long double.
__gesf2() (page 633)	Greater than or equal, float.
__gedf2() (page 634)	Greater than or equal, double.
__getf2() (page 634)	Greater than or equal, long double.

__eqsf2()

Description

Equal, float.

Prototype

```
int __eqsf2(float x,
            float y);
```

Parameters

Parameter	Description
x	Left-hand operand.
y	Right-hand operand.

Return value

Return = 0 if both operands are non-NaN and a = b (GNU three-way boolean).

__eqdf2()

Description

Equal, double.

Prototype

```
int __eqdf2(double x,
            double y);
```

Parameters

Parameter	Description
x	Left-hand operand.
y	Right-hand operand.

Return value

Return = 0 if both operands are non-NaN and a = b (GNU three-way boolean).

__eqtf2()

Description

Equal, long double.

Prototype

```
int __eqtf2(long double x,
            long double y);
```

Parameters

Parameter	Description
x	Left-hand operand.
y	Right-hand operand.

Return value

Return = 0 if both operands are non-NaN and a = b (GNU three-way boolean).

__nesf2()

Description

Not equal, float.

Prototype

```
int __nesf2(float x,
            float y);
```

Parameters

Parameter	Description
x	Left-hand operand.
y	Right-hand operand.

Return value

Return = 0 if both operands are non-NaN and a = b (GNU three-way boolean).

__nedf2()

Description

Not equal, double.

Prototype

```
int __nedf2(double x,
            double y);
```

Parameters

Parameter	Description
x	Left-hand operand.
y	Right-hand operand.

Return value

Return = 0 if both operands are non-NaN and a = b (GNU three-way boolean).

__netf2()

Description

Not equal, long double.

Prototype

```
int __netf2(long double x,
            long double y);
```

Parameters

Parameter	Description
x	Left-hand operand.
y	Right-hand operand.

Return value

Return = 0 if both operands are non-NaN and a = b (GNU three-way boolean).

__ltsf2()

Description

Less than, float.

Prototype

```
int __ltsf2(float x,
            float y);
```

Parameters

Parameter	Description
x	Left-hand operand.
y	Right-hand operand.

Return value

Return < 0 if both operands are non-NaN and a < b (GNU three-way boolean).

__ltdf2()

Description

Less than, double.

Prototype

```
int __ltdf2(double x,
            double y);
```

Parameters

Parameter	Description
x	Left-hand operand.
y	Right-hand operand.

Return value

Return < 0 if both operands are non-NaN and a < b (GNU three-way boolean).

__lttf2()

Description

Less than, long double.

Prototype

```
int __lttf2(long double x,
            long double y);
```

Parameters

Parameter	Description
x	Left-hand operand.
y	Right-hand operand.

Return value

Return < 0 if both operands are non-NaN and a < b (GNU three-way boolean).

__lesf2()

Description

Less than or equal, float.

Prototype

```
int __lesf2(float x,
            float y);
```

Parameters

Parameter	Description
x	Left-hand operand.
y	Right-hand operand.

Return value

Return ≤ 0 if both operands are non-NaN and a < b (GNU three-way boolean).

__ledf2()

Description

Less than or equal, double.

Prototype

```
int __ledf2(double x,
            double y);
```

Parameters

Parameter	Description
x	Left-hand operand.
y	Right-hand operand.

Return value

Return ≤ 0 if both operands are non-NaN and a < b (GNU three-way boolean).

__letf2()

Description

Less than or equal, long double.

Prototype

```
int __letf2(long double x,
            long double y);
```

Parameters

Parameter	Description
x	Left-hand operand.
y	Right-hand operand.

Return value

Return ≤ 0 if both operands are non-NaN and a < b (GNU three-way boolean).

__gtsf2()

Description

Greater than, float.

Prototype

```
int __gtsf2(float x,
            float y);
```

Parameters

Parameter	Description
x	Left-hand operand.
y	Right-hand operand.

Return value

Return > 0 if both operands are non-NaN and a > b (GNU three-way boolean).

__gtdf2()

Description

Greater than, double.

Prototype

```
int __gtdf2(double x,
            double y);
```

Parameters

Parameter	Description
x	Left-hand operand.
y	Right-hand operand.

Return value

Return > 0 if both operands are non-NaN and a > b (GNU three-way boolean).

__gttf2()

Description

Greater than, long double.

Prototype

```
int __gttf2(long double x,
            long double y);
```

Parameters

Parameter	Description
x	Left-hand operand.
y	Right-hand operand.

Return value

Return > 0 if both operands are non-NaN and a > b (GNU three-way boolean).

__gesf2()

Description

Greater than or equal, float.

Prototype

```
int __gesf2(float x,
            float y);
```

Parameters

Parameter	Description
x	Left-hand operand.
y	Right-hand operand.

Return value

Return ≥ 0 if both operands are non-NaN and a ≥ b (GNU three-way boolean).

__gedf2()

Description

Greater than or equal, double.

Prototype

```
int __gedf2(double x,
            double y);
```

Parameters

Parameter	Description
x	Left-hand operand.
y	Right-hand operand.

Return value

Return ≥ 0 if both operands are non-NaN and a ≥ b (GNU three-way boolean).

__getf2()

Description

Greater than or equal, long double.

Prototype

```
int __getf2(long double x,
            long double y);
```

Parameters

Parameter	Description
x	Left-hand operand.
y	Right-hand operand.

Return value

Return ≥ 0 if both operands are non-NaN and a ≥ b (GNU three-way boolean).

4.8 External function interface

This section summarises the functions to be provided by the implementor when integrating `Nuclei C Runtime Library` into an application or library.

4.8.1 I/O functions

Function	Description
__SEGGER_RTL_X_file_read() (page 635)	Read data from file.
__SEGGER_RTL_X_file_write() (page 635)	Write data to file.
__SEGGER_RTL_X_file_unget() (page 636)	Push character back to file.

__SEGGER_RTL_X_file_read()

Description

Read data from file.

Prototype

```
int __SEGGER_RTL_X_file_read(           __SEGGER_RTL_FILE *stream,
                             char      * s,
                             unsigned    len);
```

Parameters

Parameter	Description
stream	Pointer to file to read from.
s	Pointer to object that receives the input.
len	Number of characters to read from file.

Return value

≥ 0	Success.
< 0	Failure.

Additional information

This reads len octets from the file stream into the object pointed to by s.

__SEGGER_RTL_X_file_write()

Description

Write data to file.

Prototype

```
int __SEGGER_RTL_X_file_write(                 __SEGGER_RTL_FILE *stream,
                              const char      * s,
                              unsigned    len);
```

Parameters

Parameter	Description
stream	Pointer to stream to write to.
s	Pointer to object to write to stream.
len	Number of characters to write to the stream.

Return value

≥ 0	Success.
< 0	Failure.

Additional information

This writes len octets to the file stream from the object pointed to by s.

__SEGGER_RTL_X_file_unget()

Description

Push character back to file.

Prototype

```
int __SEGGER_RTL_X_file_unget(    __SEGGER_RTL_FILE *stream,
                                  int c);
```

Parameters

Parameter	Description
stream	File to push character to.
c	Character to push back to file.

Return value

= EOF	Failed to push character back.
≠ EOF	The character pushed back to the file.

Additional information

This function pushes the character c back to the file so that it can be read again. If c is EOF, the function fails and EOF is returned. One character of pushback is guaranteed; if more than one character is pushed back without an intervening read, the pushback may fail.

4.8.2 Heap protection functions

Function	Description
__SEGGER_RTL_X_heap_lock() (page 637)	Lock heap.
__SEGGER_RTL_X_heap_unlock() (page 637)	Unlock heap.

__SEGGER_RTL_X_heap_lock()

Description

Lock heap.

Prototype

```
void __SEGGER_RTL_X_heap_lock(void);
```

Additional information

This function is called to lock access to the heap before allocation or deallocation is processed. This is only required for multitasking systems where heap operations may possibly be called called from different threads.

__SEGGER_RTL_X_heap_unlock()

Description

Unlock heap.

Prototype

```
void __SEGGER_RTL_X_heap_unlock(void);
```

Additional information

This function is called to unlock access to the heap after allocation or deallocation has completed. This is only required for multitasking systems where heap operations may possibly be called called from different threads.

4.8.3 Error and assertion functions

Function	Description
__SEGGER_RTL_X_assert() (page 637)	User-defined behavior for the assert macro.
__SEGGER_RTL_X_errno_addr() (page 638)	Return pointer to object holding errno.

__SEGGER_RTL_X_assert()

Description

User-defined behavior for the assert macro.

Prototype

```
void __SEGGER_RTL_X_assert(const char * expr,
                           const char * filename,
                           int        line);
```

Parameters

Parameter	Description
expr	Stringized expression that caused failure.
filename	Filename of the source file where the failure was signaled.
line	Line number of the failed assertion.

Additional information

The default implementation of __SEGGER_RTL_X_assert() prints the filename, line, and error message to standard output and then calls *abort()* (page **??**).

__SEGGER_RTL_X_assert() is defined as a weak function and can be replaced by user code.

__SEGGER_RTL_X_errno_addr()

Description

Return pointer to object holding errno.

Prototype

```
int *__SEGGER_RTL_X_errno_addr(void);
```

Return value

Pointer to errno object.

Additional information

The default implementation of this function is to return the address of a variable declared with the __SEGGER_RTL_THREAD storage class. Thus, for multithreaded environments that implement thread-local variables through __SEGGER_RTL_THREAD, each thread receives its own thread-local errno.

It is beyond the scope of this manual to describe how thread-local variables are implemented by the compiler and any associated real-time operating system.

When __SEGGER_RTL_THREAD is defined as an empty macro, this function returns the address of a singleton errno object.

4.8.4 RTC functions

Function	Description
__SEGGER_RTL_X_set_time_of_day() (page 638)	Set RTC time.
__SEGGER_RTL_X_get_time_of_day() (page 639)	Get RTC time.

__SEGGER_RTL_X_set_time_of_day()

Description

Set RTC time.

Prototype

```
int __SEGGER_RTL_X_set_time_of_day(const struct timeval *__tp);
```

Parameters

tp - Pointer to timeval.

Return value

return 0 for success, or -1 for failure.

__SEGGER_RTL_X_get_time_of_day()

Description

Get RTC time.

Prototype

```
int __SEGGER_RTL_X_get_time_of_day(struct timeval *__tp);
```

Parameters

tp - Pointer to timeval.

Return value

return 0 for success, or -1 for failure.

4.8.5 Locale functions

Function	Description
__SEGGER_RTL_X_find_locale() (page 639)	Find locale.

__SEGGER_RTL_X_find_locale()

Description

Find locale.

Prototype

```
const __SEGGER_RTL_locale_t *__SEGGER_RTL_X_find_locale(const char *locale);
```

Parameters

locale - Pointer to zero-terminated locale name.

Return value

Returns a pointer to a locale or NULL if none can be found.

CHAPTER

FIVE

NUCLEI OPENOCD

5.1 Introduction to OpenOCD

5.1.1 Repositories and Documentation

OpenOCD Repositories

GitHub (RISC-V OpenOCD) Repository[31].

Gitee (RISC-V OpenOCD) Repository[32].

OpenFlashLoader

> **Note**
>
> The OpenOCD Flashloader enables support for various customized flash programming implementations without requiring modifications to OpenOCD itself. For detailed usage instructions, please refer to the documentation in the source code repository.

GitHub (OpenFlashLoader) Repository[33].

Gitee (OpenFlashLoader) Repository[34].

OpenOCD User Guide Location: `openocd/doc/pdf/openocd.pdf`

5.2 Getting Started

5.2.1 Checking OpenOCD Version

To determine the installed OpenOCD version:

1. Open a command prompt (Windows) or terminal (Linux)
2. Execute the command: `openocd -v`

```
nuclei@nuclei-v:~$ openocd -v
Open On-Chip Debugger 0.11.0+dev-02424-g787e48e66 (2022-12-29-08:49)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
nuclei@nuclei-v:~$ 
```

[31] https://github.com/riscv-mcu/riscv-openocd
[32] https://gitee.com/riscv-mcu/riscv-openocd
[33] https://github.com/riscv-mcu/openflashloader
[34] https://gitee.com/riscv-mcu/openflashloader

Note

Current Version Information: - Git Commit ID: 787e48e66 - Compile Date: 2022-12-29 08:49

5.2.2 Running OpenOCD

Common Command Line Parameters

Parameter	Description
-v	Display version information
-d	Set debug level to 3
-d0	Error messages only
-d1	Error and warning messages
-d2	Error, warning, and info (default)
-d3	Error, warning, info, and debug messages
-d4	All messages including low-level debug
-f file	Use specified configuration file
-s dir	Directory to search for config files
-l logfile	Redirect log output to specified file

5.3 Nuclei-Specific Features

5.3.1 CPU Information Display

The `nuclei cpuinfo` command provides detailed information about the CPU ' s capabilities, including:

- Supported instruction sets
- Available functional components
- Component configurations

This command simplifies the process of querying CPU features by automatically reading and interpreting the relevant CSR (Control and Status Register) values, eliminating the need for manual register inspection and calculation.

5.3.2 NUSPI (Nuclei SPI) Driver

The NUSPI driver provides support for Nuclei ' s SPI controller, which is utilized in Nuclei RISC-V FPGA evaluation boards and other compatible hardware.

Usage: `flash bank name nuspi base size chip_width bus_width target spi_base [simulation]`

5.3.3 Custom Driver with OpenFlashLoader

The custom driver provides compatibility with various SPI controllers and flash memory types. When using this driver, it must be combined with the OpenFlashLoader to achieve optimal functionality.

Usage: `flash bank name custom base size chip_width bus_width target spi_base flashloader_path [simulation] [sectorsize=]`

5.3.4 Nuclei-Specific CSRs

The Nuclei version of OpenOCD supports several custom CSRs (Control and Status Registers). For a complete list and detailed information, refer to:

GitHub (src/target/riscv/encoding.h)[35].

Gitee (src/target/riscv/encoding.h)[36].

5.3.5 Embedded Trace (ETrace) Support

Note

The ETrace feature is currently in the experimental stage and should not be used in production environments.

Some Nuclei CPUs include embedded trace support, enabling detailed examination of instruction execution. The trace functionality is managed through an Embedded Trace (ETrace) module integrated into the CPU ' s scan chains.

Current Implementation: - RISC-V ETrace Instruction Trace (available) - Data Trace (not yet implemented)

ETrace Commands:

1. Configuration: `nuclei etrace config etrace-addr buffer-addr buffer-size wrap` - Initializes ETrace and configures operational parameters
2. Control: - `nuclei etrace enable`: Activates ETrace functionality - `nuclei etrace disable`: Deactivates ETrace functionality - `nuclei etrace start`: Begins trace data collection - `nuclei etrace stop`: Stops trace data collection
3. Data Management: - `nuclei etrace dump filename`: Exports captured trace data to a file - `nuclei etrace clear`: Resets trace buffer pointers - `nuclei etrace info`: Displays current ETrace status

Note

- For detailed command documentation, please refer to `openocd.pdf`
- The ETrace feature is also available in Nuclei Studio IDE. Refer to the *Trace 功能的使用* (page 220) for how to use.

[35] https://github.com/riscv-mcu/riscv-openocd/blob/nuclei/2024/src/target/riscv/encoding.h#L3109
[36] https://gitee.com/riscv-mcu/riscv-openocd/blob/nuclei/2024/src/target/riscv/encoding.h#L3109

5.3.6 Debug Map Feature

> **Note**
>
> The debug map for each hardware thread (hart) is automatically read and displayed during OpenOCD initialization. Alternatively, you can access the debug map at runtime using the `examine_cpu_core` command.

For detailed documentation about the Nuclei debug map feature, please contact your application engineer.

Commands:

- `nuclei expose_cpu_core` - Configures the list of indices for `nuclei_examine_cpu_core` - Must be executed before the `init` command
- `nuclei examine_cpu_core` - Returns a 64-bit value combining `dm-custom1` and `dm-custom2` registers - Value calculation: `(dm-custom2 << 32) + dm-custom1`

5.3.7 Cross-Trigger Interface

The Cross-Trigger Interface (CTI) provides an advanced debugging mechanism that enables developers to:

- Trigger specific debugging actions
- Synchronize multiple debugging-related events

Available Commands:

- `nuclei cti halt_group on/off target_name0 target_name1 ...` - Controls halt group triggers
- `nuclei cti resume_group on/off target_name0 target_name1 ...` - Controls resume group triggers

5.3.8 Reset and Halt Command

The `init resethalt` command addresses situations where:

- The CPU becomes unresponsive due to software issues
- The debugger loses connection with the development board
- Power cycling is ineffective (particularly when code is running from flash)

This command provides a software-based solution to reset and halt the CPU without requiring physical power cycling.

5.3.9 FTDI nSCAN1 Mode Command

The `ftdi nscan1_mode` command controls Nuclei ' s Compact JTAG (cJTAG) mode functionality.

Usage: `ftdi nscan1_mode on|off`

> **Note**
>
> This command follows the same syntax and behavior as the standard `ftdi oscan1_mode` command.

5.4 Configuration File Overview

The OpenOCD configuration file defines how to establish a connection with the development board through the debug interface. Nuclei provides a sample configuration file that can be adapted to specific hardware requirements.

Example Configuration:

- Using Nuclei HBird Debugger (FTDI-based)

Reference implementation[37].

5.4.1 Debugger Speed Configuration

To adjust the debugger communication speed:

- `adapter_khz 1000`
- `adapter speed 1000`

Both commands set the debugger speed to 1000 kHz.

5.4.2 Debugger Interface Configuration

The following configuration selects and initializes the FTDI debugger interface:

```
adapter driver ftdi
ftdi vid_pid 0x0403 0x6010

transport select jtag

ftdi layout_init 0x0008 0x001b
ftdi layout_signal nSRST -oe 0x0020 -data 0x0020
ftdi layout_signal TCK -data 0x0001
ftdi layout_signal TDI -data 0x0002
ftdi layout_signal TDO -input 0x0004
ftdi layout_signal TMS -data 0x0008
ftdi layout_signal JTAG_SEL -data 0x0100 -oe 0x0100
```

Configuration Details: - FTDI chip VID/PID must match the connected hardware - JTAG transport protocol selected - Signal layout configured for HBird Debugger compatibility

5.4.3 Debugger Mode Configuration

OpenOCD supports two debugging modes:

- **JTAG Mode**: Enabled with `ftdi nscan1_mode off`
- **Compact JTAG (cJTAG) Mode**: Enabled with `ftdi nscan1_mode on`

[37] https://github.com/Nuclei-Software/nuclei-sdk/blob/master/SoC/evalsoc/Board/nuclei_fpga_eval/openocd_evalsoc.cfg

5.4.4 JTAG Link Configuration

The JTAG link configuration varies depending on the system architecture:

Single Core System

```
set _CHIPNAME0 riscv0
jtag newtap $_CHIPNAME0 cpu -irlen 5 -expected-id 0x10900a6d

set _TARGETNAME0 $_CHIPNAME0.cpu
target create $_TARGETNAME0 riscv -chain-position $_TARGETNAME0 -coreid 0
```

SMP (Symmetric Multiprocessing) System

```
set _CHIPNAME0 riscv0
jtag newtap $_CHIPNAME0 cpu -irlen 5 -expected-id 0x10900a6d

set _TARGETNAME0 $_CHIPNAME0.cpu
target create $_TARGETNAME0.0 riscv -chain-position $_TARGETNAME0 -coreid 0 -rtos␣
↪hwthread
target create $_TARGETNAME0.1 riscv -chain-position $_TARGETNAME0 -coreid 1
target create $_TARGETNAME0.2 riscv -chain-position $_TARGETNAME0 -coreid 2
target smp $_TARGETNAME0.0 $_TARGETNAME0.1 $_TARGETNAME0.2
```

AMP (Asymmetric Multiprocessing) System

```
set _CHIPNAME0 riscv0
jtag newtap $_CHIPNAME0 cpu -irlen 5 -expected-id 0x10900a6d

set _CHIPNAME1 riscv1
jtag newtap $_CHIPNAME1 cpu -irlen 5 -expected-id 0x10300a6d

set _TARGETNAME0 $_CHIPNAME0.cpu
target create $_TARGETNAME0 riscv -chain-position $_TARGETNAME0 -coreid 0

set _TARGETNAME1 $_CHIPNAME1.cpu
target create $_TARGETNAME1.0 riscv -chain-position $_TARGETNAME0 -coreid 0 -rtos␣
↪hwthread
target create $_TARGETNAME1.1 riscv -chain-position $_TARGETNAME0 -coreid 1
target smp $_TARGETNAME1.0 $_TARGETNAME1.1
```

Note

The `-rtos hwthread` option enables OpenOCD ' s pseudo RTOS functionality, which:

- Presents CPU cores (“ hardware threads ”) as threads to GDB
- Allows inspection of SMP system state through GDB commands
- Enables core-specific debugging operations: - `info threads` lists active CPU cores - `thread` switches between CPU core views - `step` and `stepi` operate on individual cores

5.4.5 Work Area Configuration

The work area is a dedicated memory region that accelerates various operations, including:

- Memory read/write operations
- Execution of small program fragments
- Flash memory operations

Configuration Example:

```
$_TARGETNAME0 configure -work-area-phys 0x08000000 -work-area-size 0x10000 -work-
↪area-backup 1
```

Note

Work Area Requirements: - Must be a readable, writable, and executable memory region - Base address (0x08000000) and size (0x10000) should be adjusted according to system requirements

5.4.6 NOR Flash Configuration

The NOR flash configuration specifies the memory mapping and controller settings:

```
set _FLASHNAME0 $_CHIPNAME0.flash
flash bank $_FLASHNAME0 nuspi 0x20000000 0 0 0 $_TARGETNAME0.0 0x10180000
```

Note

Configuration Parameters: - `nuspi`: Flash driver type (adjust as needed) - `0x20000000`: QSPI XIP address (adjust as needed) - `0x10180000`: QSPI controller base address (adjust as needed)

5.4.7 Debugger Connection Specification

When multiple debuggers are present in the debugging environment, you can specify which debugger to connect to using:

```
ftdi_serial FT4YR31I
```

Replace “FT4YR31I” with the actual serial number of your debugger.

5.4.8 Debugging Service Ports

OpenOCD provides three debugging service ports:

1. **GDB Port** (default: 3333)
2. **Telnet Port** (default: 4444)
3. **TCL Port** (default: 6666)

Configuration Example:

```
gdb_port 3333
telnet_port 4444
tcl_port 6666
```

Note

- Port numbers can be customized if the default ports are unavailable
- To disable a service, set its port to `disable`
- Ensure port numbers don ' t conflict with other services

5.4.9 Semihosting Support

OpenOCD supports ARM semihosting, which allows target programs to use host system resources. To enable:

```
arm semihosting enable
```

Note

Semihosting provides access to: - File I/O operations - Console input/output - System clock information - Other host system services

For more detailed information about how to use openocd, please check the `openocd.pdf` distributed in openocd release.

5.4.10 Target Defer Examine

In some multicore systems, the slave-target is not debuggable and needs to be unlocked by the host-target before it can be used, in which case the -defer-examine parameter is needed.

```
# Configuring the “-defer-examine” parameter
$SLAVE_TARGETNAME configure -defer-examine

# Override events, the following events will trigger reset, slave-target needs to␣
↪be initialized manually after reset
$_TARGETNAME1_0 configure -event gdb-flash-erase-start {
  init
}
$_TARGETNAME1_0 configure -event gdb-flash-write-end {
  init
}

# Initialization, since the slave-target specifies the “-defer-examine” parameter,
↪ only the other targets will be initialized.
init

# The host-target unlocks the slave-target with the command configuration registers
$HOST_TARGETNAME dmi_write/dm_write
$HOST_TARGETNAME dmi_read/dm_read

# Manually initialize slave-target
$SLAVE_TARGETNAME arp_examine
```

5.5 Frequently Asked Questions

For additional troubleshooting and common issues, refer to:

- GitHub FAQ[38].
- Gitee FAQ[39].

5.6 Low-Cost Debugger Solution

Nuclei provides an affordable debugging solution for RISC-V CPUs:

- Supports both JTAG and cJTAG protocols
- Fully compatible with Nuclei Studio
- Open-source implementation available

Dlink Repository[40].

5.7 Changelog

5.7.1 Version 2025.02

In this release, Nuclei OpenOCD bump openocd code base from 0.11 to 0.12, with this base changes, a lot of new features supported by upstream will be taken into Nuclei OpenOCD.

Known Issues:

- There is a probability that writing flash under smp multi-core architecture will fail, which can be solved by clarifying the number of SMP cores or reducing the JTAG frequency.

New Features:

- Live watch feature implementation, see *Live Watch* 功能的使用 (page 254)
- Nuclei CTI command group support
- Enhanced ETrace command group functionality, see *Trace* 功能的使用 (page 220)
- Optimized CPU information command

Improvements:

- Continuous integration and documentation enhancements
- Code organization: consolidated Nuclei commands into `nuclei_riscv.c`
- Register access optimization: replaced `vslide1down_vx` with direct `vx` register access for lite vpu case

CSR Updates:

New Custom CSRs:

Address Range	CSR Name
0x1a4~0x1af	smpuaddr4~15
0x1c0~0x1ef	smpuaddr16~63

CSR Renaming:

[38] https://github.com/riscv-mcu/riscv-openocd/wiki
[39] https://gitee.com/riscv-mcu/riscv-openocd/wikis
[40] https://github.com/Nuclei-Software/nuclei-dlink

Old Name	New Name
spmpcfg0~3	smpucfg0~3
spmpaddr0~15	smpuaddr0~15
mfp16mode	mmisc_ctl1
mecc_ctrl	mecc_ctl
mstack_ctrl	mstack_ctl

Base Version:

- Changes based on riscv-collab/riscv-openocd@f82c5a7[41].

[41] https://github.com/riscv-collab/riscv-openocd/commit/f82c5a7

CHAPTER

SIX

NUCLEI QEMU

6.1 About Nuclei QEMU

Nuclei QEMU is now developed based on QEMU version 9.0, supporting the machine features of Nuclei Evalsoc. In terms of extensions, in addition to the official upstream support, it also supports the following non ratified and custom extensions implemented by Nuclei.

- **Xxldsp**: P 0.5.4 draft extension and Nuclei custom N1/N2/N3 extensions
- **Xxlcz**: Nuclei custom code size reduction extension
- **Nice & Vnice**: Nuclei custom Nice & Vnice extensions
- **Xxlvqmacc**: Nuclei custom vpu extension
- **Zilsd & Zclsd**: riscv-zilsd Release 1.0

If you want to access the code of Nuclei QEMU, you can visit our opensource Nuclei QEMU Github Repository[42].

6.2 Design and Architecture

Previously, Nuclei QEMU supports two machine types: `nuclei_evalsoc` and `nuclei_demosoc`, but now, `nuclei_demosoc` has been abandoned on Nuclei QEMU 2025.02 and later versions.

6.2.1 Nuclei CPU Types Supported on QEMU

The following image shows the relevant information of the Nuclei CPU types supported by Nuclei QEMU.

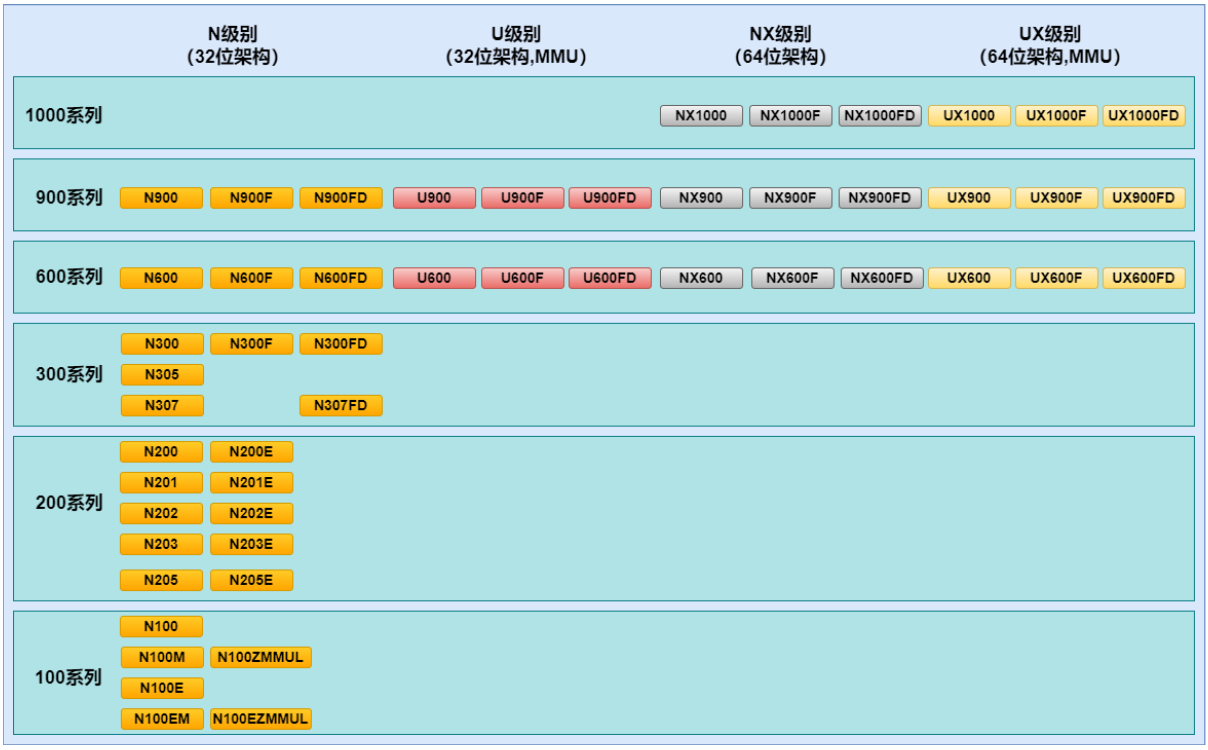
6.2.2 Address Allocation of Evalsoc on QEMU

For the sake of generality, all Nuclei IPs have been mapped and implemented on QEMU, some with fixed addresses and some dynamically initialized based on iregion information.

`Fixed`: The address is fixed by default in qemu.

`Offset`: The address is equal to iregion base address plus the corresponding offset.

[42] https://github.com/riscv-mcu/qemu/tree/nuclei/9.0



	Component	Base Address	Size	Description
Memory Resource	XIP	0x20000000	0x0200000(	XIP address space.
	DDR	0x80000000	0x0400000(	DDR address space.
	ILM	0x80000000	0x0080000(	ILM address space.
	DLM	0x90000000	0x0080000(	DLM address space.
	SRAM	0xA0000000	0x2000000(	SRAM address space.
Peripherals (Fixed)	IINFO	0	0x1000	IINFO address space.
	MROM	0x1000	0xf000	MROM address space.
	TEST	0x100000	0x10000	Space for Eval_SoC exit mechanism.
	GPIO	0x10012000	0x1000	GPIO address space.
	UART0	0x10013000	0x1000	UART0 address space.
	UART1	0x10023000	0x1000	UART1 address space.
	QSPI0	0x10014000	0x1000	QSPI0 address space.
	QSPI1	0x10024000	0x1000	QSPI1 address space.
	QSPI2	0x10034000	0x1000	QSPI2 address space.
Peripherals (Offset)	DEBUG	0x10000	0x1000	DEBUG address space.
	ECLIC	0x20000	0x10000	ECLIC address space.
	TIMER	0x30000	0x10000	TIMER address space.
	SMP	0x40000	0x1000	SMP address space.
	CIDU	0x50000	0x10000	CIDU address space.
	PLIC	0x4000000	0x4000000	PLIC address space.
	PPI	0xB0000000	/	Used to view CPU info, not implemented
	FIO	0xC0000000	/	Used to view CPU info, not implemented

In addition, Evalsoc ' s address mapping can be customized and configured through the `soc-cfg` option. Regarding the usage of this option, please see *Description of Parameters* (page 654).

6.2.3 Nuclei CPU Features Supported on QEMU

The following is the support status of Nuclei CPU features on qemu.

CPU Features	Status on QEMU
NMI	Not supported
TIMER	Supported, see *TIMER Support* (page 653)
PLIC	Supported, see *PLIC Support* (page 653)
ECLIC	Supported, see *ECLIC Support* (page 653)
CIDU	Supported, see *CIDU Support* (page 653)
PMP	Supported, see *PMP Support* (page 654)
TEE	Only CSRs Supported
WFI/WFE	Supported, see *WFI/WFE Support* (page 654)
ECC	Only CSRs Supported
CCM	Only CSRs Supported
SPMP	Not supported
SMP&CLUSTER CACHE	Supported, see *SMP&CLUSTER CACHE Support* (page 654)
UART	Supported, see *UART Support* (page 654)
GPIO	Supported, see *GPIO Support* (page 654)
QSPI	Supported, see *QSPI Support* (page 654)
TEST FINISHER	Supported, see *TEST Support* (page 654)

TIMER Support

TIMER currently supports normal access and interrupt triggering under two interrupt architectures, eclic and clint (plic) in m-mode, but the functionality in s-mode has not yet been implemented.

PLIC Support

There is already complete support for the pilc module in qemu, but when selecting **nuclei_evalsoc**, kernal needs to be passed through the `-bios` option to make it work in **PLIC** mode.

ECLIC Support

Now QEMU have been equipped with ECLIC, which is optimized based on the RISC-V standard CLIC, to manage all interrupt sources. ECLCI supports both single core and multi-core modes, but kernal needs to be passed through `-kernel` to make **nuclei_evalsoc** work in **ECLIC** mode.

CIDU Support

The CIDU is used to distribute external interrupts to the core ' s ECLIC, also it provides Inter Core Interrupt (ICI) and Semaphores Mechanism. Now QEMU supports ICI interrupt triggering and external interrupt distribution, but the semaphore mechanism needs to be improved.

PMP Support

The PMP function has been fully supported upstream, and all Nuclei CPUs in QEMU enable this by default.

WFI/WFE Support

The Nuclei processor core can support sleep mode for lower power consumption. In QEMU, WFI has been fully supported by upstream, while WFE only has CSR support.

SMP&CLUSTER CACHE Support

This module is designed to simulate Cluster Cache (CC) and Symmetric Multi-Processor (SMP), in a Nuclei MP core design (like UX900 MP core), it default integrates the Cluster Cache (CC) and SMP related module called Snoop Control Unit (SCU). However, due to the lack of complete cache support in QEMU, only register read and write as well as dynamic instantiation of CLM have been implemented for this module so far.

UART Support

Only basic data transmission and interrupt triggering have been implemented.

GPIO Support

Only basic input/output and interrupt triggering functions have been implemented.

QSPI Support

Currently, only the register mode of QSPI has been implemented in QMEU, which involves configuring relevant registers for data transmission and triggering interrupts.

TEST Support

This is an exit mechanism implemented for **nuclei_evalsoc** in QEMU. By writing different values `0x3333` / `0x5555` to `0x100000` during program execution, qemu can automatically exit in **Fail/ Pass** state. Writing `0x7777` will trigger system **reset**, initialize all devices, and run the program again.

6.3 Description of Parameters

Nuclei QEMU adds some custom features and functionalities based on the original capabilities of qemu. If you want to learn more about the usage of qemu, you can refer to the documentation at https://www.qemu.org/docs/master/.

Nuclei QEMU has several types of parameters that can be configured. You can enter `qemu-system-riscv32 --help` to view the parameters that can be configured in Nuclei QEMU.

Nuclei QEMU supports two main programs: `qemu-system-riscv32` and `qemu-system-riscv64`. `qemu-system-riscv32` is used to support 32-bit programs, while `qemu-system-riscv64` supports 64-bit programs.

This is an example of a fully functional parameter for Nuclei QEMU: `qemu-system-riscv32 -M nuclei_evalsoc,download=ddr,soc-cfg=evalsoc.json,debug=1 -cpu nuclei-n300fd, ext=_v_xxldsp,vlen=128,elen=64,s=true -m 512M -smp 1 -icount shift=0 -nodefaults -nographic -serial stdio -kernel dhrystone.elf`.

Let's describe the meaning of this complete command:

- `-M nuclei_evalsoc,download=ddr,soc-cfg=evalsoc.json,debug=1`:

  `-M` represents `machine`, which means selecting the type of machine. Currently, Nuclei QEMU has added `nuclei_evalsoc` to the existing options. This option must exist.

  `download=` is used to choose the download mode, and currently, it supports four download modes: `sram`, `flashxip`, `flash`, `ilm`, and `ddr`. If this parameter is not present, the default value is `flashxip`.

  `soc-cfg=` is an optional option to pass dynamic modifications to the initial configuration of the machine with a json file. If this parameter is not set, the default value of qemu will be used.

  Here is an example of json config file passed to `soc-cfg=`:

```
{
    "general_config": {
        "ddr": {
                "base":"0x70000000",
                "size":"2G"
        },
        "ilm": {
                "base":"0x90000000",
                "size":"0x100000"
        },
        "dlm": {
                "base":"0xA0000000",
                "size":"0x100000"
        },
        "sram": {
                "base":"0xB0000000",
                "size":"0x10000000"
        },
        "norflash": {
                "base":"0x30000000",
                "size":"32M"
        },
        "uart0": {
                "base":"0x20013000",
                "irq":"34"
        },
        "uart1": {
                "base":"0x20023000",
                "irq":"35"
        },
        "qspi0": {
                "base":"0x20014000",
                "irq":"36"
        },
        "qspi2": {
                "base":"0x20034000",
                "irq":"37"
        },
        "iregion": {
                "base":"0x1000000"
        },
        "cpu_freq":"50000000",
        "timer_freq":"32768",
        "irqmax":"100"
    },
    "download": {
        "ilm": {
                "startaddr":"0x90000000"
        },
```

(continues on next page)

(continued from previous page)

```
        "flashxip": {
                "startaddr":"0x30000000"
        },
        "flash": {
                "startaddr":"0x30000000"
        },
        "sram": {
                "startaddr":"0xB0000000"
        },
        "ddr": {
                "startaddr":"0x70000000"
        }
    }
}
```

general_config : mainly used to configure the board resource or chip base address

base: module base address, only support hex format

size: module size, support hex, dec, size string format

irq: peripheral interrupt id, dec format

download: firmware startup address

The following is a list of interrupt id for all interrupts implemented in qemu in both PLIC and ECLIC, users should follow this rule when configuring irq.

	Source	PLIC Interrupt ID	ECLIC Interrupt ID
Internal Interrupt	TIMER SW	/	3
	TIMER	/	7
	CIDU ICI	/	16
Internal Interrupt	GPIO 0 ~ 31	1 ~ 32	19 ~ 50
	UART0	33	51
	UART1	34	52
	QSPI0	35	53
	QSPI1	36	54
	QSPI2	37	55

In the above script, if there is no **download startaddr** information, the program entry will be the start address of the address range relative to the download mode. For example, when `download=ilm`, if the following configuration is not in the script,

```
"download": {
        "ilm": {
                "startaddr":"0x90000000"
        }
```

then the ilm base in **general_config** will be used as the program start address by default.

```
"general_config": {
     "ilm": {
             "base":"0x90000000",
             "size":"0x100000"
     }
```

Other configurations follow this rule as well.

> **Note**
>
> In the **general_config** JSON configuration script, the **base** attribute must coexist with either **size** or **irq**, and the format requires **base** to be written first, followed by either **size** or **irq**.

`debug=1` list the start address of the current device ' s peripherals and memory distribution information or irq info for debugging purposes. It is generally not recommended to enable this feature under normal circumstances.

- `-cpu nuclei-n300fd,ext=_v_xxldsp,vlen=128,elen=64,s=true`:

  Using the `-cpu` option, `nuclei-n300fd` represents the selectable CPU type for Nuclei, and the complete list of types can be referred to in the diagrams within the `Design and Architecture` section. This operation is necessary.

  `ext=` This parameter is optional, used to pass different riscv extension, The way to enable different extensions is to add them inside it, for example, `xxldsp` represents enable the nuclei DSP extension, `v` represents enable RISC-V V-Extension, When enabling multiple extensions, they are connected through _. Currently, Nuclei QEMU supports the following common RISC-V instruction set extension types:

Extension	Functionality
v	RISC-V V-Extension
h	RISC-V H-Extension
zicbom	RISC-V Zicbom Extension
zicboz	RISC-V Zicboz Extension
zicond	RISC-V Zicond Extension
zicsr	RV32/RV64 Zicsr Standard Extension
zifencei	RV32/RV64 Zifencei Standard Extension
zihintpause	ZiHintPause extension
zilsd	Zilsd extension (RV32 ONLY)
zclsd	Zclsd extension (RV32 ONLY)
zawrs	Zawrs extension
zfh	Zfh Extension
zfa	Zfa Extension
zfhmin	Zfhmin Extension
zfinx	Zfinx Extension
zdinx	Zdinx Extension
zca	RISC-V Zc* Extension
zcb	RISC-V Zc* Extension
zcf	RISC-V Zc* Extension
zcd	RISC-V Zc* Extension
zce	RISC-V Zc* Extension
zcmp	RISC-V Zc* Extension
zcmt	RISC-V Zc* Extension
zba	RISC-V Bitmanipulation Extension
zbb	RISC-V Bitmanipulation Extension
zbc	RISC-V Bitmanipulation Extension
zbkb	RISC-V Bitmanipulation Extension
zbkc	RISC-V Bitmanipulation Extension
zbkx	RISC-V Bitmanipulation Extension
zbs	RISC-V Bitmanipulation Extension
zk	RISC-V Scalar Crypto Extension
zkn	RISC-V Scalar Crypto Extension
zknd	RISC-V Scalar Crypto Extension
zkne	RISC-V Scalar Crypto Extension
zknh	RISC-V Scalar Crypto Extension
zkr	RISC-V Scalar Crypto Extension

continues on next page

Table 6.1 – continued from previous page

Extension	Functionality
zks	RISC-V Scalar Crypto Extension
zksed	RISC-V Scalar Crypto Extension
zksh	RISC-V Scalar Crypto Extension
zkt	RISC-V Scalar Crypto Extension
zve32x	RISC-V V-Extension
zve32f	RISC-V V-Extension
zve64x	RISC-V V-Extension
zve64f	RISC-V V-Extension
zve64d	RISC-V V-Extension
zvfh	RISC-V V-Extension
zvfhmin	RISC-V V-Extension
zhinx	Zhinx Extension
zhinxmin	Zhinxmin Extension
smaia	Smaia Extension
ssaia	Ssaia Extension
sscofpmf	Sscofpmf Extension
sstc	Sstc Extension
svadu	Svadu Extension
svinval	Svinval Extension
svnapot	Svnapot Extension
svpbmt	Svpbmt Extension
xxldsp	Nuclei DSP Extension based on P-ext 0.5.4 + default 8 EXPD instructions
xxldspn1x	Xxldsp + Nuclei N1 extension
xxldspn2x	Xxldspn1x + Nuclei N2 extension
xxldspn3x	Xxldspn2x + Nuclei N3 extension
xxlcz	Nuclei code size reduction extension
xxlvqmacc	Nuclei custom vpu extension

vlen=128,elen=64: The VLEN and ELEN are only effective when the V extension instructions of RISC-V are enabled. The default value of VLEN is 128, and it must be a multiple of 2 when set, with a value range of [128, 1024]. The default value of ELEN is 64, and ELEN must also be a multiple of 2, with a value range of [8, 64].

s=true: This parameter is optional, If you wish for RISC-V to support the S (supervisor) privilege mode, you can add s=true to the parameters to meet this requirement. Nuclei QEMU currently only supports interrupt handling in M-privilege mode.

- `-m 512M`: To set the DDR size in QEMU, if the DDR size is not passed with `-m`, then the JSON config will be used to determine the size, and lastly, if neither is specified, it will initialize with 32MB.

> **Note**
>
> The following is the current default qemu memory size configuration, **xip: 32MB**, **ddr:64MB**, **ilm: 8MB**, **dlm: 8MB**, **sram: 512MB**. You can change the size of the DDR by using **-m size**. When **-m 128M** or no `-m` is passed, the default DDR size configured in the JSON or the size initialized by the program will be used. If the DDR size is configured too large and the computer does not have enough memory to allocate, an error such as `qemu-system-riscv32: cannot set up guest memory 'riscv.evalsoc.ram.sram'` may occur.

- `-smp 1`: Nuclei Qemu currently supports up to 64 CPUs. If this parameter is not set, it defaults to 1.
- `-icount shift=0`: This parameter is optional, Qemu TCG Instruction Counting. By enabling this option, you can enable qemu ' s instruction count. For more detailed information, refer to https://www.qemu.org/docs/master/devel/tcg-icount.html
- `-nodefaults`: QEMU is used to disable all default devices and configurations, and some custom parameters

and commands can be passed.

- `-nographic`: Disable qemu ’ s graphical interface and redirect standard output to the console.
- `-serial stdio`: Direct standard output to the console.
- `-kernel or -bios`: Choose the boot mode for the firmware. By default, programs on nuclei-sdk load using the `-kernel` mode, while on Linux, they load using the `-bios` mode. In the design of Nuclei Qemu, `-kernel` enables the use of **ECLIC**. For bare metal or RTOS, `-kernel` is used to transfer ELF file, while `-bios` is used to enable **PLIC+CLINT** timers, which are more suitable for Linux applications.

6.4 Use Nuclei QEMU in Nuclei SDK

Setup Tools and Environment

1. Download the nuclei-sdk[43], checkout to `master` branch.
2. Download RISC-V GNU Toolchain form Nuclei Download Center[44].
3. Download Nuclei Qemu form Nuclei Download Center[45].
4. Set up the system environment variables to ensure that the directories containing `riscv64-unknown-elf-gcc` and `qemu-system-riscv32` are included in the global system variable environment.

Example

If you want to use QEMU on Nuclei-SDK. The example here uses the CPU of the nx900fd, but other CPU types can also be used for testing. The example is xxldsp.

First, you need to configure the toolchain, nuclei-sdk, and qemu environments according to the documentation https://doc.nucleisys.com/nuclei_sdk/quickstart.html

```
# Enter the example folder of xxldsp
cd nuclei-sdk/application/baremetal/demo_dsp/
# Clear the compilation cache
make clean
# Compile the program for the nx900fd, set the download mode to ILM, and enable␣
↪the xxldsp extension
make CORE=nx900fd SOC=evalsoc DOWNLOAD=ilm ARCH_EXT=_xxldsp dasm
# Automatically generate qemu running commands and execute the program
make CORE=nx900fd SOC=evalsoc DOWNLOAD=ilm ARCH_EXT=_xxldsp run_qemu
```

Where **ARCH_EXT** can be used to pass the extension name. Under normal circumstances, you should see the final output `NMSIS_TEST_PASS`, which indicates that all test cases have passed successfully.

Support for Nuclei SDK Cases on QEMU

`Y` - Successfully run and consistent with hardware

`N` - Successfully run but inconsistent with hardware

`F` - Failed

[43] https://github.com/Nuclei-Software/nuclei-sdk
[44] https://nucleisys.com/download.php
[45] https://nucleisys.com/download.php

Cases	SMP=1	SMP>1	Description (Additional compilation parameters and running status)
benchmark/ coremark	Y		
benchmark/ dhrystone	Y		
benchmark/ whetstone	Y		
cpuinfo/	Y		
demo_cache/	F		QEMU does not support cache emulation.
demo_cidu/		Y	SMP,XLCFG_CIDU,eg:SMP=1 XXLCFG_CIDU=1
demo_clint_timer/	Y		
demo_dsp/	Y		
demo_eclic/	Y		
demo_nice/	Y		
demo_plic/	Y	Y	XLCFG_PLIC, eg:XLCFG_PLIC=1
demo_pmp/	N		Not meeting expectations when TRIGGER_PMP_VIOLATION_MODE=LOAD_EXCEPTION.
demo_profiling/	Y		
demo_smode_ecli	F		Eclic does not yet support S mode.
demo_smpu/	F		XLCFG_SMPU, eg:XLCFG_SMPU=1, SPMU has not yet been implemented in qemu.
demo_spmp/	F		XLCFG_SPMP, eg:XLCFG_SPMP=1, SPMP has not yet been implemented in qemu.
demo_stack_checl	N		Only read and write access to CSRs.
demo_timer/	Y		
demo_vnice/	Y		
helloworld/	Y		
lowpower/	Y		
smphello/		Y	SMP, eg:SMP=4
freertos/demo/	Y		
freertos/ smpdemo/		N	SMP, eg:SMP=4, all tasks run on core0.
rtthread/demo/	Y		
rtthread/msh/	Y		
threadx/demo/	Y		
ucosii/demo/	Y		

And Nuclei QEMU and Nuclei SDK are deeply integrated in Nuclei Studio, you can also use it in Nuclei Studio, see *Nuclei Studio IDE* (page 3).

6.5 Use Nuclei QEMU in Nuclei Linux SDK

Nuclei QEMU can also used to boot and test RISC-V Linux Kernel using emulated Nuclei EvalSoC, please check documentation here https://github.com/Nuclei-Software/nuclei-linux-sdk#booting-linux-on-nuclei-qemu .

An example of a typical Nuclei QEMU running Nuclei Linux SDK is as follows:

```
qemu-system-riscv64 -M nuclei_evalsoc,download=flashxip,soc-cfg=soc.json -cpu␣
↪nuclei-ux900fd,ext= -smp 8 -m 2G -bios freeloader_qemu.elf -nographic -drive␣
↪file=disk.img,if=sd,format=raw
```

This command sets up QEMU to emulate a Nuclei processor and environment specifically for the Nuclei Linux SDK. Here ' s a breakdown of the parameters:

- `qemu-system-riscv64`: This is the QEMU emulator for the RISC-V 64-bit architecture.

- `-M nuclei_evalsoc`: Specifies the machine type for nuclei_evalsoc.
- `download=flashxip`: The download mode of firmware, which is an optional parameter. If not set, the default download mode is flashxip.
- `soc-cfg=evalsoc.json`: optional, additional configuration scripts can customize the interrupt information and memory address information of peripherals. For details, see Description of Parameters.
- `-cpu nuclei-ux900fd`: Selects the Nuclei UX900FD CPU model for emulation.
- `-ext=`: You can pass the extensions supported by riscv, and connect multiple extensions with _, eg. `_zba_zbb_zbc_zbs_zicond`.
- `-smp 8`: Enables Symmetric Multi-Processing (SMP) with 8 CPU cores.
- `-m 2G`: Allocates 2GB of RAM to the virtual machine.
- `-bios freeloader_qemu.elf`: Specifies the BIOS or bootloader to use, in this case a freeloader named freeloader_qemu.elf specifically for QEMU.
- `-nographic`: Disables graphical output, making QEMU run in a text-only mode.
- `-drive file=disk.img,if=sd,format=raw`: Attaches a virtual disk image named `disk.img` to the virtual machine, using the SD card interface (if=sd) and a raw file format (format=raw). This disk image likely contains the Nuclei Linux SDK filesystem.

6.6 Changelog

6.6.1 Version 2025.02

- Synchronize to the upstream QEMU 9.0
- Remove the Nuclei Demosoc support
- Add new cpu support for the Nuclei 200 and 1000 series
- Add Nuclei custom Xxlvqmacc extension support
- Add basic support for Nuclei gpio, qspi, test finisher, etc
- Zilsd/Zclsd extensions sync to upstream releases 1.0
- Add support for rv32/64 linux-user
- The maximum number of cpus supported by the Nuclei evalsoc smp has been adjusted from 16 to 64
- Fixed failure to respond to external interrupt in ECLIC
- Fixed Nuclei systimer support in CLINT mode

6.7 Known Issues

6.7.1 LiteOS-M is not able to run on Nuclei Qemu

This issue still existed in 2025.02 version, see https://github.com/riscv-mcu/qemu/issues/6

CHAPTER

SEVEN

NUCLEI MODEL

7.1 About Nuclei Near Cycle Model

The `Nuclei Near Cycle Model` (referred to as *xlmodel*) is a co-simulation using **SystemC TLM-2 combined with xlspike**. Xlspike uses spike as the RISC-V ISA simulator and adds support for Nuclei ' s N/NX/UX RISC-V processors. SystemC establishes the TLM 2.0 interaction relationships among the components under **Nuclei EvalSoC**.

- The *xlmodel* can be obtained in **Nuclei Studio 2025.02**.
- The *xlmodel* is supported on both **Linux and Windows** system.
- The *xlmodel* is a **near cycle model** that can test the performance of firmware with different ISA configurations.
- The *xlmodel* has SystemC built-in and uses version 2.3.4 by default.
- The *xlmodel* has built-in **gprof** functionality, allowing for a more intuitive display of the function call stack, the cycle count proportion of each function within the test code.
- The *xlmodel* can easily extend **user-defined** instructions, namely **Nuclei NICE instructions**.
- Xlspike supports RV32IMAFDCBPV and RV64IMAFDCBPV ISA.

Since this model is also integrated in Nuclei Studio, you can use it do to

- **Profiling and Code Coverage**
  - https://doc.nucleisys.com/nuclei_studio_supply/18-demonstrate_NICE_VNICE_acceleration_of_the_Nuclei_Model_through_profiling/
  - https://doc.nucleisys.com/nuclei_studio_supply/19-rapid_verification_of_NICE_VNICE_acceleration_with_Nuclei_Model_and_NICE_Wizard/
  - *Coverage*、*Profiling 和 Call Graph 使用* (page 203)
- **NICE Instruction Design and Modeling**
  - *Nuclei NICE Wizard* (page 241)

7.2 Design and Architecture

7.2.1 SystemC components

Brief description of the *xlmodel* SystemC components:

- Evalsoc: SystemC modeling of EvalSoC, including the binding relationships of all components and load image functionality
- Cluster: Modeling of single-core or multi-core CPU, using xlspike internally to execute instructions
- Bus: Simple bus manager

- Memory: Simple memory
- EvalUart: Nuclei EvalSoC UART
- EvalQspi: Nuclei EvalSoC QSPI

7.2.2 Address Allocation of Evalsoc in xlmodel

For the sake of generality, some Nuclei IP cores have been mapped and implemented in *xlmodel*. Some of these IP cores use fixed addresses, while others are dynamically initialized based on iregion information.

`Fixed`: The address is fixed by default in *xlmodel*.

`Offset`: The address is equal to iregion base address plus the corresponding offset.

	Component	Base Address	Size	Description
Memory Resource	XIP	0x20000000	0x02000000	XIP address space.
	DDR	0x80000000	0x04000000	DDR address space.
	ILM	0x80000000	0x00800000	ILM address space.
	DLM	0x90000000	0x00800000	DLM address space.
	SRAM	0xA0000000	0x20000000	SRAM address space.
Peripherals (Fixed)	MROM	0x1000	0xf000	MROM address space.
	UART0	0x10013000	0x1000	UART0 address space.
	QSPI0	0x10014000	0x1000	QSPI0 address space.
	QSPI1	0x10024000	0x1000	QSPI1 address space.
	QSPI2	0x10034000	0x1000	QSPI2 address space.
Peripherals (Offset)	ECLIC	0x20000	0x10000	ECLIC address space.
	TIMER	0x30000	0x10000	TIMER address space.
	PLIC	0x4000000	0x4000000	PLIC address space.

In addition, Evalsoc's address mapping can be customized and configured through the `--mem=<str>` parameter. Regarding the usage of this parameter, please see *Description of Parameters* (page 667).

7.2.3 Nuclei CPU Features Supported in xlmodel

The following is the support status of Nuclei CPU features in *xlmodel*.

CPU Features	Status in xlmodel
NMI	Supported
TIMER	Supported, see *TIMER Support* (page 665)
PLIC	Supported, see *PLIC Support* (page 665)
ECLIC	Supported, see *ECLIC Support* (page 665)
CIDU	Not supported
PMP	Supported
TEE	Supported
WFI/WFE	Not supported
ECC	Only CSRs Supported
CCM	Only CSRs Supported
SPMP	Not supported
SMP&CLUSTER CACHE	Not supported

TIMER Support

TIMER currently supports normal access and interrupt triggering under two interrupt architectures, eclic and clint (plic) in m-mode, but the functionality in s-mode has not yet been implemented.

PLIC Support

The PLIC interrupt architecture supports M-mode and S-mode, but it only supports single-core mode and does not support multi-core mode.

ECLIC Support

Now *xlmodel* have been equipped with ECLIC, which is optimized based on the RISC-V standard CLIC, to manage all interrupt sources. The ECLIC interrupt architecture supports M-mode and S-mode, but it only supports single-core mode and does not support multi-core mode.

7.2.4 Nuclei SDK Cases Supported in xlmodel

`Y` - Successfully run and consistent with hardware

`N` - Successfully run but inconsistent with hardware

`F` - Failed

Cases	SMP=	SMP>	Additional compilation parameters	Running Status
benchmark/ coremark/	Y			
benchmark/ dhrystone/	Y			
benchmark/ whetstone/	Y			
cpuinfo/	N			
demo_cache/	F		XLCFG_CCM, eg:XLCFG_CCM=1	xlmodel does not support cache emulation.
demo_cidu/	F	F	SMP,XLCFG_CIDU, eg:SMP=1 XXLCFG_CIDU=1	xlmodel does not support cidu.
demo_clint_ti	Y			
demo_dsp/	Y			
demo_eclic/	Y			
demo_nice/	Y			
demo_plic/	Y		XLCFG_PLIC, eg:XLCFG_PLIC=1	
demo_pmp/	Y			
demo_profiling	F			xlmodel already implements its own profiling, so there is no need to run this case.
demo_smode_	Y		XLCFG_TEE, eg:XLCFG_TEE=1	
demo_smpu/	Y		XLCFG_SMPU, eg:XLCFG_SMPU=1	
demo_stack_c	N			xlmodel only implements CSR read and write.
demo_timer/	Y			
demo_vnice/	Y			
helloworld/	Y			
lowpower/	Y			
smphello/		Y	SMP, eg:SMP=4	
freertos/ demo/	Y			
freertos/ smpdemo/		F	SMP, eg:SMP=4, all tasks run on core0.	xlmodel does not support SMP ECLIC.
rtthread/ demo/	Y			
rtthread/ msh/	Y			
threadx/ demo/	Y			
ucosii/demo/	Y			

And *xlmodel* and Nuclei SDK are deeply integrated in Nuclei Studio, you can run Nuclei SDK test code using *xlmodel* in Nuclei Studio, please refer to the *Nuclei Near Cycle Model* (page 248).

7.3 Description of Parameters

7.3.1 model help

Frequently used command line parameters

parameter	description
--version	display version info
--machine=<str>	machine type config, defaults to 'nuclei_evalsoc'
--cpu=<str>	core config, defaults to 'n300fd'
--ext=<str>	RISC-V arch extensions config, defaults to NULL
--mem=<str>	memory map config, mem="start_addr0:size0,start_addr1:size1…"
--timeout=<n>	expected real execution time(s); otherwise, it is unlimited
--bpu=<str>	core bpu type config, defaults to --bpu=n300 and can be set to n900
--smp=<n>	SMP system core number configuration, with a maximum of 16
--trace=<n>	whether generate the trace file, --trace=1 means generating
--gprof=<n>	whether to use profiling, --gprof=1 means using profiling
--varch=<str>	RISCV Vector uArch config, defaults to --varch=vlen:128,elen:64
--funcmode=<n>	Whether to only use the function model which instructions do not have cycle count in
--log=<str>	logging system level, defaults to --log=info and can be set to error, tlm, debug
--logdir=<str>	the directory to save trace and gprof files

You need to pass the different parameters above based on the results you want to obtain and the specific test code you are running.

7.3.2 parameter usage

When you want to use the model, you can select the executable file as follows:

- For Linux: `bin/xl_cpumodel`
- For Windows: `bin/xl_cpumodel.exe`

The following parameter examples are based on a Linux system. If you are using a Windows system, simply replace **xl_cpumodel** with **xl_cpumodel.exe**.

The default test code ELF files are provided in the `tests` directory.

- --cpu=<core_type>: Each test code needs to be run with this parameter. To see all Nuclei cores that are supported by *xlmodel*, refer to the Supported Nuclei Processor Cores section[46]. This is an example of running an ELF file with *nx900* core using the *xlmodel*:

```
./xl_cpumodel --cpu=nx900 ../tests/demotimer/demotimer_nx900.elf
```

[46] https://doc.nucleisys.com/nuclei_sdk/develop/buildsystem.html#core

```
xuzt@whml1:/Local/xuzt/modelNuclei/model/model_new/nuclei_cpu_model/build$ ./xl_cpumodel --cpu=nx900 ../tests/demotimer/demotimer_nx900.elf

        SystemC 2.3.4-Accellera --- May 16 2024 16:02:03
        Copyright (c) 1996-2022 by all Contributors,
        ALL RIGHTS RESERVED
[XLMODEL-INFO] filename[0]: ../tests/demotimer/demotimer_nx900.elf
[XLMODEL-INFO] Found the following .elf files:
[XLMODEL-INFO] ../tests/demotimer/demotimer_nx900.elf
[XLMODEL-INFO] Created Cluster0
[XLMODEL-INFO] start pc: 0x80000200
[XLMODEL-INFO] rv64 file isa: rv64imac
[XLMODEL-INFO] argv[0]: -t
[XLMODEL-INFO] argv[1]: -pl
[XLMODEL-INFO] argv[2]: -m0x70000000:0x90000000,0x20000000:0x10000000,0x60000000:0x8000000,0x10014000:0xc000,0x14000000:0x100000
[XLMODEL-INFO] argv[3]: +permissive-off
[XLMODEL-INFO] argv[4]: ../tests/demotimer/demotimer_nx900.elf
Nuclei SDK Build Time: Mar 27 2024, 10:30:49
Download Mode: ILM
CPU Frequency 692715 Hz
CPU HartID: 0
init timer and start
MTimer IRQ handler 1
MTimer IRQ handler 2
MTimer IRQ handler 3
MTimer IRQ handler 4
MTimer IRQ handler 5
MTimer SW IRQ handler 1
MTimer SW IRQ handler 2
MTimer SW IRQ handler 3
MTimer SW IRQ handler 4
MTimer SW IRQ handler 5
MTimer msip and mtip interrupt test finish and pass

[XLMODEL-INFO] total run 431608 instruction

Info: /OSCI/SystemC: Simulation stopped by user.
[XLMODEL-INFO] Total elapsed real time: 2.895097s
[XLMODEL-INFO] Press Enter to finish
```

- `--ext=`: This parameter is used to pass different riscv extension, the way to enable different extensions is to add them inside it. For example, `_xxldsp` represents enable the nuclei DSP extension, `v` represents enable RISC-V V Extension. Currently, *xlmodel* supports the following common RISC-V instruction set extension types:

Extension	Functionality
v	RISC-V V Extension
h	RISC-V H-Extension
_zicbom	RISC-V Zicbom Extension
_zicboz	RISC-V Zicboz Extension
_zicond	RISC-V Zicond Extension
_zicsr	RV32/RV64 Zicsr Standard Extension
_zifencei	RV32/RV64 Zifencei Standard Extension
_zihintpause	ZiHintPause extension
_zilsd	Zilsd extension (RV32 ONLY)
_zcmlsd	Zcmlsd extension (RV32 ONLY)
_zawrs	Zawrs extension
_zfh	Zfh Extension
_zfa	Zfa Extension
_zfhmin	Zfhmin Extension
_zca	RISC-V ZC* Extension
_zcb	RISC-V ZC* Extension
_zcf	RISC-V ZC* Extension
_zcd	RISC-V ZC* Extension
_zcmp	RISC-V ZC* Extension
_zcmt	RISC-V ZC* Extension
_zba	RISC-V Bitmanipulation Extension
_zbb	RISC-V Bitmanipulation Extension
_zbc	RISC-V Bitmanipulation Extension
_zbkb	RISC-V Bitmanipulation Extension
_zbkc	RISC-V Bitmanipulation Extension
_zbkx	RISC-V Bitmanipulation Extension
_zbs	RISC-V Bitmanipulation Extension
_zk	RISC-V Scalar Crypto Extension
_zkn	RISC-V Scalar Crypto Extension
_zknd	RISC-V Scalar Crypto Extension
_zkne	RISC-V Scalar Crypto Extension

continues on next page

Table 7.1 – continued from previous page

Extension	Functionality
_zknh	RISC-V Scalar Crypto Extension
_zkr	RISC-V Scalar Crypto Extension
_zks	RISC-V Scalar Crypto Extension
_zksed	RISC-V Scalar Crypto Extension
_zksh	RISC-V Scalar Crypto Extension
_zkt	RISC-V Scalar Crypto Extension
_zve32x	RISC-V V Extension
_zve32f	RISC-V V Extension
_zve64x	RISC-V V Extension
_zve64f	RISC-V V Extension
_zve64d	RISC-V V Extension
_zvfh	RISC-V V Extension
_zvfhmin	RISC-V V Extension
_sscofpmf	Sscofpmf Extension
_sstc	Sstc Extension
_svinval	Svinval Extension
_svnapot	Svnapot Extension
_svpbmt	Svpbmt Extension
_xxldsp	Nuclei DSP Extension based on P-ext 0.5.4 + default 8 EXPD instructions
_xxldspn1x	Xxldsp + Nuclei N1 extension
_xxldspn2x	Xxldspn1x + Nuclei N2 extension
_xxldspn3x	Xxldspn2x + Nuclei N3 extension
_xxlcz	Nuclei code size reduction extension

This is an example of running an ELF file with *_zba_zbb_zbc_zbs_xxldspn1x* extension using the *xlmodel*:

```
./xl_cpumodel --cpu=n300fd --ext=_zba_zbb_zbc_zbs_xxldspn1x ../tests/demodsp/demo_
↪dsp_n300fd.elf
```

- `--varch=vlen:n,elen:n`: The VLEN and ELEN are only effective when the V extension instructions of RISC-V are enabled. Note that VLEN and ELEN must comply with the RISC-V Vector specifications. Example:

```
./xl_cpumodel --cpu=ux900fd --ext=v --varch=vlen:128,elen:64 ../tests/rvv_conv_
↪f32/rvv_conv_f32.elf
./xl_cpumodel --cpu=ux900fd --ext=v --varch=vlen:256,elen:64 ../tests/rvv_conv_
↪f32/rvv_conv_f32.elf
./xl_cpumodel --cpu=ux900fd --ext=v --varch=vlen:512,elen:64 ../tests/rvv_conv_
↪f32/rvv_conv_f32.elf
./xl_cpumodel --cpu=ux900fd --ext=v --varch=vlen:1024,elen:64 ../tests/rvv_
↪conv_f32/rvv_conv_f32.elf
```

- `--smp=n`: *xlmodel* currently supports up to 16 CPUs. If this parameter is not set, then uses 1 CPU. Running the 4-core SMP example is as follows:

```
./xl_cpumodel --cpu=nx900 --smp=4 ../tests/smphello_4core/smphello_nx900.elf
```

- `--mem=start_addr0:size0,start_addr1:size1...`: when you compile the test code using custom sections, you need to pass the memory map of the SoC, i.e., the starting address and sizes of each section, as parameters to the *xlmodel*. Example:

```
./xl_cpumodel --cpu=nx900 --mem="0x70000000:0x90000000,0x20000000:0x10000000" .
↪./tests/democlic_swirq_high/demo_eclic_swirq_high.elf
```

- `--bpu=xxx`: The *xlmodel* can select different **BPU strategies** based on Nuclei core types, which will affect the cycle count of branch and jump instructions. *xlmodel* currently supports the `--bpu=n300` and `--bpu=n900` parameters.

  - The `--bpu=n300` parameter is applicable to Nuclei cores up to and including N300 (<= N300):

    ```
    ./xl_cpumodel --cpu=n300fd --ext=_zba_zbb_zbc_zbs_xxldspn1x --bpu=n300 ../tests/demodsp/demo_dsp_n300fd.elf
    ```

  - The `--bpu=n900` parameter is applicable to Nuclei cores from N300 onwards (> N300):

    ```
    ./xl_cpumodel --cpu=nx900 --bpu=n900 ../tests/demotimer/demotimer_nx900.elf
    ```

- `--trace=1`: *xlmodel* currently supports outputting instruction trace streams during execution, The `<elf-name>.rvtrace` file will be generated in the path specified by the `--logdir=<path>` parameter. If `--logdir=<path>` is not configured, it will be generated in the current execution path. Example:

  ```
  ./xl_cpumodel --cpu=nx900 --trace=1 ../tests/demoeclic_swirq_low/demo_eclic_swirq_low.elf                         // rvtrace file in current execution path
  ./xl_cpumodel --cpu=nx900 --trace=1 --logdir=../log ../tests/demoeclic_swirq_low/demo_eclic_swirq_low.elf        // rvtrace file in User-defined path
  ```

You can obtain information such as the instruction count, cycle count, the associated hart, pc, opcode, disassembly code for each instruction in generated `<elf-name>.rvtrace` file.

- `--gprof=1`: This parameter is used to enable the built-in **gprof** functionality of *xlmodel*. The `gprof<n>.gmon` and `gprof<n>.log` files will be generated in the path specified by the `--logdir=<path>` parameter. If `--logdir=<path>` is not configured, it will be generated in the current execution path. Example:

  ```
  ./xl_cpumodel --cpu=nx900 --gprof=1 ../tests/whet/whet_nx900.elf          // gprof files in current execution path
  ./xl_cpumodel --cpu=nx900 --gprof=1 --logdir=../log ../tests/whet/whet_nx900.elf         // gprof files in User-defined path
  ```

You can obtain the `gprof<n>.gmon` and `gprof<n>.log` files when the simulation is complete, where n represents the hart ID.

To use them further, you need to import them into the IDE, then you can refer to the model usage guide in the Nuclei Studio for detailed instructions on using **gprof**.

- `--timeout=<n>`: You can pass this parameter to set the real execution duration for *xlmodel*. When the timeout period is reached or when *xlmodel* finishes running the test code, the `gprof<n>.gmon` and `gprof<n>.log` files will be generated if the `--gprof=1` parameter is enabled. An example of specifying a 20-second simulation is as follows:

  ```
  ./xl_cpumodel --cpu=nx900 --timeout=20 --gprof=1 ../tests/cmk/cmk_nx900.elf
  ```

- `--log=xxx`: The *xlmodel* has multiple log levels, listed from least to most detailed as *error*, *info*, *debug*, *tlm*. By default, it provides basic log information at the *info* level, which can be changed by passing `--log=xxx`.

  When `--log=error` is selected, it only outputs error messages generated during the xlmodel runtime:

  ```
  ./xl_cpumodel --cpu=nx900 --log=error ../tests/demotimer/demotimer_nx900.elf
  ```

  When `--log=debug` is selected, the following options provide additional detailed information:

  - `--trace=1`: The `<elf-name>.rvtrace` file will contain detailed trace information, including register updates, exceptions, and CSRs:

    ```
    ./xl_cpumodel --cpu=nx900 --log=debug --trace=1 ../tests/demotimer/demotimer_nx900.elf
    ```

  - `--gprof=1`: It will output pc jump information for jump instructions, exceptions, and interrupts to the terminal:

```
./xl_cpumodel --cpu=nx900 --log=debug --gprof=1 ../tests/demotimer/
↪demotimer_nx900.elf
```

When `--log=tlm` is selected, it includes all the features of `--log=debug` and additionally outputs TLM bus read and write information to the terminal:

```
./xl_cpumodel --cpu=nx900 --log=tlm ../tests/demotimer/demotimer_nx900.
↪elf
```

7.4 NICE support

7.4.1 NICE build

If you need to validate your custom **NICE** instructions, you need to contact Nuclei Support to obtain software package of model (*xlmodel_nice*).

The directory structure of *xlmodel_nice* is as follows, you need to implement the **NICE** instructions in *nice/nice.cc*.

nice directory	description
nice	header and source files for the NICE interface
systemc	SystemC 2.3.4 header files and static libraries
xl_model	xlmodel header files and library files
xl_spike	xlspike header files and library files
tests	simple test codes
CMakeLists.txt	CMake file required for compilation

After implementing the **NICE** instruction, you need to recompile *xlmodel_nice*.

nice build for Linux

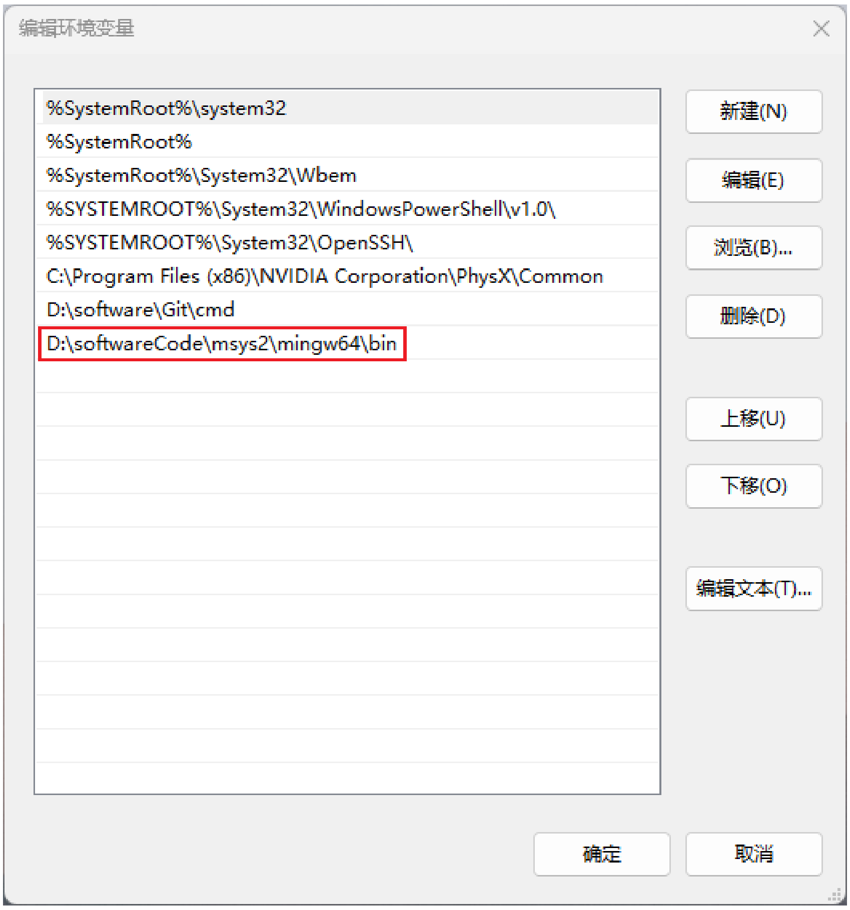
```
# Install essential compilation tools
sudo apt install build-essential cmake
# Check the tools have been installed successfully
gcc -v && g++ -v && cmake --version
# Change to the root directory of the xlmodel_nice package
cd <xlmodel_nice root directory>
mkdir build && cd build
cmake -DCMAKE_BUILD_TYPE=Release ..
make -j$(nproc)
```

nice build for Windows

To compile *xlmodel_nice* on Windows, you need to download a Windows-compatible GCC tool, such as **MinGW64**. You can download **MSYS2** to easily obtain the MinGW64 toolchain, which simplifies the installation and management of MinGW64 on Windows.

Below are the steps to use MSYS2 ' s MinGW64 to compile *xlmodel_nice* on Windows:

1. Install the latest version of MSYS2 from https://www.msys2.org/, and then add the MinGW64 toolchain path to the **environment variables**:



2. Install MinGW64 toolchain, CMake, and other basic compilation tools in the **MSYS2 terminal**:

```
pacman -S base-devel mingw-w64-x86_64-gcc mingw-w64-x86_64-cmake
```

3. Compile *xlmodel_nice* in the **MinGW64 terminal**:

```
# Check the tools have been installed successfully
gcc -v && g++ -v && cmake --version
# Change to the root directory of the xlmodel_nice package
cd <xlmodel_nice root directory>
mkdir build && cd build
cmake -G "Unix Makefiles" -DCMAKE_BUILD_TYPE=Rlease ..
make -j$(nproc)
```

Note

If you need to import and compile *xlmodel_nice* in Nuclei Studio (*Nuclei Studio use xlmodel* (page 248)), you also need to configure the **MSYS2** environment as described above first.

7.4.2 NICE example

If you are unfamiliar with how to implement the NICE instruction, refer to the implementation in *xlmodel_nice/nice/nice.cc* for custom **Nuclei NICE/VNICE** instructions.

The test example for Nuclei NICE instruction features are located in *tests/demonice*:

```
./xl_cpumodel --cpu=nx900 ../tests/demonice/demonice_nx900.elf
```

The test example for Nuclei VNICE instruction features are located in *tests/demovnice*:

```
./xl_cpumodel --cpu=nx900fd --ext=v ../tests/demovnice/demovnice_nx900fd.elf
```

7.5 SystemC integration

If you need to integrate *xlmodel* into a third-party SoC simulation platform, you need to contact Nuclei AE for the software integration package of the model (*xlmodel_integration*).

xlmodel_integration mainly includes the source code of *SystemC components* (page 663). You can extract the parts you need and use them on third-party platforms with relatively little adaptation.

The main directory structure of *xlmodel_integration* is as follows:

directory	description
src	All SystemC components, Cluster and Timer are mandatory to integrate, while others be integrated as needed
inc	All header files related to xlmodel, which are integrated as needed
systemc	SystemC 2.3.4 header files and static libraries, which are integrated as needed
cycle	The near cycle lib file, must be integrated
profiling	The xlmodel profiling lib file, which is integrated as needed
xlspike	xlspike header files and library files, must be fully integrated
tests	simple test codes
nice	header and source files for the NICE interface, must be fully integrated
CMakeLists.txt	CMake file required for compilation

Here are the integration steps instructions:

1. Find the Cluster class in *inc/cluster.h*:
   - The Cluster class contains two types of variables:
     - **Internal variables**: Initialized to default values in the constructor ' s member initialization list.
     - **User-defined variables**: Initialization in the constructor ' s member list depends on your **firmware, CPU configurations and extra simulation feature needs**.
   - Usage recommendations:
     - You can inherit the Cluster class in your SystemC platform class to use base methods of your platform.
2. Find the Cluster class in *src/cluster.cpp*:
   - Usage recommendations:
     - If you do not need to replace the SystemC sockets and TLM methods in the Cluster class, you can directly use the constructor and methods in the Cluster class without any modification.
     - Adjust the user-defined variables in the constructor ' s member initialization list.

    - If your platform implements TLM transmission from CPU to Bus, you can substitute the read/write implementations into `cpu_read_tlm()` and `cpu_write_tlm()`.

3. Use External Interrupts:
    - To connect User-defined external interrupts to the CPU:
        - Configure the number of external interrupts by updating `ext_irq_num` during Cluster instantiation.
        - Bind your device ' s interrupt socket to the CPU.
        - Use the device socket ' s `b_transport` method to pass the **irq_id** to the CPU.

7.6 Changelog

7.6.1 Version 2025.02

Known Issues:

- Does not support smp ECLIC, CIDU and CACHE emulation

New Features:

- Nuclei 200/300/900/1000 are supported, and support cpu core name are aligned with Nuclei SDK 0.7.1 release.
- Added support for Windows system, now both windows and linux system are supported.
- Added important parameters such as `--cpu`, `--ext`, and `--mem`, and changed the `--time` parameter to `--timeout`
- Added rough cycle calculation for vector instructions

Improvements:

- Improved the CPU, Timer, and UART models
- Improved the *xlmodel_nice* software package
- Resolved the issue where some instructions could not be profiled
- Adapted to run *demo_vnice*, *demo_plic*, and multiple RTOS cases in nuclei-sdk

CHAPTER

EIGHT

CHANGELOG

Nuclei Tools official releases can be found in https://nucleisys.com/download.php#tools, you can download it from there.

You can also find our **Nuclei Studio Supply Documents** in https://doc.nucleisys.com/nuclei_studio_supply/, which is used for application notes using Nuclei Studio and Nuclei Tools.

8.1 2025.02

This is 2025.02 release of Nuclei Tools.

For this release pdf doc, please check https://doc.nucleisys.com/nuclei_tools/nuclei_tool_user_guide.pdf

Find release note below:

- Nuclei Studio 2025.02[47] : Add a lot new features such as Live watch, Modeling integration, flash programming, and Nice Wizard etc.
- Nuclei RISC-V Toolchain 2025.02[48] : Bump to GCC 14 and LLVM 19.
- Nuclei QEMU 2025.02[49] : Bump to Qemu 9.0.
- Nuclei OpenOCD 2025.02[50] : Bump to OpenOCD 0.12.0

Click https://github.com/Nuclei-Software/nuclei-tool-guide/issues/18 for this release known issues.

8.2 2024.06

This is 2024.06 release of Nuclei Tools.

This release introduced *About Nuclei Near Cycle Model* (page 663) first release which can be used in Ubuntu 20.04 environment.

For this release pdf doc, please check https://doc.nucleisys.com/nuclei_tools/nuclei_tool_user_guide_2024.06.pdf

Find release note below:

- Nuclei Studio 2024.06[51]
- Nuclei RISC-V Toolchain 2024.06[52]
- Nuclei QEMU 2024.06[53]

[47] https://github.com/Nuclei-Software/nuclei-studio/releases/tag/2025.02
[48] https://github.com/riscv-mcu/riscv-gnu-toolchain/releases/tag/nuclei-2025.02
[49] https://github.com/riscv-mcu/qemu/releases/tag/nuclei-2025.02
[50] https://github.com/riscv-mcu/riscv-openocd/releases/tag/nuclei-2025.02
[51] https://github.com/Nuclei-Software/nuclei-studio/releases/tag/2024.06
[52] https://github.com/riscv-mcu/riscv-gnu-toolchain/releases/tag/nuclei-2024.06
[53] https://github.com/riscv-mcu/qemu/releases/tag/nuclei-2024.06

- Nuclei OpenOCD 2024.06[54]

  Click https://github.com/Nuclei-Software/nuclei-tool-guide/issues/4 for this release known issues.

8.3 2024.02

This release is a bugfix release for 2023.10, which fix many issues reported by 2023.10.

See https://github.com/Nuclei-Software/nuclei-tool-guide/issues/1

8.4 2023.10

This is 2023.10 release of Nuclei Tools.

Find release note below:

- Nuclei Studio 2023.10[55]
- Nuclei RISC-V Toolchain 2023.10[56]
- Nuclei QEMU 2023.10[57]
- Nuclei OpenOCD 2023.10[58]

  Click https://github.com/Nuclei-Software/nuclei-tool-guide/issues/1 for this release known issues.

[54] https://github.com/riscv-mcu/riscv-openocd/releases/tag/nuclei-2024.06
[55] https://github.com/Nuclei-Software/nuclei-studio/releases/tag/2023.10
[56] https://github.com/riscv-mcu/riscv-gnu-toolchain/releases/tag/nuclei-2023.10
[57] https://github.com/riscv-mcu/qemu/releases/tag/nuclei-2023.10
[58] https://github.com/riscv-mcu/riscv-openocd/releases/tag/nuclei-2023.10

CHAPTER NINE

GLOSSARY

API
(Application Program Interface) A defined set of routines and protocols for building application software.

ISR
(Interrupt Service Routine) Also known as an interrupt handler, an ISR is a callback function whose execution is triggered by a hardware interrupt (or software interrupt instructions) and is used to handle high-priority conditions that require interrupting the current code executing on the processor.

XIP
(eXecute In Place) a method of executing programs directly from long term storage rather than copying it into RAM, saving writable memory for dynamic data and not the static program code.

CHAPTER

TEN

APPENDIX

- **Nuclei Tools and Documents**: https://nucleisys.com/download.php
- **Nuclei Software Opensource Organization**: https://github.com/Nuclei-Software
- **RISC-V MCU Opensource Organization**: https://github.com/riscv-mcu
- **Nuclei Toolchain Repo**: https://github.com/riscv-mcu/riscv-gnu-toolchain
- **Nuclei OpenOCD Repo**: https://github.com/riscv-mcu/riscv-openocd
- **Nuclei QEMU Repo**: https://github.com/riscv-mcu/qemu
- **Nuclei SDK**: https://github.com/Nuclei-Software/nuclei-sdk
- **NMSIS**: https://github.com/Nuclei-Software/NMSIS
- **Nuclei RISC-V IP Products**: https://www.nucleisys.com/product.php
- **RISC-V MCU Community Website**: https://www.riscv-mcu.com/
- **Nuclei RISC-V CPU Spec**: https://doc.nucleisys.com/nuclei_spec
- **RISC-V ELF psABI Document**: https://github.com/riscv-non-isa/riscv-elf-psabi-doc#navigation
- **RISC-V ISA Specfications(Ratified)**: https://riscv.org/technical/specifications
- **RISC-V Architecture Profiles**: https://github.com/riscv/riscv-profiles
- **RISC-V Bitmanip(B) Extension Spec**: https://github.com/riscv/riscv-bitmanip
- **RISC-V Packed SIMD(P) Extension Spec**: https://github.com/riscv/riscv-p-spec
- **RISC-V Cryptography(K) Extension Spec**: https://github.com/riscv/riscv-crypto
- **RISC-V Vector(V) Extension Spec**: https://github.com/riscv/riscv-v-spec
- **RISC-V Vector Intrinsic API Spec**: https://github.com/riscv-non-isa/rvv-intrinsic-doc
- **RISC-V ISA Extension Spec Status**: https://wiki.riscv.org/display/HOME/Specification+Status
- **Nuclei Bumblebee Core Document**: https://github.com/nucleisys/Bumblebee_Core_Doc

CHAPTER

ELEVEN

INDICES AND TABLES

- genindex
- search

INDEX