芯来科技
NUCLEI

Nuclei Studio

IDE 使用说明

Release 2024.06

版权声明

联系我们

若您有任何疑问，请通过电子邮件 support@nucleisys.com 联系芯来科技。

修订历史

版本号	修订日期	修订的章节	修订的内容
1.0	2019/10/19	N/A	1. 初始版本
1.1	2020/2/10	N/A	1. 修订了若干文字错误
1.2/ 2020.08	2020/08/06	ALL	1. 全新的 Nuclei Studio IDE 开发工具 2. 采用 Project Wizard 方式全面支持 Nuclei SDK 3. 增加 Nuclei SDK 工程设置工具 4. 当前 Nuclei SDK 集成版本为 v0.2.5 5. 当前 Nuclei 工具链版本为 2020.07 6. Nuclei Studio 2020.08 版本正式发布 7. 集成 IDE Plugins 版本为 1.0.0
1.3/ 2020.09	2020/09/01	1、3、4、5、6	1. 增加 Linux 环境使用介绍。 2. 增加使用 J-Link 调试运行项目 3. IDE 不再需要单独安装 JDK 环境 4. 增加支持 HummingBird E203 模板工程，集成 Hbird SDK V0.1.0 5. 当前 Nuclei SDK 集成版本为 v0.2.6 6. 当前 Nuclei 工具链版本为 2020.08 7. Nuclei Studio 2020.09 版本正式发布 8. 集成 IDE Plugins 版本为 1.0.2
1.3/ 2020.09	2020/09/04	6.2.2	1. 修订 J-Link 的 Device 选项描述
1.4/ 2021.02	2021/02/02	8、10	1. 新增常见问题章节 2. 新增 QEMU 仿真调试章节

版本号	修订日期	修订的章节	修订的内容
			3. 增加支持 gd32vf103c_longan_nano 开发板模板工程 4. 当前 OpenOCD 版本更新为 v0.10.0-15 5. 当前 Eclipse CDT 版本更新为 2020.09 6. 当前 Nuclei SDK 集成版本为 v0.3.0 7. 当前 Hbird SDK 集成版本为 v0.1.2 8. Nuclei Studio 2021.02 版本正式发布 9. 集成 IDE Plugins 版本为 1.0.8
1.5/ 2021.09	2021/02/09		1. 修改第 3 章 Nuclei Studio 的介绍 2. 修改第 4.1 章 Nuclei Studio 创建工程 3. 修改第 10 章 对 QEMU 的说明 4. 修改 4.4.2 章设置 Makefile 路径和 Build 选项 5. 新增常见问题 6. Nuclei Studio 2021.09 版本正式发布 7. 集成 IDE Plugins 版本为 1.1.0
1.6/ 2022.01	2022/01/17	全部	1. 更新的部分插图 2. 正式发布 Nuclei Package(NPK)功能以及介绍如何使用 3. 修改第 3 章 关于 SDK Configuration Tools 功能的描述
1.7/ 2022.04	2022/04/1		1. 更新的部分插图 2. 新增第 12 章
1.8/ 2022.08	2022/08/30		1. 新增了 4.2 章 2. 新增 9.18 章
1.9/ 2022.12	2022/12/27		1.更改 4.1 章节 2.更改 5.章，其中新增了 5.2 章，5.3、5.4、5.5 章为原有章节

版本号	修订日期	修订的章节	修订的内容
			3.更改 8.章节 4.新增 9.19 章节
2.0/ **2023.10**	2023/11/1		1.新增 2.1 章节 2.更改 4.1.2 章节 3.更改 8.章节 4.新增第 9 章节 5.新增第 13.20 章节 6.版本变更内容参见 2.3 2023.10 版更新说明
2.1/ **2024.02.dev**	2024/02/28		1.修增 2.1 章节 2.新增第 8.1.3 章节 3.新增第 10 章节 4.新增第 11 章节 5.新增第 13.23 章节 6.版本变更内容参见 2.2 2024.02.dev 版更新说明
2.2/ **2024.06**	2024/07/02		1.修增 2.1 章节 2.修改第 4.1.1 章节 3.修改第 7 章节 4.新增第 9 章节 5.修改第 10 章节 6.修改第 11 章节 7.新增第 13 章节 8.版本变更内容参见 2.1 2024.06 版更新说明

目　录

版权声明 **2**

联系我们 **2**

修订历史 **1**

目　录 **4**

图片清单 **12**

1. NUCLEI STUDIO IDE 简介 **24**

2. NUCLEI STUDIO 更新说明 **25**

2.1. 2024.06 版更新说明 26

2.1.1. 升级 Eclipse CDT 版本 *26*

2.1.2. 升级 RISC-V Toolchain、OpenOCD、QEMU 版本 *26*

2.1.3. 新增对 U600 和 UX1000 的支持 *26*

2.1.4. 优化 NPK 软件包管理 *26*

2.1.5. 增加和优化部分编译选项 *26*

2.1.6. 优化调式模式切换 *27*

2.1.7. 优化和完善 DLink Debug 调试 *27*

2.1.8. 集成 Terapines ZCC Lite 编译器 *27*

2.1.9. 新增 LST View 工具 *27*

2.1.10. 优化和完善 Gprof 功能 *27*

2.1.11. 优化和完善 Gcov 功能 *27*

2.1.12. 新增 Call Graph 功能 *28*

2.1.13. 新增 Nuclei Near Cycle Model 支持 *28*

2.2. 2024.02.DEV 版更新说明 28

2.2.1. 升级 Eclipse CDT 版本 *28*

2.2.2. 新增对 N100 的支持 *28*

2.2.3. 新增批量转换 Gcc13 工程工具 *28*

2.2.4. 优化和完善 Trace 功能 *28*

2.2.5. 优化和完善 RVProf 功能 *28*

2.2.6. 新增对 DLlink Debug 的支持 *29*

2.3. 2023.10 版更新说明 29

2.3.1. 升级 Eclipse CDT 版本 *29*

2.3.2. 升级 build-tools 版本 *30*

2.3.3. 支持 GCC 13 和 Clang 17 ..... *30*

2.3.4. RISC-V 指令扩展使用变更 *33*

2.3.5. NPK 包的使用变更 *37*

2.3.6. 升级 OpenOCD *38*

2.3.7. 升级 QEMU *38*

2.3.8. 新增了 elf 文件查看器 *39*

2.3.9. 新增 Code Coverage 和 Profiling 功能 *40*

2.3.10. 新增 trace 功能 *40*

2.3.11. Nuclei Settings 功能优化 *40*

2.3.12. 新增指定工作空间快速打开 *43*

2.4. 2022.12 版更新说明 43

3. NUCLEI STUDIO 下载与安装 44

3.1. NUCLEI STUDIO IDE 下载 44

3.2. NUCLEI STUDIO IDE 安装 45

3.3. NUCLEI STUDIO IDE 启动 46

4. NUCLEI STUDIO NPK 功能 48

4.1. 通过 NPK 创建工程 49

4.1.1. NPK 软件包管理 49

4.1.2. 创建 NPK 示例工程 53

4.1.3. SDK Configuration Tools 更改工程配置 57

4.1.3.1 SDK Configuration Tools 57

4.1.3.2 Nuclei Settings 菜单 58

4.2. 通过 NPK 导入工具 59

5. 创建工程 63

5.1. 通过模板自动创建项目 64

5.2. 通过应用关联文件导入工程 71

5.2.1. 将 Nuclei Studio IDE 写入到注册表 71

5.2.2. 通过应用关联文件导入工程 72

5.3. 从已有项目直接导入创建新项目 73

5.4. 无模板手动创建项目 77

5.4.1. 手动创建项目 77

5.4.2. 配置项目的 nuclei_sdk 83

5.4.3. 配置项目的编译和链接选项 87

5.4.4. 配置项目的包含路径和文件 95

5.5. 基于已有的 MAKEFILE 创建项目 98

5.5.1. 手动新建项目 *98*

5.5.2. 设置 Makefile 路径和 Build 选项 *100*

6. 编译工程 102

6.1. 编译工具 104

6.1.1. Nuclei GNU Toolchain *104*

6.1.2. Nuclei LLVM Toolchain *106*

6.1.3. Terapines ZCC *108*

6.2. NUCLEI SDK 工程设置工具 110

6.3. NUCLEI STUDIO 中编译 HELLO WORLD 项目 115

7. 调试运行工程 118

7.1. 调试模式管理 118

7.2. 使用蜂鸟调试器结合 OPENOCD 调试运行项目 120

7.2.1. 安装蜂鸟调试器驱动 *121*

7.2.1.1 在 Windows 系统中安装驱动 121

7.2.1.2 在 Linux 系统中安装驱动 122

7.2.2. Debug Configuration *124*

7.2.2.1 使用 Nuclei Studio 生成的 Debug Configuration 124

7.2.2.2 新建并配置 Debug Configuration 127

7.2.3. 在原型开发板上调试程序 *131*

7.2.4. 下载运行程序 *136*

7.3. 使用 J-LINK 调试运行项目 138

7.3.1. 安装 J-Link 驱动并导入 RTT 文件 *138*

7.3.2. Debug Configuration *143*

7.3.2.1 使用 Nuclei Studio 生成的 Debug Configuration 144

7.3.2.2 新建并配置 Debug Configuration 147

7.3.3. 在原型开发板上调试程序 *151*

7.3.4. 下载运行程序 *156*

7.4. 使用 DLink 调试运行项目 158

8. 导入旧版本 NUCLEI STUDIO 创建的工程 166

8.1. Nuclei Studio 2023.10 版导导入旧工程 166

8.1.1. 手动导入 GCC 10 工具链 *167*

8.1.2. 通过工具将工程转换成支持 gcc 13 的工程 *168*

8.1.3. 批量将工程转换成支持 gcc 13 的工程 *169*

8.2. Nuclei Studio 2022.12 之后版本导入旧工程 172

8.3. Nuclei Studio 2022.12 之前版本导入旧工程 176

9. LST VIEW 177

10. CODE COVERAGE 和 PROFILING 功能 180

10.1. 关于 Code Coverage 功能 181

10.2. 关于 Profiling 功能 181

10.3. 关于 Call Graph 功能 182

10.3.1. Radial View *182*

10.3.2. Tree View *183*

10.3.3. Level View *184*

10.3.4. Aggregate View *185*

10.4. COVERAGE、PROFILING 和 CALL GRAPH 使用 186
10.4.1. 通过串口使用 *186*
10.4.2. 通过 Semihosting 使用 *198*
11. TRACE 功能的使用 206
11.1. TRACE 界面介绍 206
11.2. TRACE 的使用 211
11.2.1. 在单核应用中使用 Trace *211*
11.2.2. 在 SMP 多核应用中使用 Trace *217*
11.2.3. 在 AMP 多核应用中使用 Trace *222*
11.2.4. 查看脱机 Trace *224*
12. RVPROF 功能的使用 225
12.1. 测试环境 225
12.1.1. 准备测试 NPK 软件或者工具包 *226*
12.1.2. 创建 rvprof 测试工程 *226*
12.2. 查看 RVPROF 的结果 230
13. 使用 NUCLEI NEAR CYCLE MODEL 仿真性能分析 235
13.1. 创建测试工程 236
13.2. 运行工程并生成性能分析结果 238
14. 使用 NUCLEI QEMU 仿真调试 242
15. 常见问题 245
15.1. NUCLEI STUDIO 启动慢 245
15.2. NUCLEI STUDIO 编译程序很慢 246

15.3. 找不到 New Nuclei Risc-V C/C++ Project 菜单 246
15.4. 打开/关闭 Launch Bar 247
15.5. 使用 Launch Bar 248
15.6. 不使用 Launch Bar 进行运行/调试 249
15.7. Debug 页面查看寄存器 250
15.8. Debug 页面结束进程 250
15.9. Nuclei Studio 工具栏中各按键功能 250
15.10. 显示其他窗口 252
15.11. 恢复默认窗口布局 252
15.12. 对比历史文件 253
15.13. 新建工程时可能出现报错 253
15.14. 新增 Include 路径出现缓存 253
15.15. 设置页面栏目找不到 254
15.16. 开发板下载速度很慢 254
15.17. Linux 环境下多用户使用 Nuclei Studio 254
15.18. 设备管理器中识别出两个串口 255
15.19. Linux 下使用时报 Could not determine GDB version after sending:riscv-nuclei-elf-gdb –version,response:的错误 256
15.20. 在 linux 下使用 QEMU 时报错 257
15.21. 工程编译链接 C 库找不到符号报错 257
15.22. 编译工程报错 fatal error: rvintrin.h: No such file or directory 261
15.23. Debug 时报错 Error: Couldn't find an available hardware trigger. 262
16. NUCLEI STUDIO 升级 263
16.1. GCC/OpenOCD 等工具链的安装 263
16.2. IDE Plugins 升级 264

17. 创建并贡献 NUCLEI STUDIO 组件包 267

18. 其他未注明或者遇到的版本问题 268

图片清单

图 2-1 WORKSPACE 弹出不兼容的警告 30
图 2-2 BUILD-TOOLS 更新到 4.4 版 30
图 2-3 GCC 和 CLANG 的目录结构 31
图 2-4 工程对 GCC 13 的支持 32
图 2-5 项目对 CLANG 17 的支持 33
图 2-6 创建工程时使用 RISC-V 扩展 34
图 2-7 项目中对 RISC-V 扩展的支持 35
图 2-8 NUCLEI SETTINGS 中对 RISC-V 扩展的支持 36
图 2-9 QEMU 中对 RISC-V 扩展的支持 36
图 2-10 创建工程时选择合适的工具链 37
图 2-11 组件包所适配的 NUCLEI STUDIO 版本号 38
图 2-12 QEMU 8.0 所在的目录 39
图 2-13 ELF 文件编辑器查看.ELF 文件 39
图 2-14 ELF 文件编辑器查看.O 文件 40
图 2-15 NUCELI SETTINGS 页面修改 41
图 2-16 SELECT C RUNTIME LIBRARY 在新版 IDE 中已不存在 42
图 2-17 SHARED 项目 NUCLEI SETTINGS(ARM) 42
图 2-18 SHARED 项目 NUCLEI SETTINGS(RISCV) 42
图 2-19 WORK.NUWORKSPACE 文件 43
图 3-1 NUCLEI STUDIO IDE 软件包的下载界面 45
图 3-2 NUCLEI STUDIO IDE 压缩包文件内容 46
图 3-3 双击“NUCLEI STUDIO.EXE”启动 NUCLEI STUDIO 47
图 3-4 设置 NUCLEI STUDIO 的 WORKSPACE 目录 47

图 3-5 第一次启动 Nuclei Studio 界面 48
图 4-1 进入 SDK Manage 49
图 4-2 导入资源包 50
图 4-3 资源包管理 50
图 4-4 NPK 安装成功 51
图 4-4 提示所需依赖存在并下载 52
图 4-4 提示所需依赖不存在 53
图 4-5 使用菜单栏创建项目 54
图 4-6 使用菜单栏创建项目 54
图 4-7 选择建立项目类型 55
图 4-8 选择程序模板及相关设置项 56
图 4-9 打开 SDK 设置工具 57
图 4-10 修改工程设置 58
图 4-11 清理工程并重新编译 58
图 4-12 Nuclei Settings 菜单 59
图 4-13 npk.yml 文件示例 60
图 4-14 Variables 查看 npk tools 对应参数 61
图 4-15 Variables 查看 npk tools 对应参数 62
图 5-1 从菜单创建项目 64
图 5-2 Project Explorer 视图创建项目 65
图 5-3 选择建立项目模型 66
图 5-4 新建项目及模板选择 68
图 5-5 完成模板创建项目 68
图 5-6 完成项目编译 69

图 5-7 开启调试配置面板 70
图 5-8 自动生成的 OpenOCD 配置 71
图 5-9 install.bat/install.sh 文件 72
图 5-10 用户授权 72
图 5-11 应用关联文件 test.nuproject 73
图 5-12 位于 github 上面的项目包 73
图 5-13 选择导入的项目 74
图 5-14 选择导入的方式 75
图 5-15 选择需要导入的项目 76
图 5-16 识别出要导入的项目 76
图 5-17 显示项目的文件结构 77
图 5-18 新建 C/C++ Project 78
图 5-19 设置 C Project 项目名和类型 79
图 5-20 设置 Hello World 项目的基本信息 80
图 5-21 设置项目的调试或者发布属性 81
图 5-22 创建完成 Hello World 项目的界面 82
图 5-23 打开新建文件夹 83
图 5-24 完成新建文件夹 83
图 5-25 打开工程设置选项页面 84
图 5-26 在弹出窗口进入 hello_world 项目的文件夹位置 85
图 5-27 添加 nuclei_sdk 文件夹 85
图 5-28 刷新工程 86
图 5-29 新建项目目录下的文件夹 87
图 5-30 配置 Target Processor 选项 89

图 5-31 配置 Optimization 选项 90
图 5-32 配置 Debugging 选项 91
图 5-33 配置链接脚本 92
图 5-34 配置链接的 General 选项 93
图 5-35 配置 Libraaies 选项 94
图 5-36 配置链接的 Miscellaneous 选项 95
图 5-37 设置 GNU RISC-V Cross C Assembler 的 Includes 栏目包含路径 97
图 5-38 设置 GNU RISC-V Cross C Compiler 的 Includes 栏目包含路径 98
图 5-39 新建基于 Makefile 的工程 99
图 5-40 设置工程选项 100
图 5-41 配置 Makefile 路径 100
图 5-42 配置 Build 选项 101
图 5-43 配置工具链路径 102
图 6-1 NulceiStudio 支持多款编译器 103
图 6-2 创建一个支持 GCC 13 编译的工程 105
图 6-3 配置工具链路径 106
图 6-4 创建一个支持 Clang 17 编译的工程 107
图 6-5 配置工具链路径 108
图 6-6 创建一个支持 zcc 编译的工程 109
图 6-7 配置工具链路径 110
图 6-8 打开修改编译选项弹窗 1 112
图 6-9 打开修改编译选项弹窗 2 113
图 6-10 打开修改编译选项弹窗 3 114
图 6-11 一键修改编译选项 114

图 6-12 清理工程并重新编译工程 115
图 6-13 对 HELLO_WORLD 项目单击右键选择“CLEAN PROJECT” 116
图 6-14 单击锤子图标对 HELLO_WORLD 项目进行编译 117
图 6-15 查看编译后的生成文件 118
图 7-1 LAUNCH BAR 118
图 7-2 点击*.LAUNCH 文件切换调试模式 119
图 7-3 RUN AS/DEBUG AS 切换调试模式 120
图 7-4 LAUNCH BAR 切换调试模式 120
图 7-5 在设备管理器中查询 COM 串口号 122
图 7-6 开发板连接至虚拟机中 124
图 7-7 查看 GCC 的依赖是否完整 124
图 7-8 LAUNCH 文件 125
图 7-9 RUN AS/DEBUG AS 126
图 7-10 在 LAUNCH BAR 中查看 DEBUG CONFIGURATIONS 126
图 7-11 DEBUG CONFIGURATIONS 127
图 7-12 单击“RUN CONFIGURATION”进行下载 128
图 7-13 添加新的 GDB OPENOCD DEBUGGING 129
图 7-14 GDB OPENOCD DEBUGGING 栏目下的 HELLO_WORLD 项目 DEBUG 129
图 7-15 配置 HELLO_WORLD_DEMO DEBUG 的参数 1 130
图 7-16 配置 HELLO_WORLD_DEMO DEBUG 的参数 2 131
图 7-17 通过 NUCLEI STUDIO 打开串口调试助手并配置参数 132
图 7-18 设置串口参数 132
图 7-19 打开 TERMINAL 终端 133
图 7-20 打开串口 133

图 7-21 单击“DEBUG CONFIGURATION”进行下载 134
图 7-22 切换至 DEBUG 模式 135
图 7-23 下载完成后进入调试界面 136
图 7-24 下载程序到开发板中运行 137
图 7-25 串口调试助手界面输出的 HELLO WORLD 137
图 7-26 下载完成单击红色按钮断开连接 138
图 7-27 打开工程所在目录 139
图 7-28 新建“SEGGER”文件夹 140
图 7-29 压缩包内文件内容 140
图 7-30 复制后 SEGGER 文件夹内容 141
图 7-31 注释不使用的头文件 141
图 7-32 添加 SEGGER 文件夹路径至 INCLUDE 中 142
图 7-33 打开 EXCLUDE FROM BUILD 143
图 7-34 移除 WRITE.C 文件 143
图 7-35 LAUNCH 文件 144
图 7-36 RUN AS/DEBUG AS 145
图 7-37 在 LAUNCH BAR 中查看 DEBUG CONFIGURATIONS 146
图 7-38 DEBUG CONFIGURATIONS 147
图 7-39 新建 J-LINK DEBUG CONFIGURATION 148
图 7-40 确保 PROJECT 和 C/C++ APPLICATION 与工程一致 149
图 7-41 设置 DEBUGGER 栏目内容 150
图 7-42 设置 STARTUP 栏目内容 151
图 7-43 J-LINK 引脚图 152
图 7-44 J-LINK 实物连接图 153

图 7-45 设置 RTT VIEWER 154
图 7-46 切换至下载模式 155
图 7-47 下载完成后进入调试页面 156
图 7-48 下载程序到开发板中运行 157
图 7-49 打印输出 HELLO WORLD 158
图 7-50 下载完成单击红色按钮断开连接 158
图 7-51 DLINK 实物图 158
图 7-52 RUN CONFIGURATIONS 的配置页面 159
图 7-53 下载 GD 32 DFU DRIVERS 159
图 7-54 CUSTOM DEBUGGING 配置页面 161
图 7-55 CUSTOM DEBUGGING 配置页面 162
图 7-56 DLINK 处理工作状态 163
图 7-57 DLINK_GDBSERVER.CFG 内容样例 163
图 7-58 DLINK 处理 DEBUG 中 164
图 7-59 联接串口输出 165
图 7-60 DLINK 调试时串口输出 165
图 8-1 工程导入后编译报错 167
图 8-2 手动导入 GCC 10 168
图 8-3 CONVERT GCC 10 PROJECT TO GCC 13 168
图 8-4 CONVERT PROJECT TO GCC 13 OK 169
图 8-5 打开 CONVERT ALL 170
图 8-6 选择需要转换的工程 171
图 8-7 转换结果 172
图 8-8 选择 EXPORT 菜单 173

图 8-9 EXPORT 工程 174
图 8-10 指定导出的目录及相应文件 175
图 8-11 查看导出的工程 176
图 8- 12 修改 NUCLEI STUDIO 工具链 177
图 9- 1 打开 SHOW VIEW 178
图 9- 2 打开 LST VIEW 179
图 9- 3 LST VIEW 上的菜单 179
图 9- 4 LST VIEW 打开 LST 文件 180
图 9- 5 LST VIEW 中 LST 与源码联动 180
图 10- 1 RADIAL VIEW 183
图 10- 2 TREE VIEW 184
图 10- 3 LEVEL VIEW 185
图 10- 4 AGGREGATE VIEW 186
图 10- 5 创建测试工程 187
图 10- 6 设置-PG 188
图 10- 7 编译测试工程 188
图 10- 8 运行工程并查看 PROFILING 信息 189
图 10- 9 SELECT ALL 并使用 PARSE AND GENERATE HEXDUMP 190
图 10- 10 解析 PROFILING 信息并生成对应文件 191
图 10- 11 查看生成的对应文件 192
图 10- 12 打开.GCDA 文件 193
图 10- 13 打开 GCOV 工具 194
图 10- 14 在 NUCLEI STUDIO 中查看 CODE COVERAGE 194
图 10- 15 在 NUCLEI STUDIO 中查看 CODE COVERAGE 直方图 195

图 10- 16 在 Nuclei Studio 中查看 profiling 信息 196
图 10- 17 在 Nuclei Studio 中查看 profiling 信息 197
图 10- 18 在 Nuclei Studio 中查看 profiling 信息直方图 197
图 10- 19 在 Nuclei Studio 中查看 Call Graph 198
图 10- 20 创建一个启用 Semihosting 的工程 199
图 10- 21 设置-pg --coverage 属性 200
图 9- 5 修改参数以通过 Semihost 写入文件 200
图 10- 23 编译测试工程 201
图 10- 24 运行工程 202
图 10- 25 查看生成的文件 203
图 10- 26 在 Nuclei Studio 中查看 gcov 204
图 10- 27 在 Nuclei Studio 中查看 gprof 205
图 10- 28 在 Nuclei Studio 中查看 Call Graph 206
图 10- 1 在 Nuclei Studio 中 Trace 的视图 207
图 10- 2 Trace setting 的视图 209
图 10- 2 Set Current Debug hart Configuration 210
图 10- 3 N900 的单核应用 helloworld 212
图 10- 4 helloworld 工程进入 Debug 模式 213
图 10- 5 设置 Trace Configuration 214
图 10- 6 dump 到的 trace 文件 215
图 10- 7 Set current debug hart configuration 216
图 10- 8 trace 与源码、反汇编码联动 216
图 10- 9 以文本方式查看 trace 217
图 10- 10 SMP 多核应用中设置 Trace Configuration 218

图 10- 11 SMP 多核应用中对 THREAD #1 START TRACE 219
图 10- 12 SMP 多核应用中对 THREAD #2 START TRACE 220
图 10- 13 SMP 多核应用中查看 HARDID=0 的 TRACE 221
图 10- 14 多核应用中查看 HARDID=1 的 TRACE 222
图 10- 15 AMP 多核应用中 CORE 0 设置 TRACE 配置 223
图 10- 16 AMP 多核应用中 CORE 1 设置 TRACE 配置 224
图 10- 2 SET CURRENT DEBUG HART CONFIGURATION 225
图 11- 1 RVPROF 测试环境 226
图 11- 2 安装 RVPROF 测试包及依赖包 227
图 11- 3 选择最新的 NUCLEI_SDK 228
图 11- 4 创建 RVPROF 测试工程 229
图 11- 5 RVPROF 测试工程的 LAUNCH 文件 230
图 11- 6 CYCLE MODEL 启动及 LOG 输出 231
图 11- 7 PERFETTO 启动本地服务 232
图 11- 8 打开 PERFETTO TRACE VIEWER 233
图 11- 9 手动载入 JSON 文件 234
图 11- 10 RVPROF TRACE 结果展示 235
图 13- 1 创建任意测试工程 236
图 13- 2 打开 RUN CONFIGURATIONS 237
图 13- 3 配置 RUN CONFIGURATION 237
图 13- 4 配置 RUN CONFIGURATION 238
图 13- 5 运行程序 239
图 13- 6 程序在 NUCLEI NEAR CYCLE MODEL 中运行成功执行 239
图 13- 7 查看生成的文件 240

图 13- 8 通过 PERFETTO 查看*.RVTRACE 240
图 13- 9 通过 GPROF 查看生成的*.GPROF 文件 241
图 13- 10 查看 CALL GRAPH 文件 241
图 12- 1 选择 QEMU 进行仿真调试 243
图 12- 2 QEMU 支持的 CPU 型号 244
图 12- 3 QEMU 仿真调试界面 245
图 13-1 NEW NUCLEI RISC-V C/C++ PROJECT 菜单 246
图 13-2 视图重置 247
图 13-3 切换当前视图为 C/C++视图 247
图 13- 4 关闭 LAUNCH BAR 248
图 13- 5 LAUNCH BAR 功能介绍 248
图 13- 6 不使用 LAUNCH BAR 首次进行调试 250
图 13- 7 DEBUG 模式查看功能窗口 252
图 13- 8 对比历史文件 253
图 13- 9 显示隐藏栏目 254
图 13- 10 设备管理器中显示两个串口 256
图 13-11 报错提示 256
图 12-12 查看 GDB 缺失的依赖 256
图 13- 13 因为 LIBRARIES 内部依赖而导致的编译报错 258
图 13- 14 整 LIBRARIES 的顺序或者添加多个链接 259
图 13- 15 修改 COMMAND LINE PATTERN 内容 260
图 13- 16 修改 COMMAND LINE PAATERN 内容 261
图 13- 17 FATAL ERROR: RVINTRIN.H: NO SUCH FILE OR DIRECTORY 262
图 13- 18 COULDN'T FIND AN AVAILABLE HARDWARE TRIGGER. 263

图 14- 1 IDE 工具链路径 264
图 14- 2 打开安装软件弹窗 265
图 14- 3 打开已安装界面 266
图 14- 4 选择“NUCLEI STUDIO”更新 267

1. Nuclei Studio IDE 简介

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公司处理器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 已经充分与 Nuclei SDK 整合，完全满足 SDK 的需要，可以方便快捷地新建模板工程，快速修改工程设置选项。内置 Nuclei SDK 源代码，可根据模板需求自动添加。

2. Nuclei Studio 更新说明

2.1.2024.06 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。

2.1.1.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。

2.1.2.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 。

2.1.3.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。

2.1.4.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 4.1.1 章节内容。

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。

2.1.5.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 15.21 章节。

2.1.6.优化调式模式切换

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，优化了调式模式的切换，具体内容参见第 7.1 章节内容。

2.1.7.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。具体内容可以查看第 7.4 章节内容。

2.1.8.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 6.1.3 章节内容。

2.1.9.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，具体请参见第 9 章节内容。

2.1.10.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。具体请参见第 10 章节内容。

2.1.11.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。具体请参见第 10 章节内容。

2.1.12.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。具体请参见第 10 章节内容。

2.1.13.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，具体请参见第 13 章节内容。

2.2.2024.02.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。

2.2.1.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。

2.2.2.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。

2.2.3.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。具体内容参见第 8.1.3 章节内容。

2.2.4.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，具体内容参见第 10 章节内容；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。

2.2.5.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 11 章节内容。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。

2.2.6.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。

2.3.2023.10 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。

2.3.1.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

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。

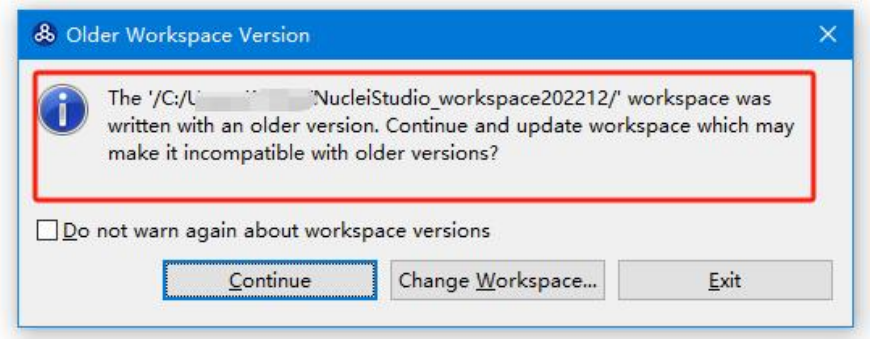


图 2- 1 workspace 弹出不兼容的警告

2.3.2.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。

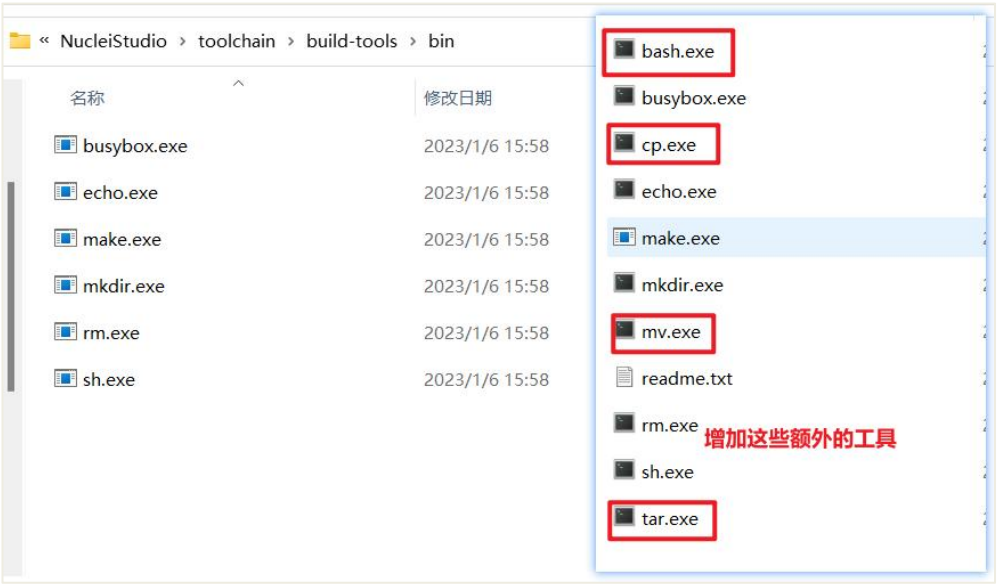


图 2- 2build-tools 更新到 4.4 版

2.3.3.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，具体使用可以参考章节 8 内容。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

图 2- 3 GCC 和 Clang 的目录结构

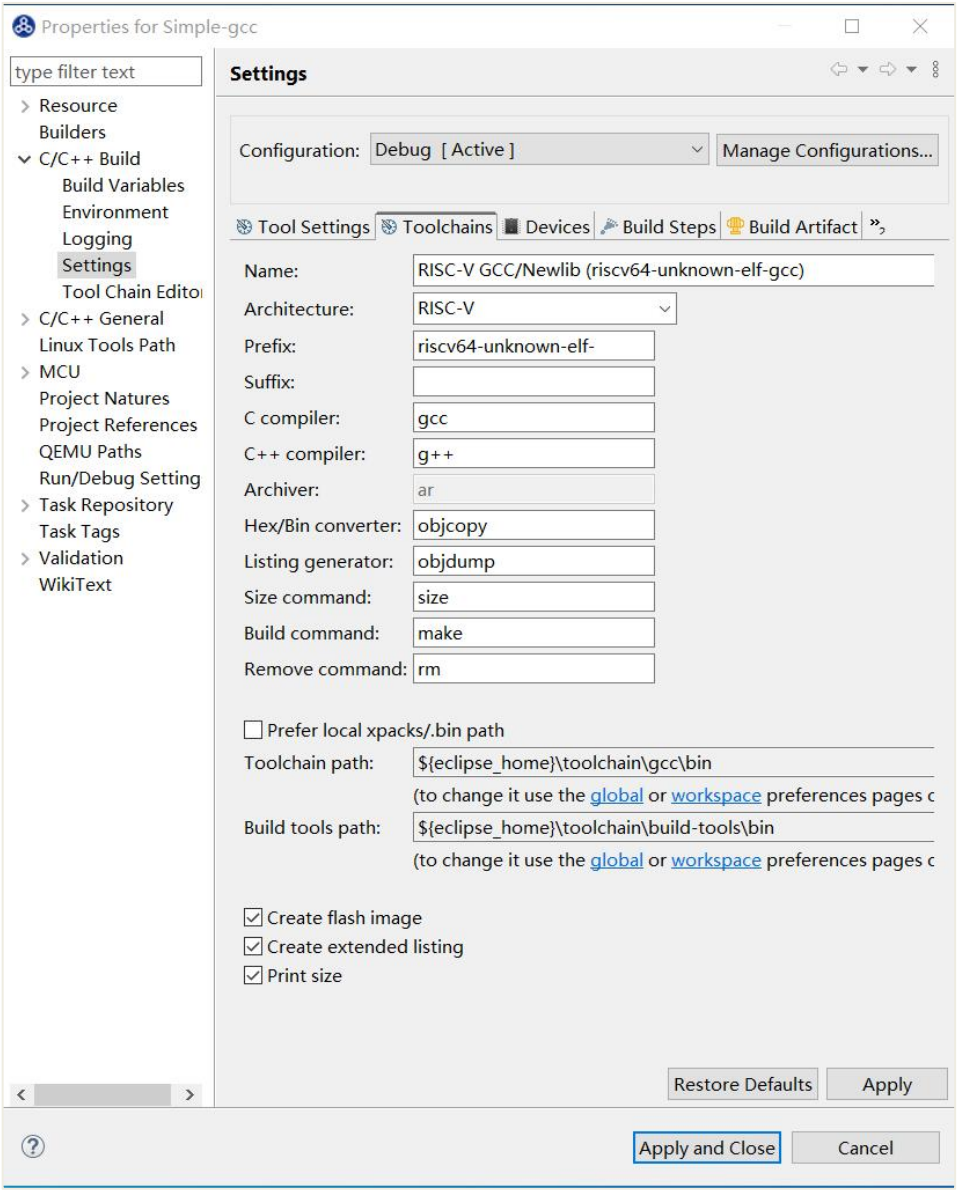


图 2-4 工程对 GCC 13 的支持

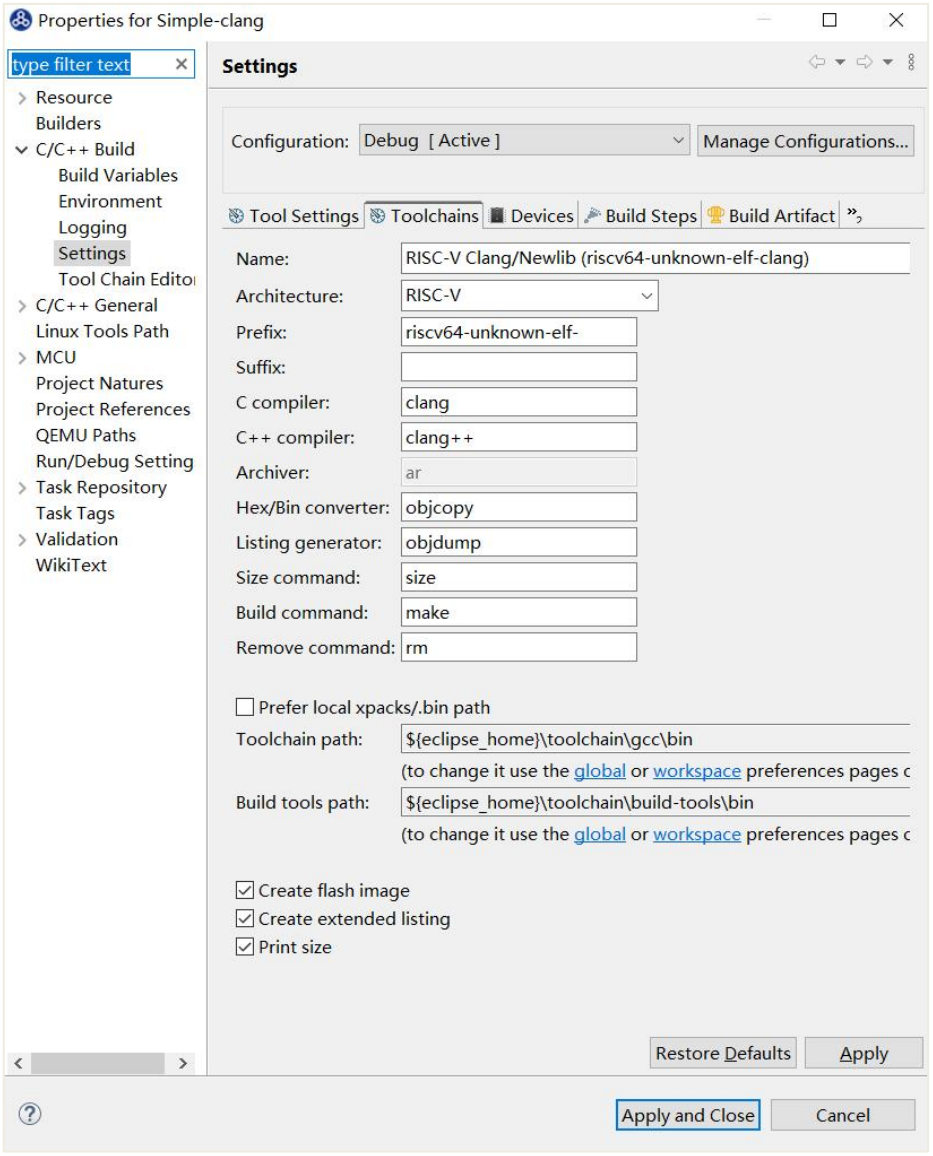


图 2- 5 项目对 Clang 17 的支持

2.3.4.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:

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** 和 **C/C++ Build->Settings->Target Processor** 中均有体现，同时在 QEMU 的配置中也会有相对应的设置。

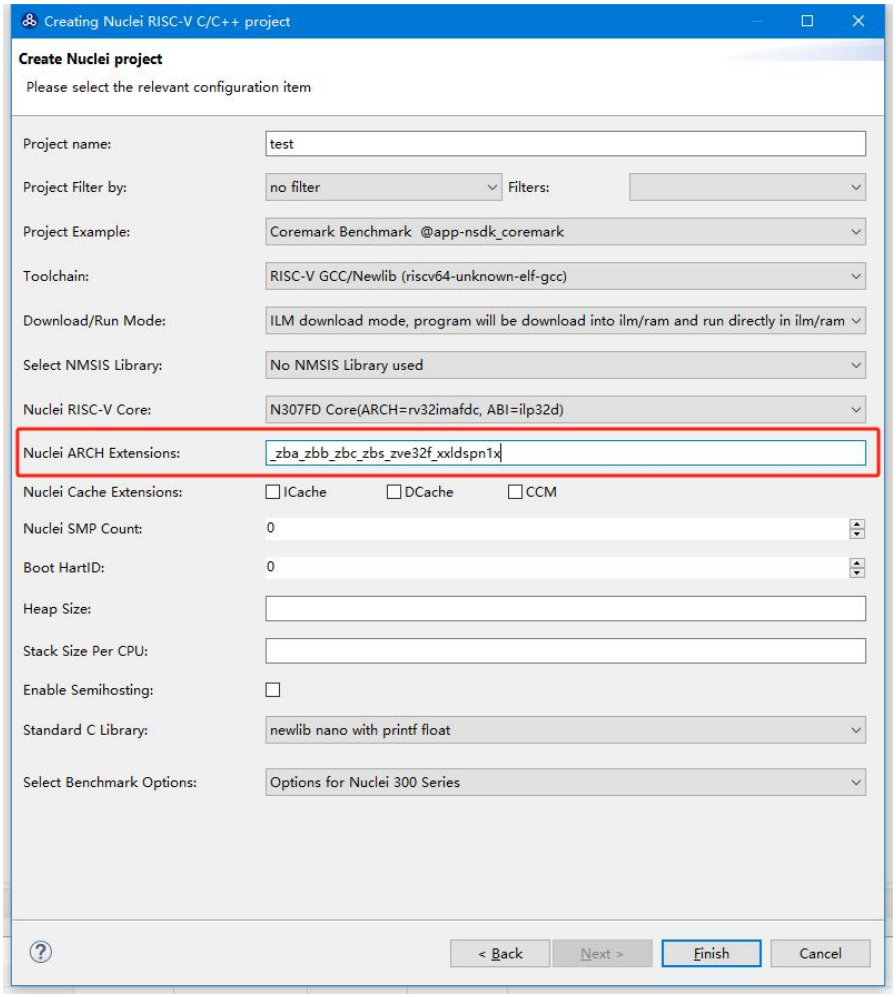


图 2-6 创建工程时使用 RISC-V 扩展

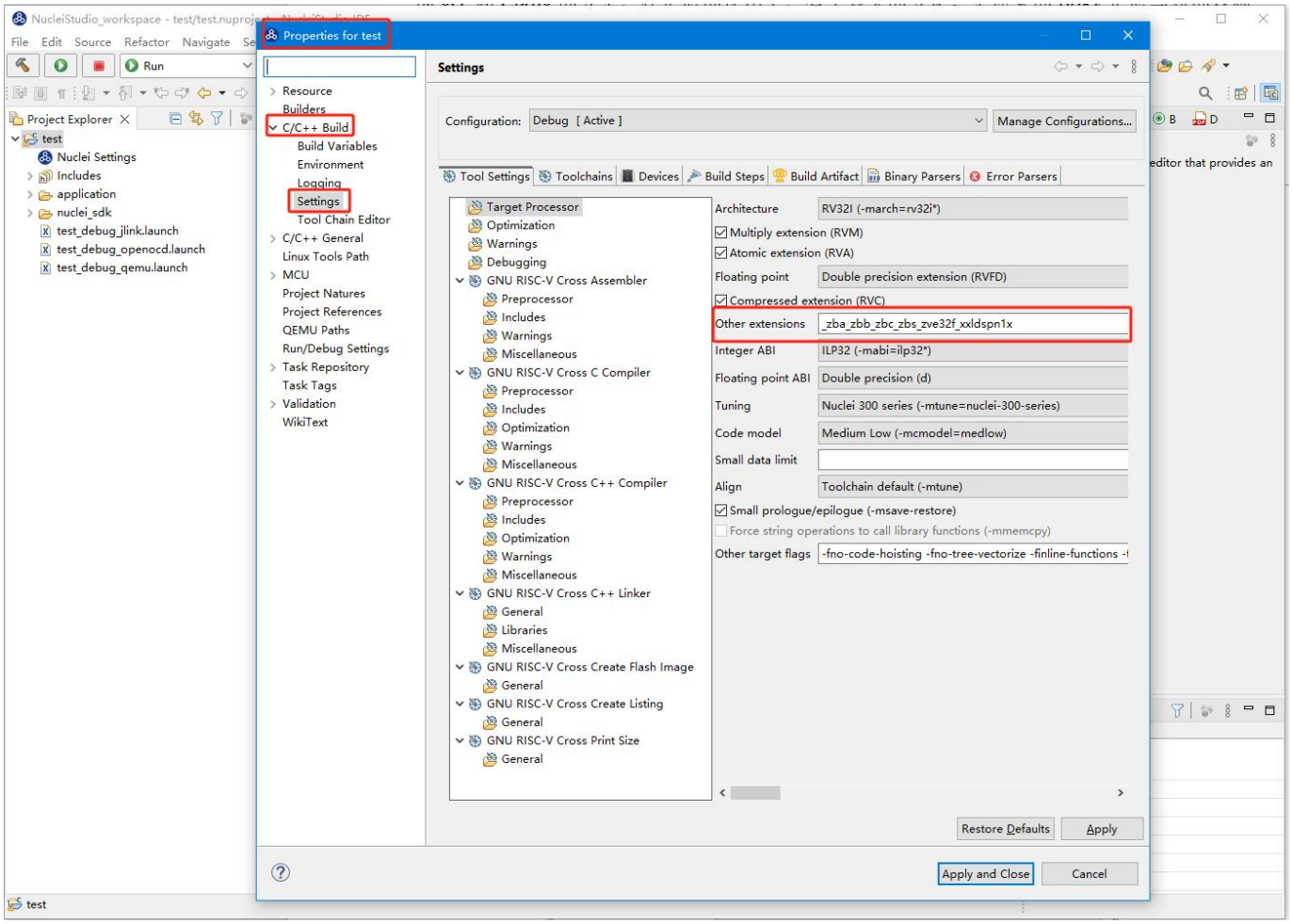


图 2-7 项目中对 RISC-V 扩展的支持

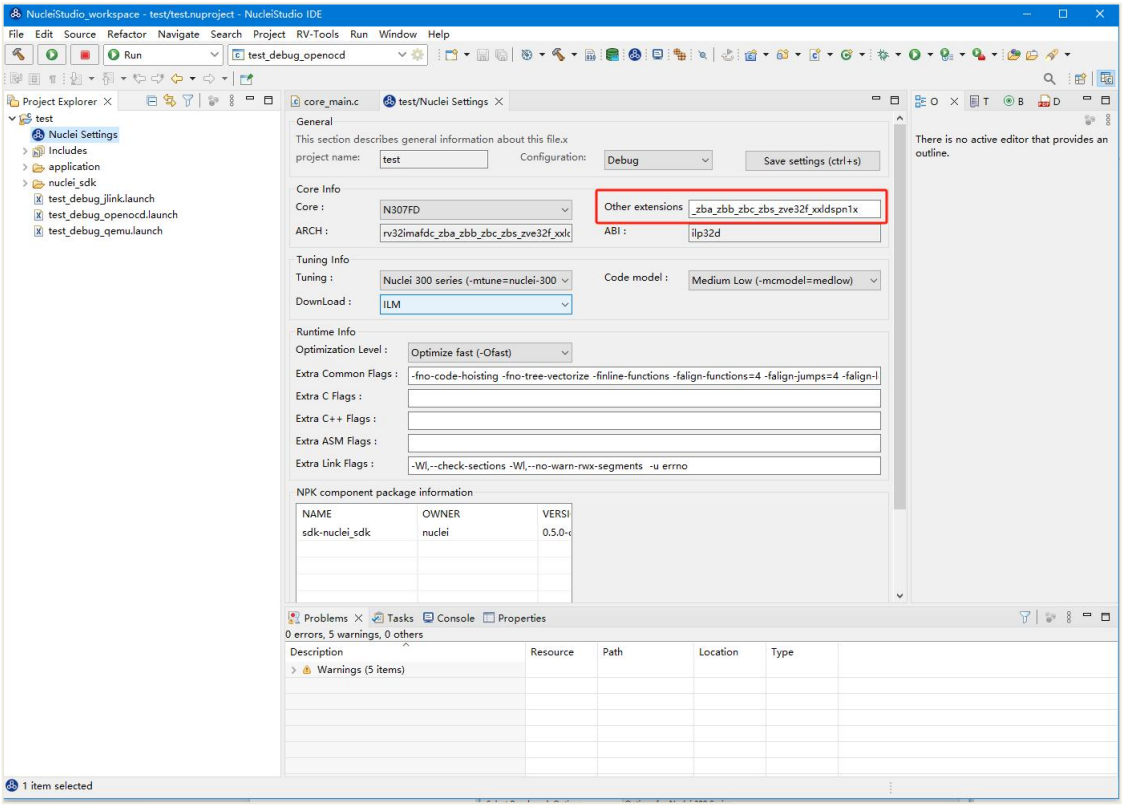


图 2- 8 Nuclei Settings 中对 RISC-V 扩展的支持

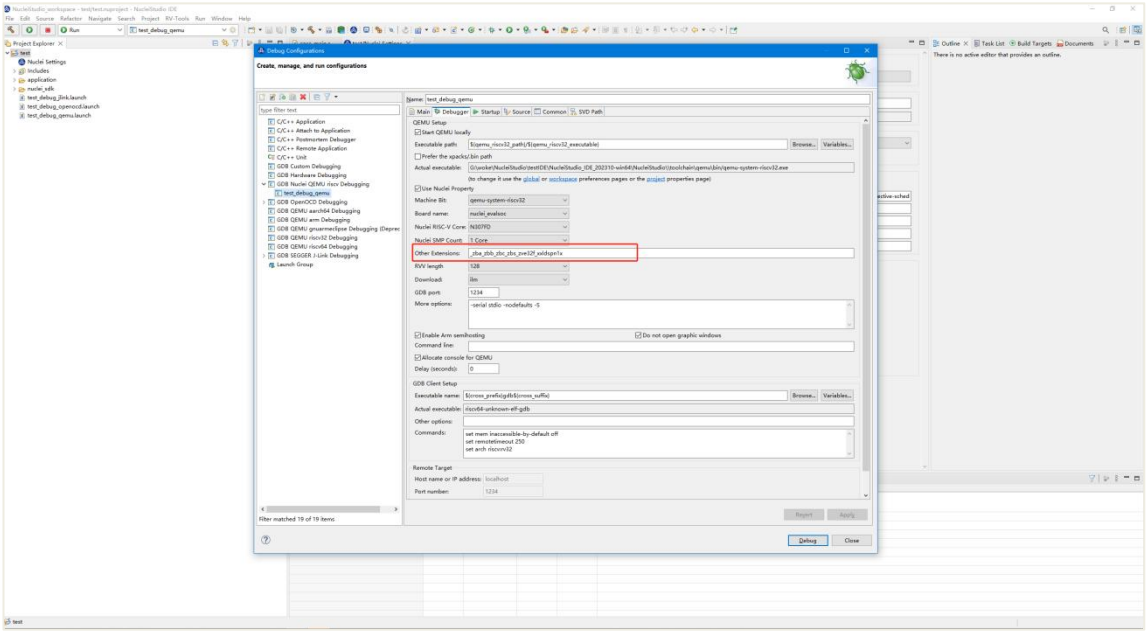


图 2- 9 QEMU 中对 RISC-V 扩展的支持

2.3.5.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**，具体可能参考章节 8.1 内容。

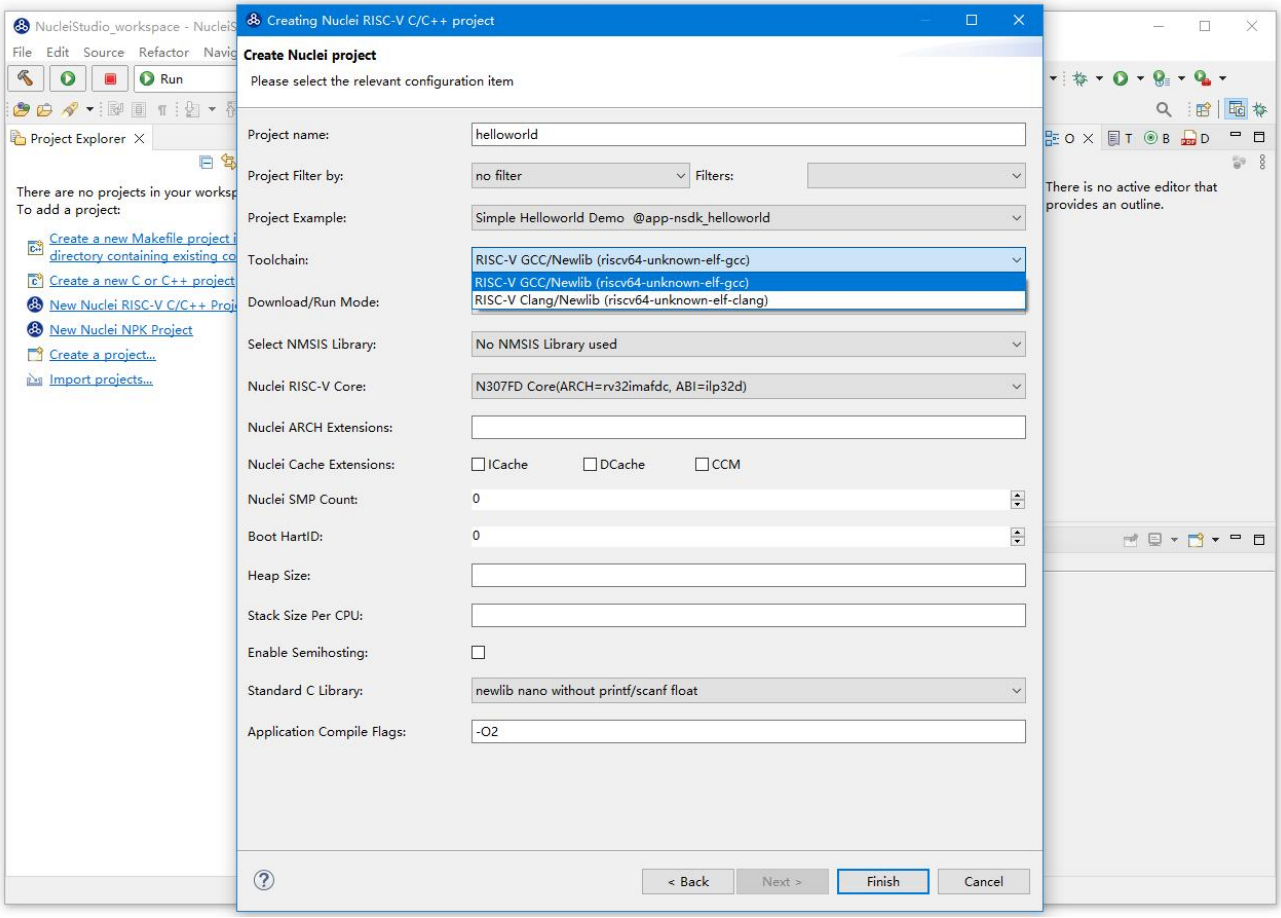


图 2- 10 创建工程时选择合适的工具链

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。

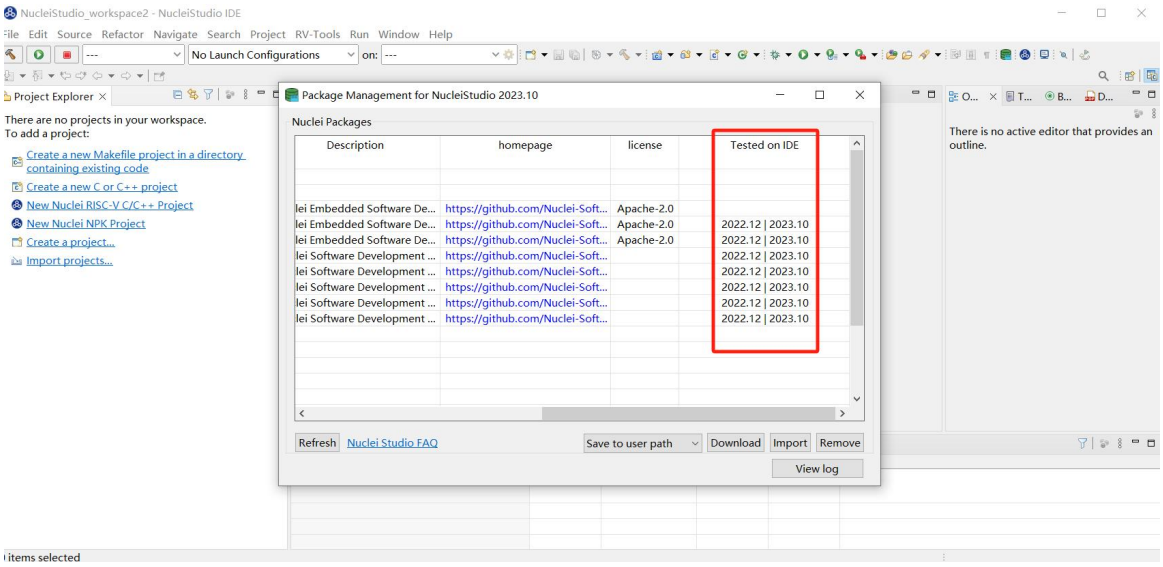


图 2-11 组件包所适配的 Nuclei Studio 版本号

2.3.6.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

2.3.7.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

图 2- 12 QEMU 8.0 所在的目录

2.3.8.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。

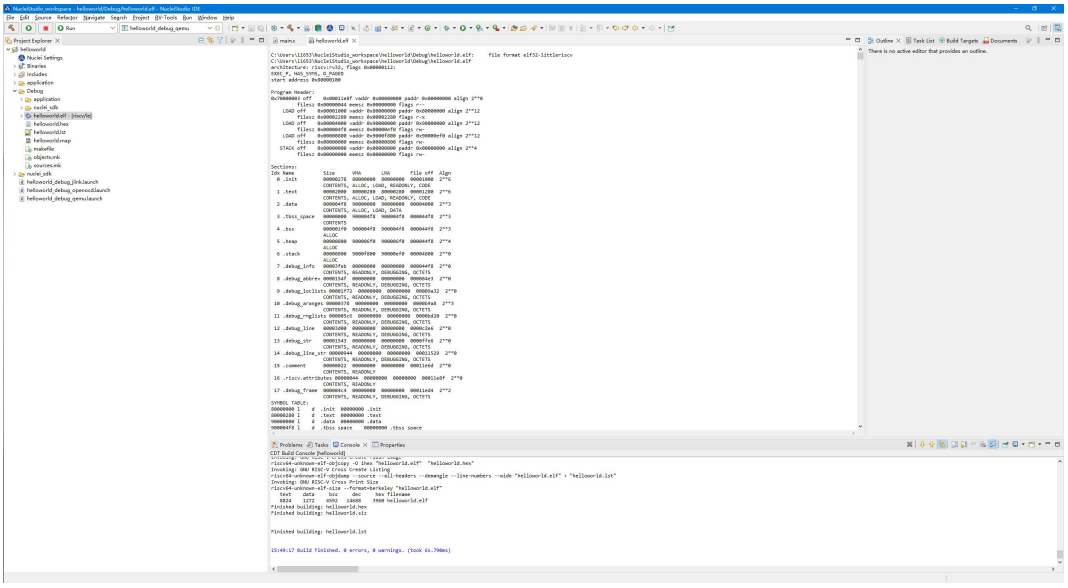


图 2- 13 elf 文件编辑器查看.elf 文件

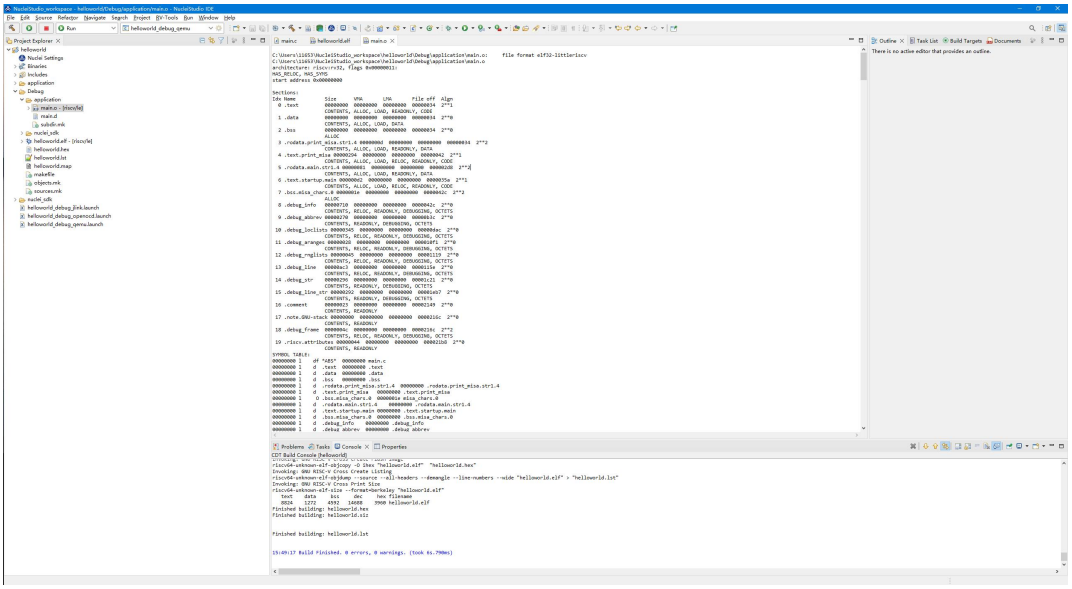


图 2- 14 elf 文件编辑器查看.o 文件

2.3.9.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，具体参考第 9 章节内容。

2.3.10.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。

2.3.11.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，可以参考章节 2.1.4 内容。

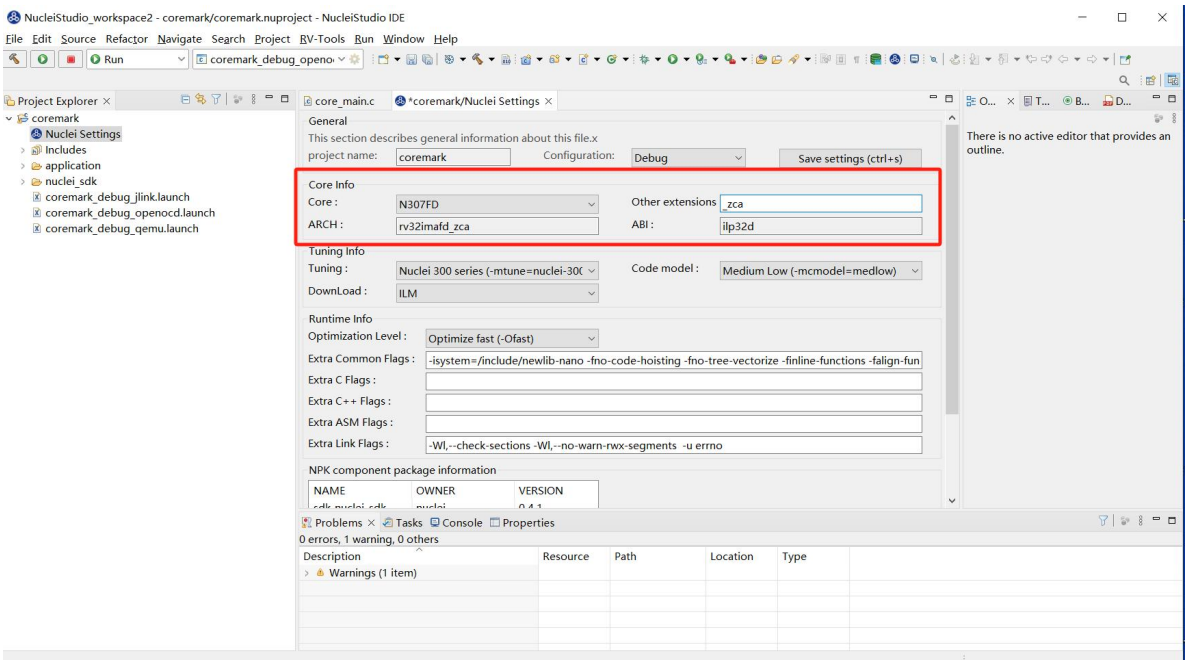


图 2-15 Nuceli Settings 页面修改

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。

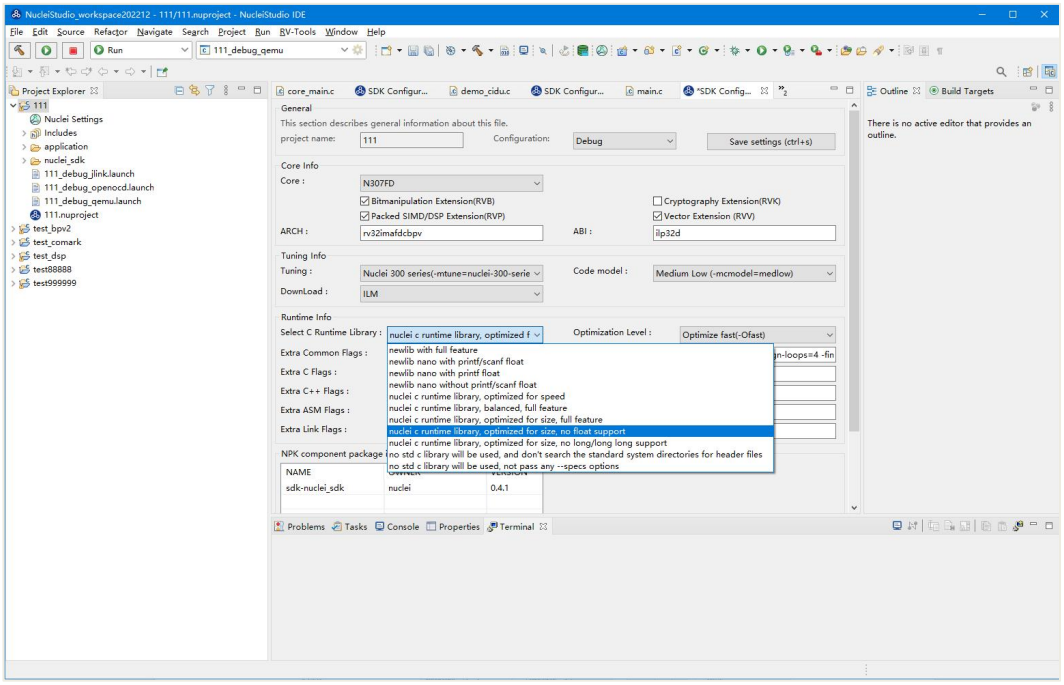


图 2-16 Select C Runtime Library 在新版 IDE 中已不存在

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。

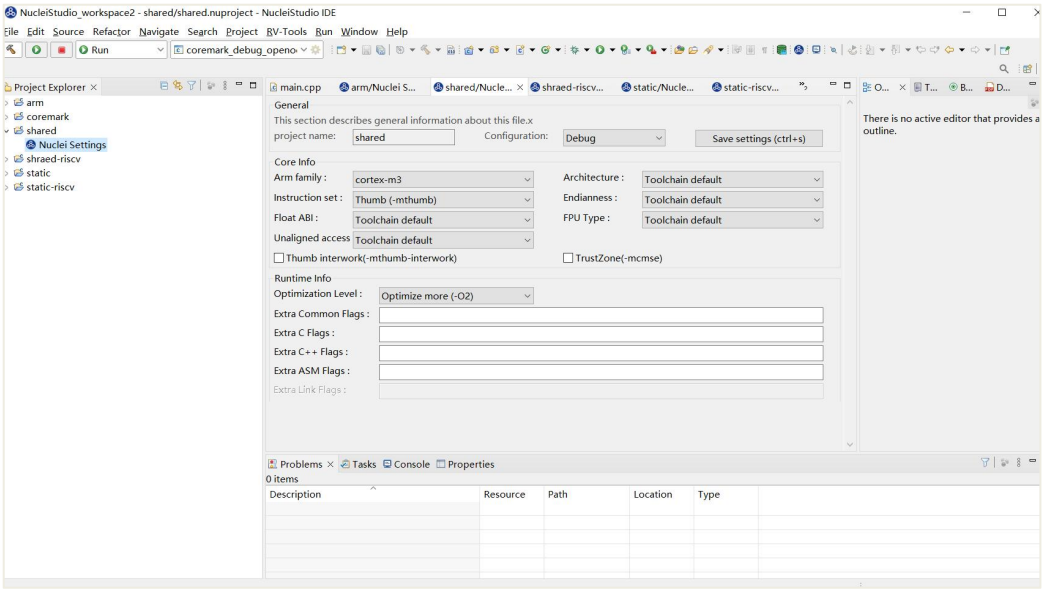


图 2-17 Shared 项目 Nuclei Settings(Arm)

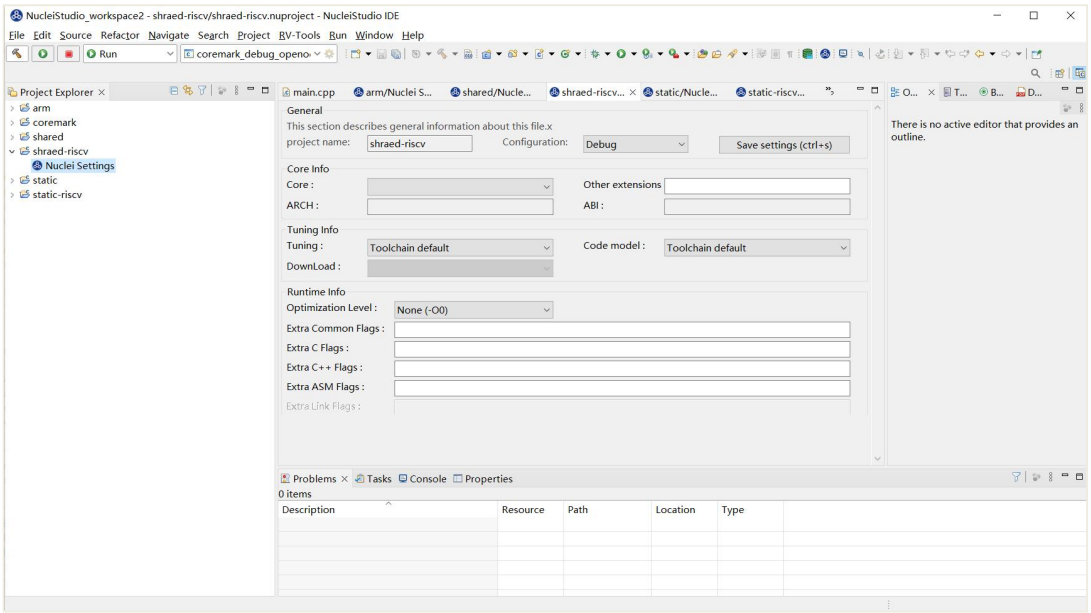


图 2-18 Shared 项目 Nuclei Settings(Riscv)

2.3.12.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。

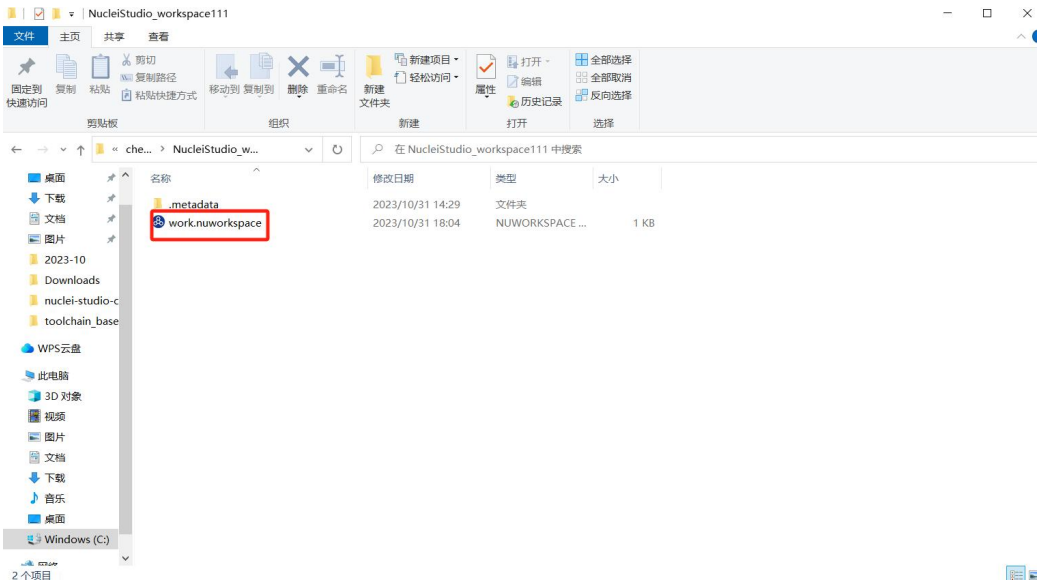


图 2-19 work.nuworkspace 文件

2.4.2022.12 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。

3. Nuclei Studio 下载与安装

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，如图 3-1 所示。

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。

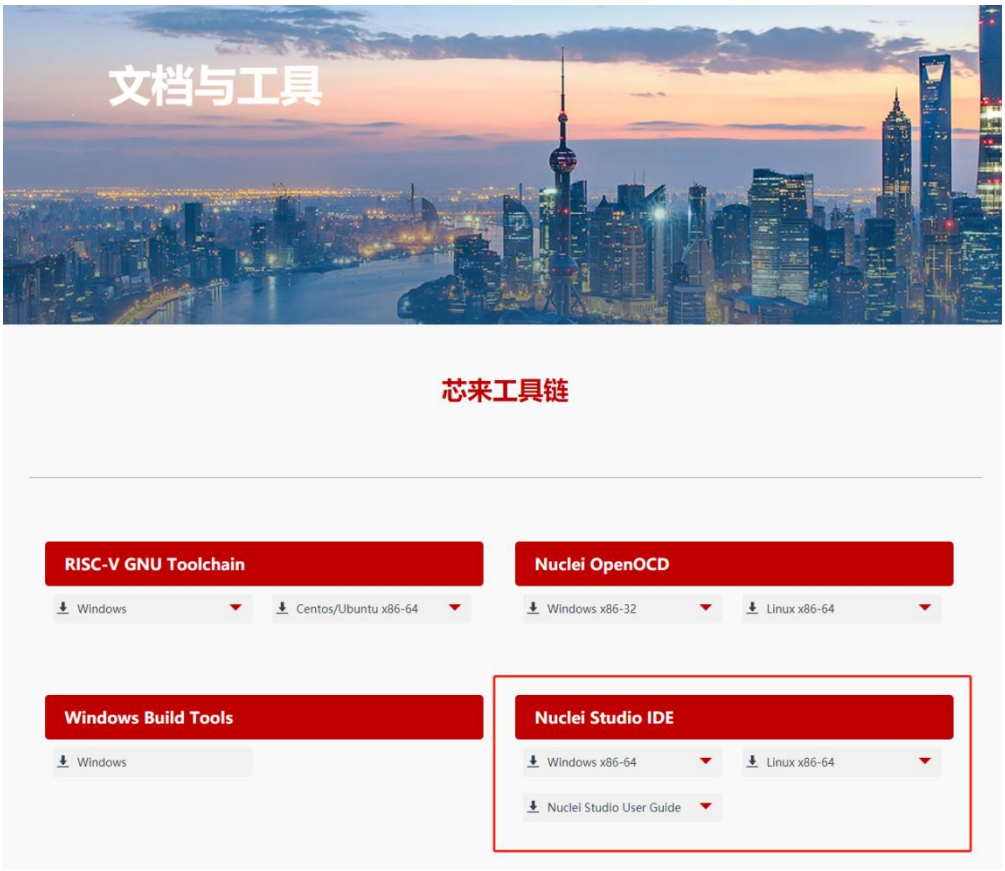


图 3- 1 Nuclei Studio IDE 软件包的下载界面

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如图 3- 2（解压路径中不可包含中文），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。

图 3- 2Nuclei Studio IDE 压缩包文件内容

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，如图 3-3 所示。

■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，如图 3-4 所示。

■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 界面如图 3-5 所示。

■**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。**

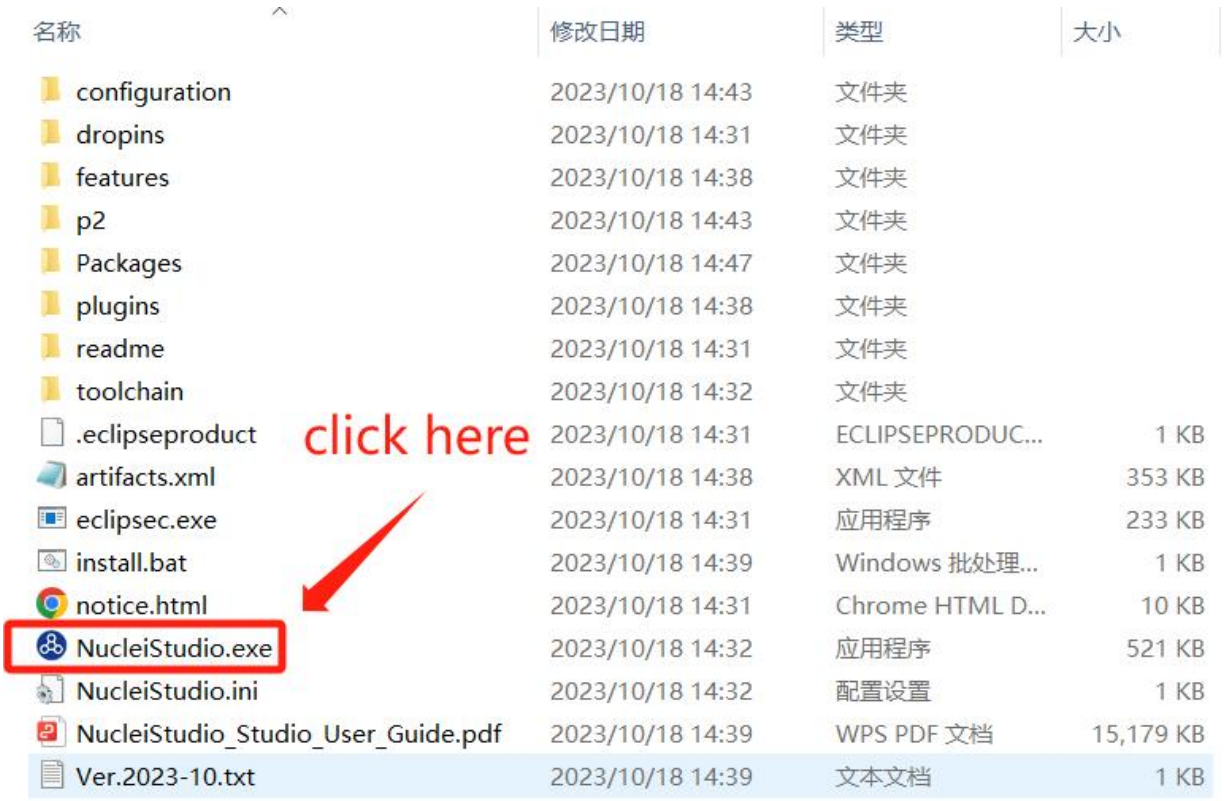


图 3- 3 双击“Nuclei Studio.exe”启动 Nuclei Studio

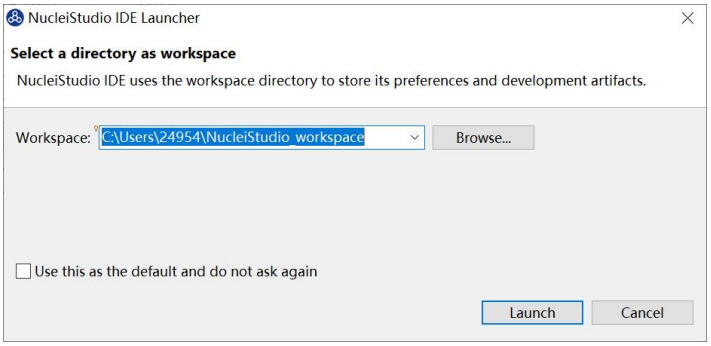


图 3- 4 设置 Nuclei Studio 的 Workspace 目录

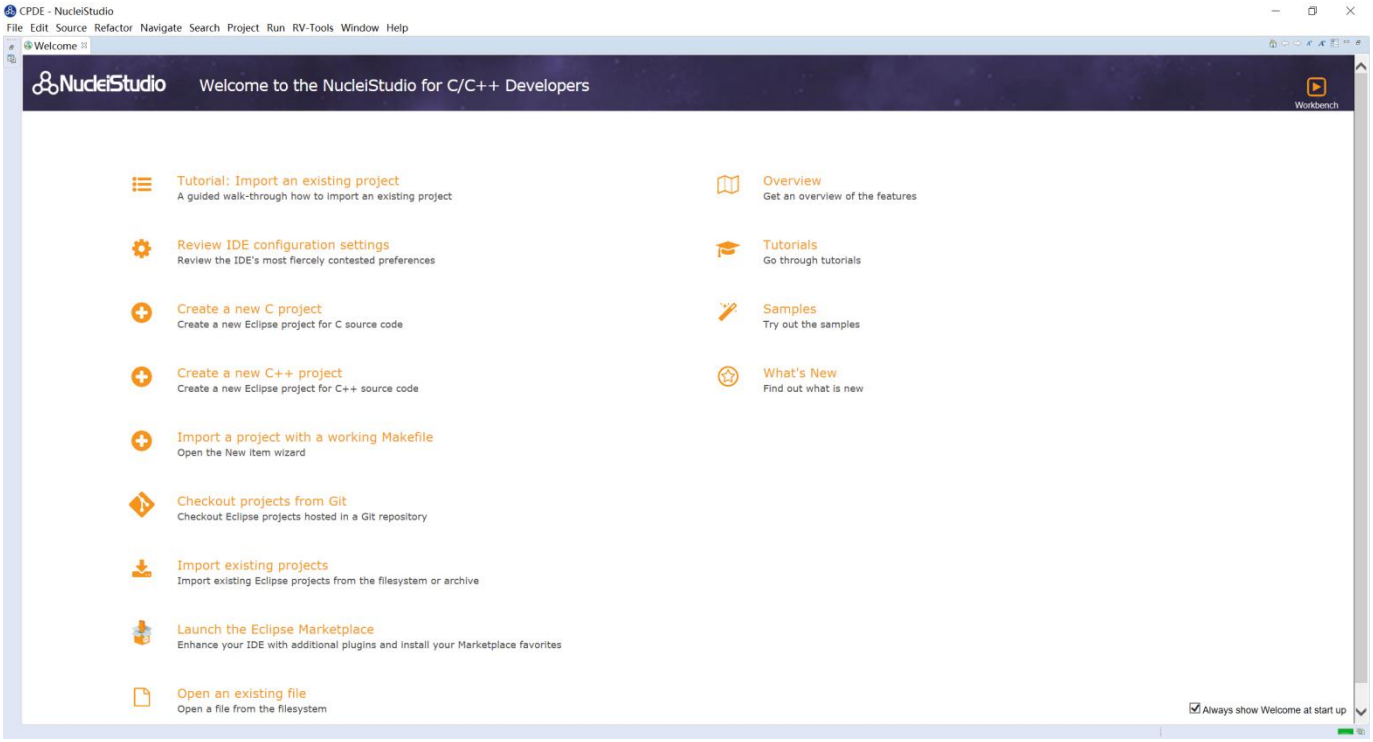


图 3-5 第一次启动 Nuclei Studio 界面

4.Nuclei Studio NPK 功能

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。

4.1.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。

4.1.1.NPK 软件包管理

■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 管理如图 4-3。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 0.3.7 版本，最新版本为 0.5.0，请使用最新版本进行安装，最新版本支持 gcc13 工具链，之前版本支持的是 gcc10，安装完成后 npk 状态变为 Installed。

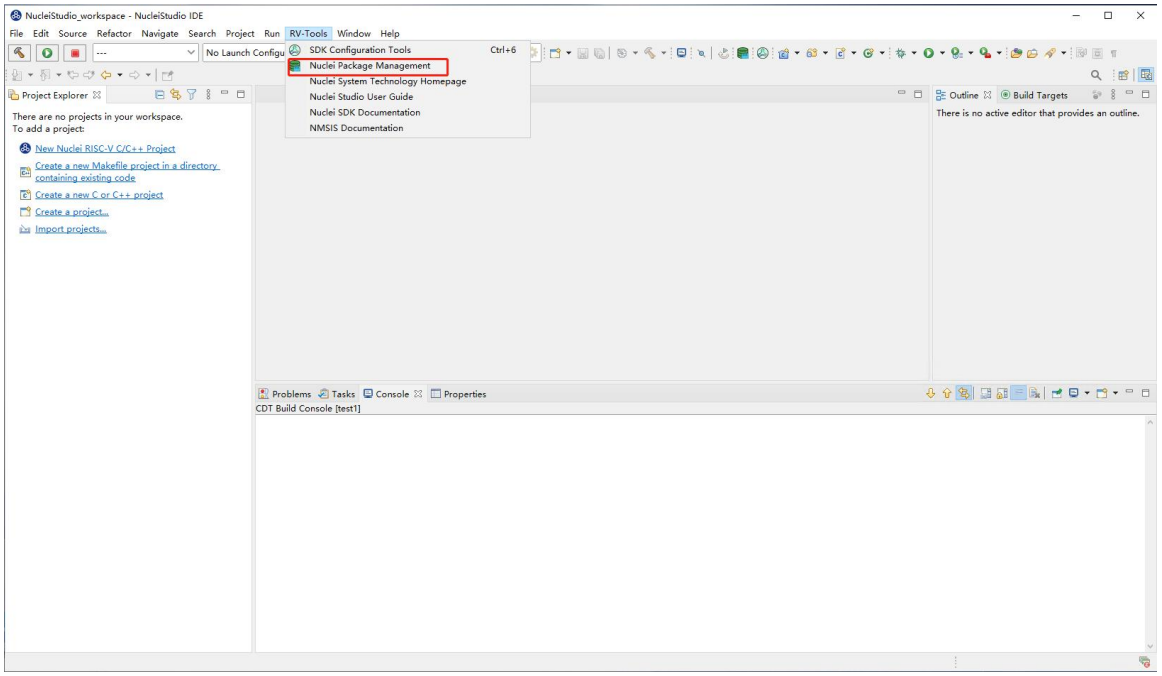


图 4-1 进入 SDK Manage

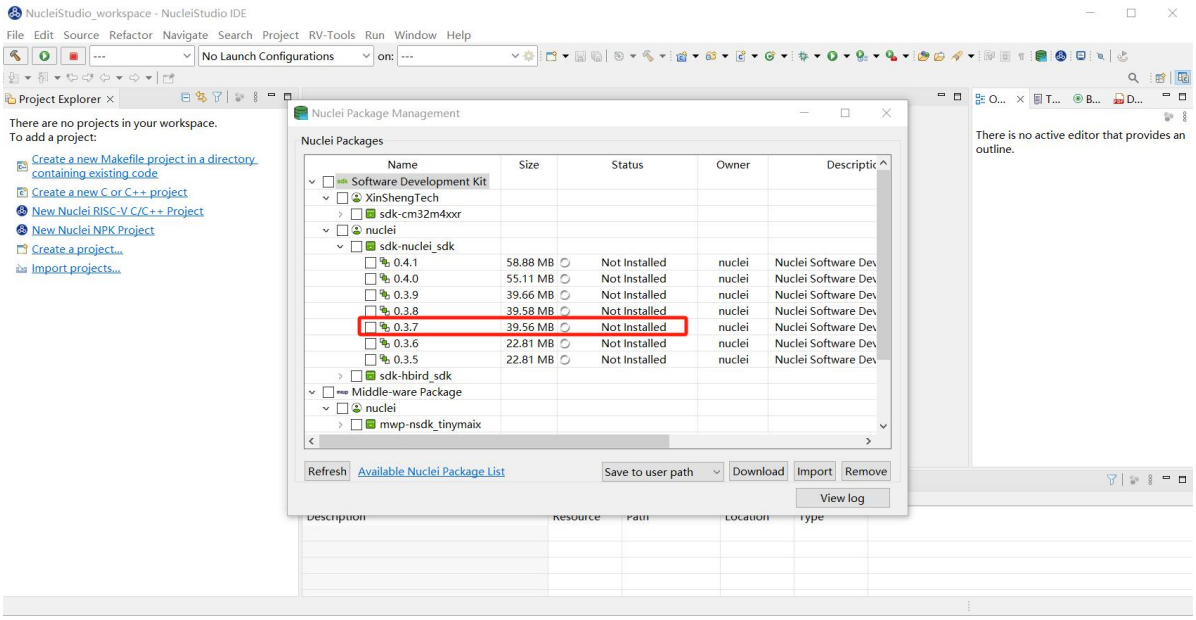


图 4-2 导入资源包

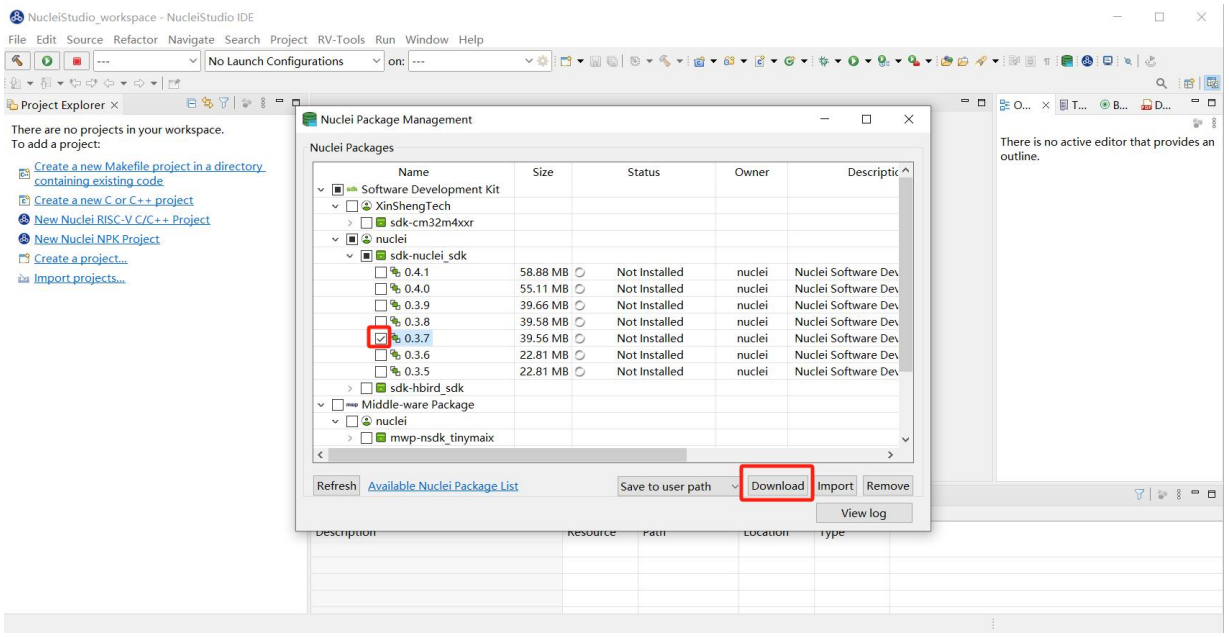


图 4-3 资源包管理

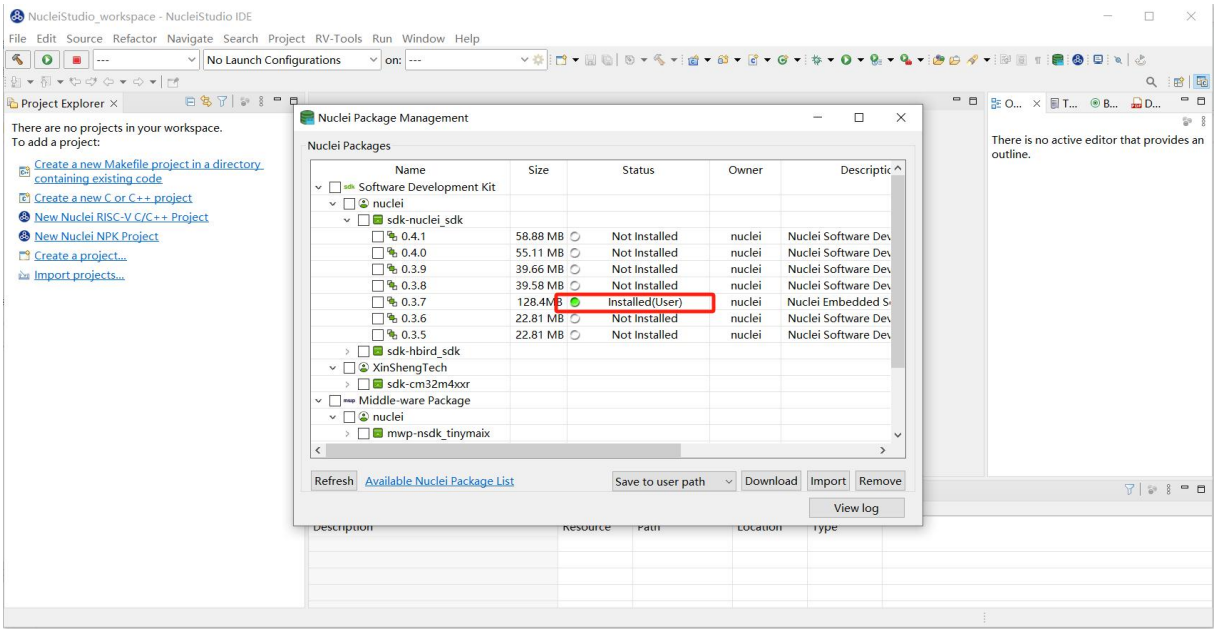


图 4-4 NPK 安装成功

在 NucleiStudio 2024.06 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。

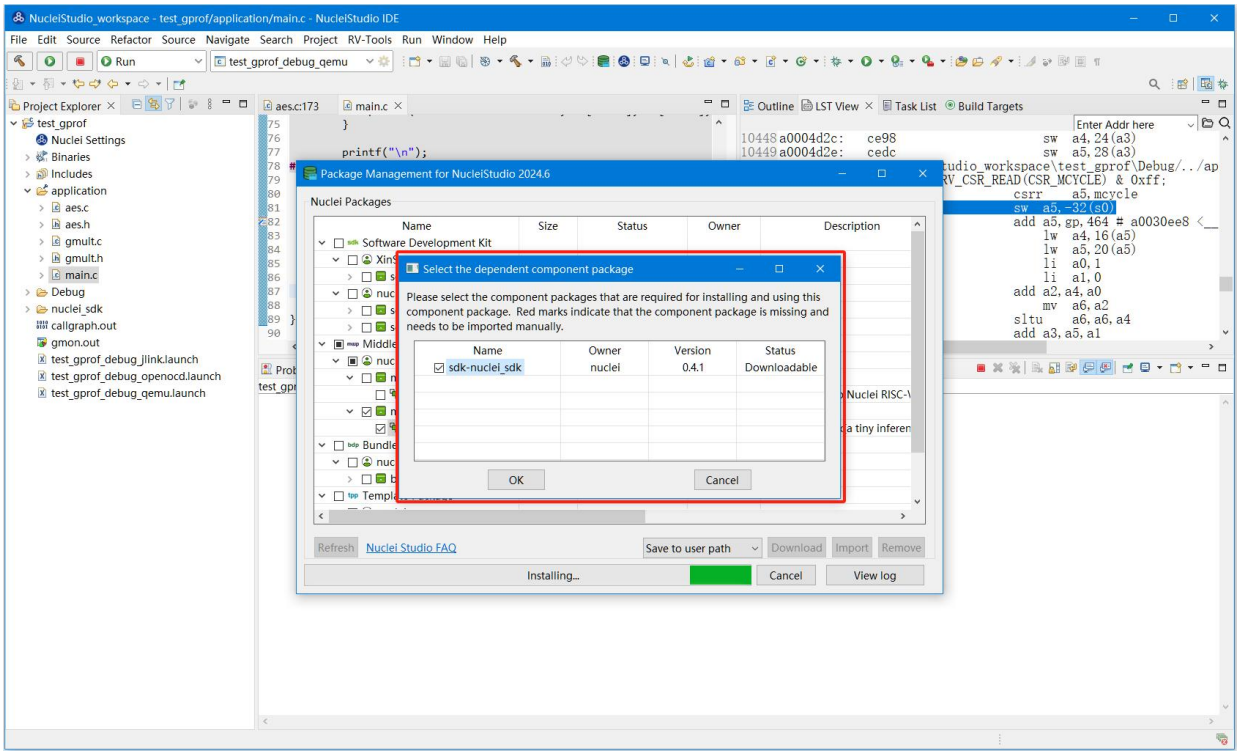


图 4-4 提示所需依赖存在并下载

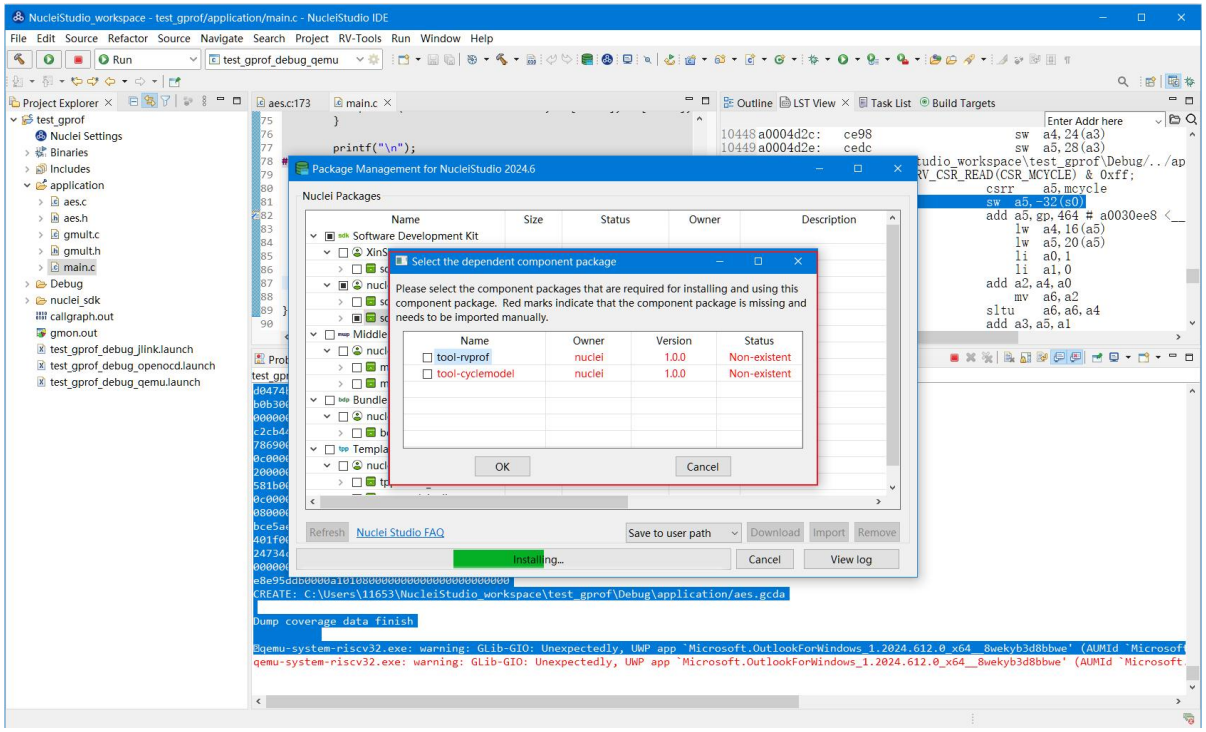
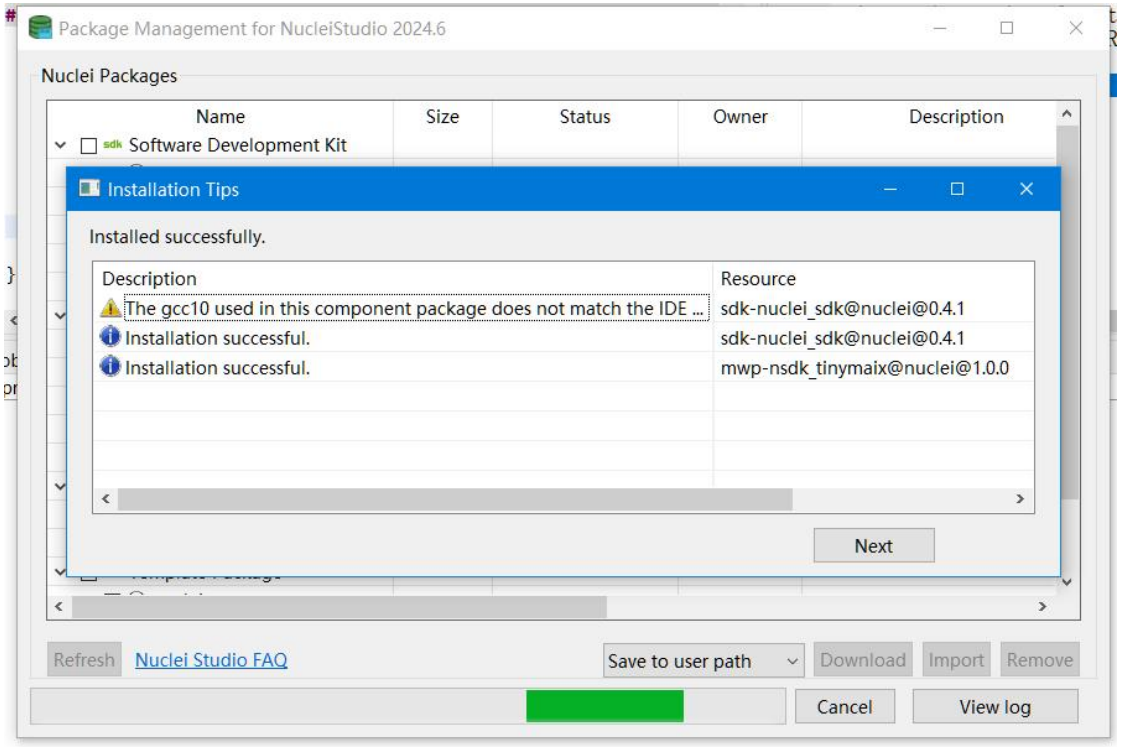


图 4-4 提示所需依赖不存在

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。



4.1.2.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”，如 图 4- 5。也可以在 Project Explorer 视图中选中“New Nuclei RISC-V C/C++ Project”

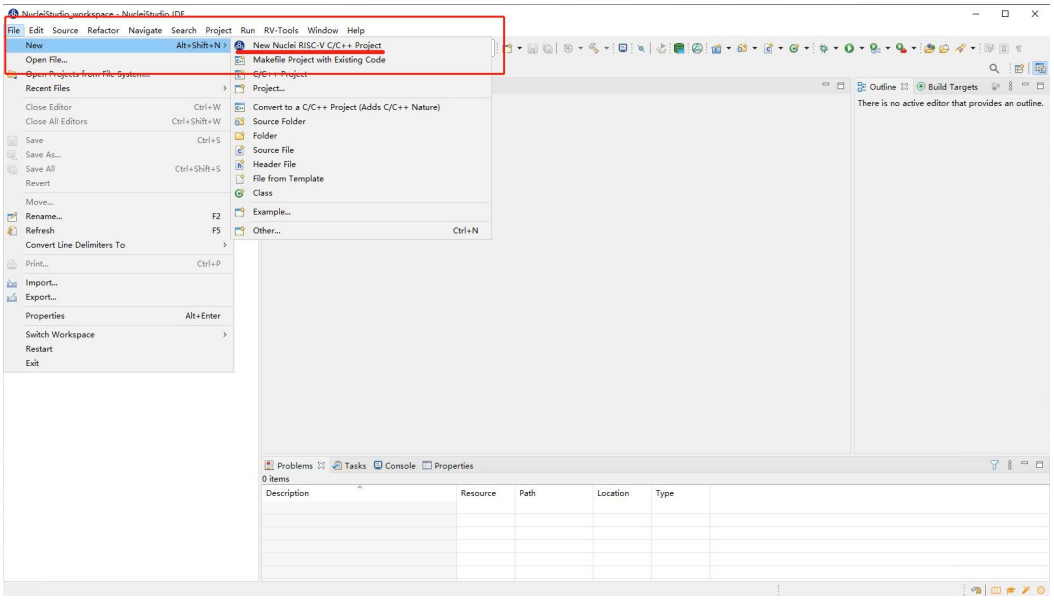


图 4-5 使用菜单栏创建项目

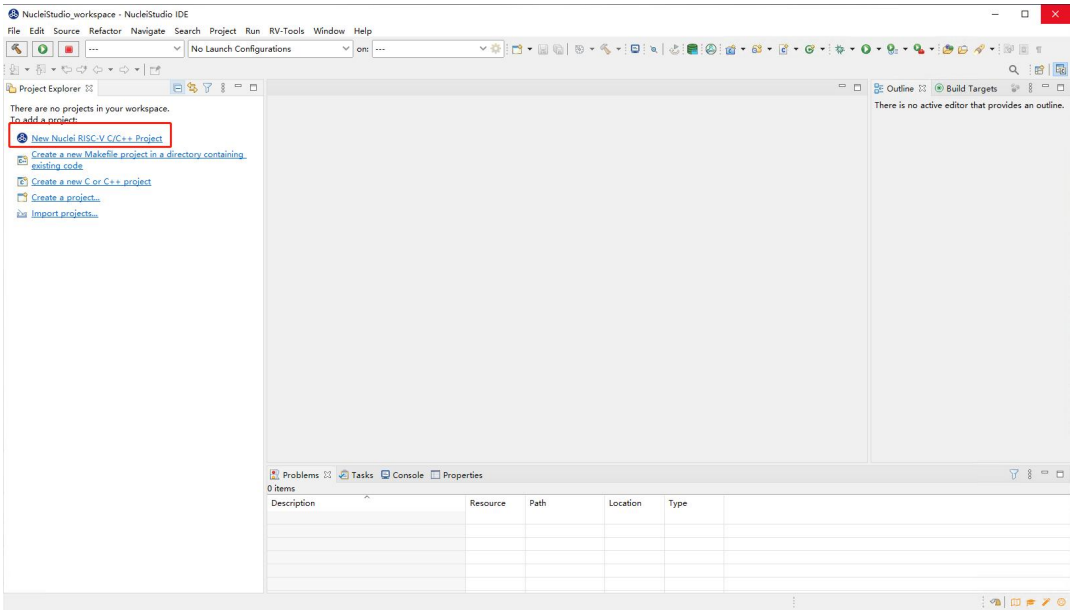


图 4-6 使用菜单栏创建项目

■如图 4-7 所示，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。（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）

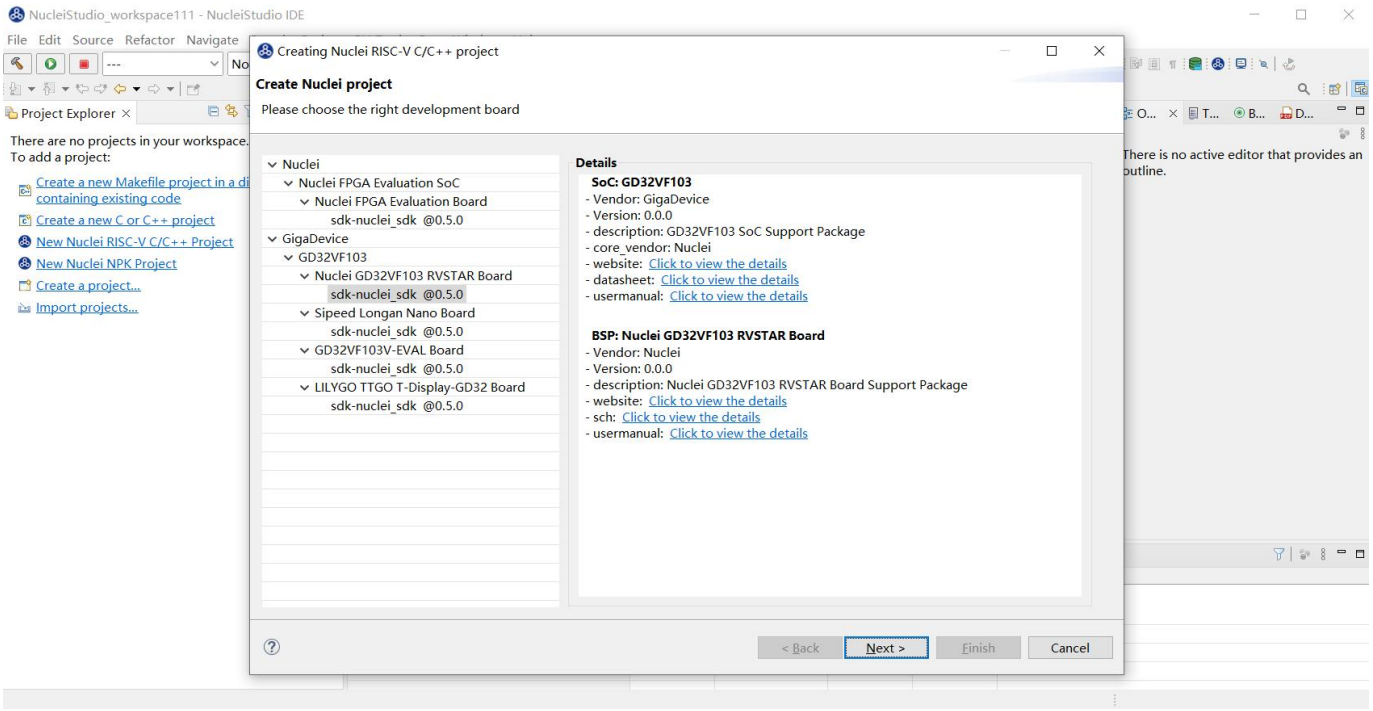


图 4-7 选择建立项目类型

■进入具体的项目配置页如图 4-8 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。

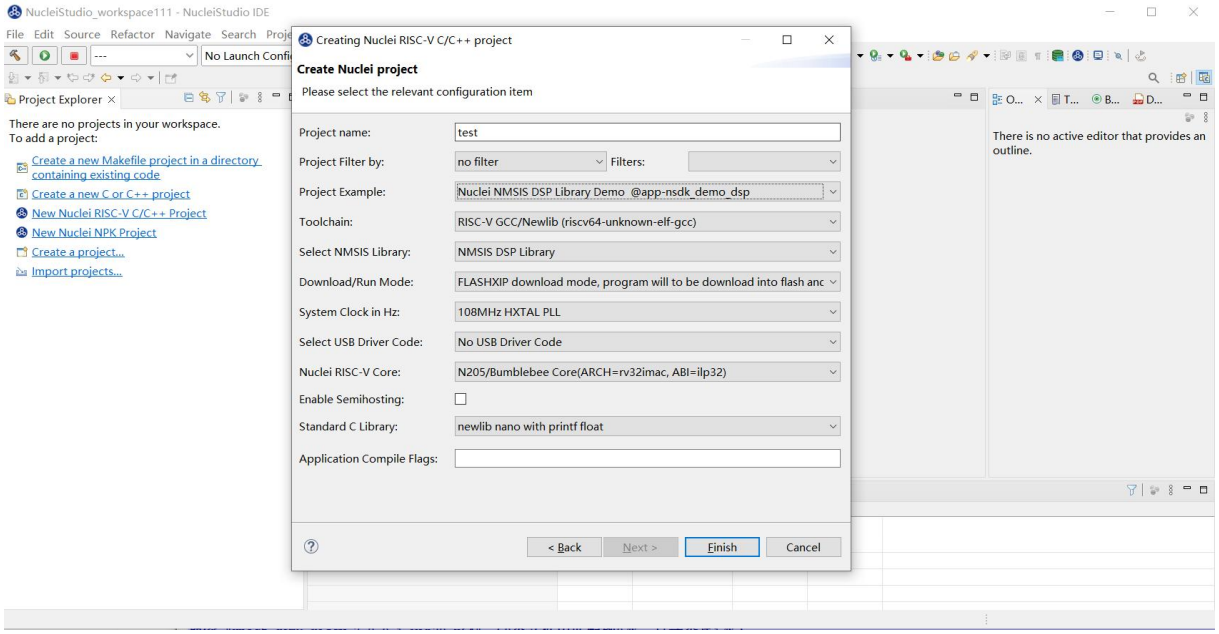


图 4-8 选择程序模板及相关设置项

●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”。

4.1.3.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。

4.1.3.1 SDK Configuration Tools

新建好的工程如图 4-9 所示，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。如图 4-10，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。如图 4-11 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。

- **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。

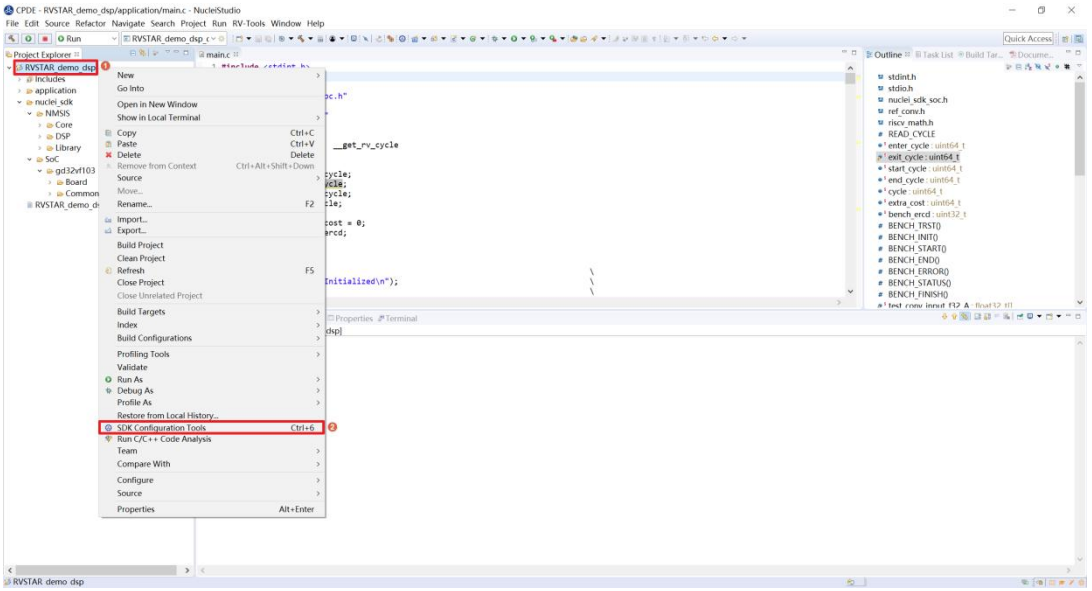


图 4-9 打开 SDK 设置工具

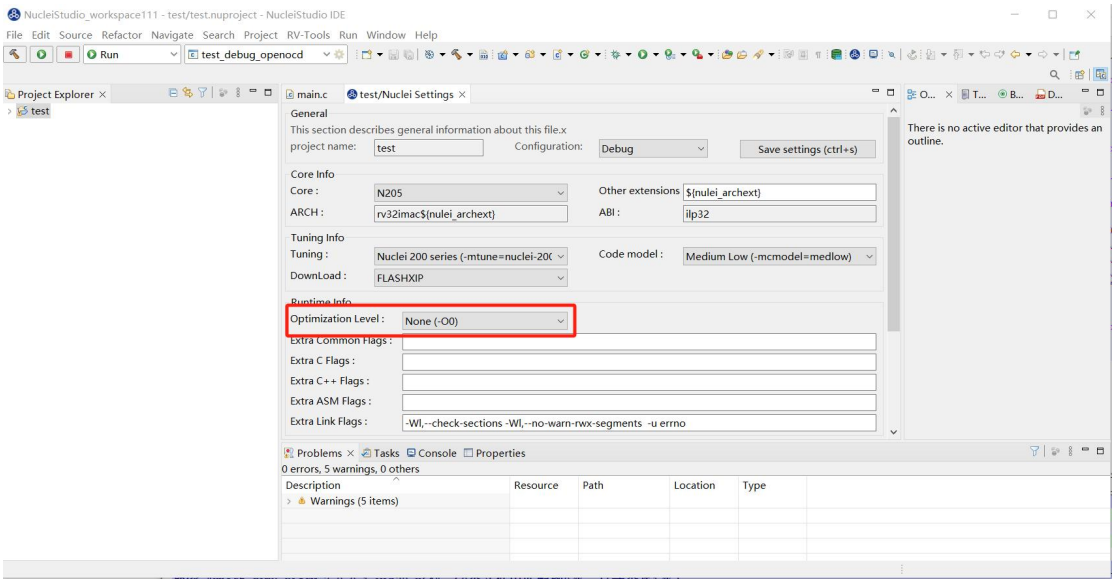


图 4-10 修改工程设置

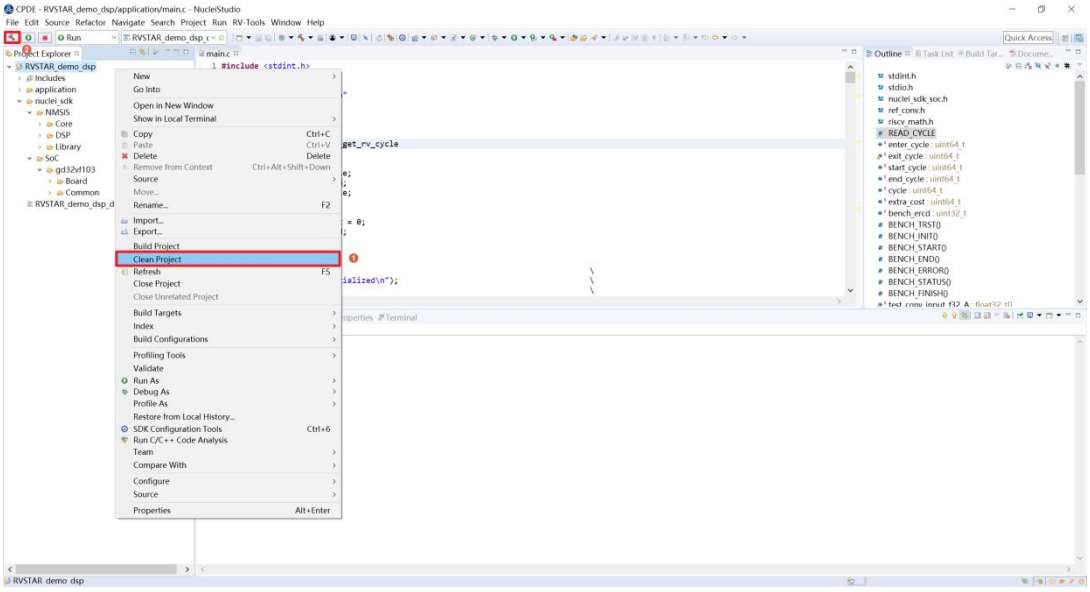


图 4- 11 清理工程并重新编译

4.1.3.2 Nuclei Settings 菜单

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。

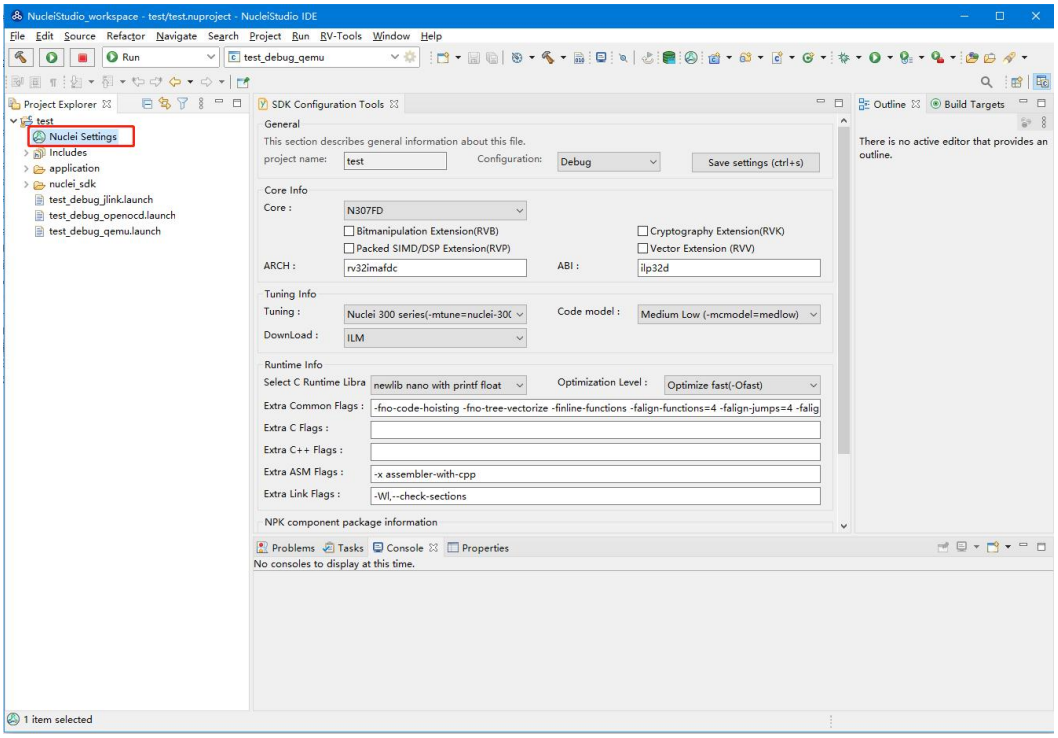


图 4-12 Nuclei Settings 菜单

4.2.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 4.1.章的内容，

将 npk tools 导入到 ide 中，或共享到 www.rvmcu.com 网站上。

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

图 4-13 npk.yml 文件示例

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。

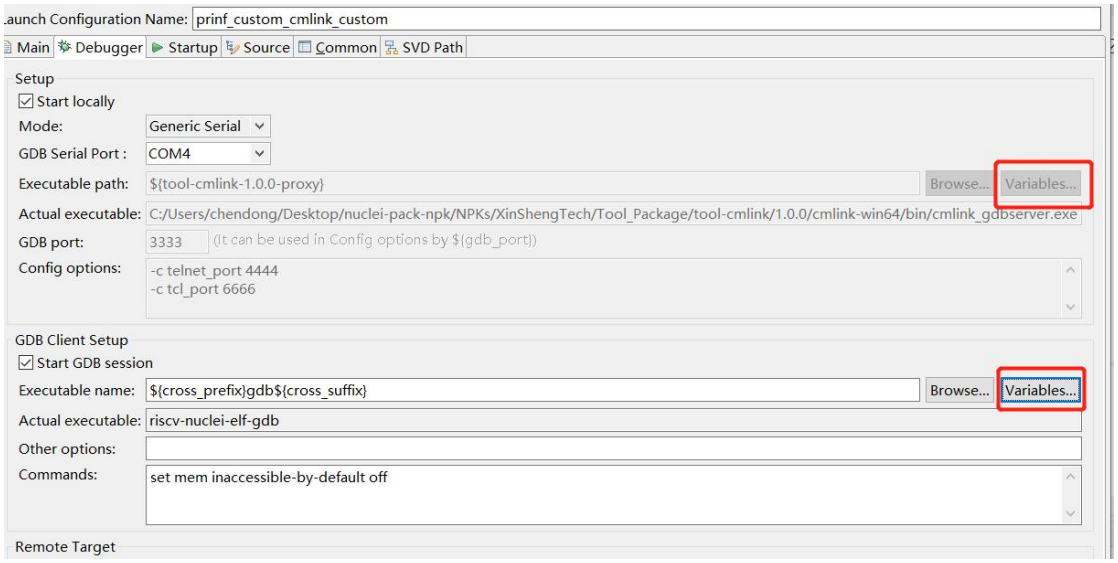


图 4-14 Variables 查看 npk tools 对应参数

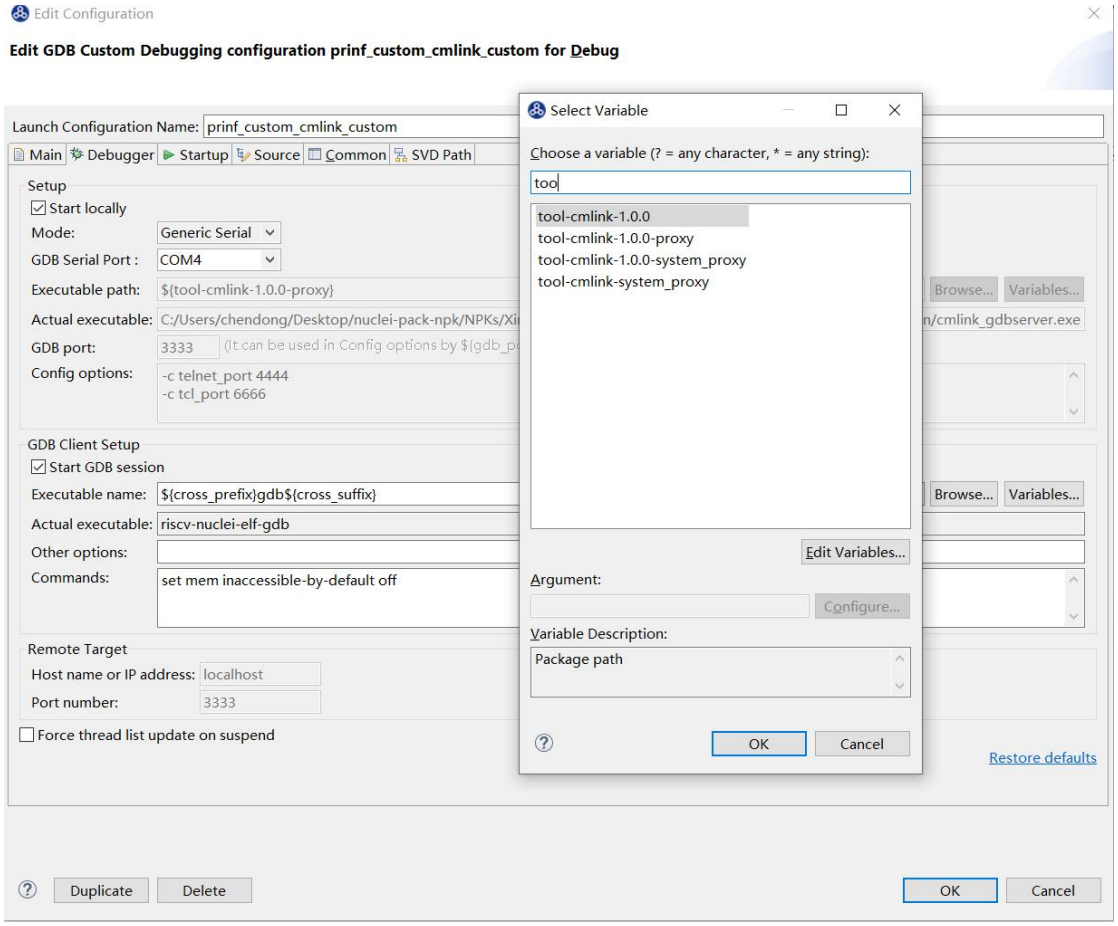


图 4-15 Variables 查看 npk tools 对应参数

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}

5. 创建工程

这里以开发板为 Nuclei FPGA Evaluation Board，评估处理器内核为 N307(rv32imafc)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：

■使用模板自动创建项目：

●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。

5.1.通过模板自动创建项目

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”，如图 5-1。也可以在 Project Explorer 视图中选中“New Nuclei RISC-V C/C++ Project”，如图 5-2。

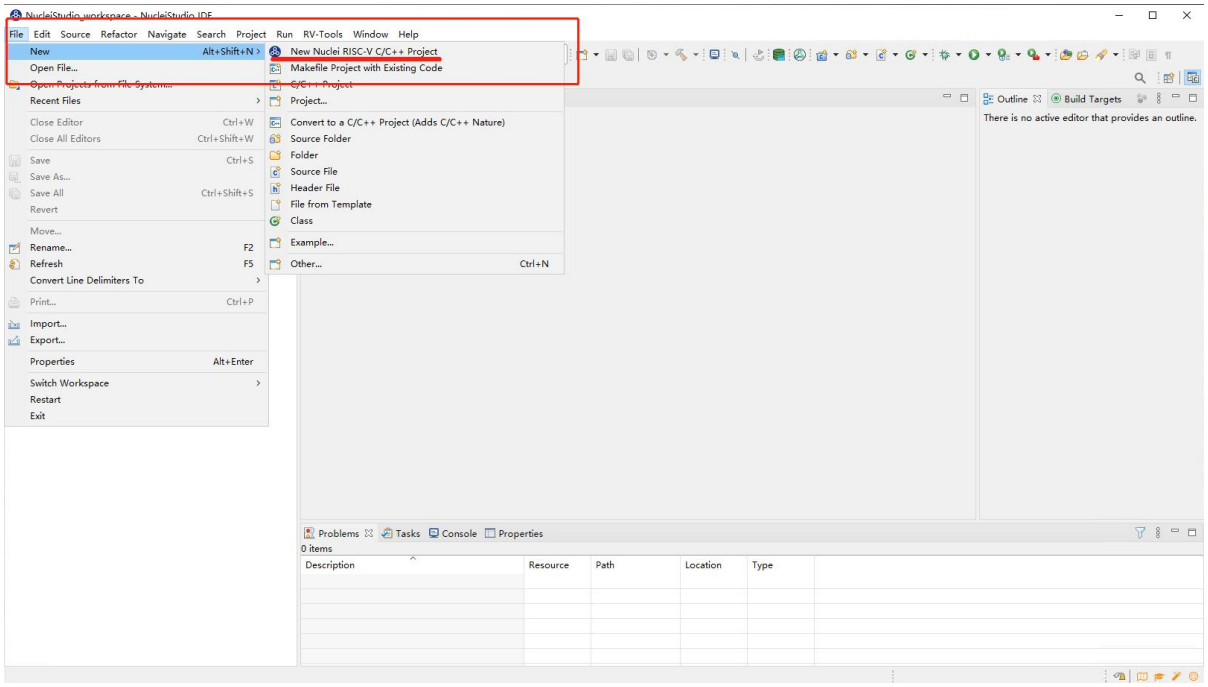


图 5-1 从菜单创建项目

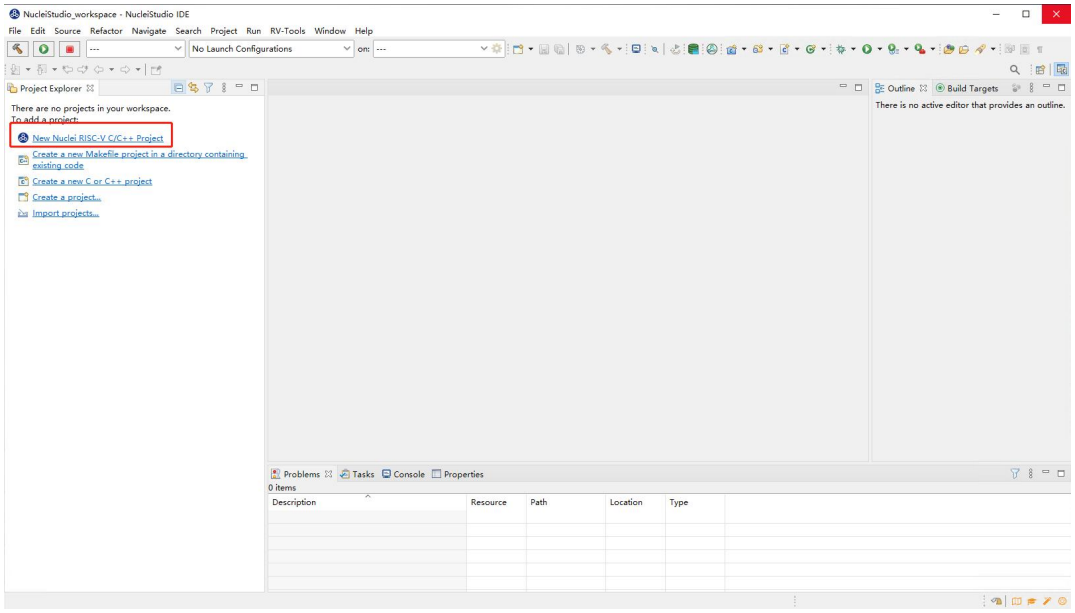


图 5-2Project Explorer 视图创建项目

■如图 5-3 所示，在弹出的窗口中选择项目类型，这里我们在 Nuclei FPGA 板，内核是 N307，SDK 为 nuclei_sdk@0.3.9 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。

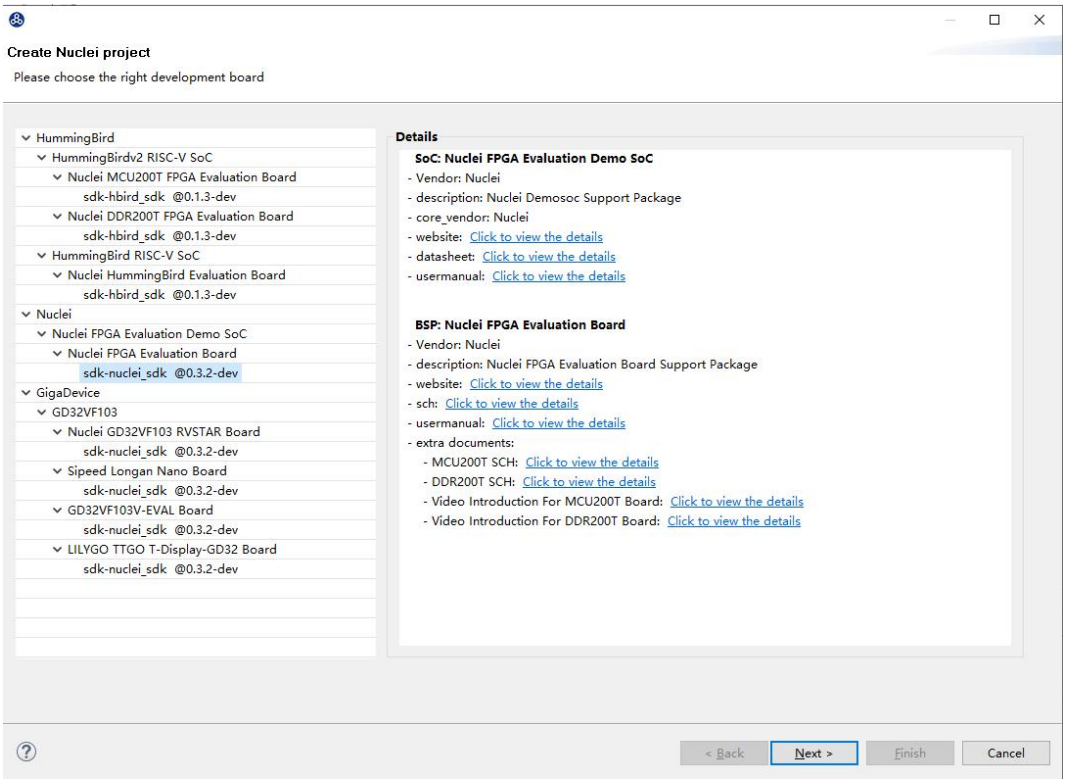


图 5-3 选择建立项目模型

■如下图所示，在弹出的窗口中设定如下参数（**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**）。

●Project name：项目命名。这里设置为“1_helloworld”

●Project Example：选 Helloworld。

●Toolchains：我们使用 Nuclei GUN Toolchain。

●我们的内核是 N307，所以“Core”选择“N307”。

●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 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 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 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。

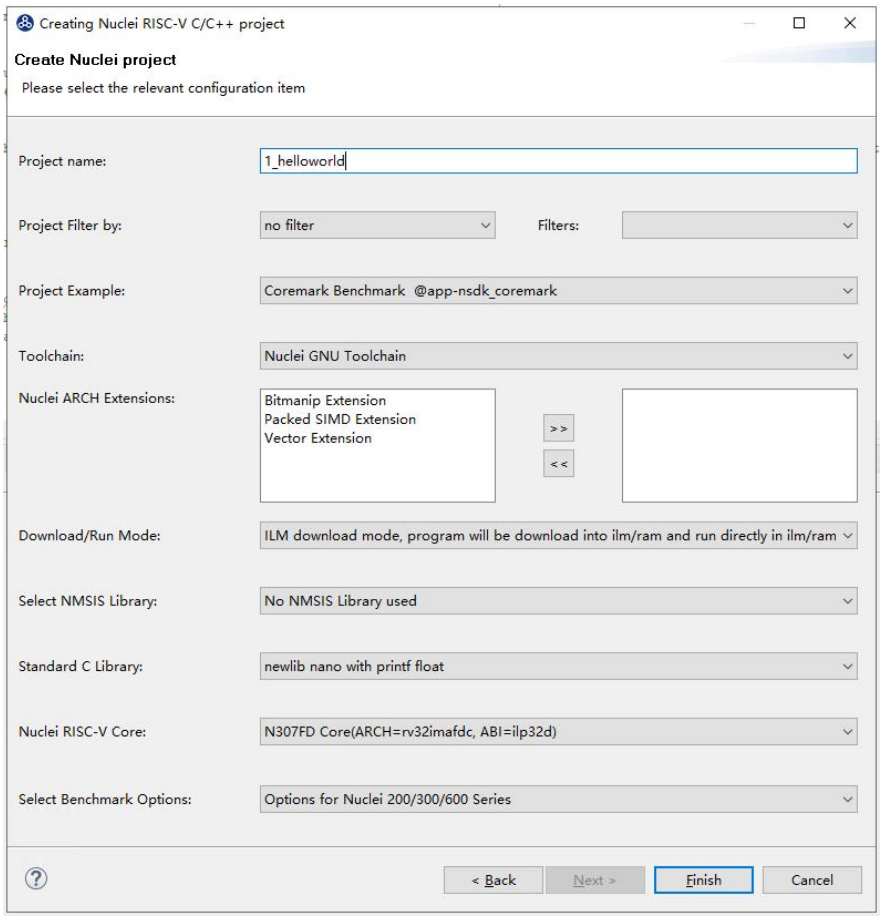


图 5-4 新建项目及模板选择

■如下图所示，使用模板自动创建 Hello World 项目已经完成。

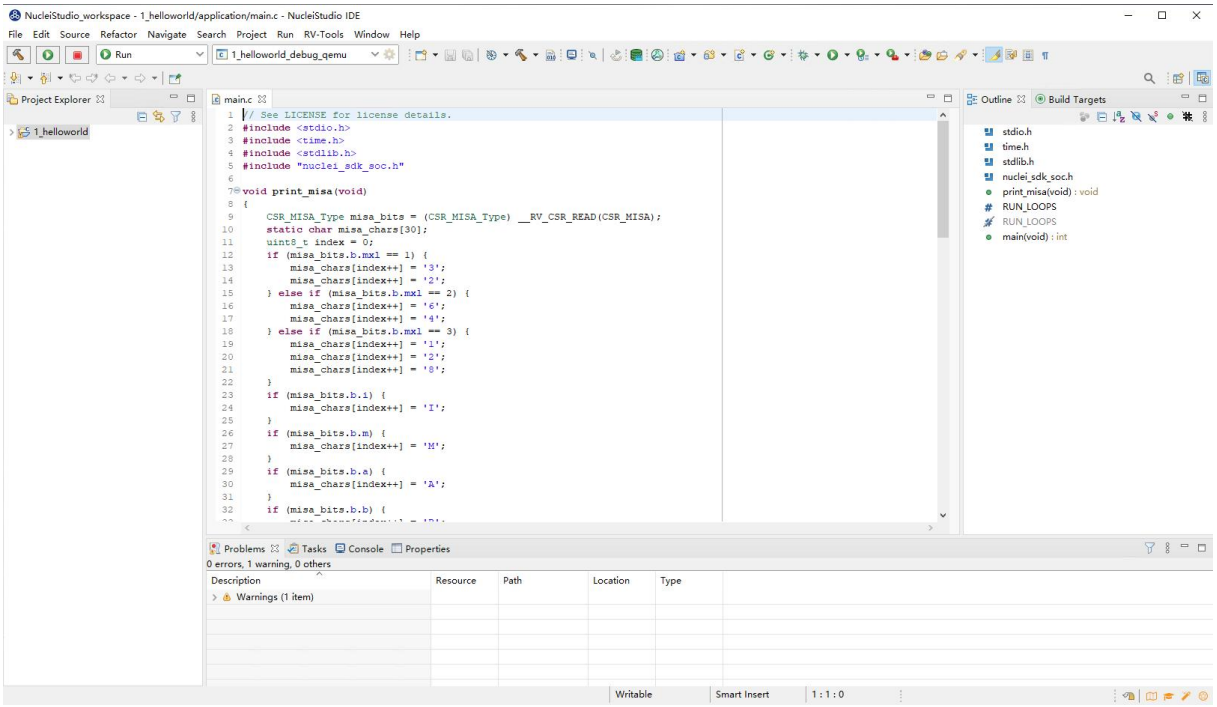


图 5-5 完成模板创建项目

■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，编译后如图 5-6。

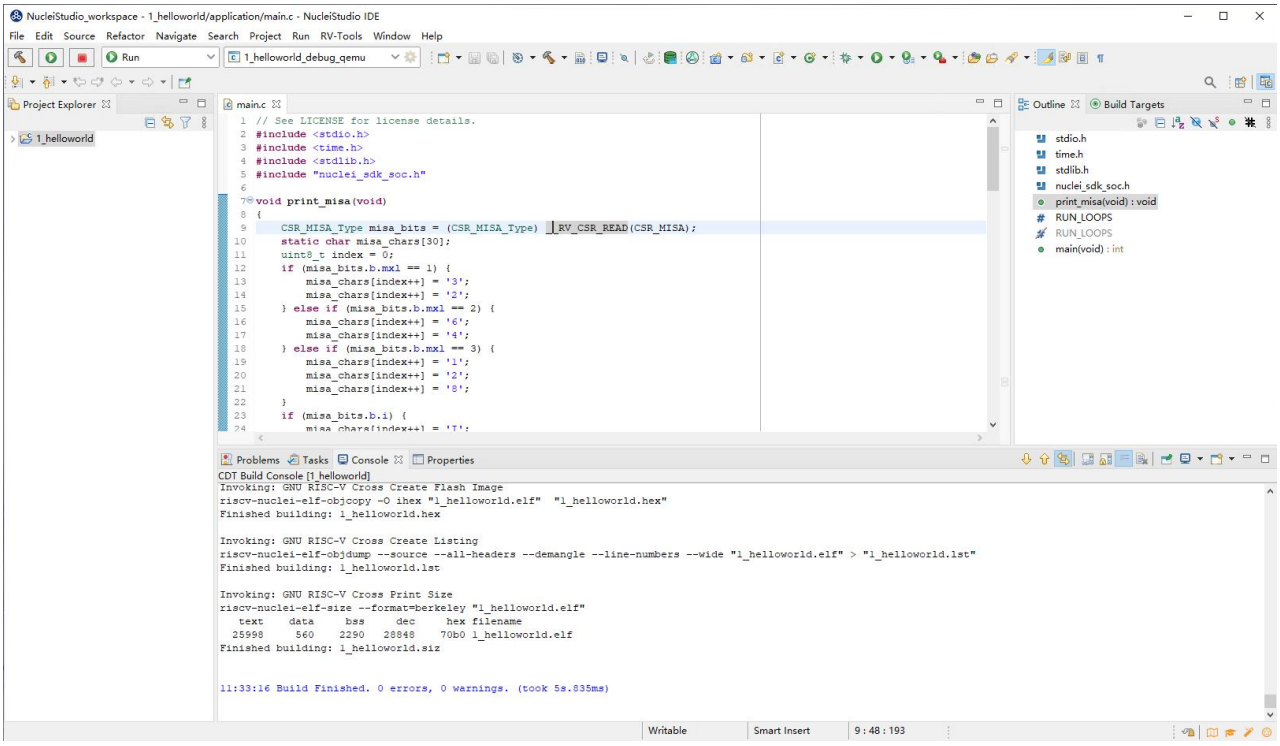


图 5-6 完成项目编译

■如图 5-7 所示，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。

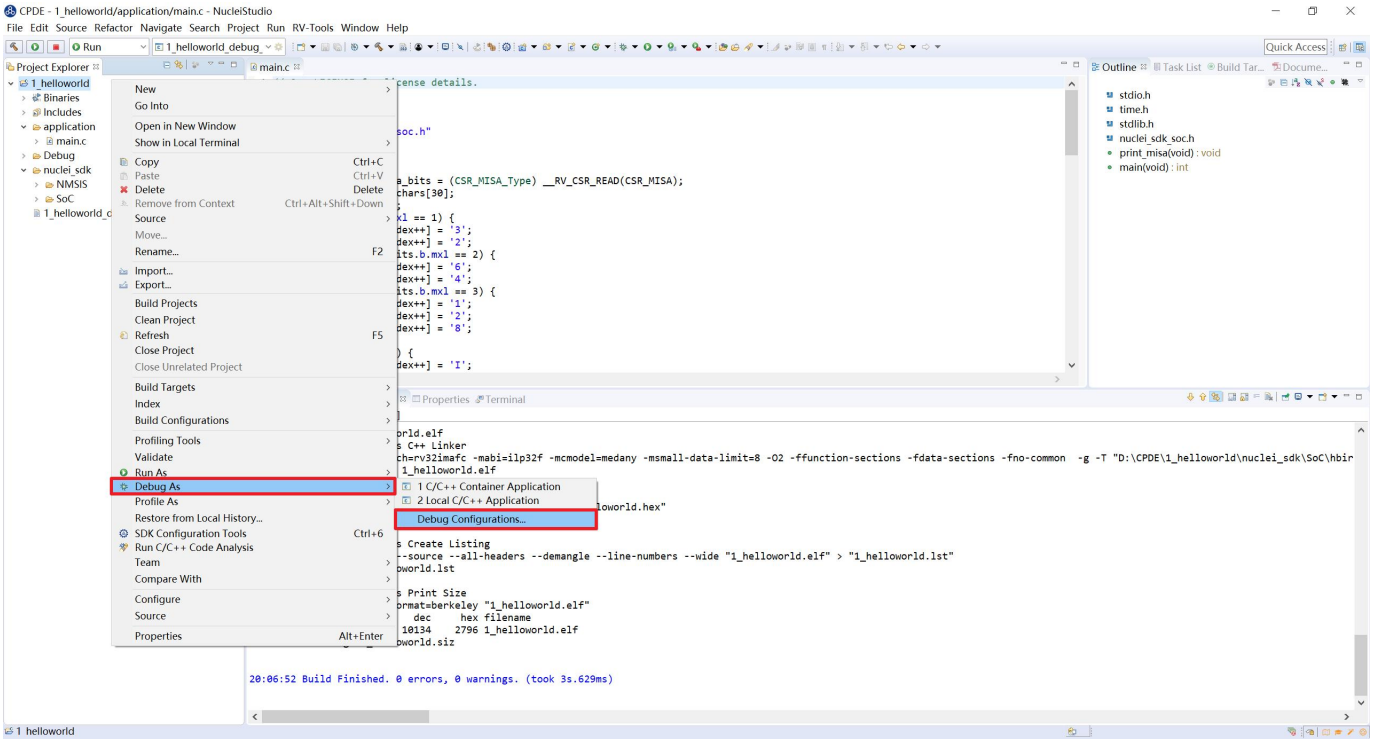


图 5-7 开启调试配置面板

■如图 5-8 所示，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，可以查看第 7 章进行详细了解。

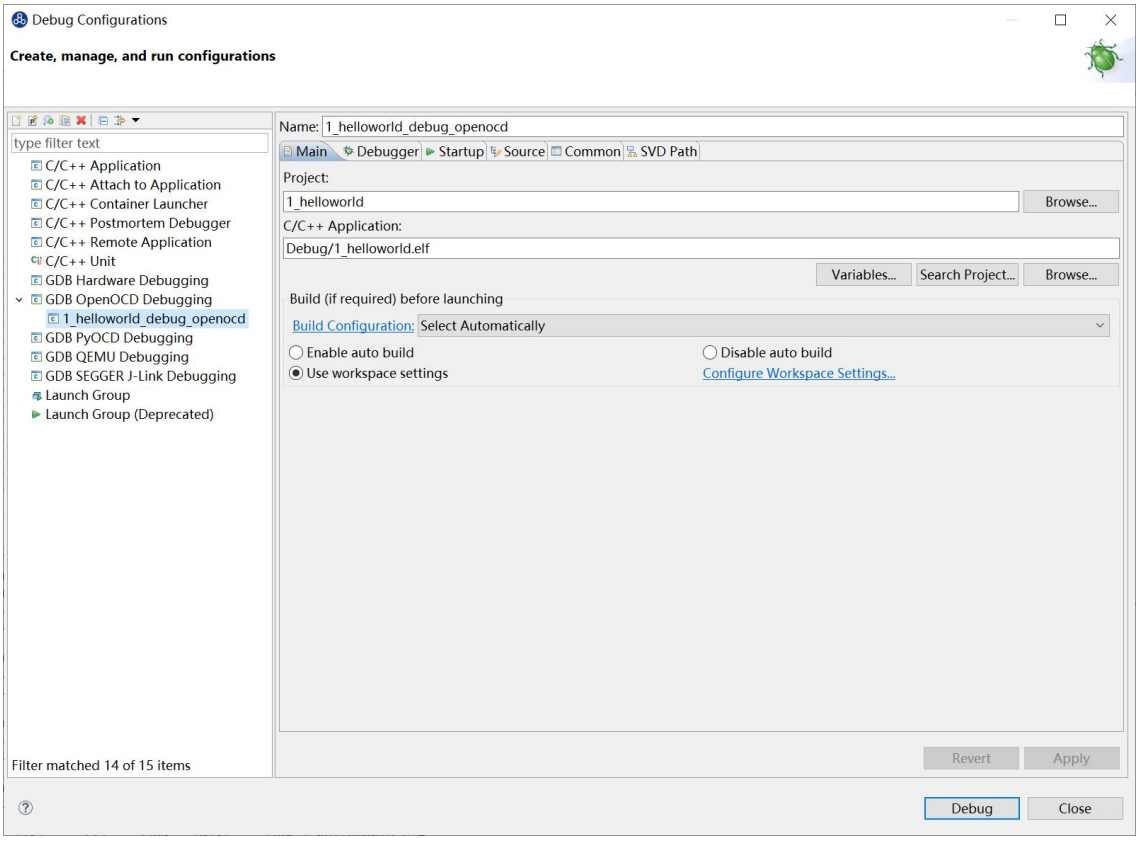


图 5- 8 自动生成的 OpenOCD 配置

5.2.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。

5.2.1.将 Nuclei Studio IDE 写入到注册表

下载 Nuclei Studio IDE 2022.12 版，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。

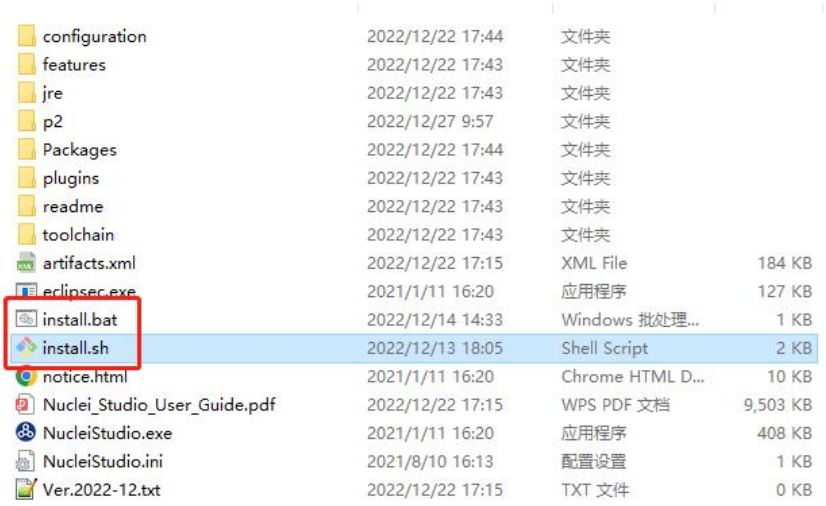


图 5-9 install.bat/install.sh 文件

图 5-10 用户授权

5.2.2.通过应用关联文件导入工程

Nuclei Studio IDE 2022.12 版创建工程 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。

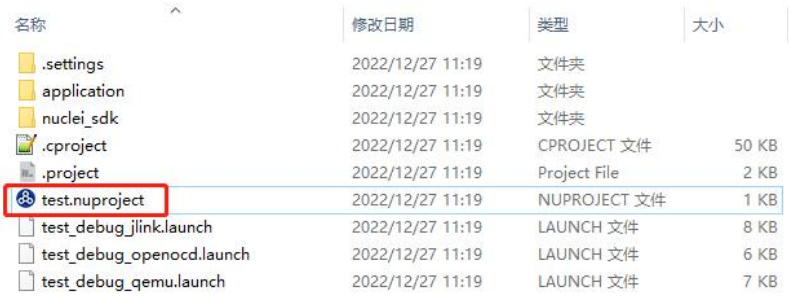


图 5-11 应用关联文件 test.nuproject

5.3.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）。如图 5-12 所示，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，相关内容会在下一节中具体介绍。

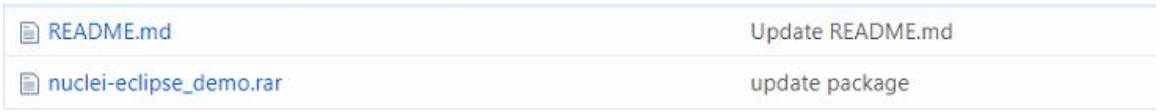


图 5-12 位于 github 上面的项目包

将 nuclei-eclipse_demo.rar 压缩包下载解压后，内容如图 5-13 所示，分别为：

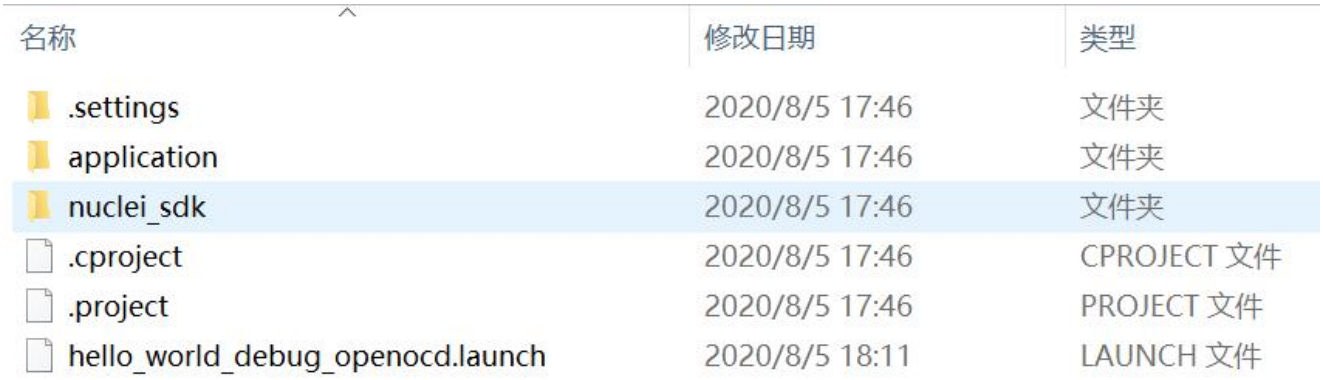


图 5-13 选择导入的项目

■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 5-14 所示，选择“Existing Project into WorkSpace”后，点击“Next”。

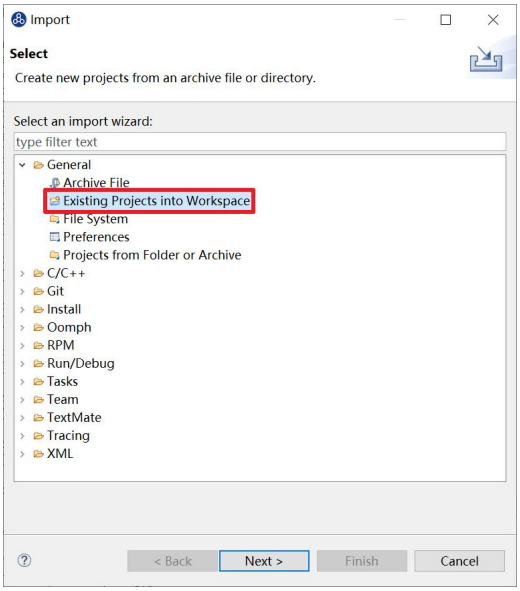


图 5-14 选择导入的方式

■点击“Browse”，选择需要导入的项目路径，如图 5-15 所示，

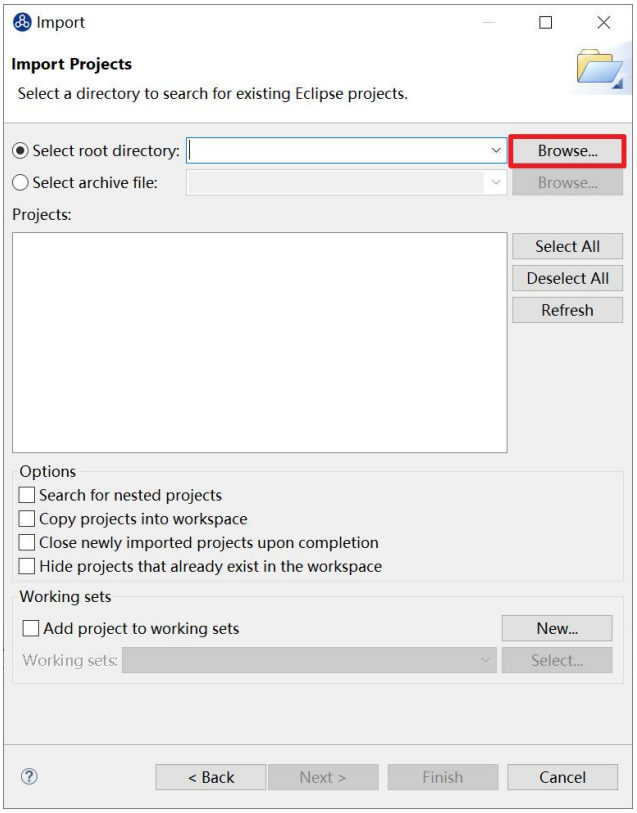


图 5-15 选择需要导入的项目

■需要的导入的项目成功被 IDE 识别，点击“Finish”如图 5-16 所示

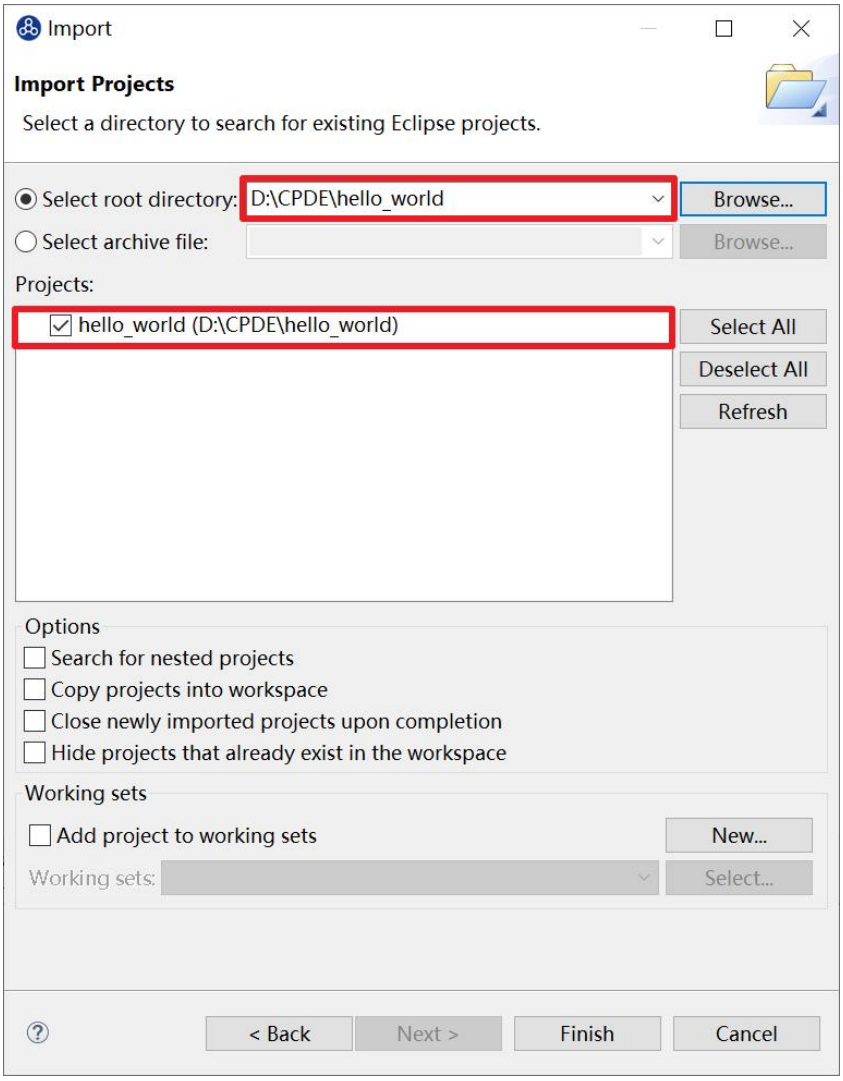


图 5-16 识别出要导入的项目

■ 在 IDE 的项目资源管理器中显示导入项目的目录结构如图 5-17 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。

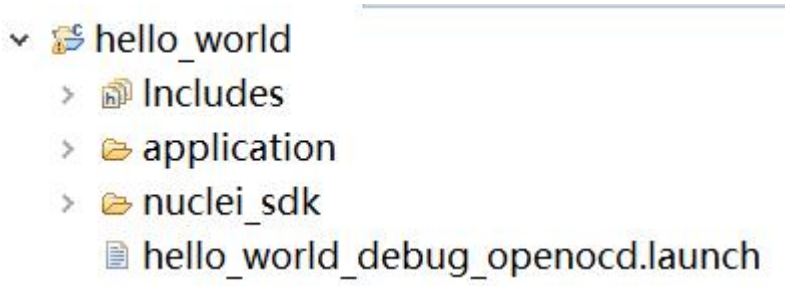


图 5-17 显示项目的文件结构

5.4.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（**不建议使用，建议使用章节 4.1 NPK 方式**）。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。

5.4.1.手动创建项目

■在菜单栏中选择“File—> New —> C/C++ Project”。如图 5-18 所示。

■如图 5-19 所示，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

■如图 5-200 所示，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。

■如图 5-211 所示，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 Hello World 项目的创建。

■创建完成的 Hello World 项目界面如图 5-22 所示。

■新建一个 application 文件夹。如图 5-23，在工程处右击选择 New —> Folder，输入 application，点击“Finish”完成新建工程，如图 5-24。将 main.c 拖入 application 文件夹完成文件分类。

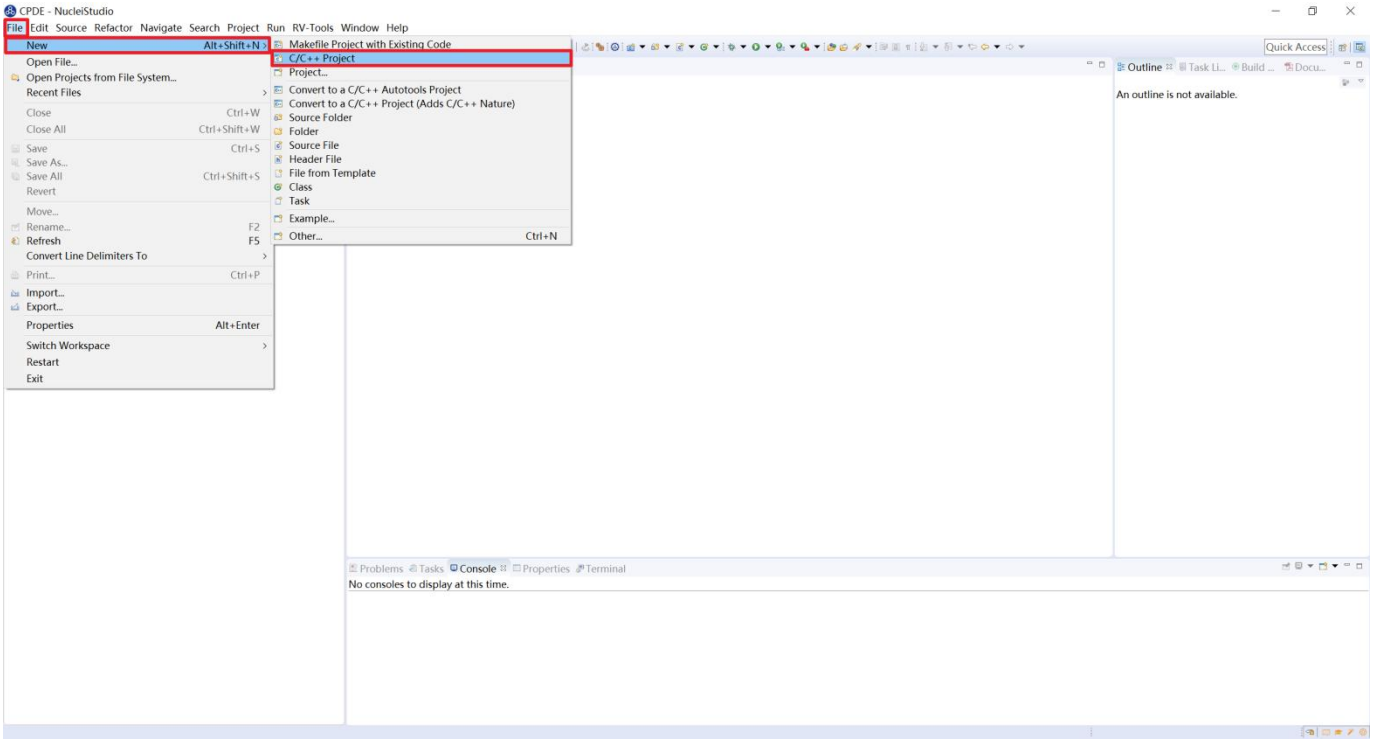


图 5-18 新建 C/C++ Project

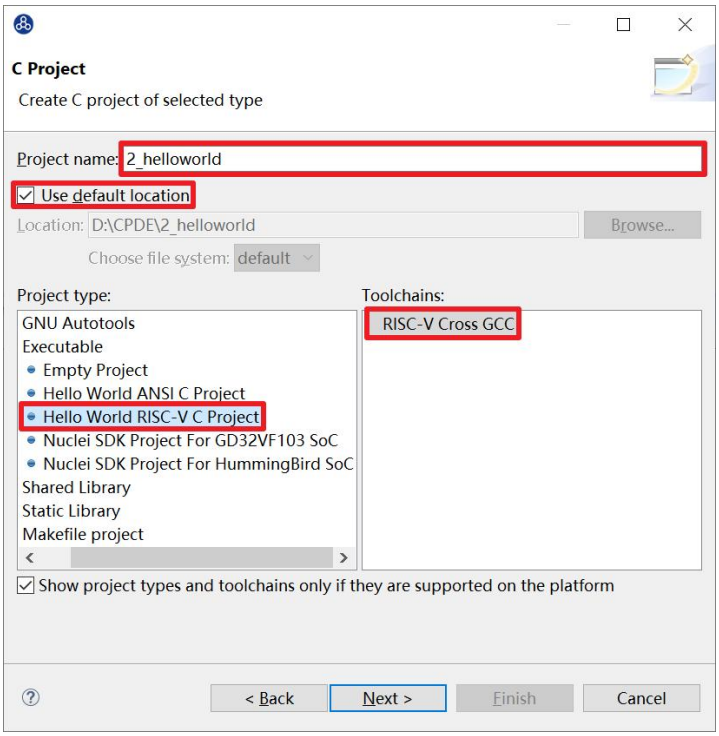


图 5- 19 设置 C Project 项目名和类型

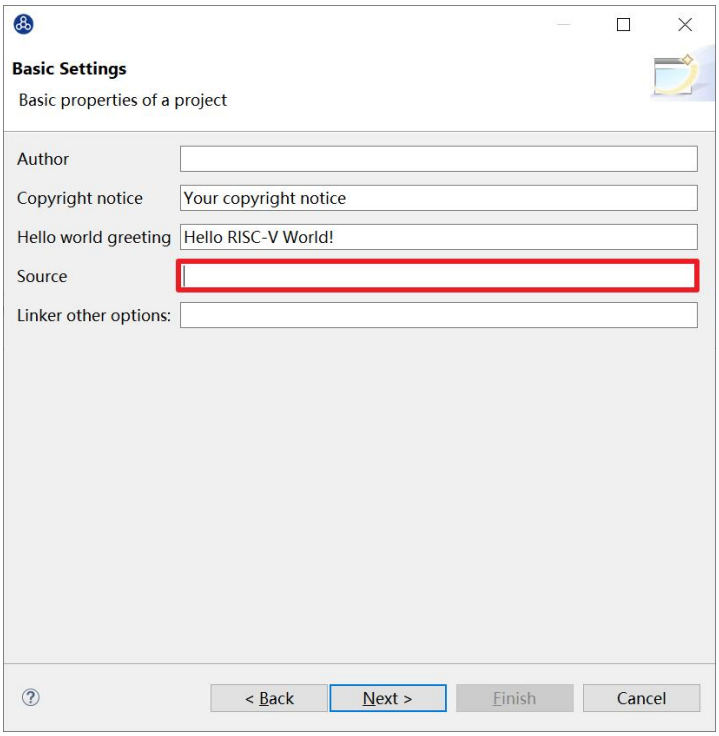


图 5- 20 设置 Hello World 项目的基本信息

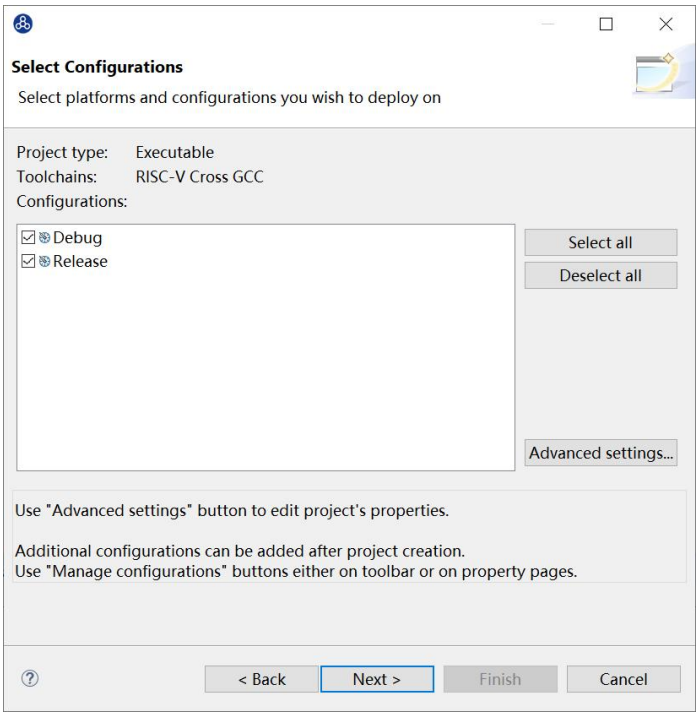


图 5-21 设置项目的调试或者发布属性

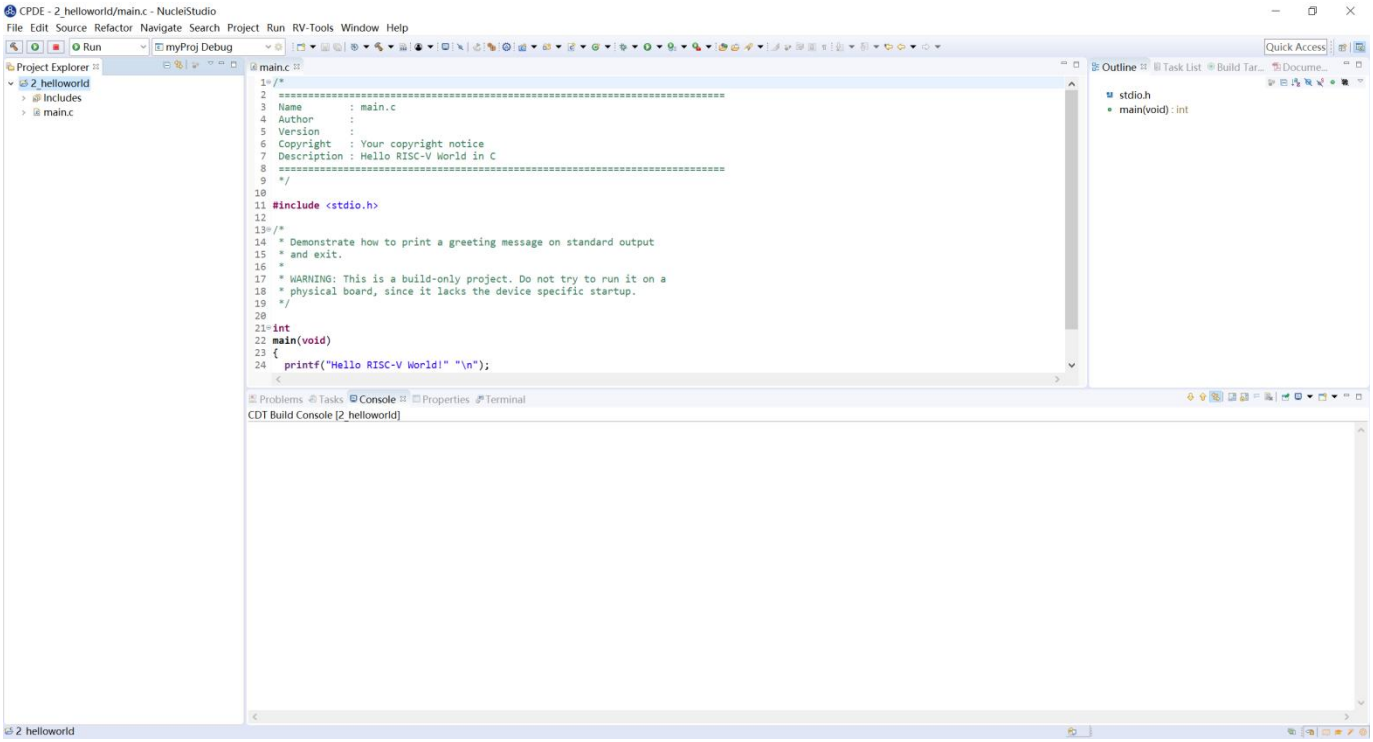


图 5- 22 创建完成 Hello World 项目的界面

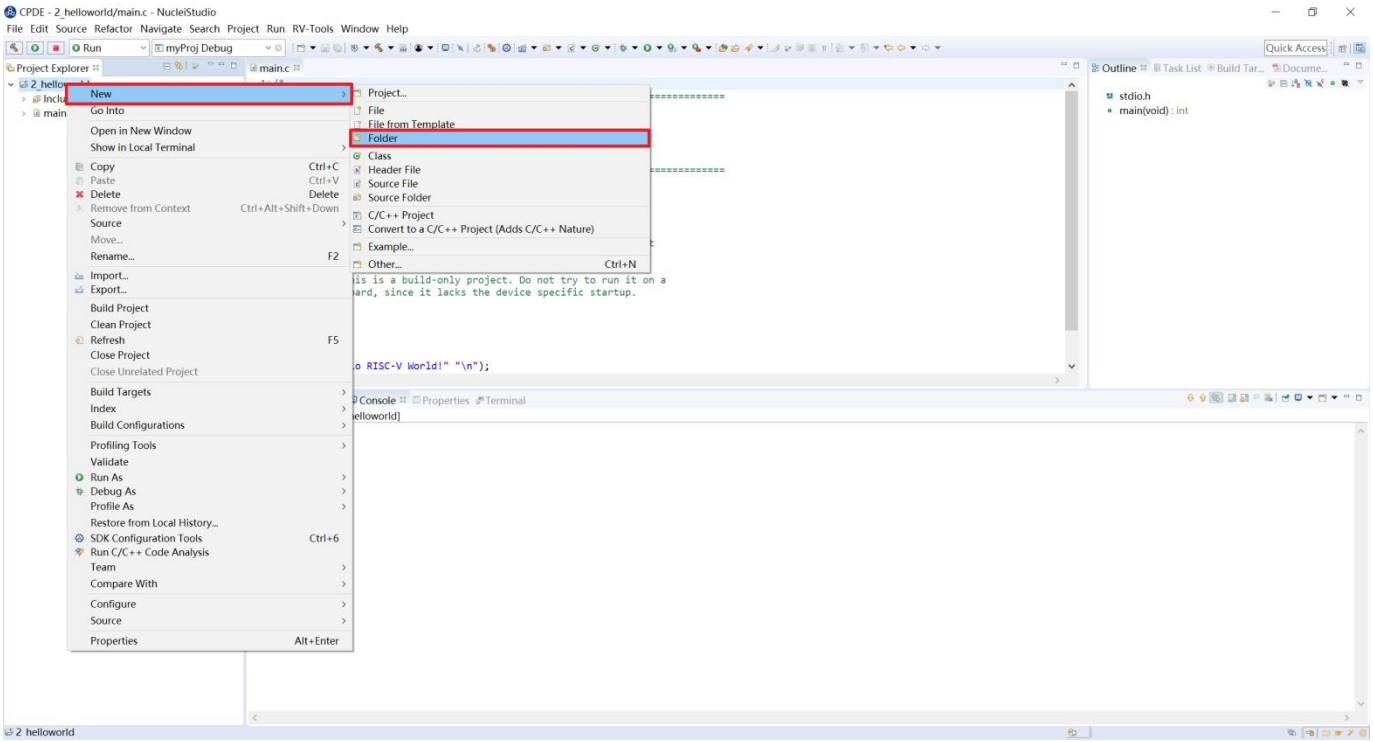


图 5-23 打开新建文件夹

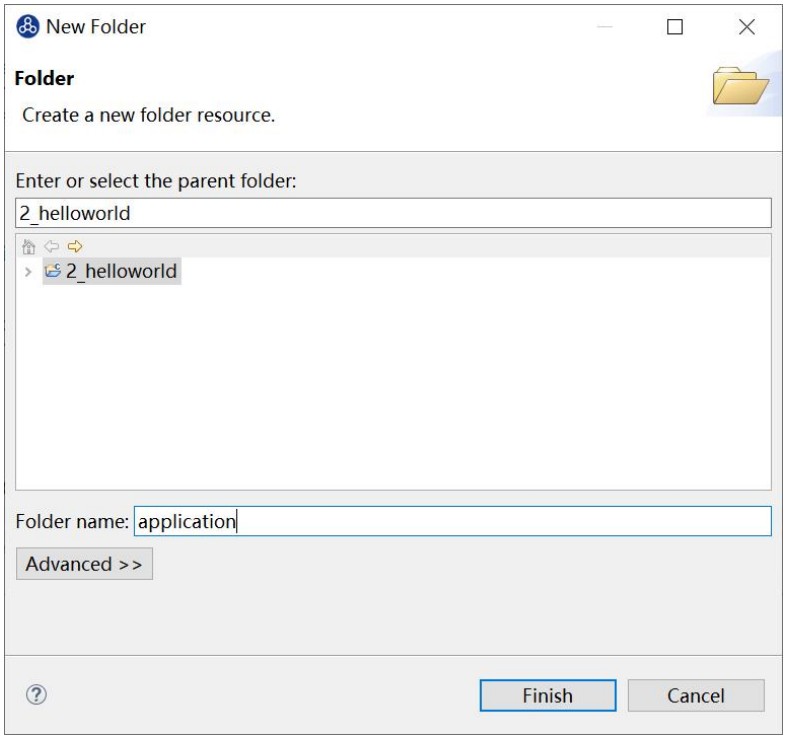


图 5-24 完成新建文件夹

5.4.2.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。
  - 如图 5-25 所示，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。

◆如图 5-26 所示，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。

◆将图 5-12 中介绍的 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，如图 5-27 所示。

◆回到 Nuclei Studio，在 Project Explorer 栏中选中 2_helloworld 项目，单击鼠标右键，选择“Refresh”，如图 5-28 所示。

◆Refresh 之后 2_helloworld 项目的下便可以看到 nuclei_sdk 文件夹，如图 5-29 所示，至此便完成了 nuclei_sdk 源文件的导入。

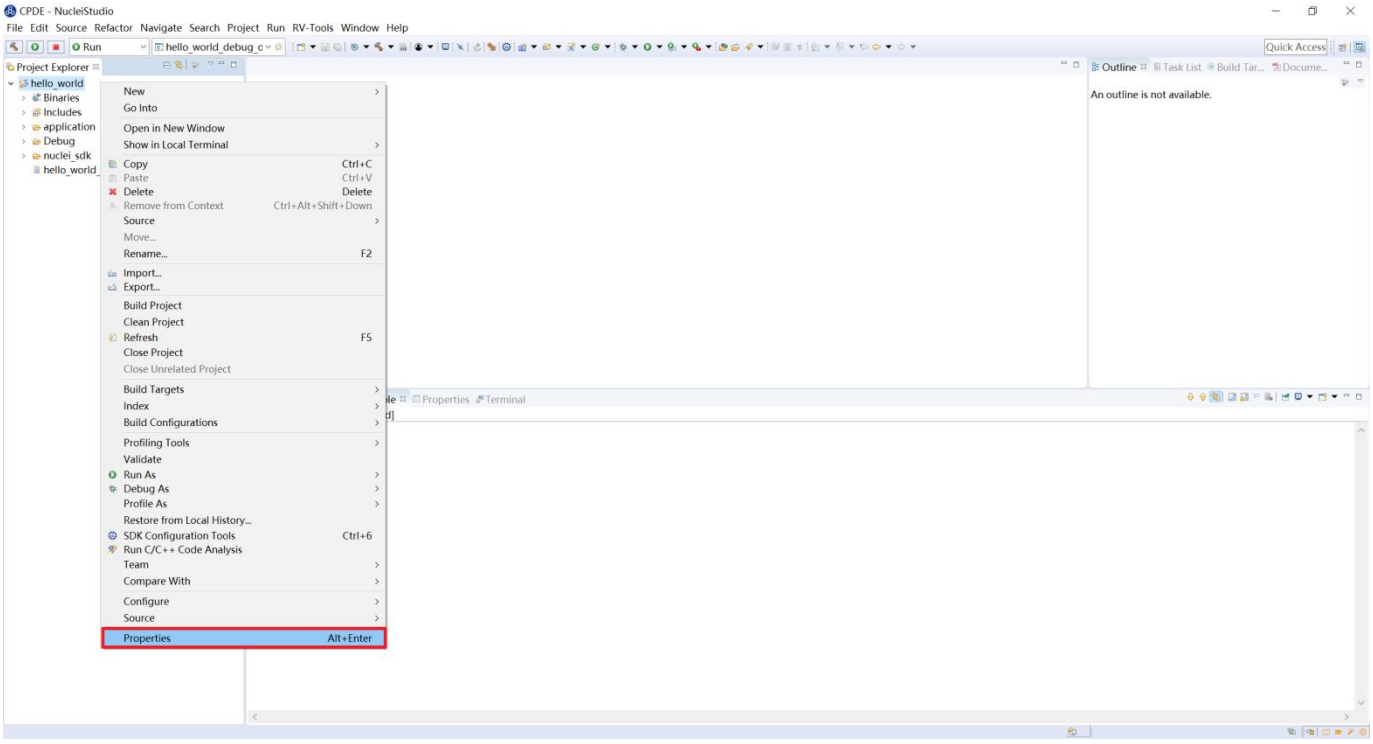


图 5-25 打开工程设置选项页面

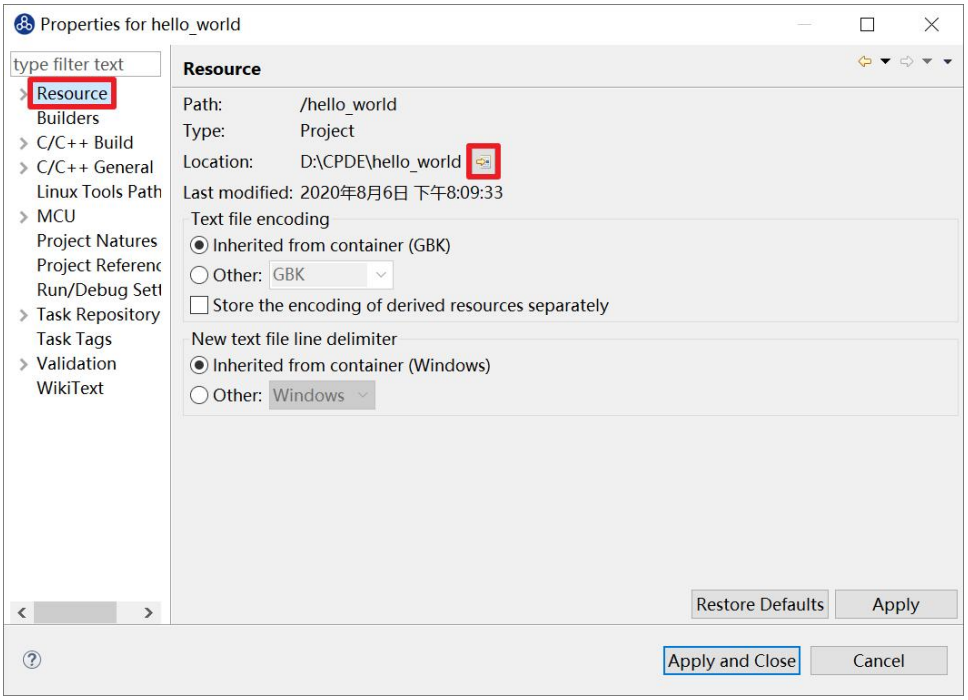


图 5- 26 在弹出窗口进入 hello_world 项目的文件夹位置

图 5- 27 添加 nuclei_sdk 文件夹

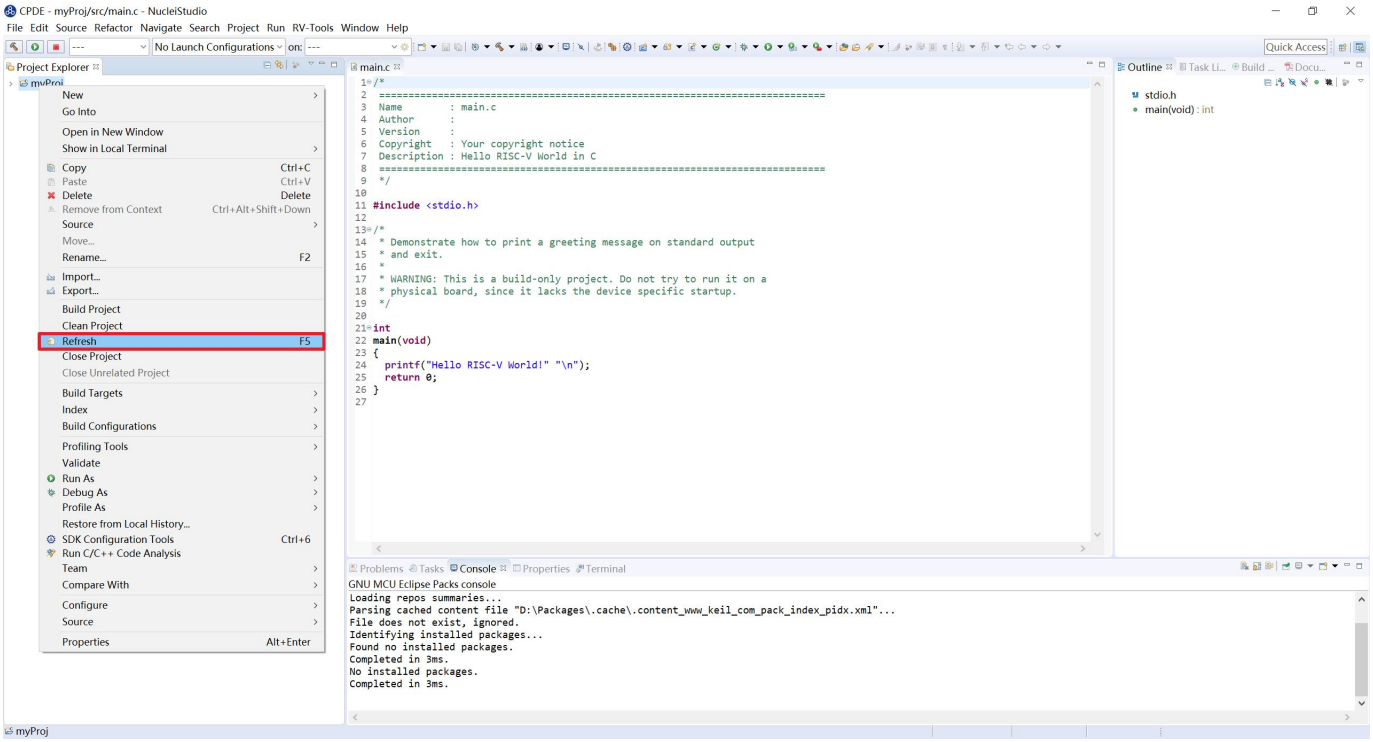


图 5-28 刷新工程

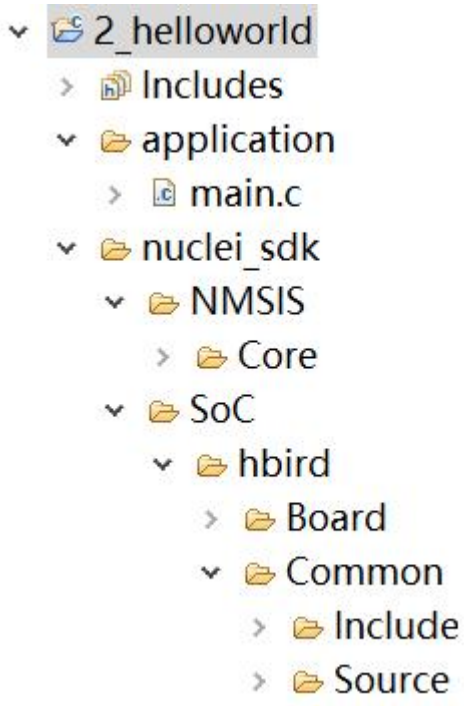


图 5-29 新建项目目录下的文件夹

5.4.3.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。

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。
- 如图 5-300 所示，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。

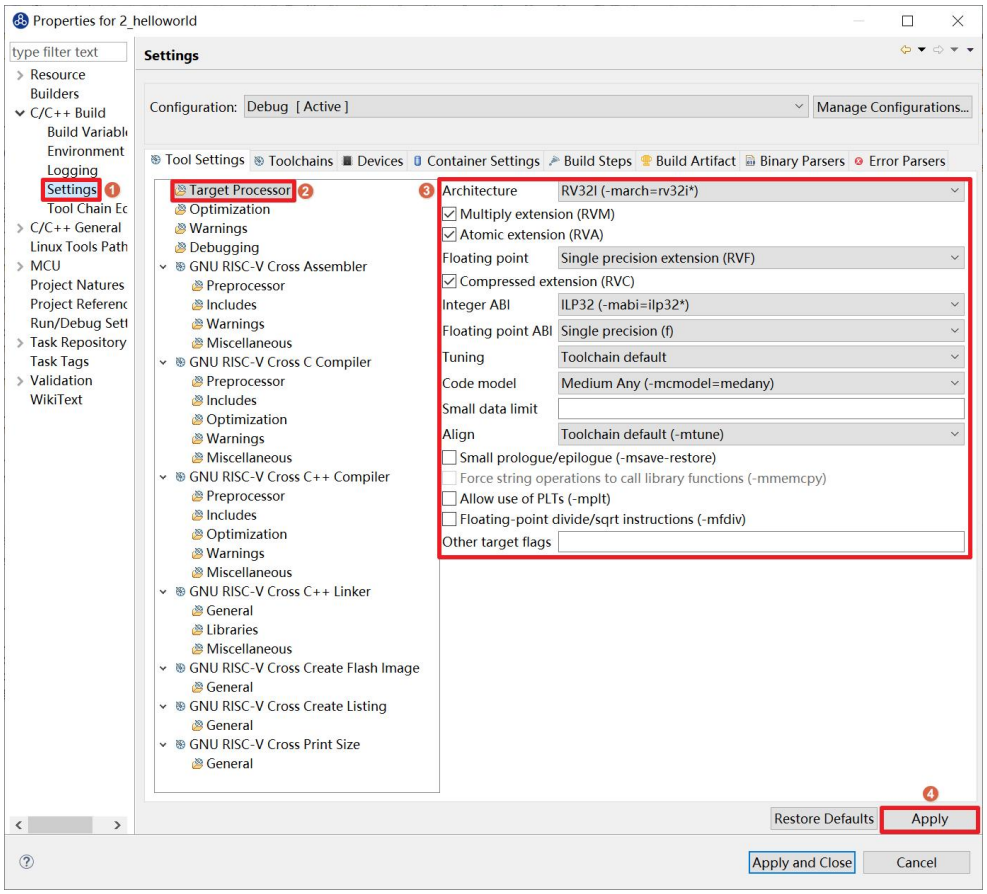


图 5- 30 配置 Target Processor 选项

■如图 5- 311 所示，选中“Optimization”，按照图所示勾选配置选项。

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

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)

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。

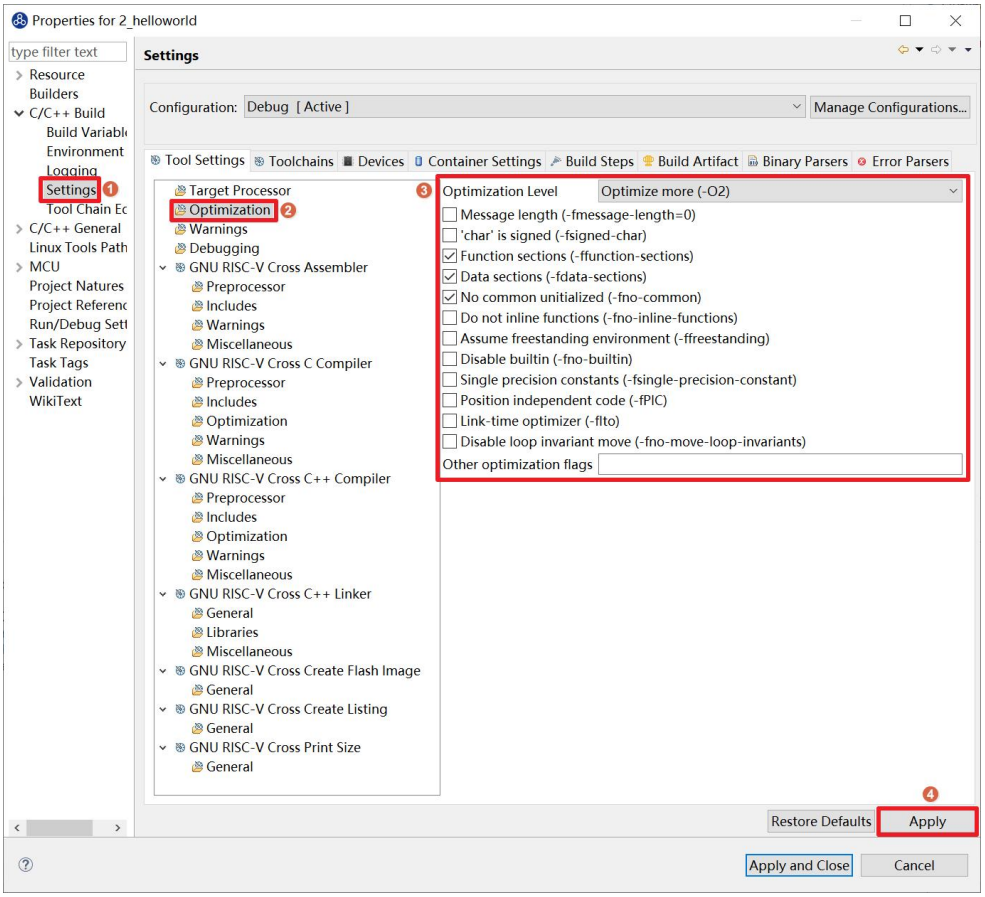


图 5-31 配置 Optimization 选项

■如图 5-32 所示，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。

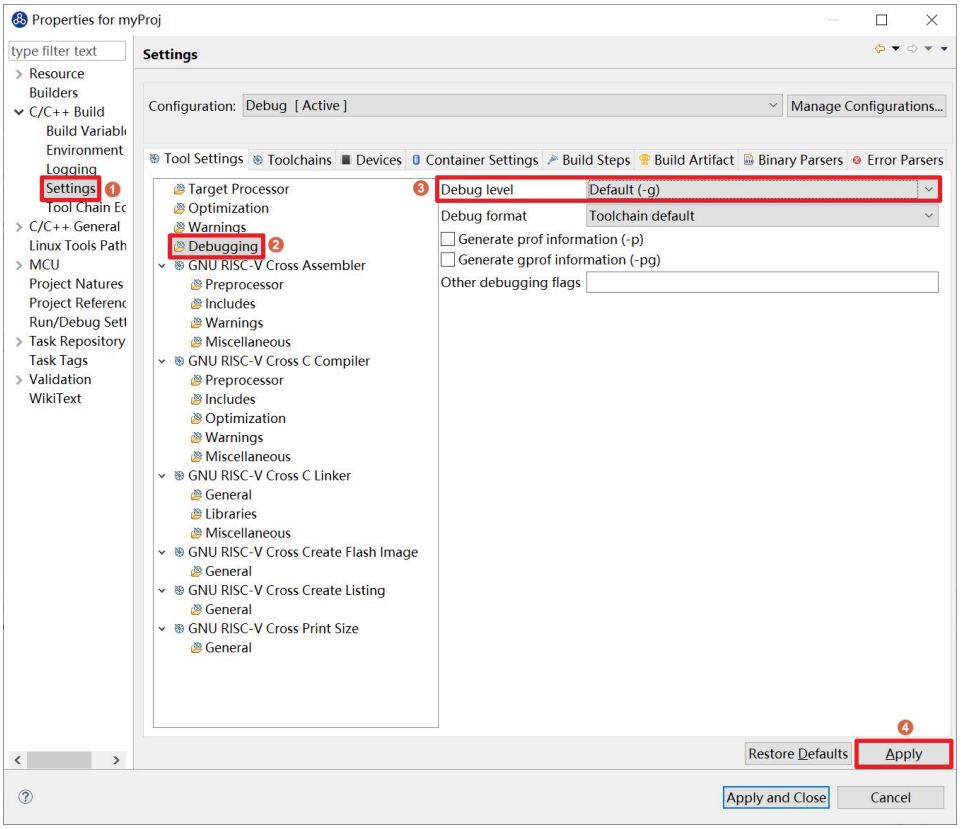


图 5- 32 配置 Debugging 选项

■选中 GNU RISC-V Cross C Linker 的 General。

●如图 5- 33 所示，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。

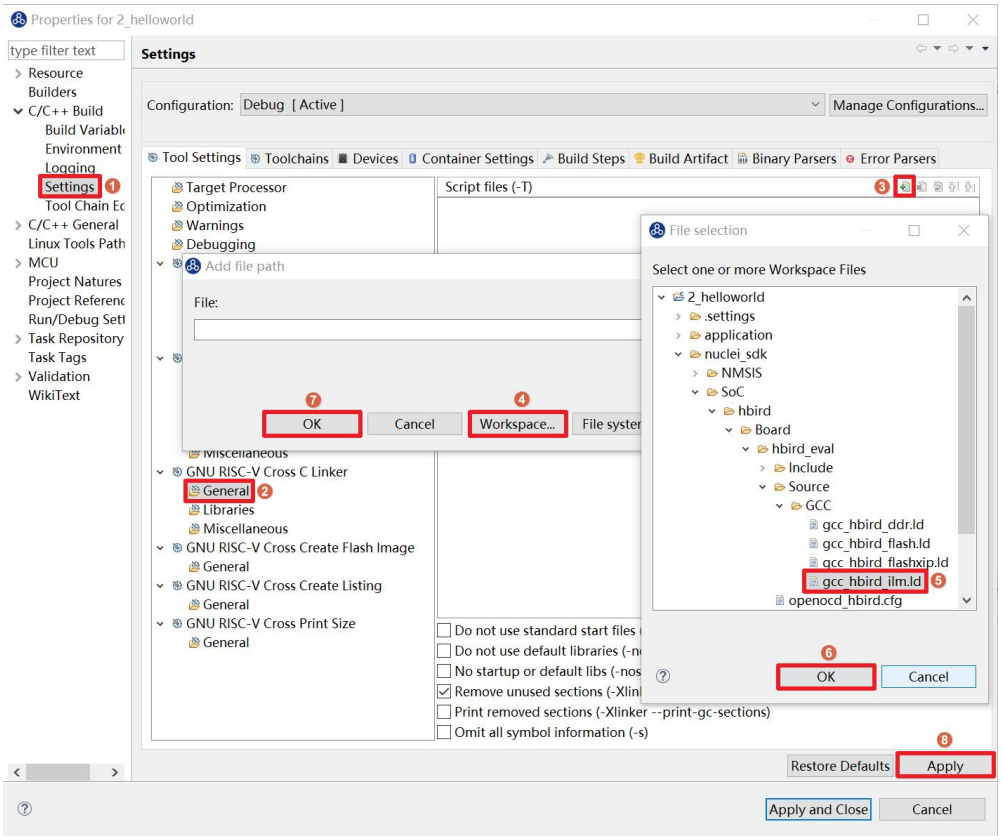


图 5-33 配置链接脚本

■如图 5-34 所示，按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。

●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。

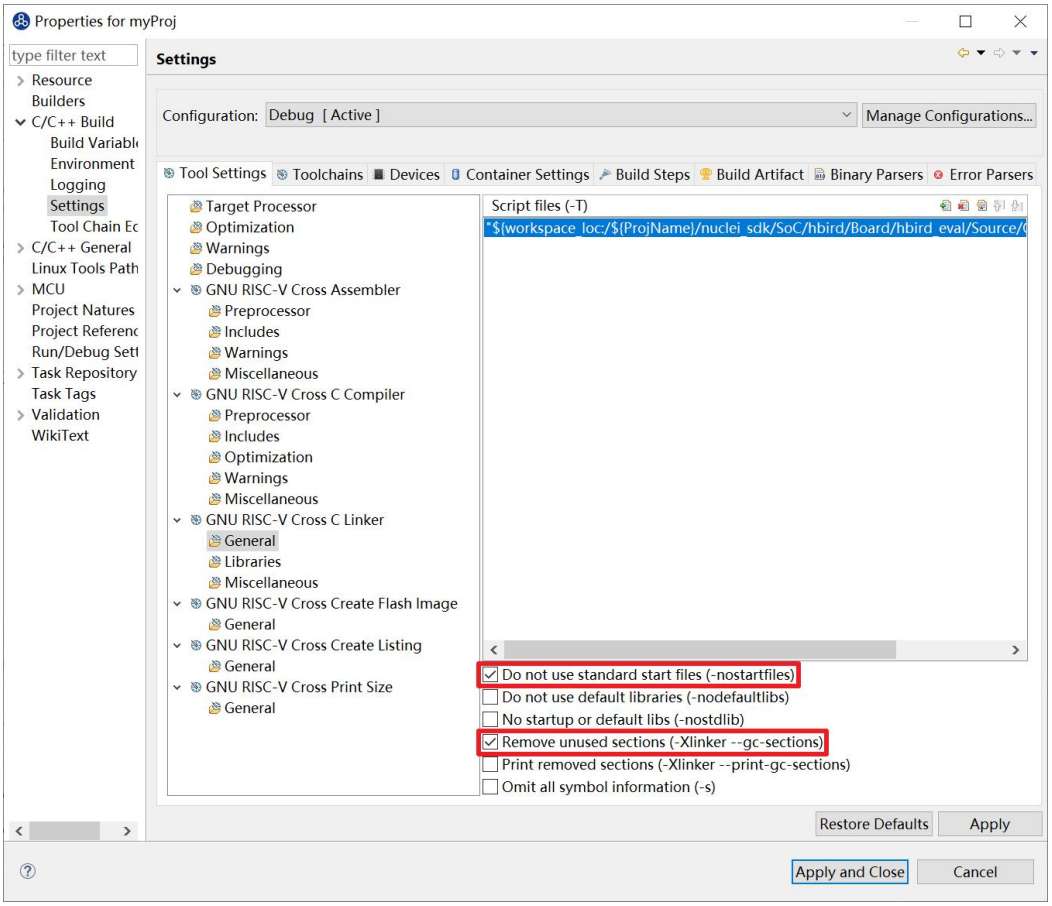


图 5-34 配置链接的 General 选项

■如图 5-35 所示，选中 GNU RISC-V Cross C++ Linker 的 Libraries，按照图所示填写编译配置选项。

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。

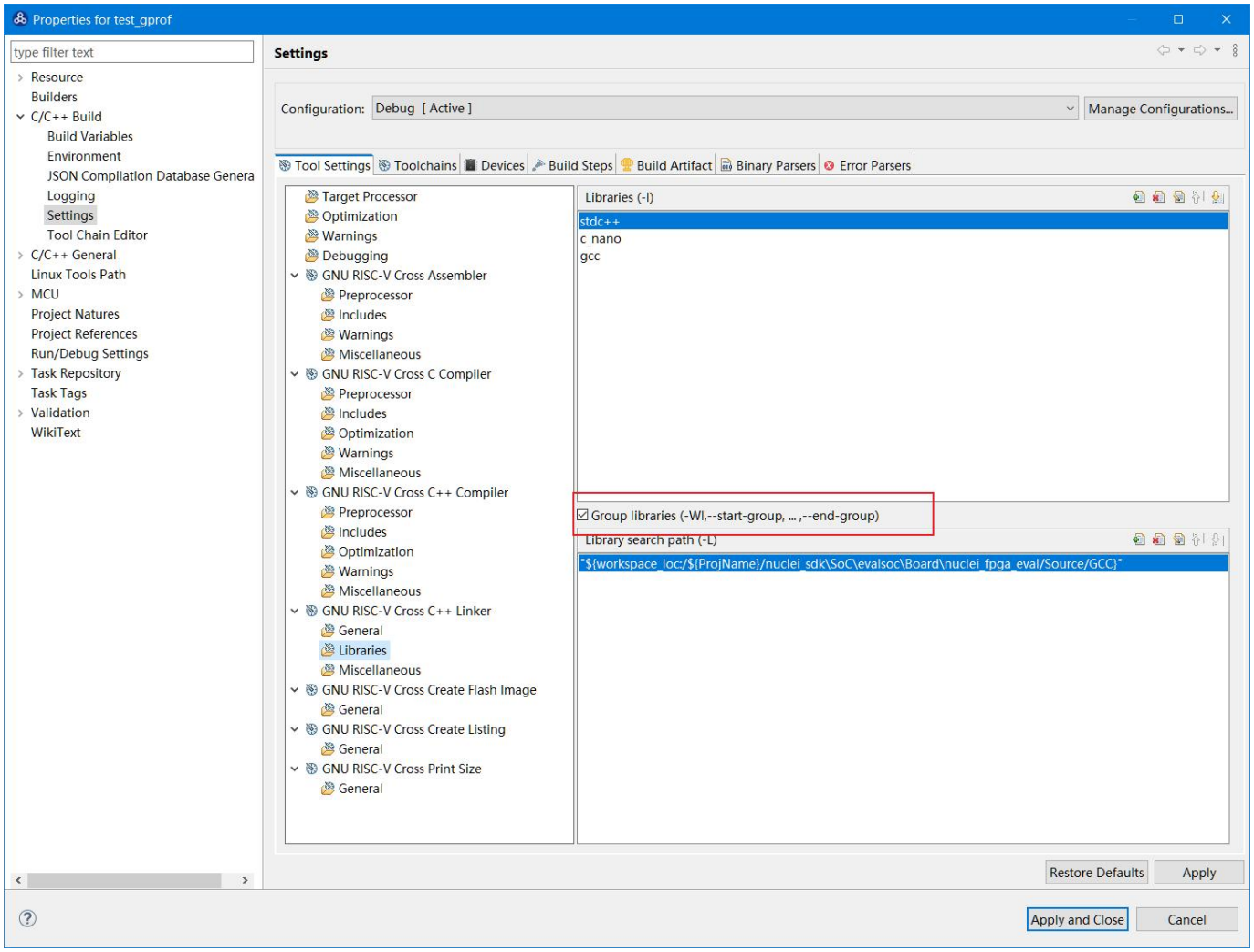


图 5- 35 配置 Libraaies 选项

■如图 5- 36 所示，选中 GNU RISC-V Cross C Linker 的 Miscellaneous，按照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。

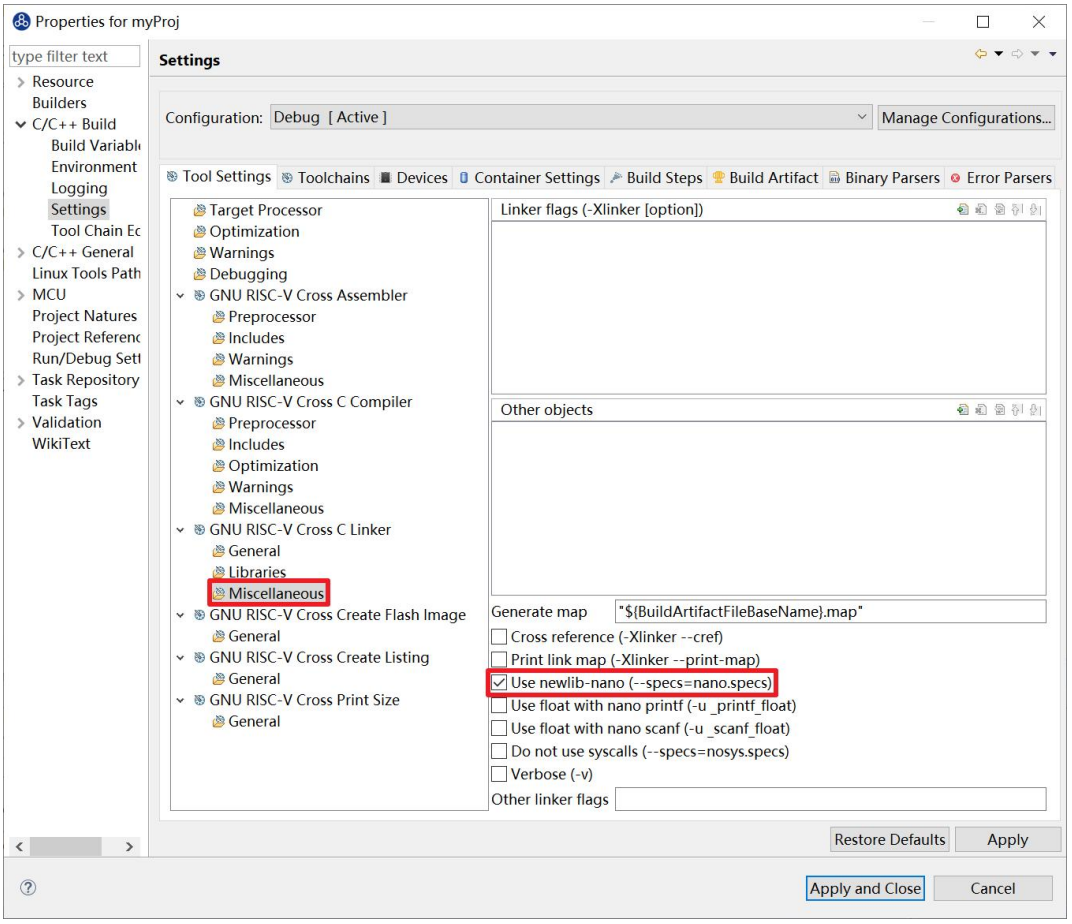


图 5- 36 配置链接的 Miscellaneous 选项

5.4.4.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。

■如图 5- 25 所示，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。

■如图 5-36 所示，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图 5-37 所示。

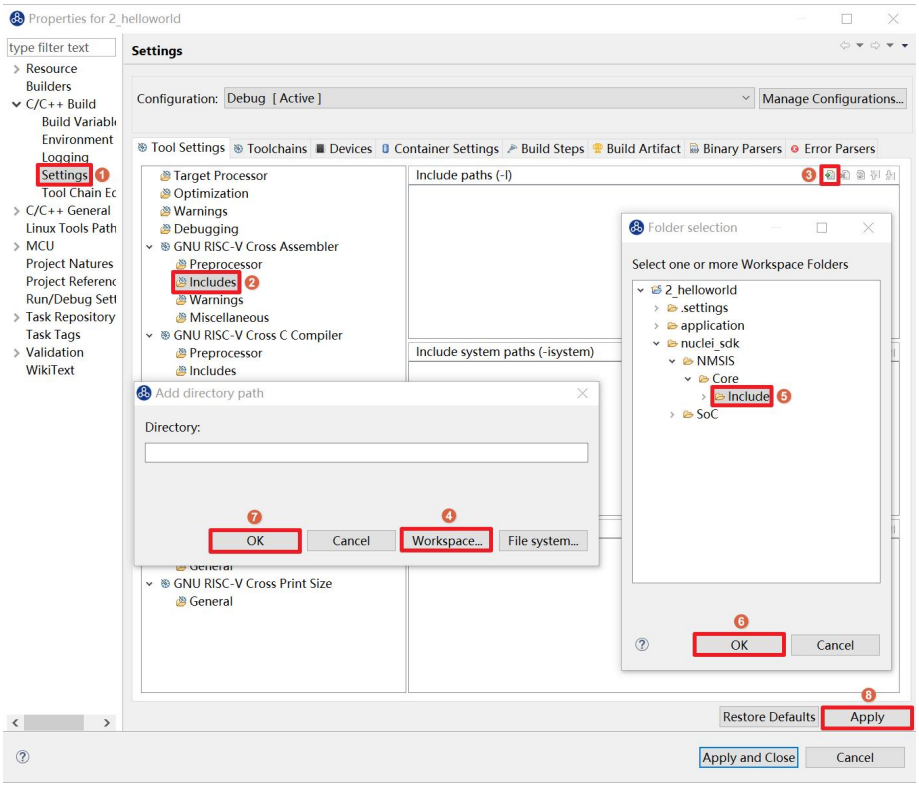


图 5- 37 设置 GNU RISC-V Cross C Assembler 的 Includes 栏目包含路径

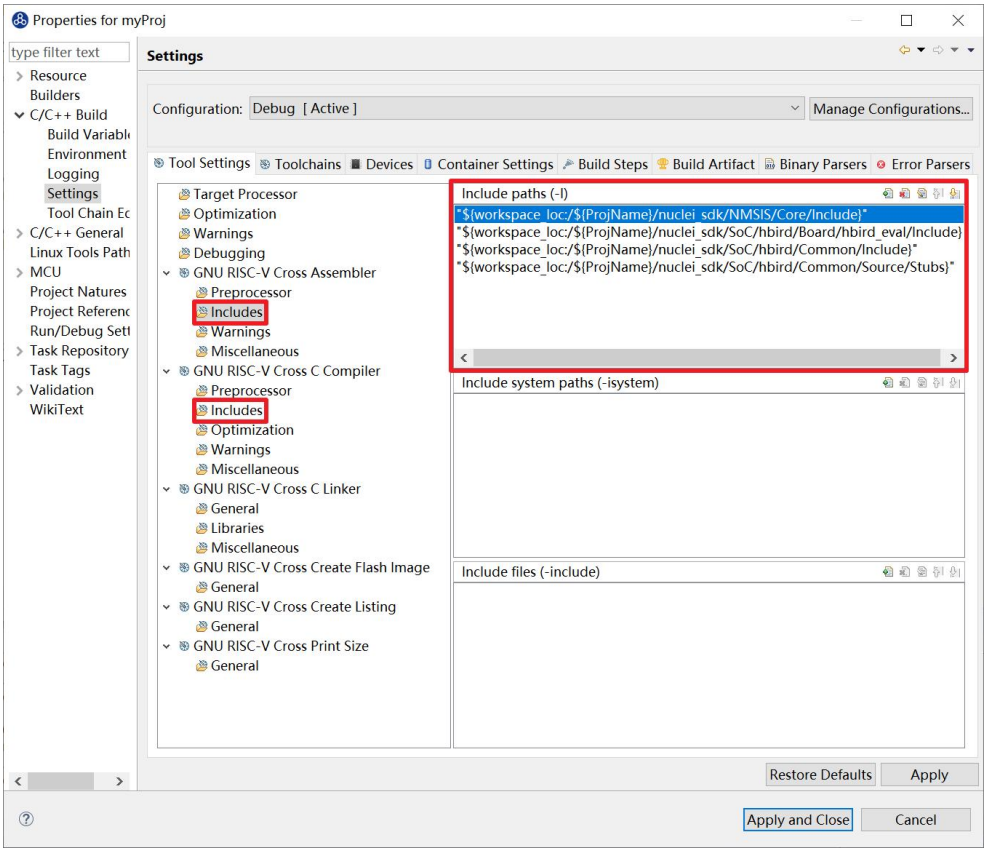


图 5- 38 设置 GNU RISC-V Cross C Compiler 的 Includes 栏目包含路径

5.5.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。

5.5.1.手动新建项目

在菜单栏中选择“File—> New —> Makefile Project with Existing Code”。如图 5- 38 所示。

如图 5- 39 所示，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。

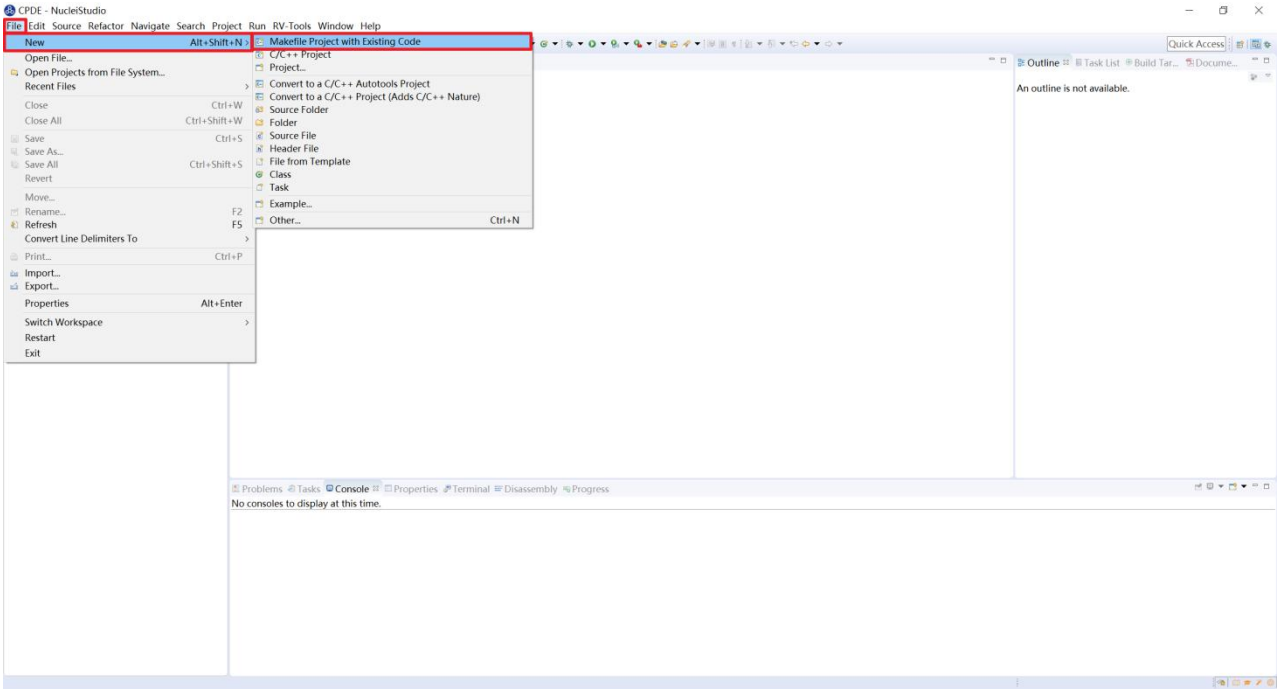


图 5- 39 新建基于 Makefile 的工程

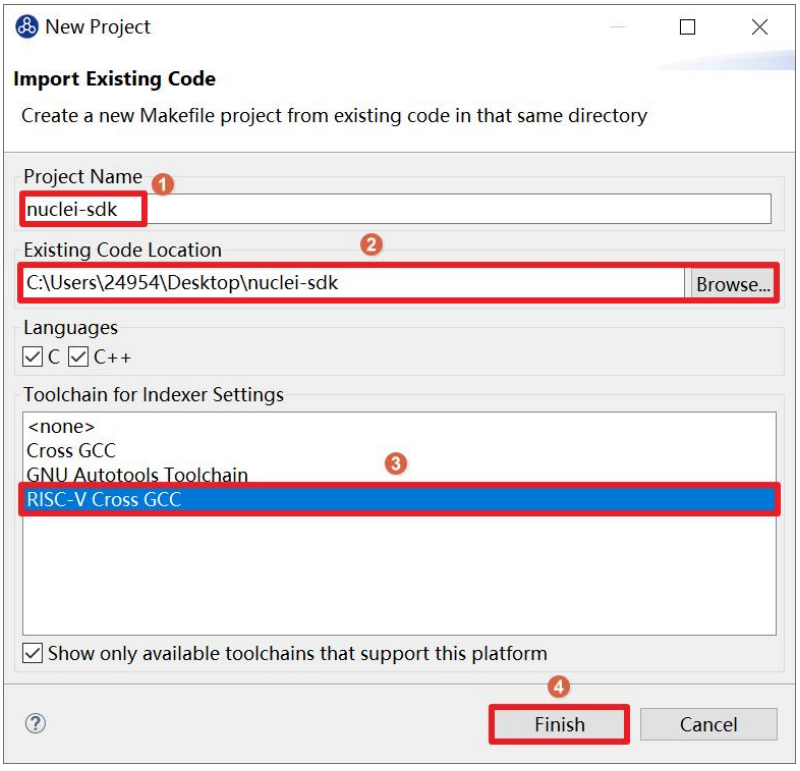


图 5-40 设置工程选项

5.5.2.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，如图 5-400，选择“C/C++ Build”，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。

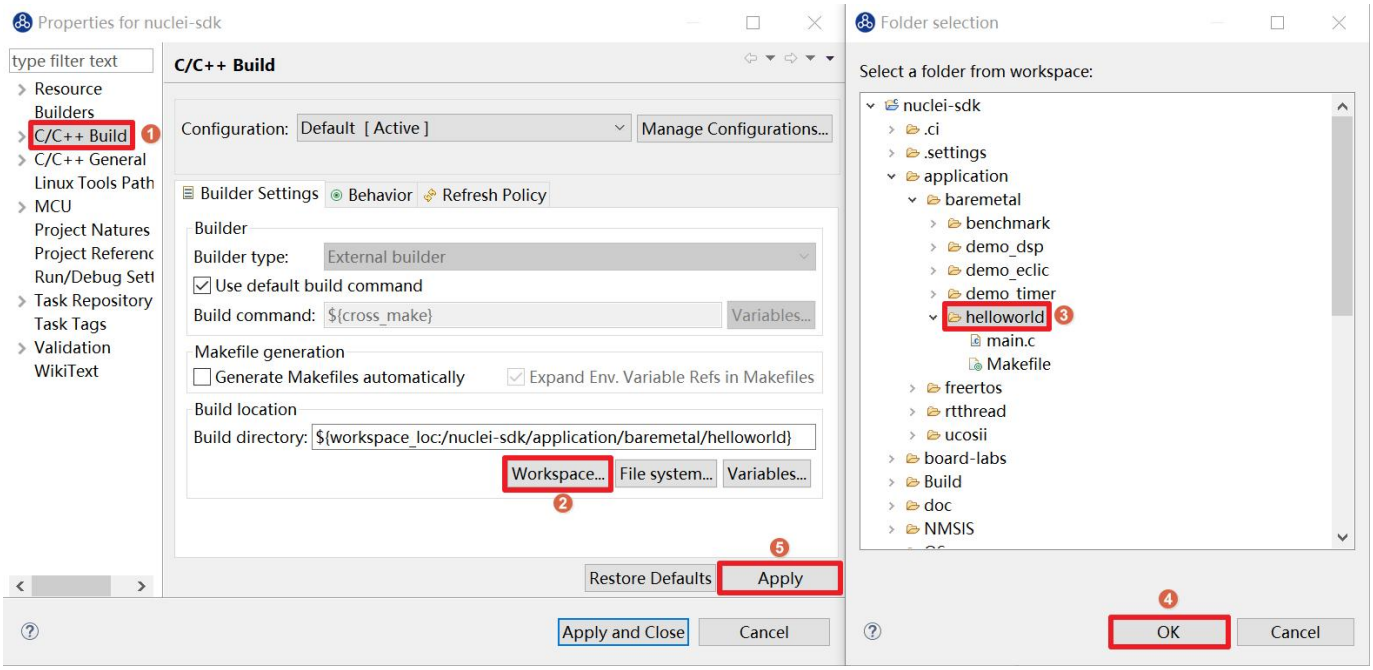


图 5-41 配置 Makefile 路径

如图 5-411，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。

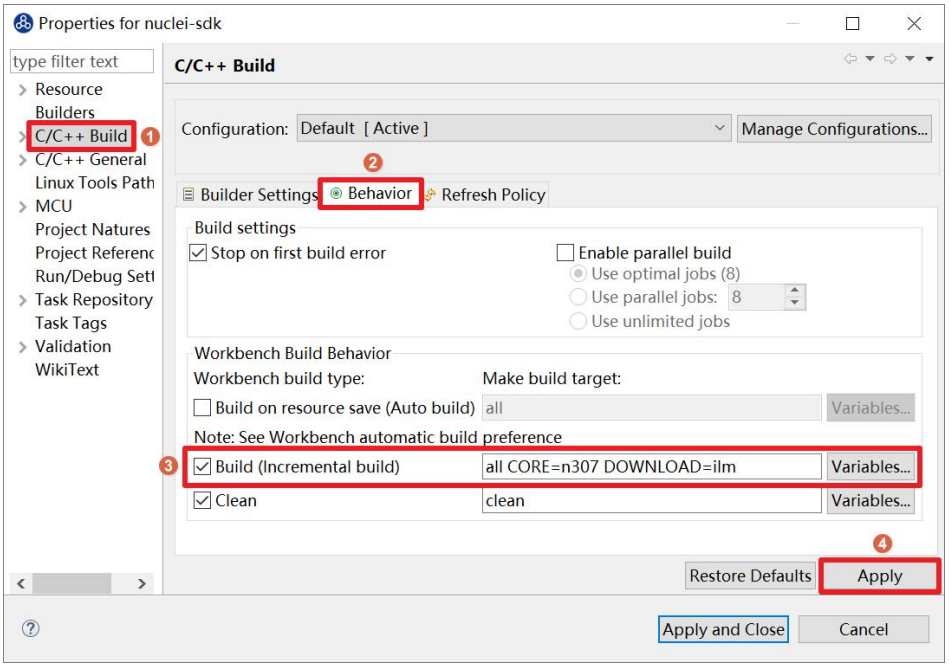


图 5-42 配置 Build 选项

在完成上述操作后，打开工具链配置页，如图 5-412，按图中配置，点击“Apply”保存修改。

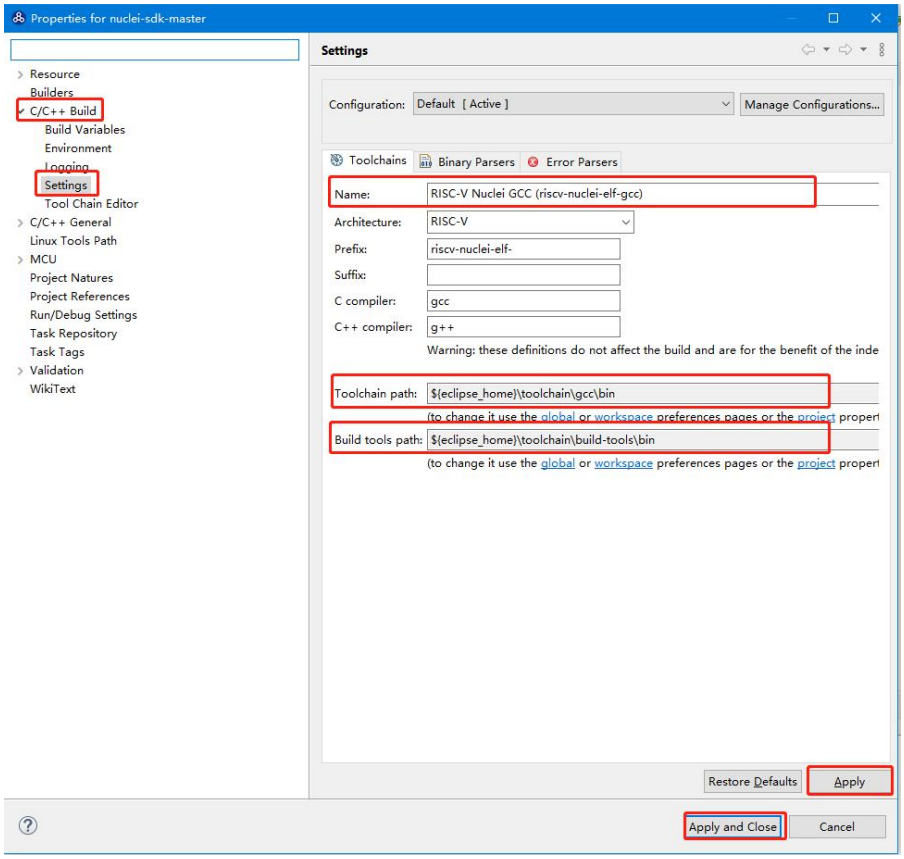


图 5-43 配置工具链路径

编译运行以及下载的详细内容请参考 5、6 章。

6.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。

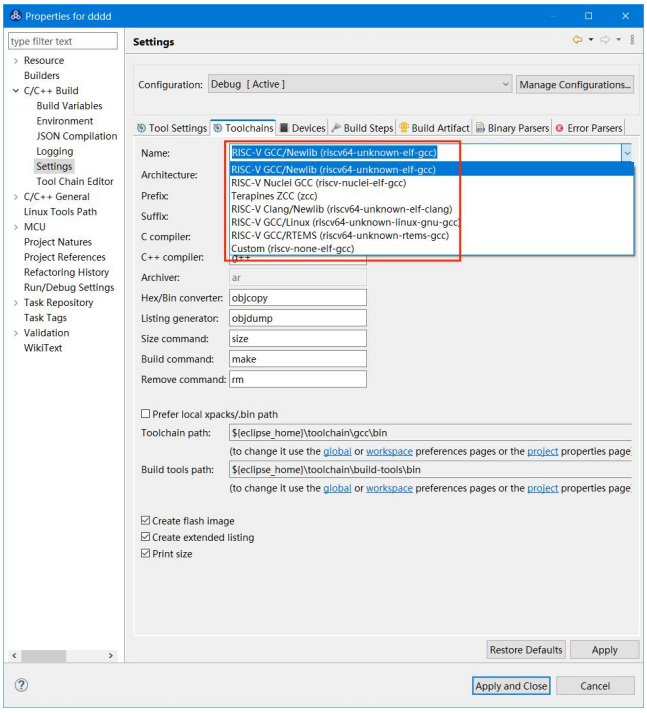


图 6- 1 NulceiStudio 支持多款编译器

■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。

6.1.编译工具

6.1.1.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 8.1 章节内容。

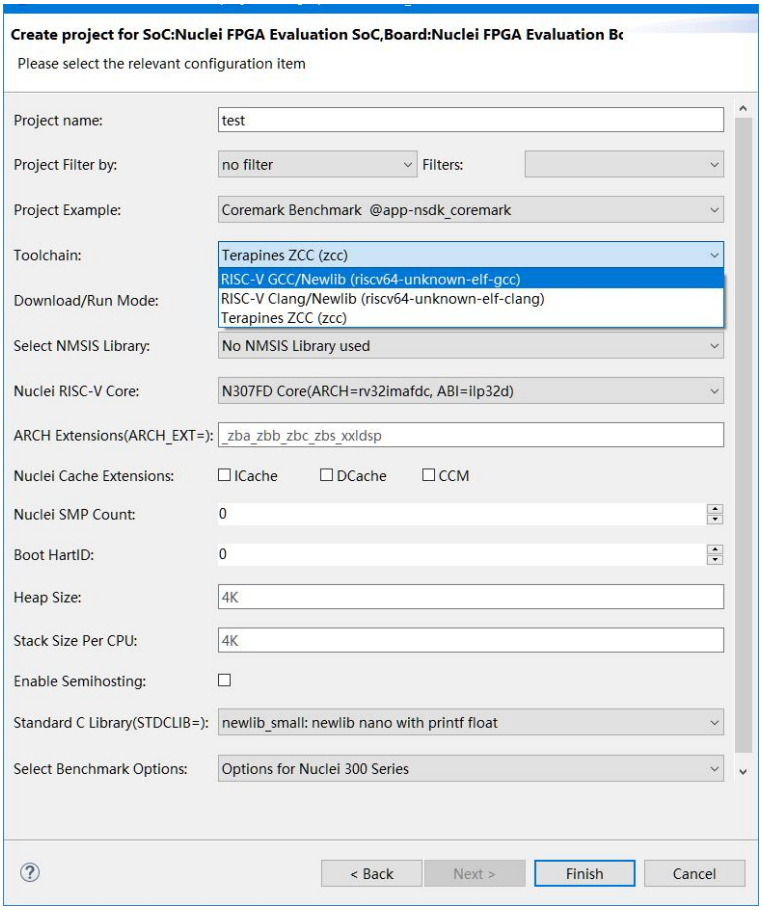


图 6-2 创建一个支持 GCC 13 编译的工程

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。

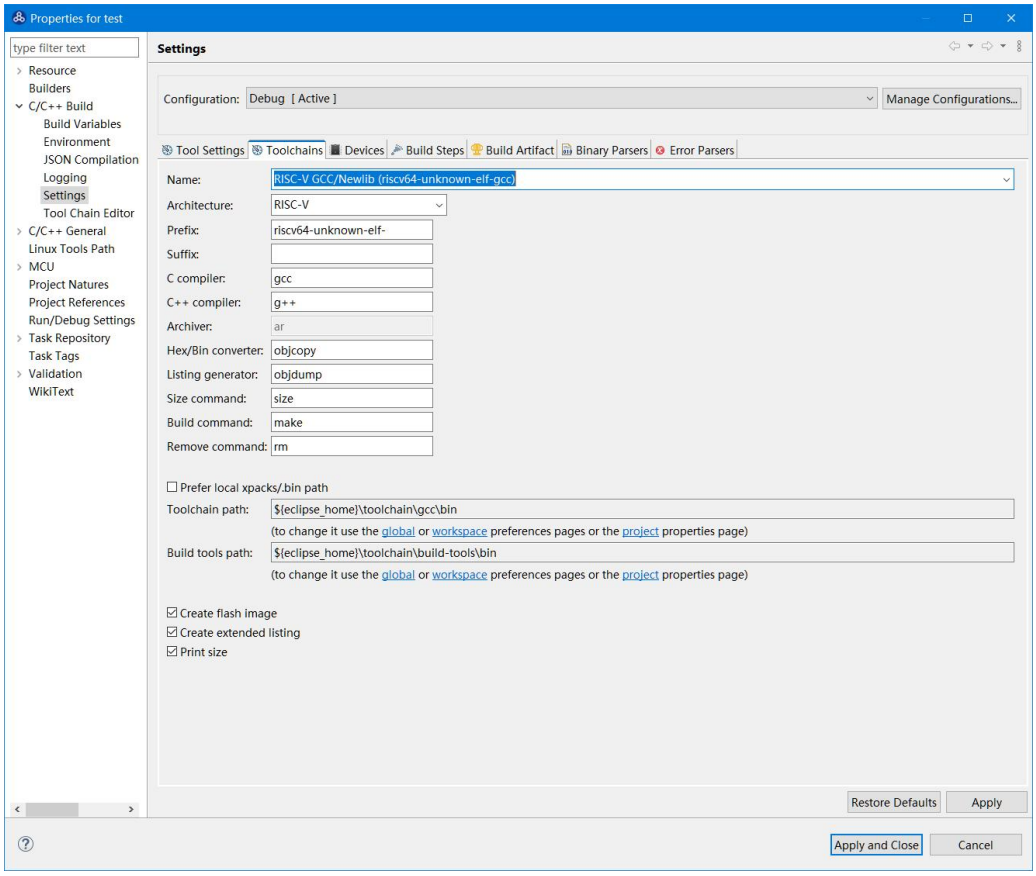


图 6-3 配置工具链路径

6.1.2.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。

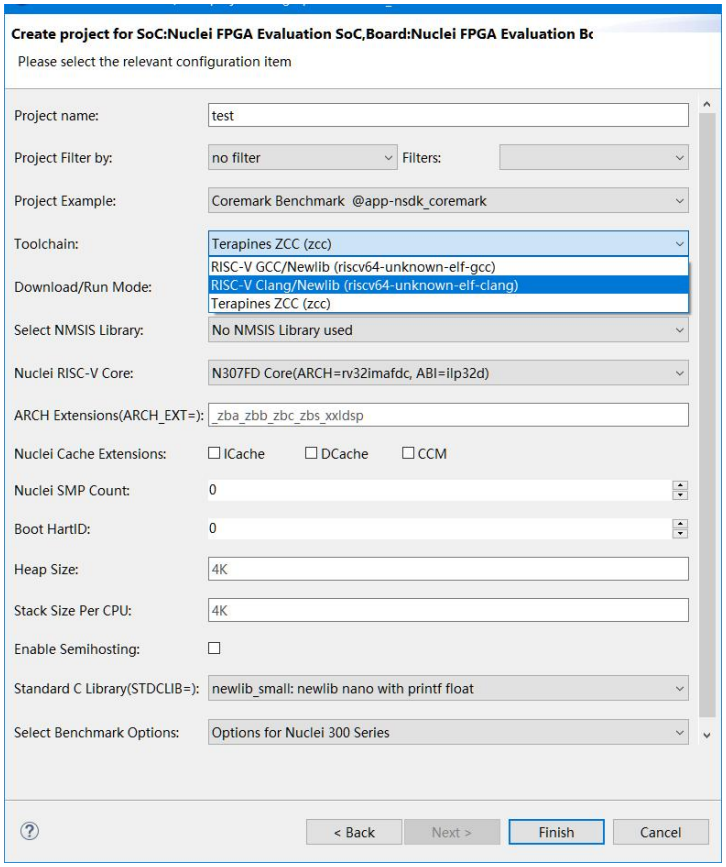


图 6-4 创建一个支持 Clang 17 编译的工程

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。

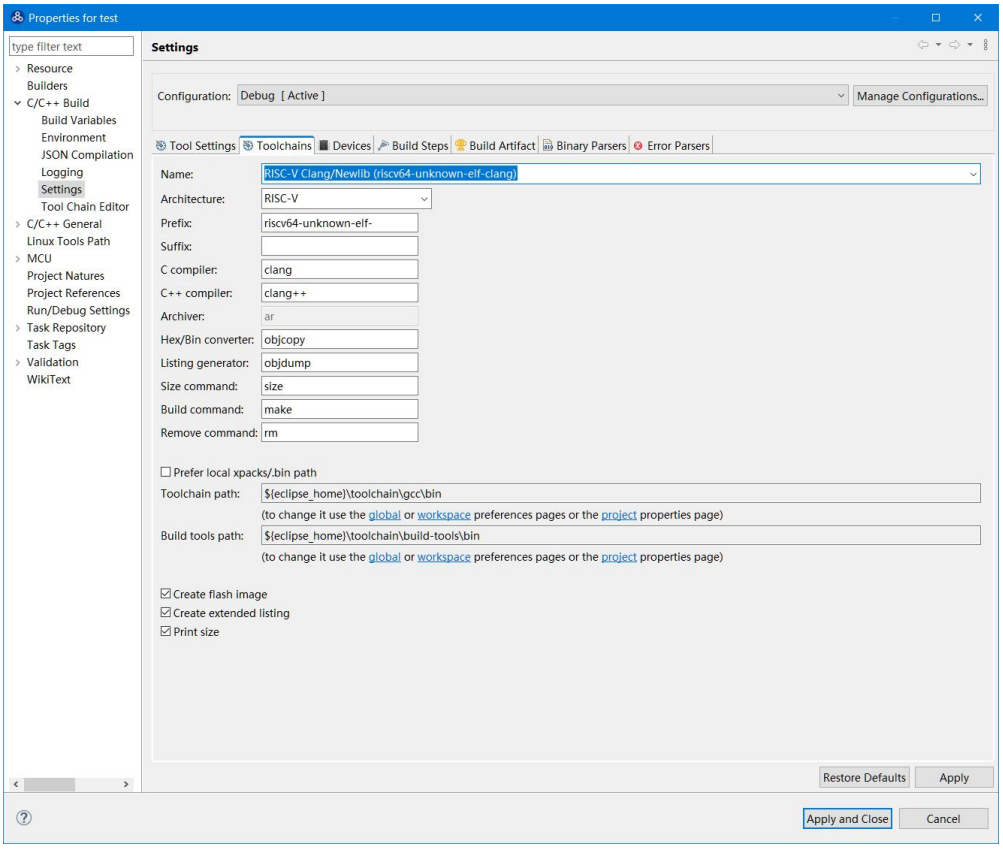


图 6-5 配置工具链路径

6.1.3.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

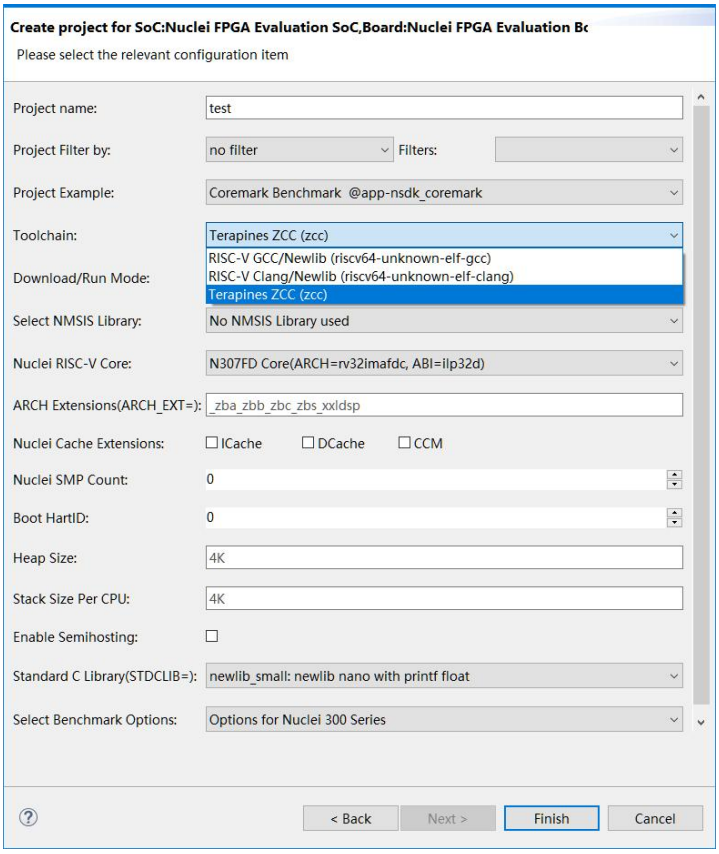


图 6-6 创建一个支持 ZCC 编译的工程

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。

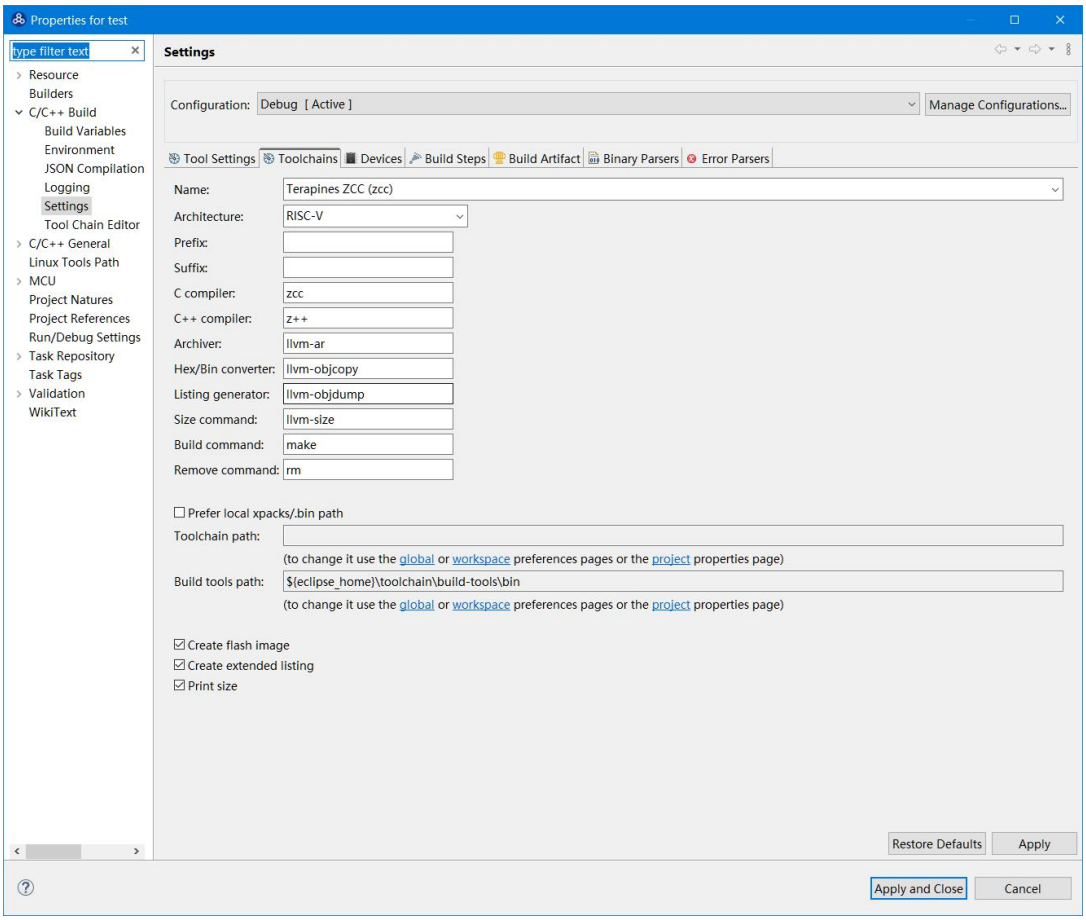


图 6-7 配置工具链路径

6.2.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。**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。**详细使用步骤如下:

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，如图 6-2，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，如图 6-3，右击打开右键菜单，选择 SDK Configuration Tools。如图 6-4 为工程设置工具弹窗，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)，**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)，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，之后如图 6-5，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。

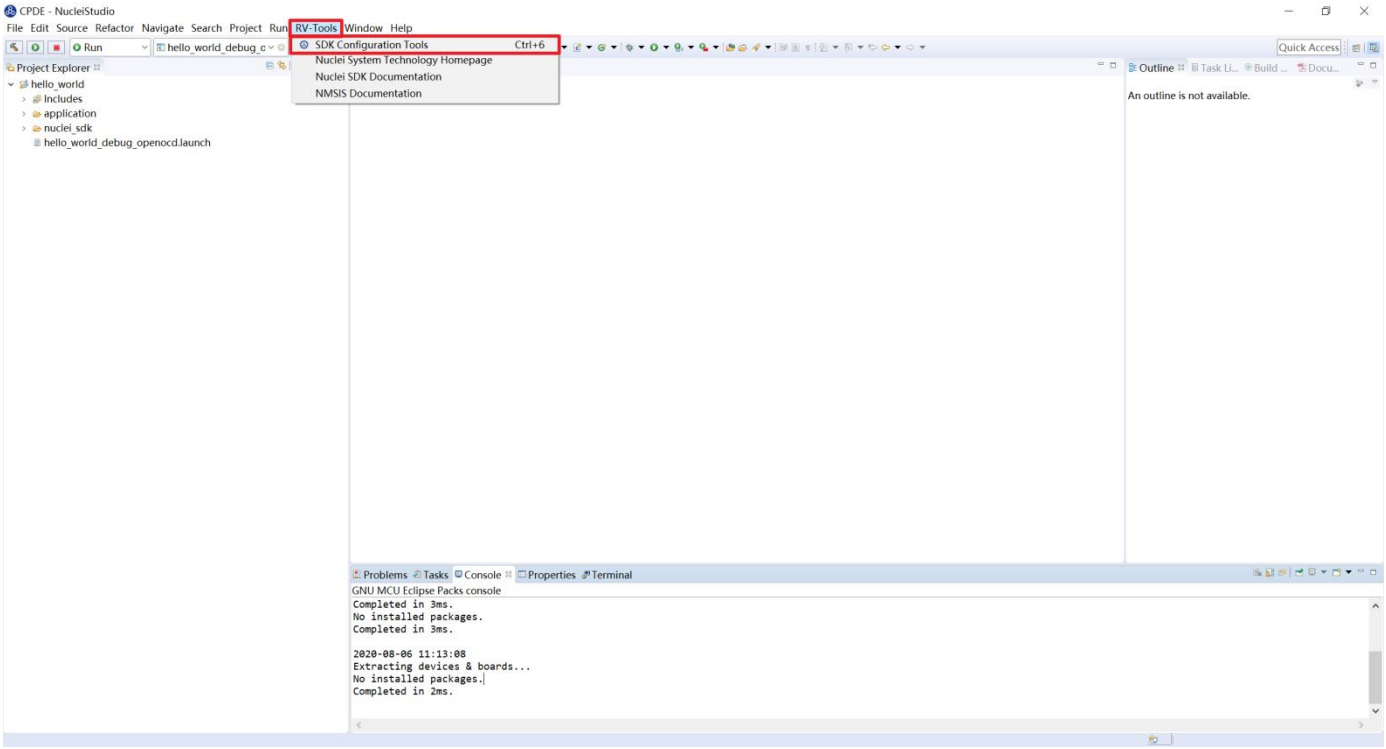


图 6-8 打开修改编译选项弹窗 1

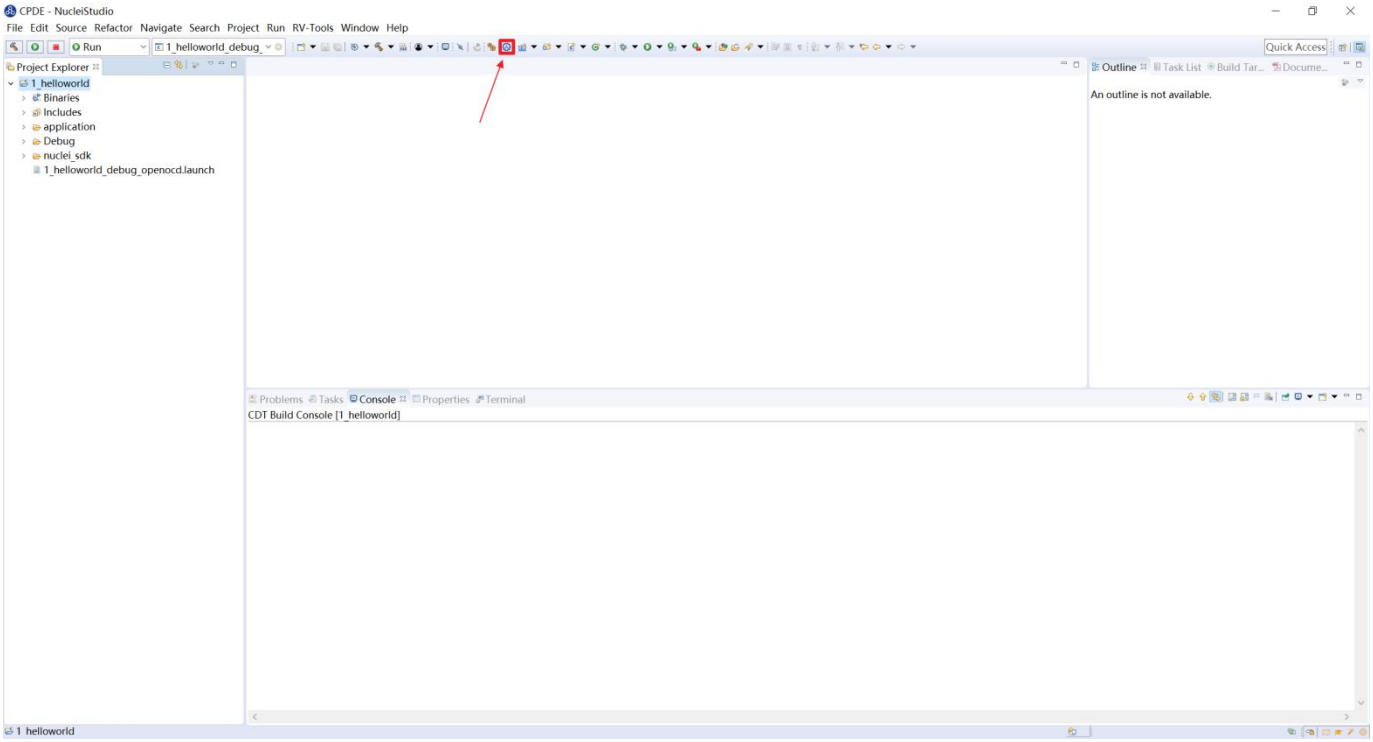


图 6-9 打开修改编译选项弹窗 2

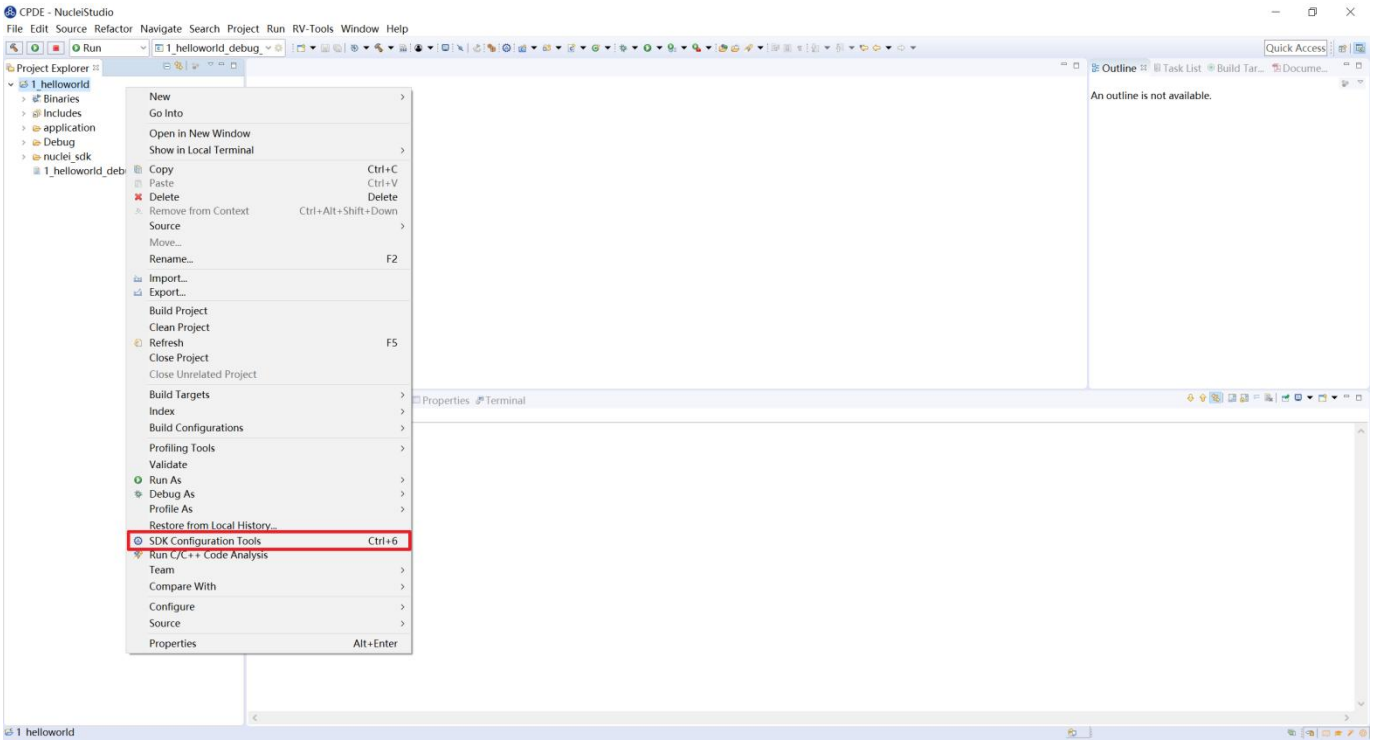


图 6- 10 打开修改编译选项弹窗 3

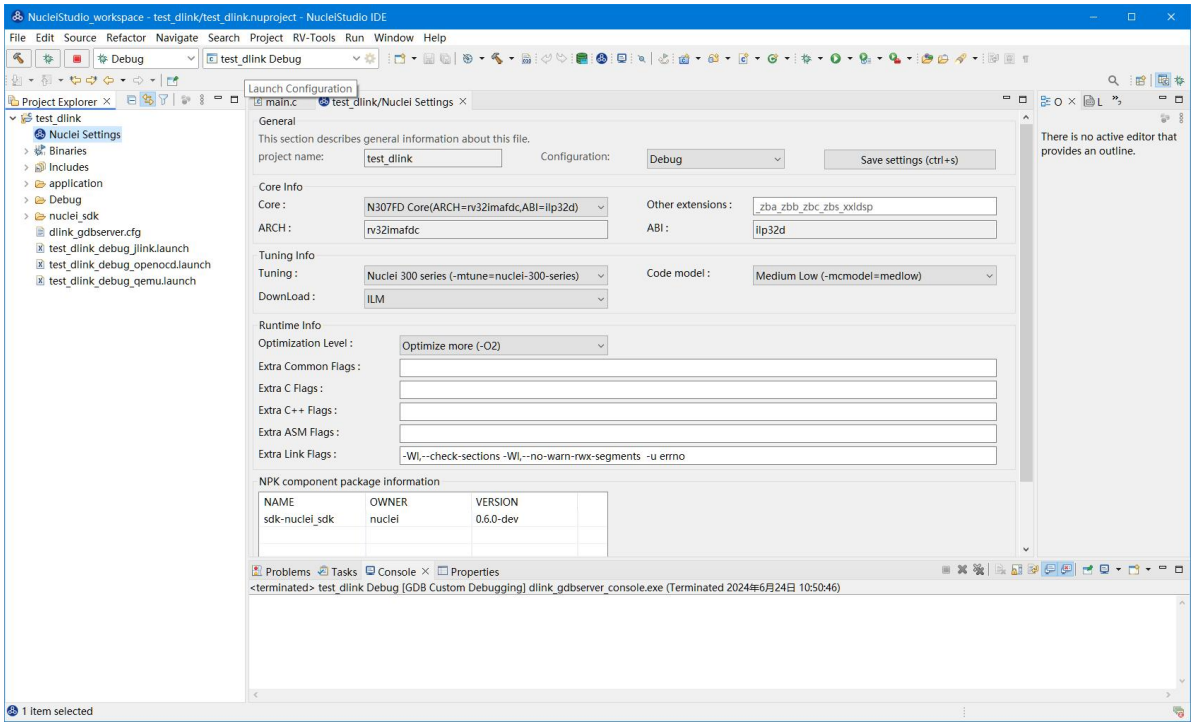


图 6- 11 一键修改编译选项

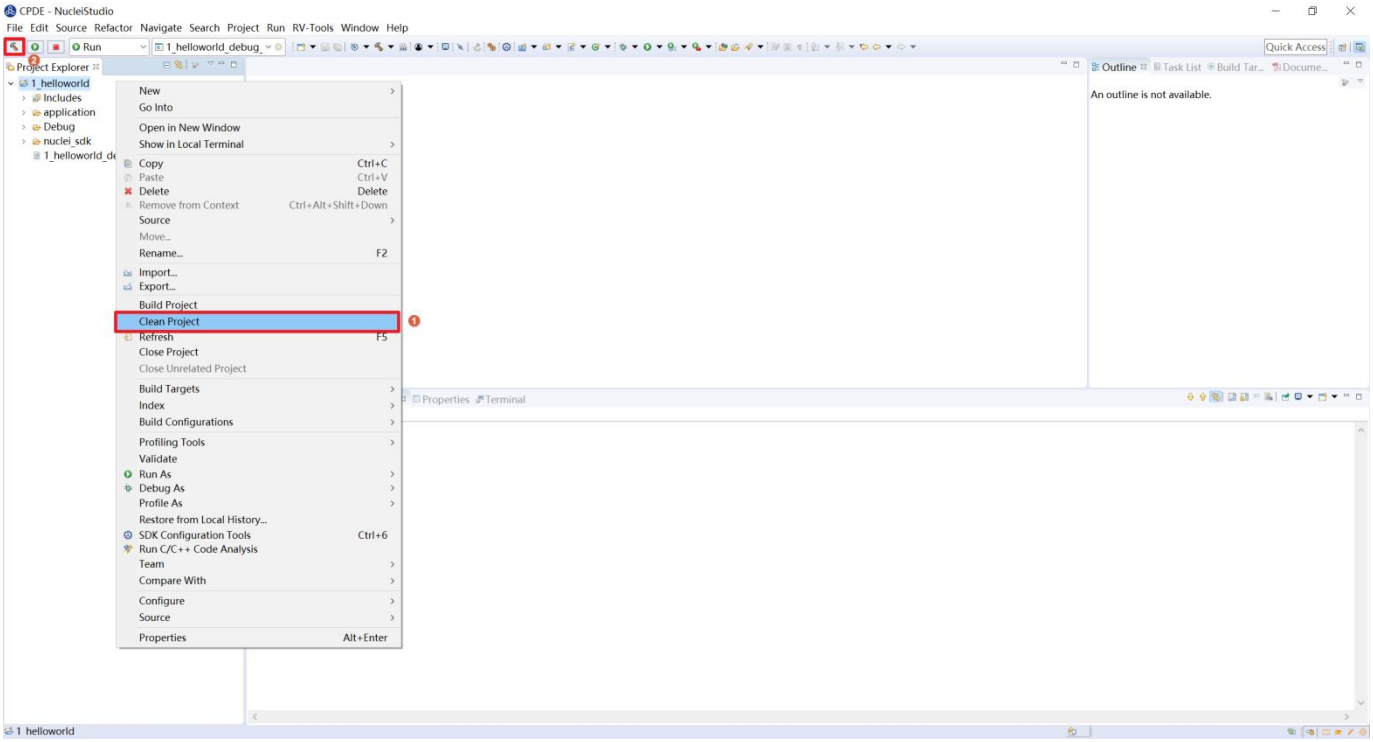


图 6-12 清理工程并重新编译工程

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，如图 7-11，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。

6.3.Nuclei Studio 中编译 Hello World 项目

在 Nuclei Studio 中编译的步骤如下。

- 为了保险起见，建议先将项目清理一下。如图 6-6 所示，在 Project Explorer 栏中选中 hello_world 项目，单击鼠标右键，选择“Clean Project”。

- 如图 6-7 所示，单击菜单上的锤子按钮，开始对项目进行编译。如果编译成功，则显示如图 6-7 所示，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。

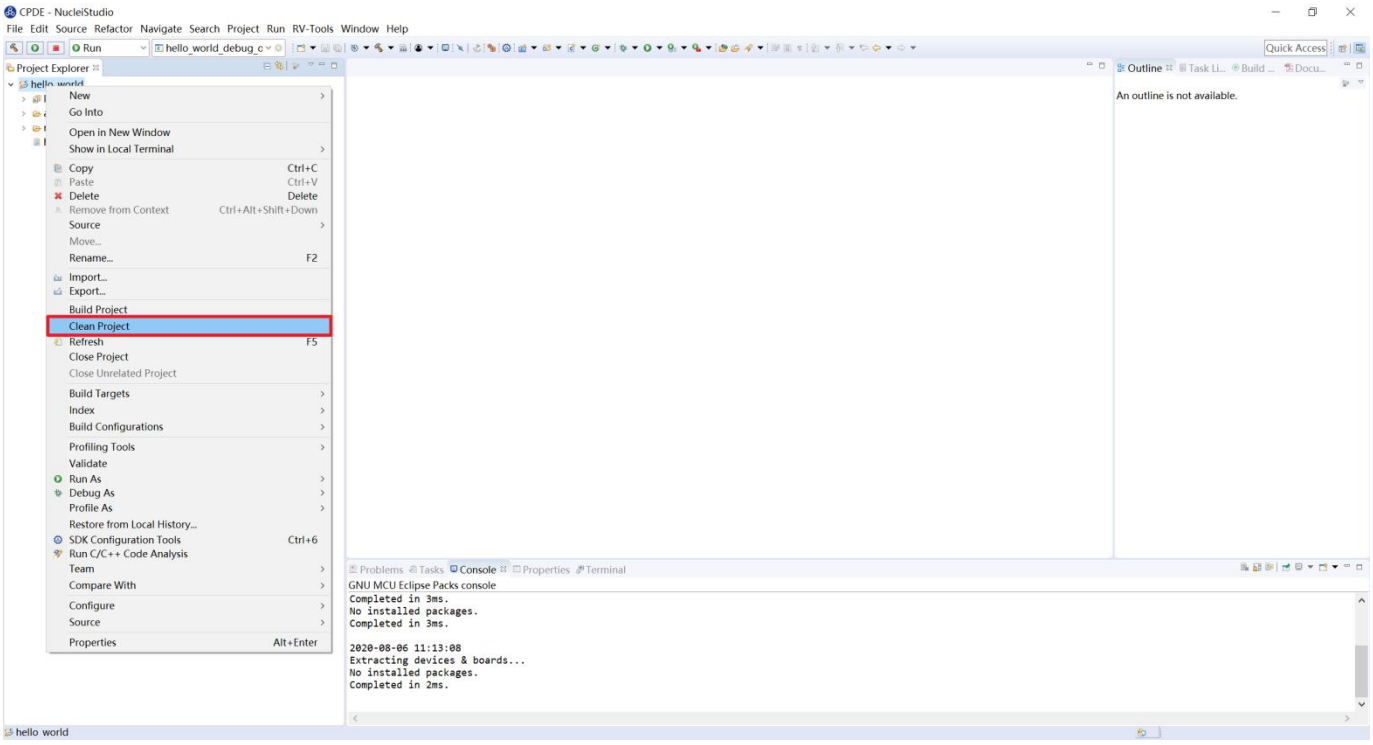


图 6- 13 对 hello_world 项目单击右键选择“Clean Project”

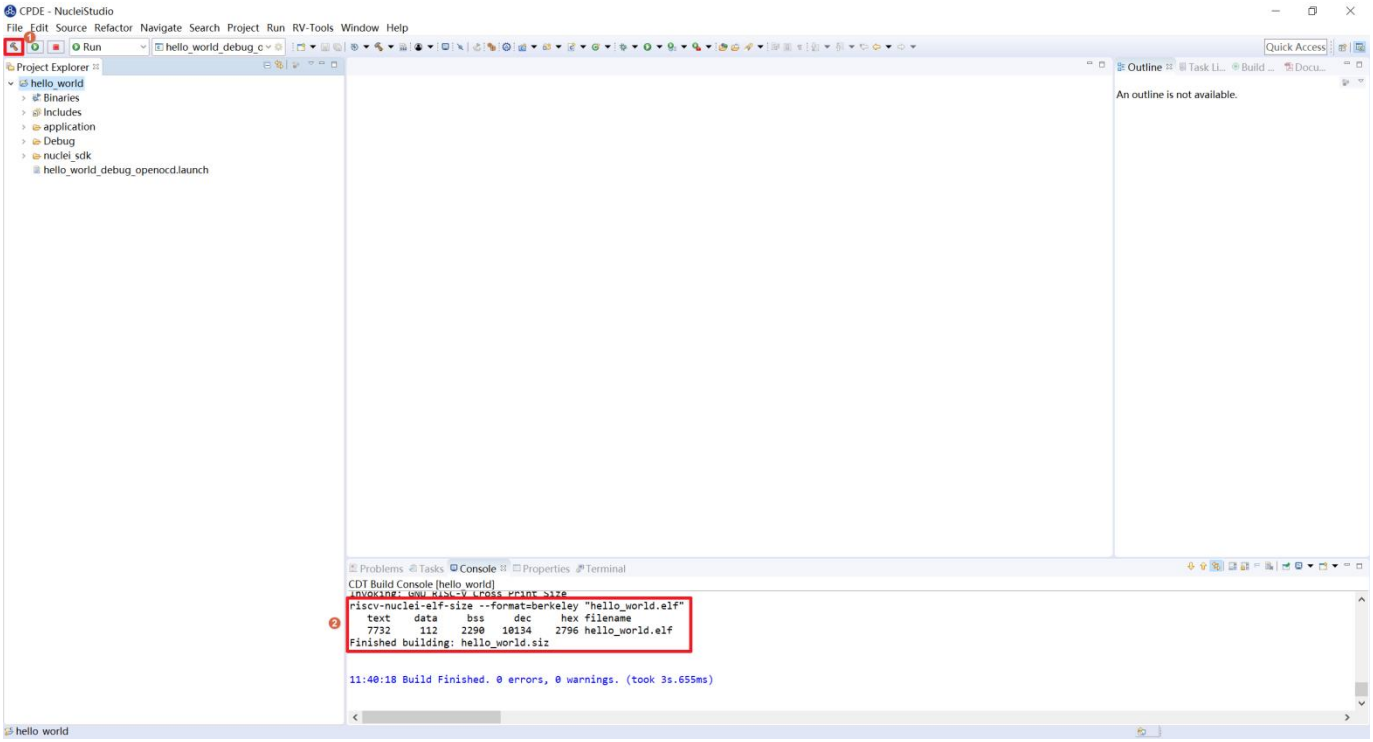


图 6-14 单击锤子图标对 hello_world 项目进行编译

编译成功后可以看到增加了 Debug 文件夹，如图 6-8，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。

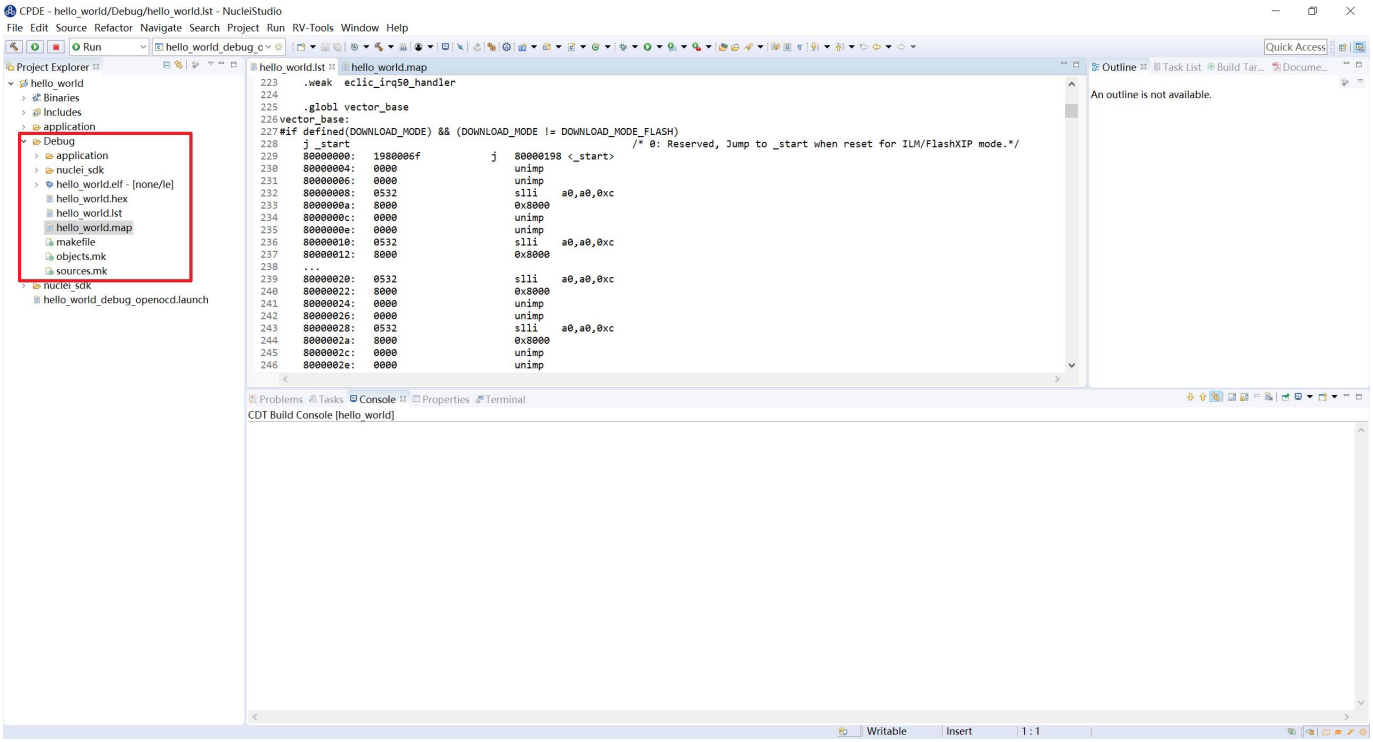


图 6- 15 查看编译后的生成文件

7. 调试运行工程

7.1.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。

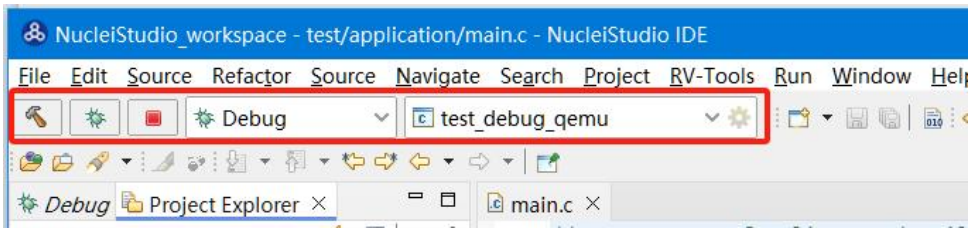


图 7- 1 Launch Bar

■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。

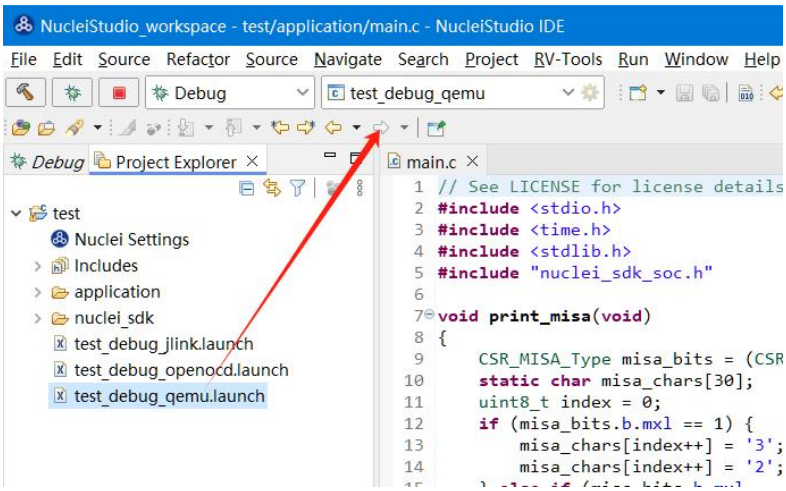


图 7- 2 点击*.launch 文件切换调试模式

■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。

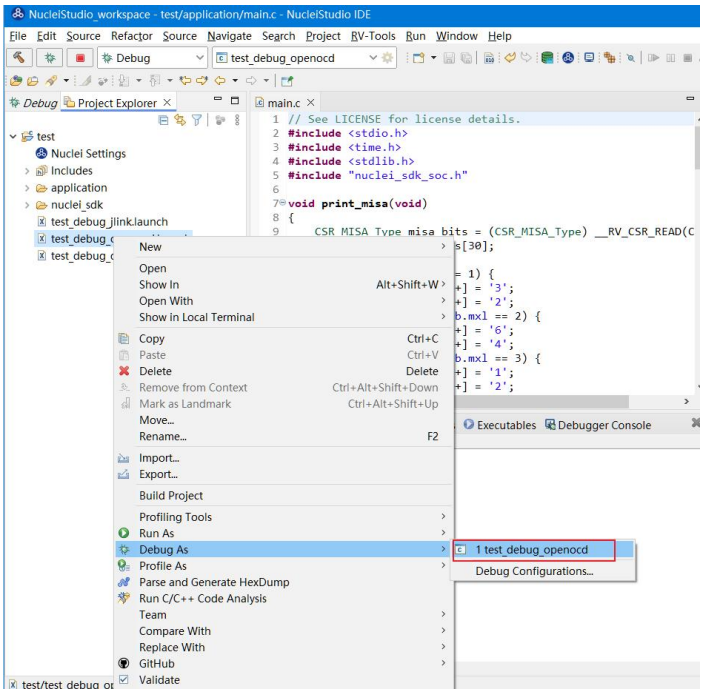


图 7-3 Run as/Debug as 切换调试模式

■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。

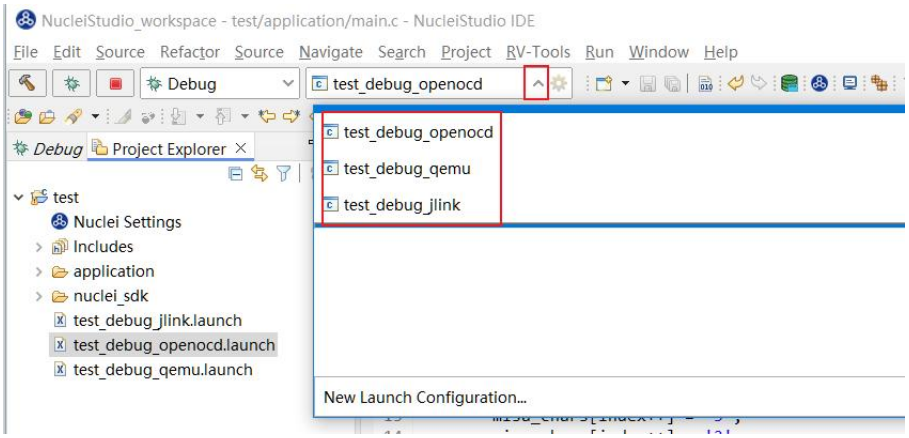


图 7-4 Launch Bar 切换调试模式

7.2.使用蜂鸟调试器结合 OpenOCD 调试运行项目

7.2.1.安装蜂鸟调试器驱动

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。

7.2.1.1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。注意：如果是第一次使用蜂鸟调试器，为了使得主机 PC 的 Windows 系统能够识别蜂鸟调试器的 USB，需要安装驱动，双击如图 3-2 所示的 Nuclei Studio_IDE 文件包中的 HBird-Driver.exe 即可完成此驱动的安装。

- 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。

- 如图 7-1 所示，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。

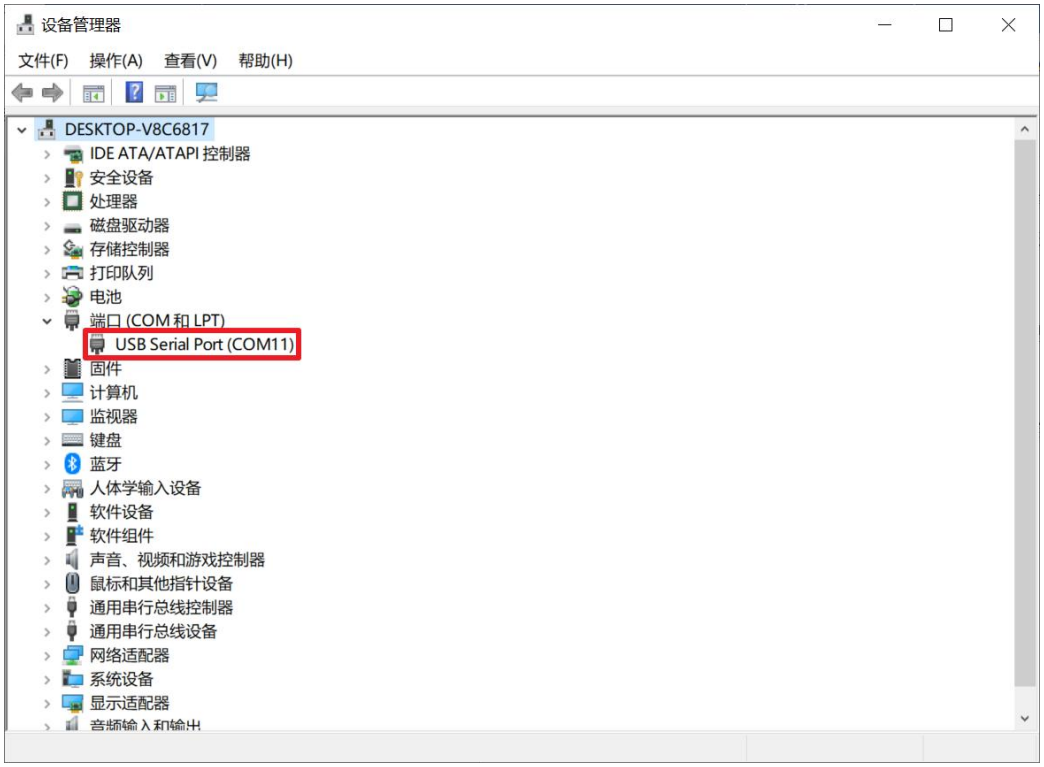


图 7-5 在设备管理器中查询 COM 串口号

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。

7.2.1.2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，如图 7-2，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。

图 7-6 开发板连接至虚拟机中

■9：查看 gcc 的依赖是否完整

cd Nuclei Studio/toolchain/gcc/bin/

ldd ./riscv-nuclei-elf-gdb

```
linux-vdso.so.1 (0x00007fffc83f3000)
libtinfo.so.5 => not found
libncursesw.so.5 => not found
libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f4df6d08000)
libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007f4df6c21000)
libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f4df6c1c000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f4df69f2000)
/lib64/ld-linux-x86-64.so.2 (0x00007f4df6d1e000)
```

图 7-7 查看 gcc 的依赖是否完整

如图有依赖需要安装，可以执行 sudo apt install libncursesw5 libtinfo5 进行安装

7.2.2.Debug Configuration

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

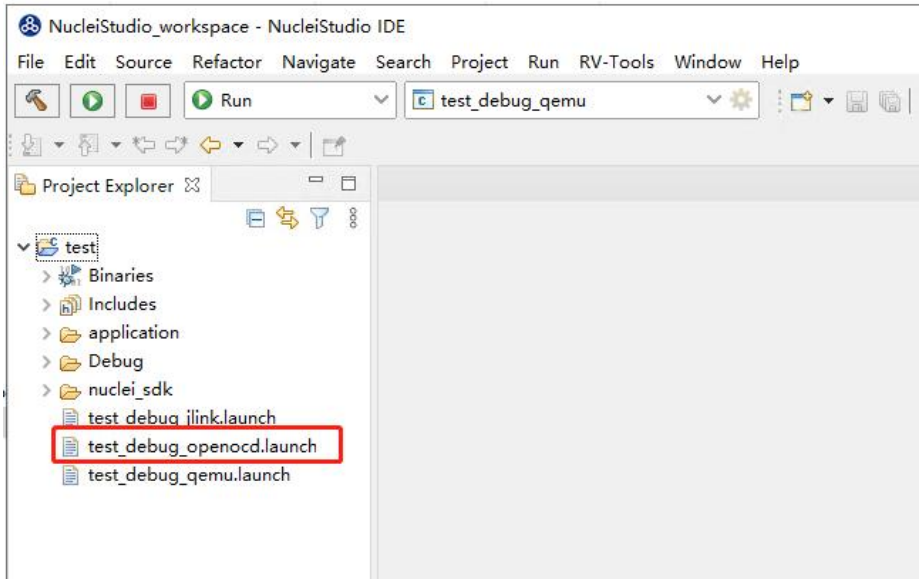


图 7-8 Launch 文件

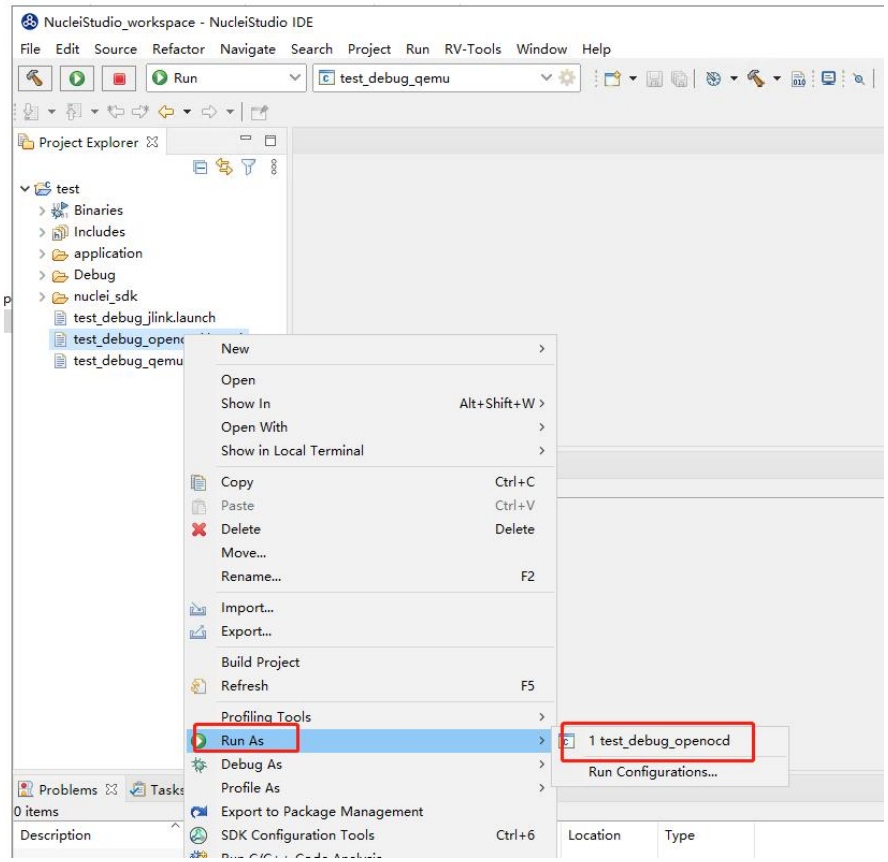


图 7-9 Run as/Debug as

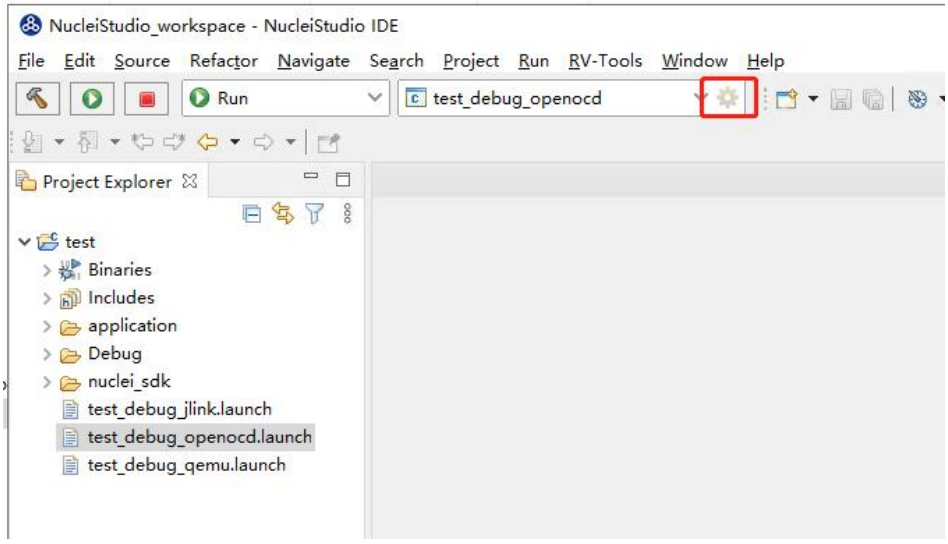


图 7-10 在 Launch Bar 中查看 Debug Configurations

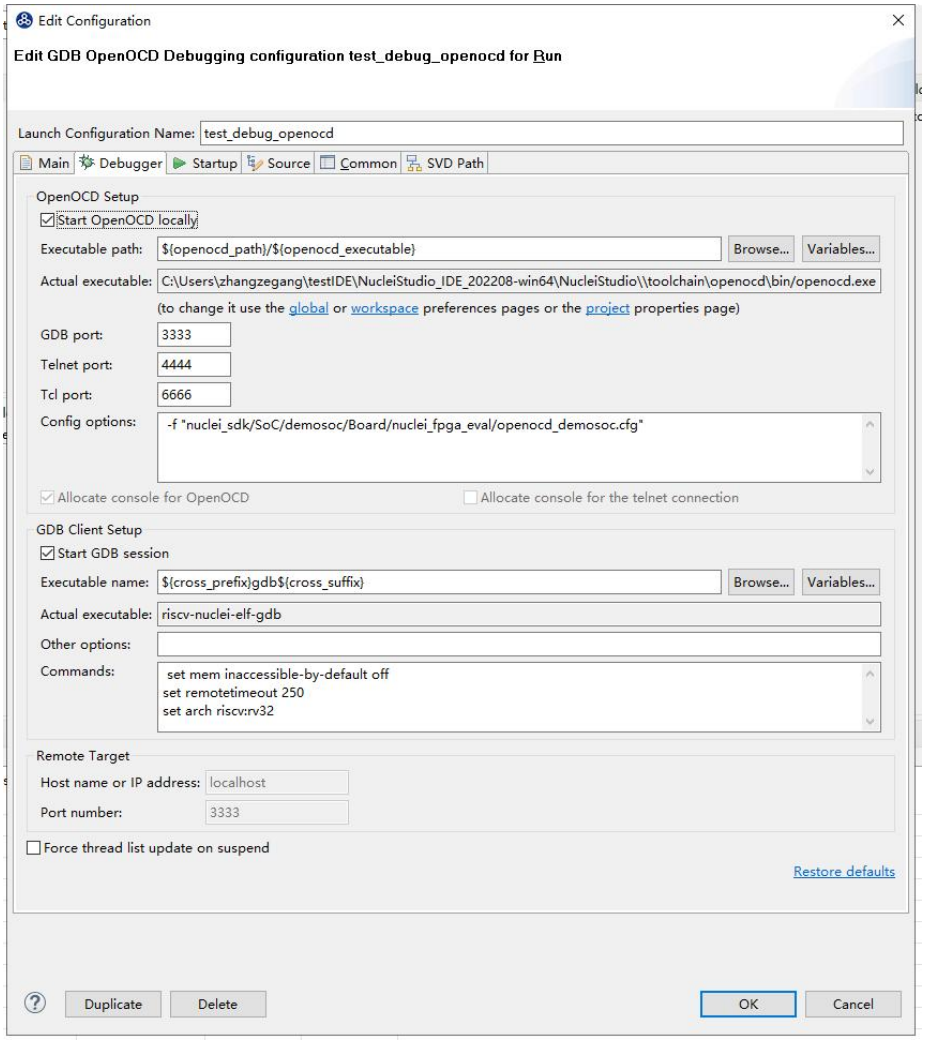


图 7-11 Debug Configurations

7.2.2.2 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。

- 如图 7-8 所示，在菜单栏中选择“Run—>Debug Configurations”。
- 如图 7-9 所示，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”，如图 7-10 所示。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。

■如图 7-11 所示，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*

■如图 7-12 所示，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。

■如图 7-12，完成后点击右下方“Apply”保存设置。

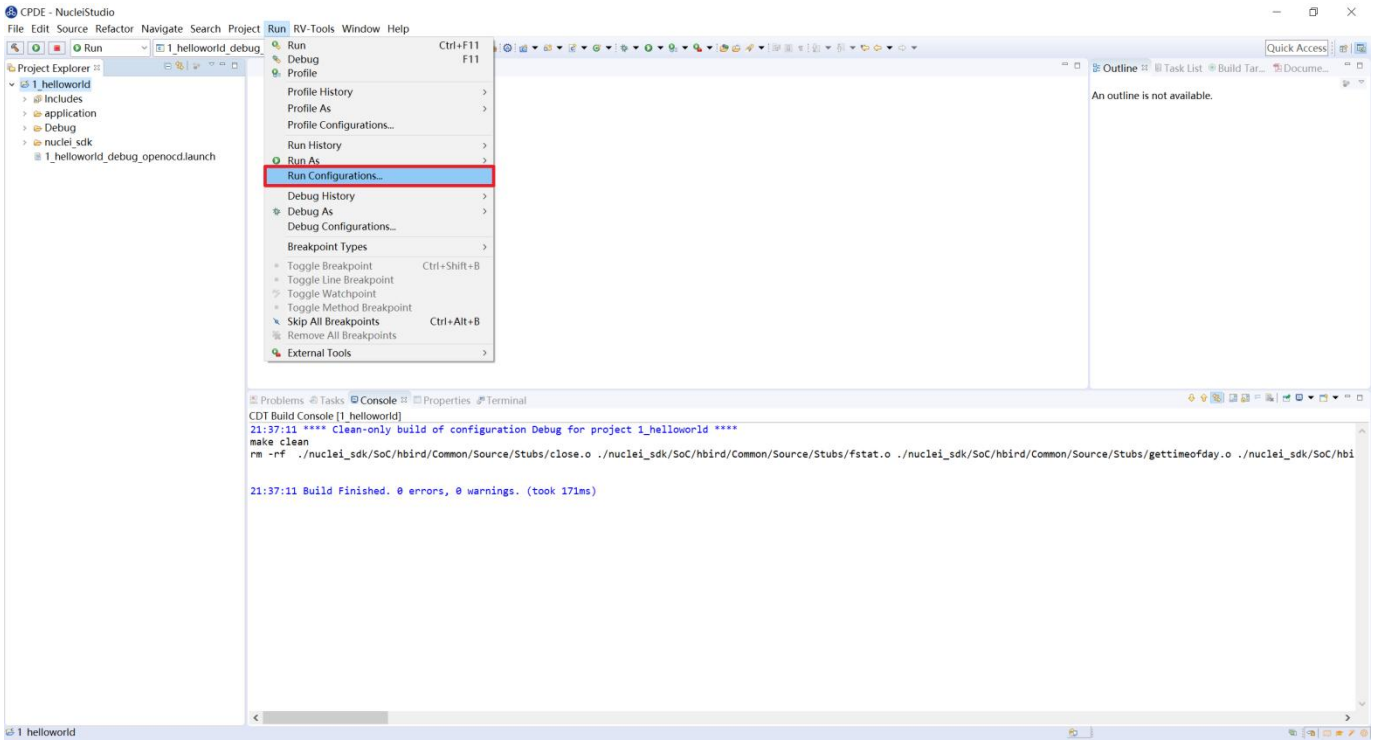


图 7-12 单击“Run Configuration”进行下载

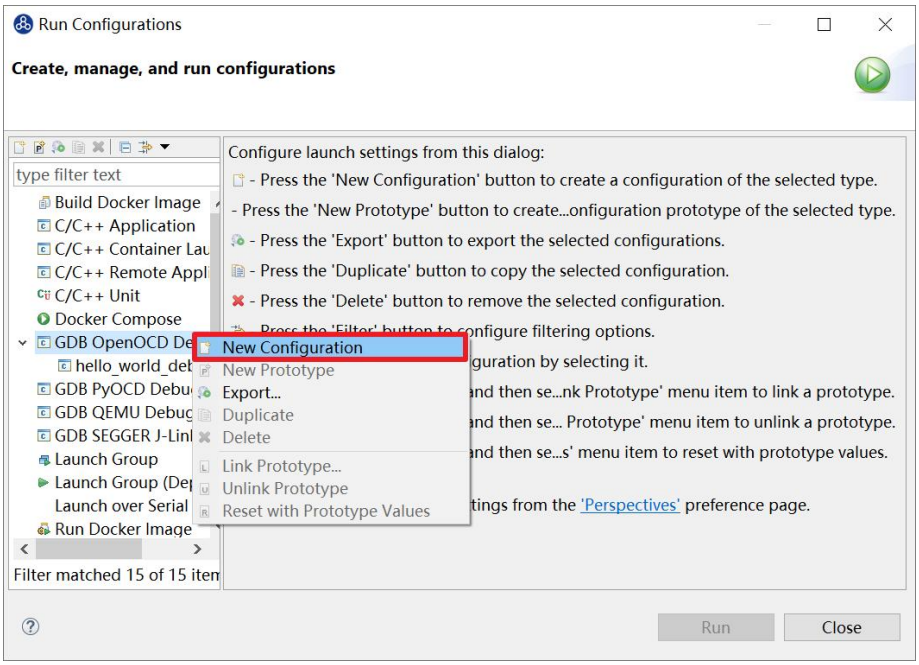


图 7- 13 添加新的 GDB OpenOCD Debugging

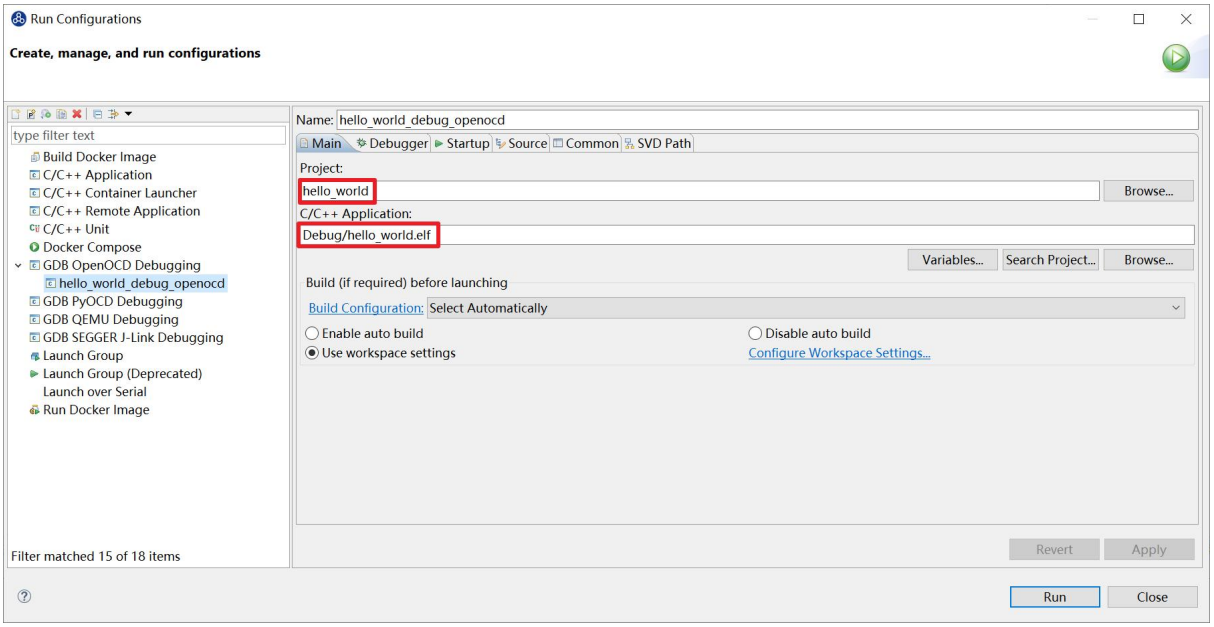


图 7- 14 GDB OpenOCD Debugging 栏目下的 hello_world 项目 Debug

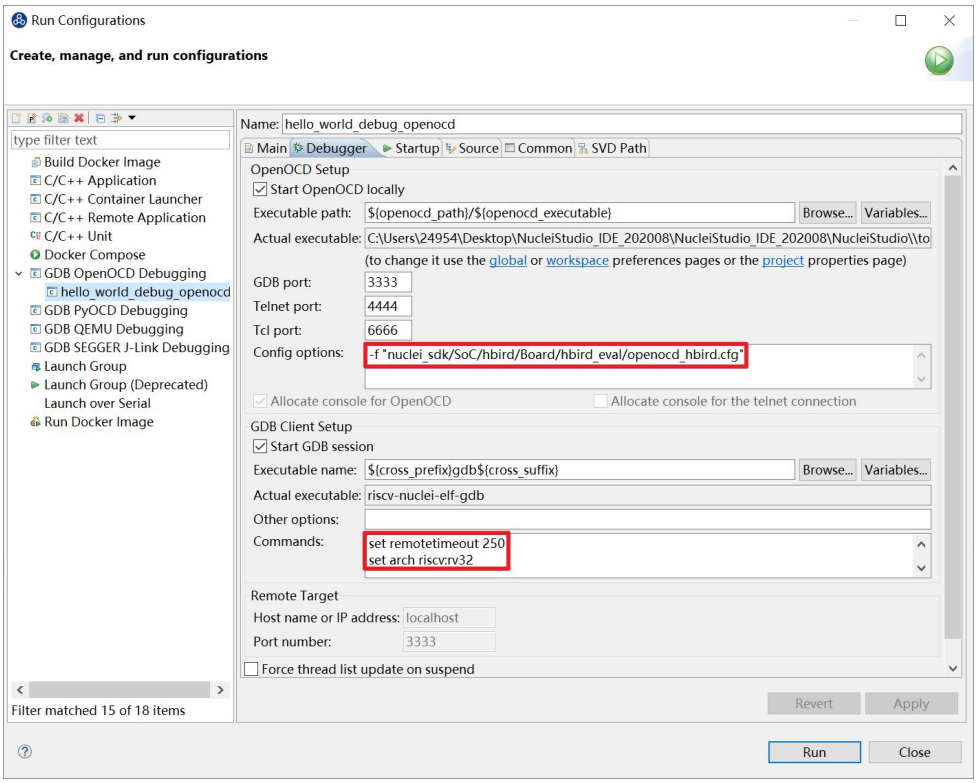


图 7- 15 配置 hello_world_demo Debug 的参数 1

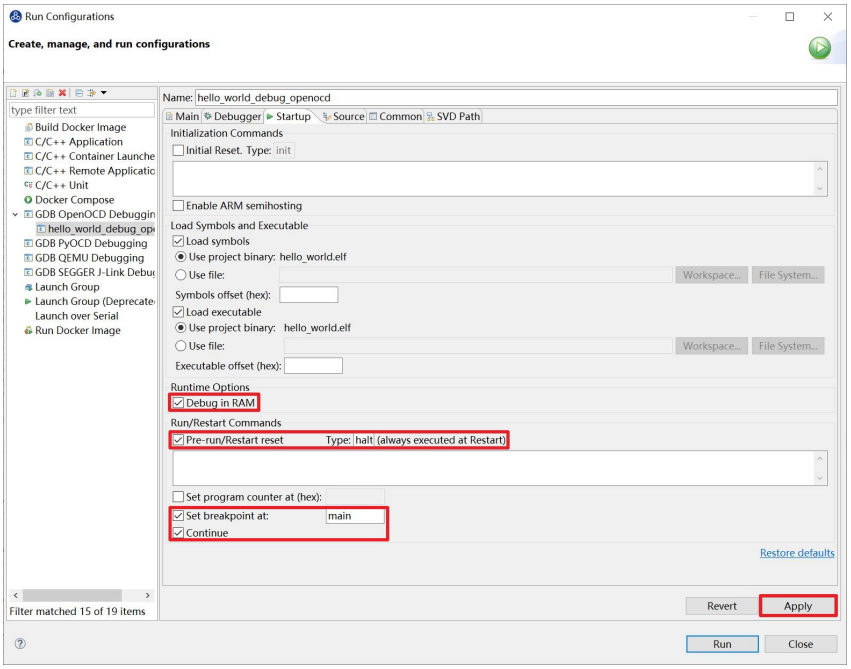


图 7-16 配置 hello_world_demo Debug 的参数 2

7.2.3.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。如图 7-14 所示，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。

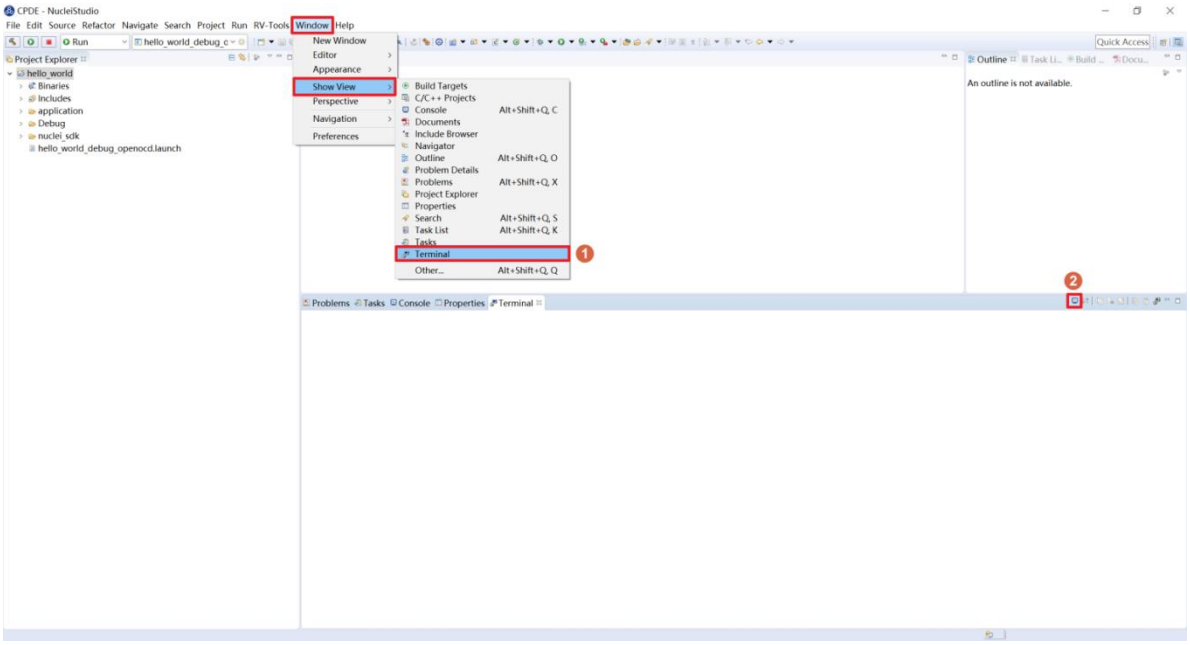


图 7- 17 通过 Nuclei Studio 打开串口调试助手并配置参数

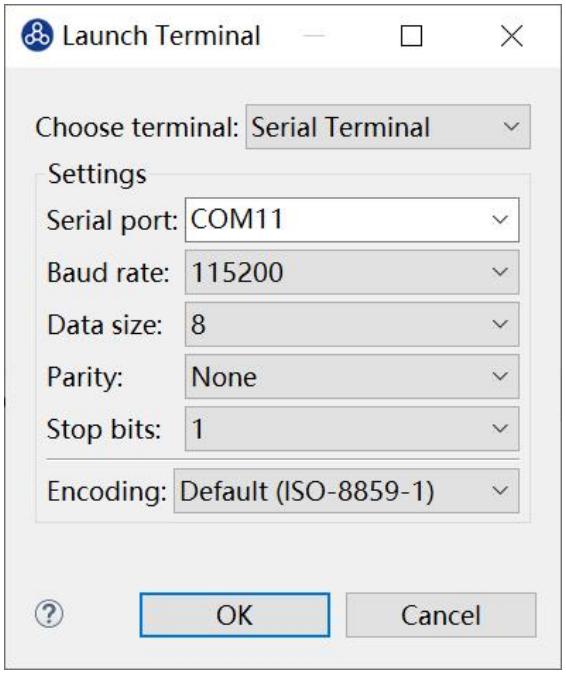


图 7- 18 设置串口参数

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，如图 7-13 所示，点击显示器图标打开串口设置选项。如图 7-15 所示，choose terminal 选择 Local Terminal，点击 OK 打开 Terminal 终端。如图 7-16 所示，输入“minicom /dev/ttyUSB1 115200”打开串口，即可在 Nuclei Studio 中查看串口打印信息。

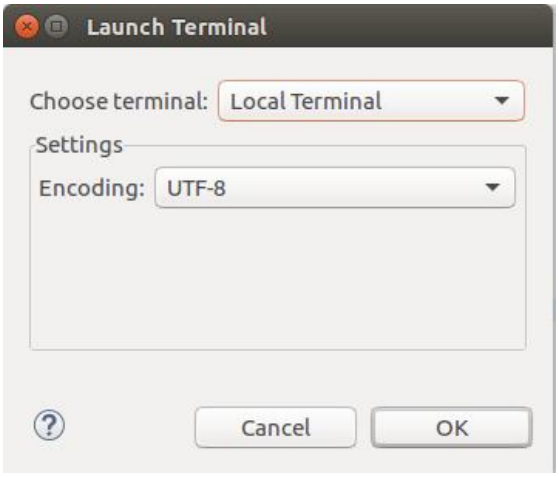


图 7-19 打开 Terminal 终端

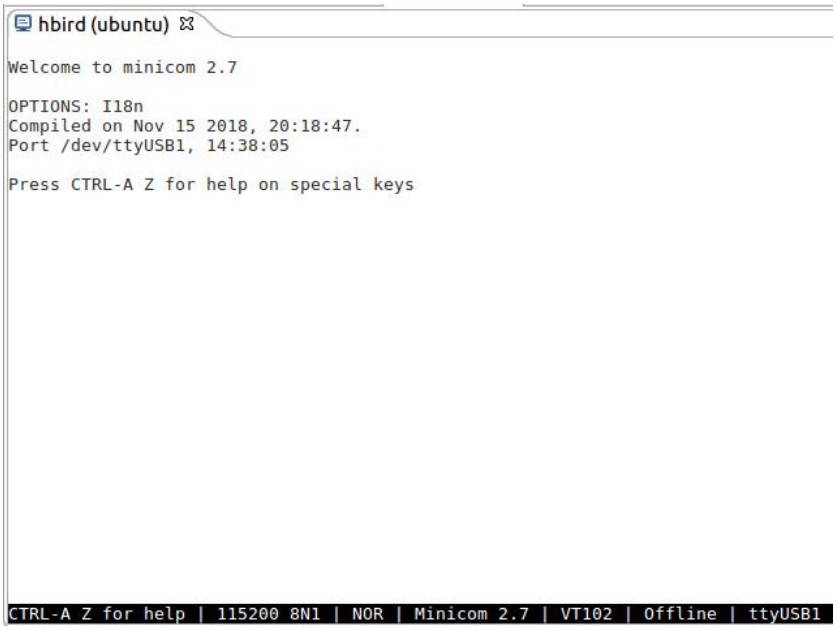


图 7-20 打开串口

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，**注意 demosoc 在 Nuclei SDK 0.5.0 中被移除，请使用 evalsoc 作为替代**。

- 确保 6.2 节中的 Debug 设置内容正确，可以打开 Debug 设置选项确认。如图 7-17 所示，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，在图 7-18 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。

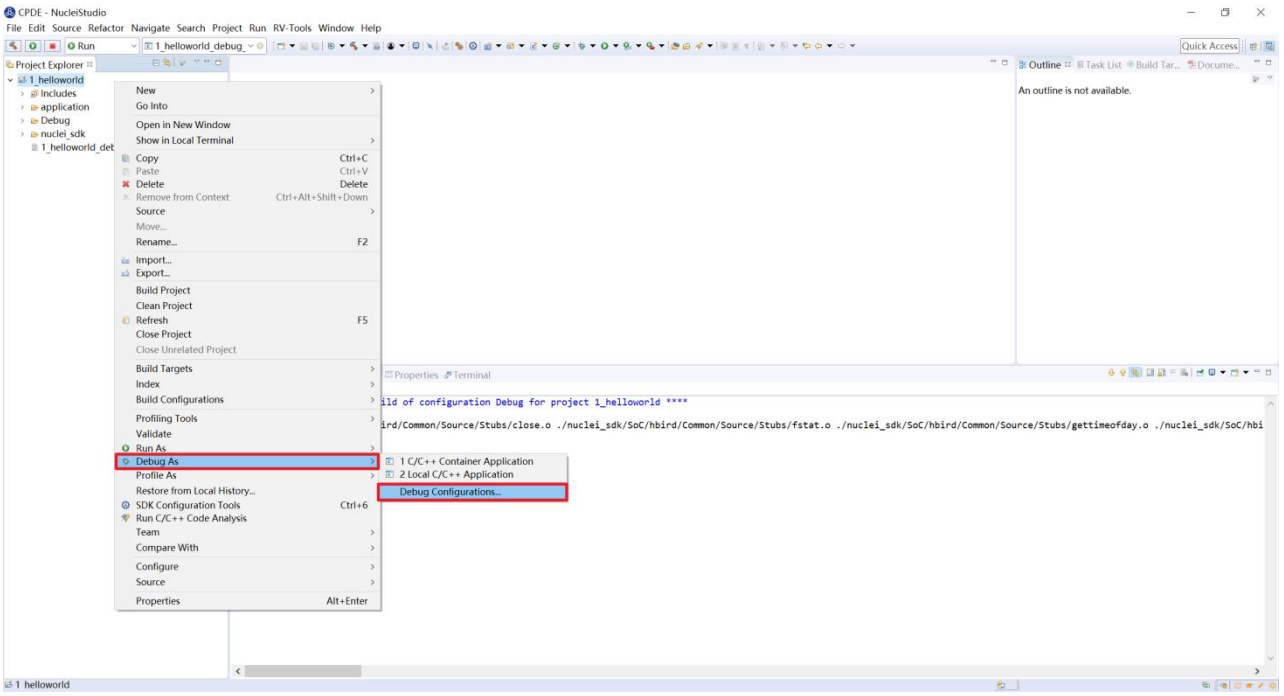


图 7-21 单击“Debug Configuration”进行下载

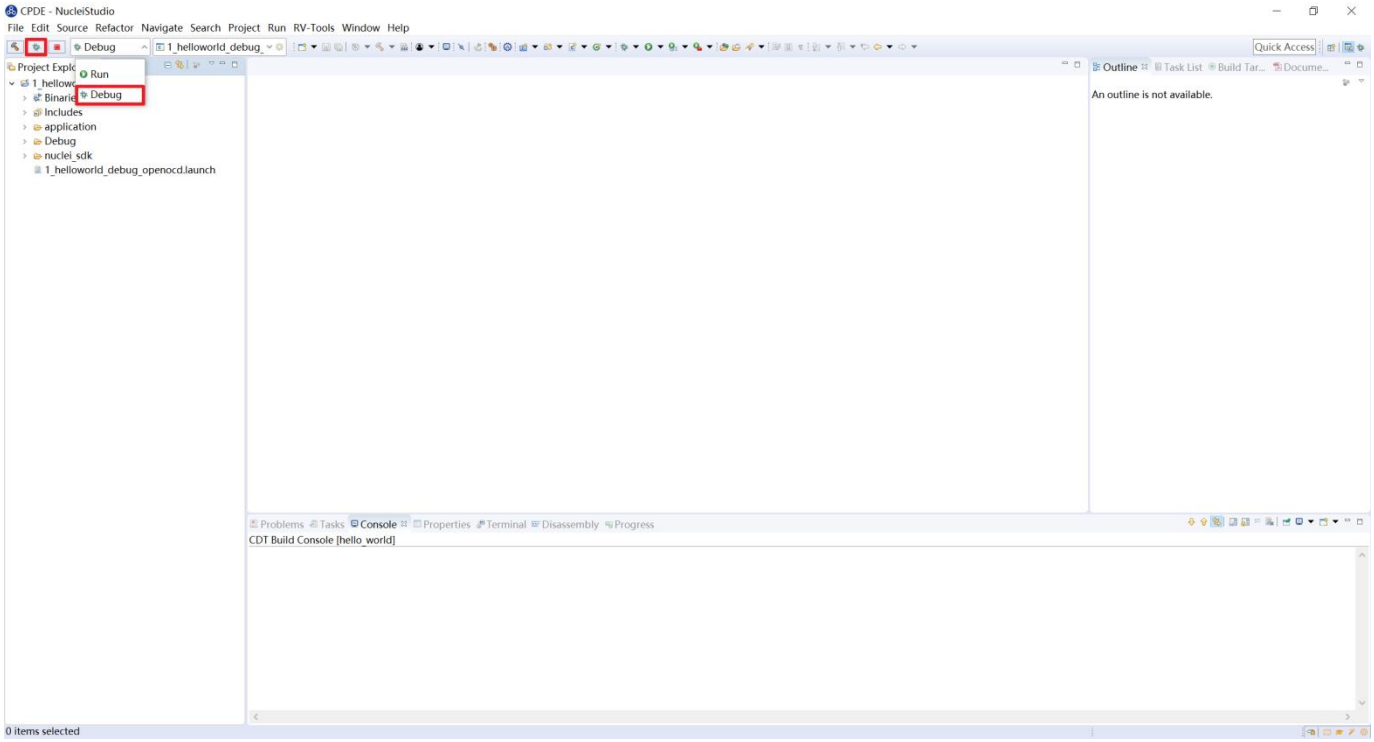


图 7-22 切换至 Debug 模式

■如果下载成功，则如图 7-19 所示，并且会启动调试界面。

●如图 7-19 的 1 号标注位置，这里功能包括单步，运行，汇编级调试等。

●如图 7-19 的 2 号标注位置，这个箭头表示当前程序运行位置。

●如图 7-19 的 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 7-19 的 4 号标注位置，这里可以切换编辑模式和调试模式。

●如图 7-19 的 5 号标注位置，这里是函数内变量显示的位置。

●如图 7-19 的 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 7-19 的 7 号标注位置，点击这里红色按钮可以退出调试模式。

●如图 7-19 的 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。

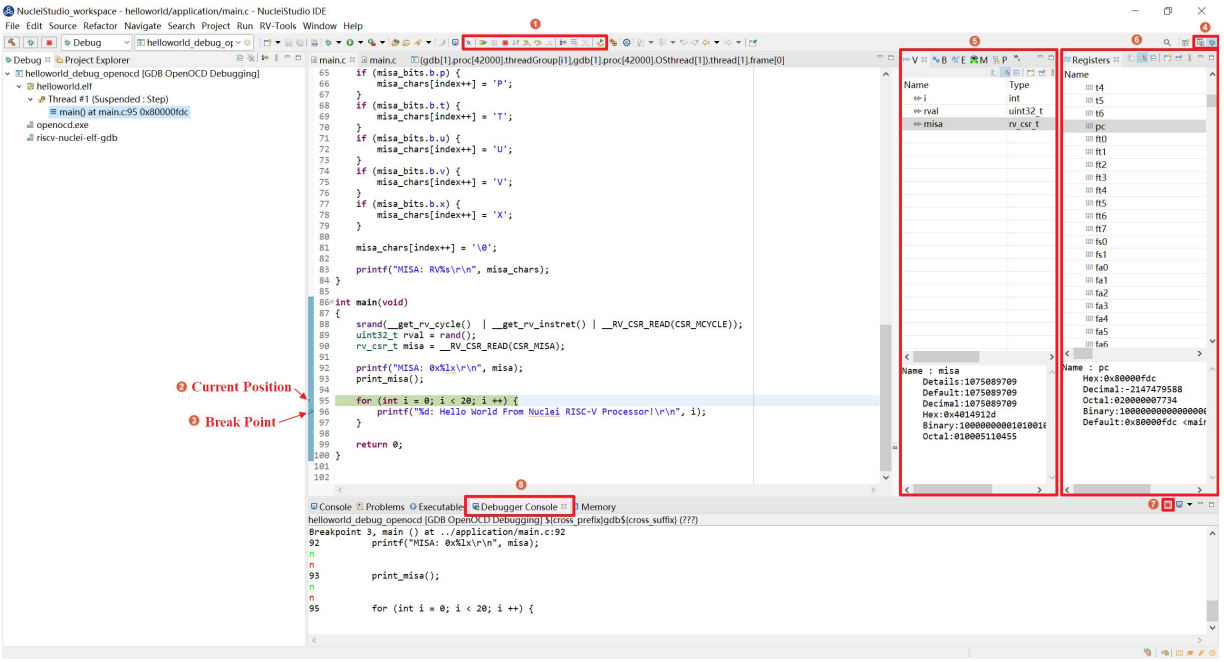


图 7-23 下载完成后进入调试界面

7.2.4.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。如图 7-20，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。如图 7-21，可以看到串口正确打印出 helloworld 等信息。需要断开连接，如图 7-22，在 console 栏目下点击红色按钮断开连接。

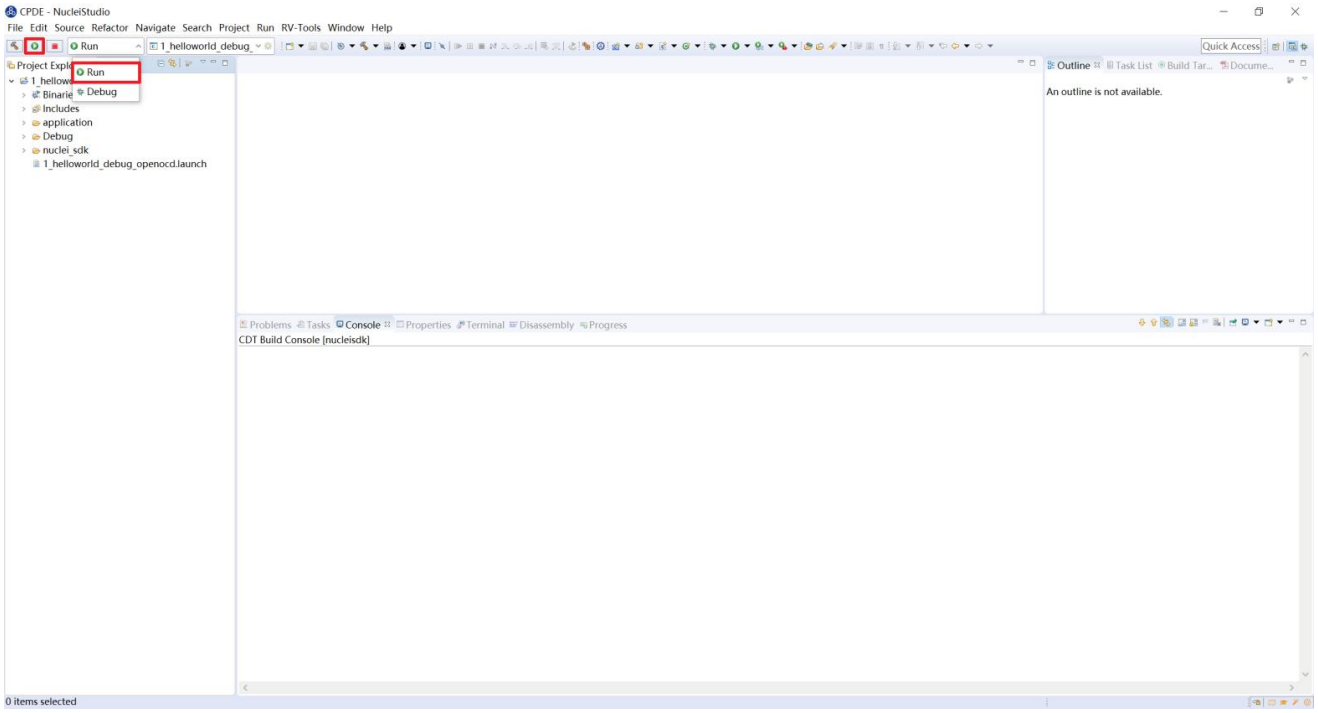


图 7- 24 下载程序到开发板中运行

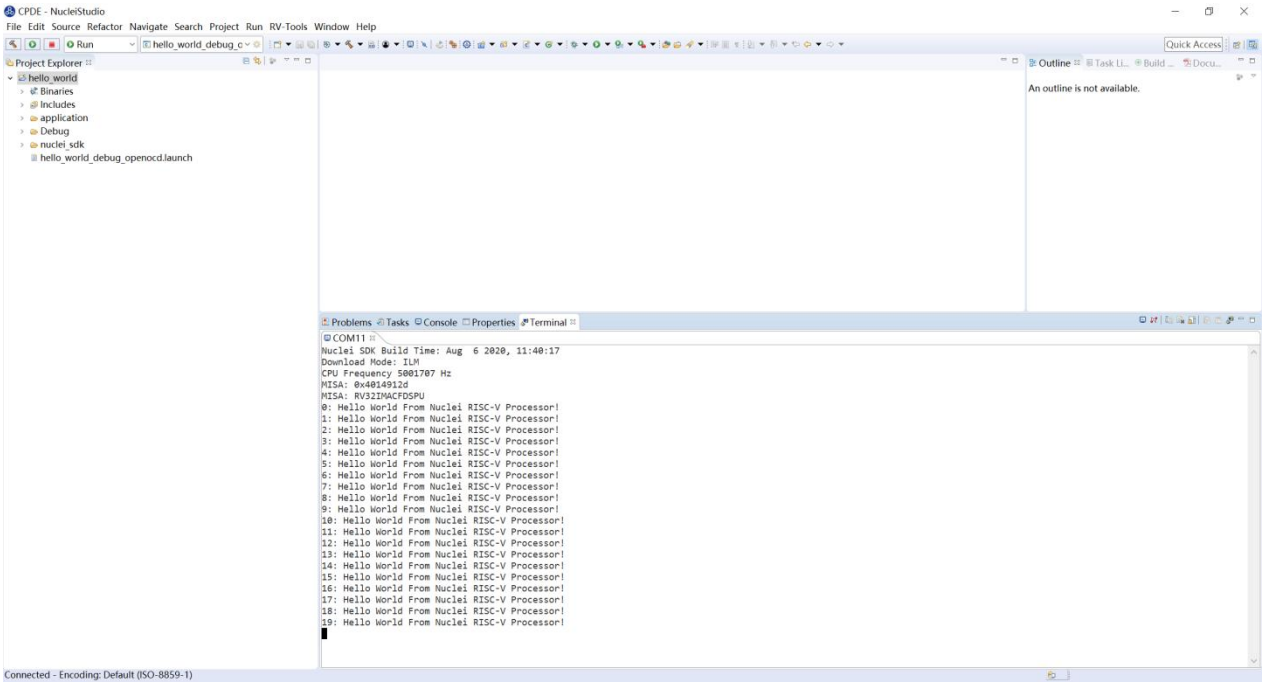


图 7- 25 串口调试助手界面输出的 Hello World

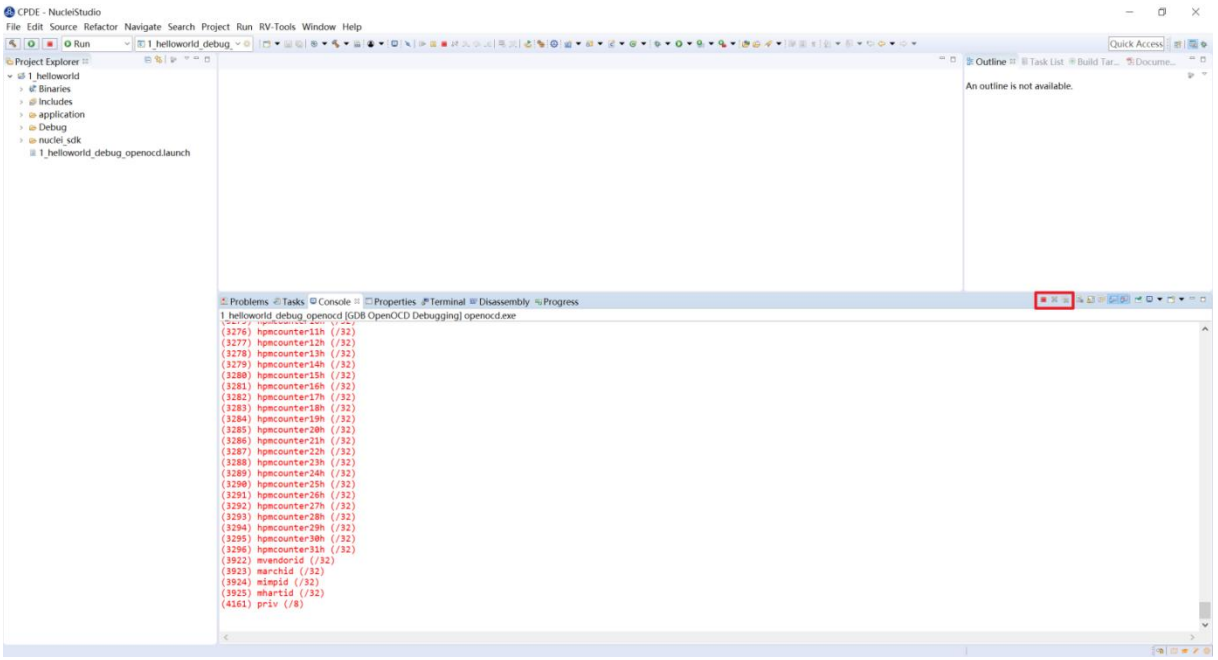


图 7- 26 下载完成单击红色按钮断开连接

7.3.使用 J-Link 调试运行项目

7.3.1.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，如图 7- 23，在“Resource”选项点击红框标注的图标快速打开工程所在的目录。如图 7- 24，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。

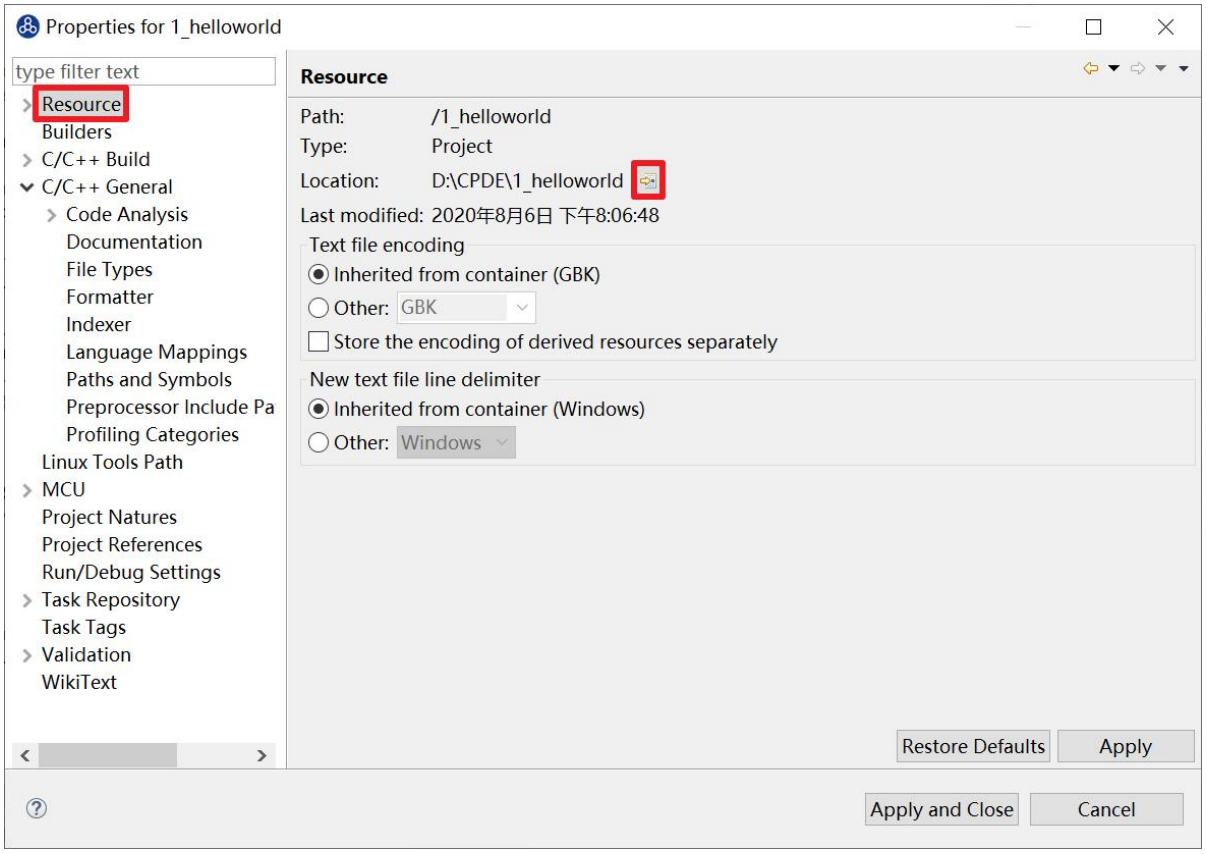


图 7- 27 打开工程所在目录

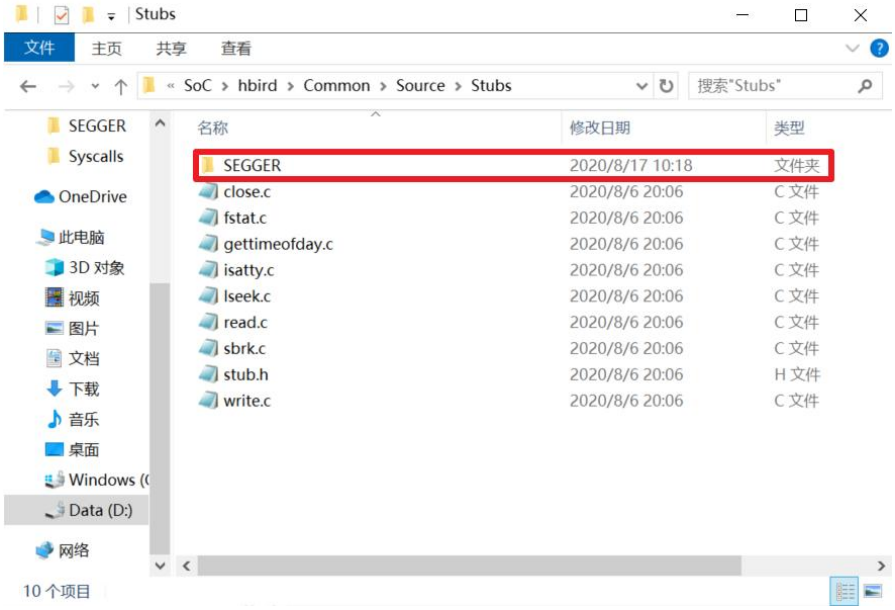


图 7-28 新建“SEGGER”文件夹

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）。解压缩后文件内容如图 7-25，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，最后如图 7-26 所示。在 IDE 中打开“SEGGER_RTT_Syscalls_GCC.c”，如图 7-27，注释“#include <reent.h>”所在的这一行。

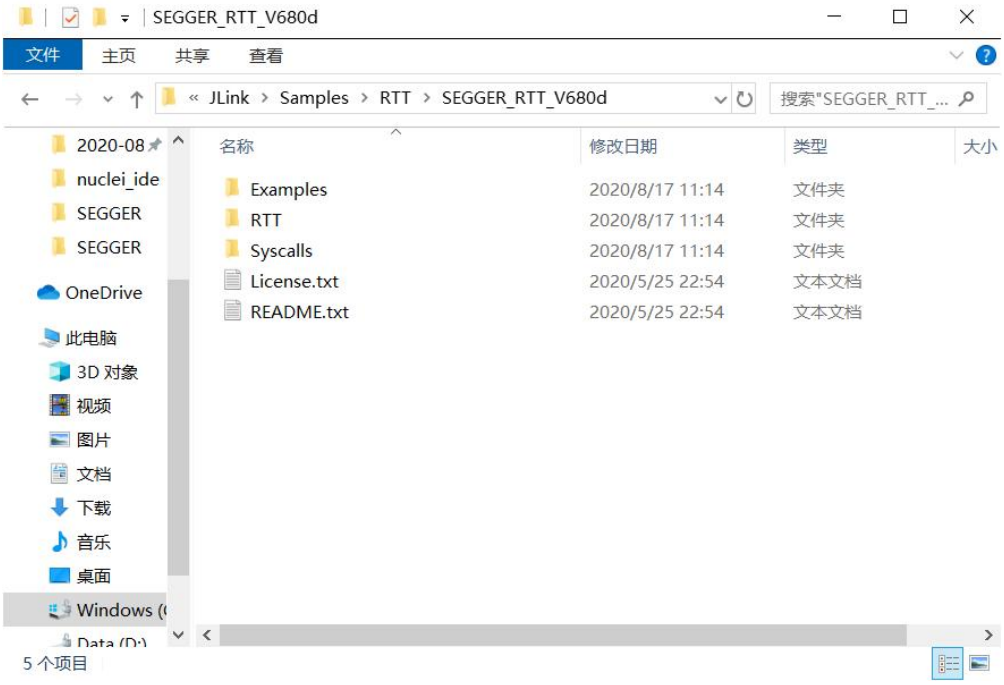


图 7-29 压缩包内文件内容

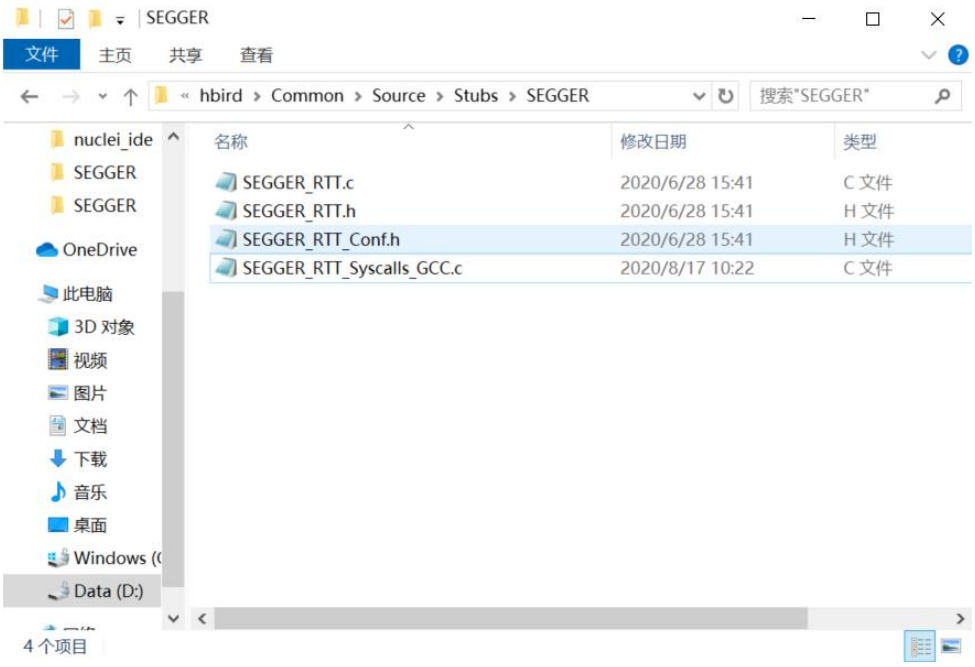


图 7-30 复制后 SEGGER 文件夹内容

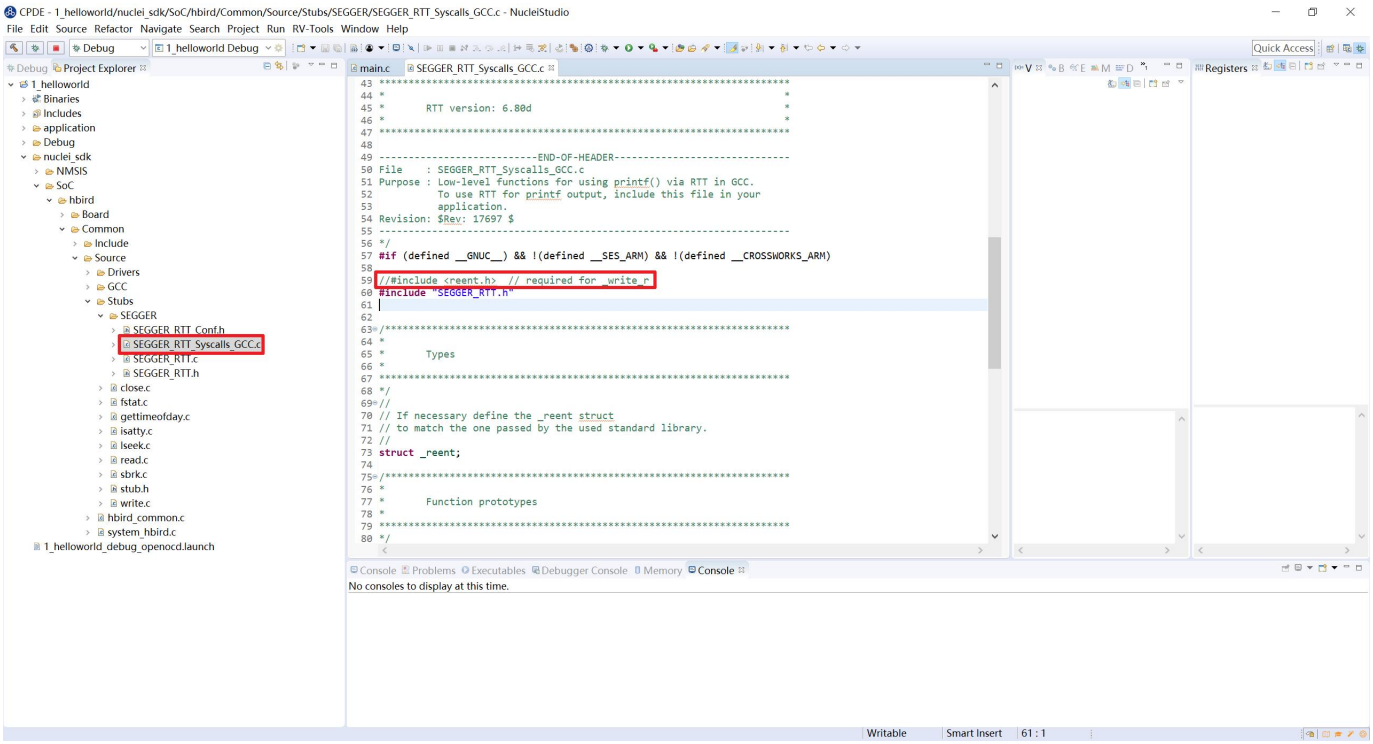


图 7-31 注释不使用的头文件

文件添加完成后添加 SEGGER 文件夹路径至 include，如图 7-28，打开当前工程的设置页面，添加 SEGGER 文件夹路径至 include 中。

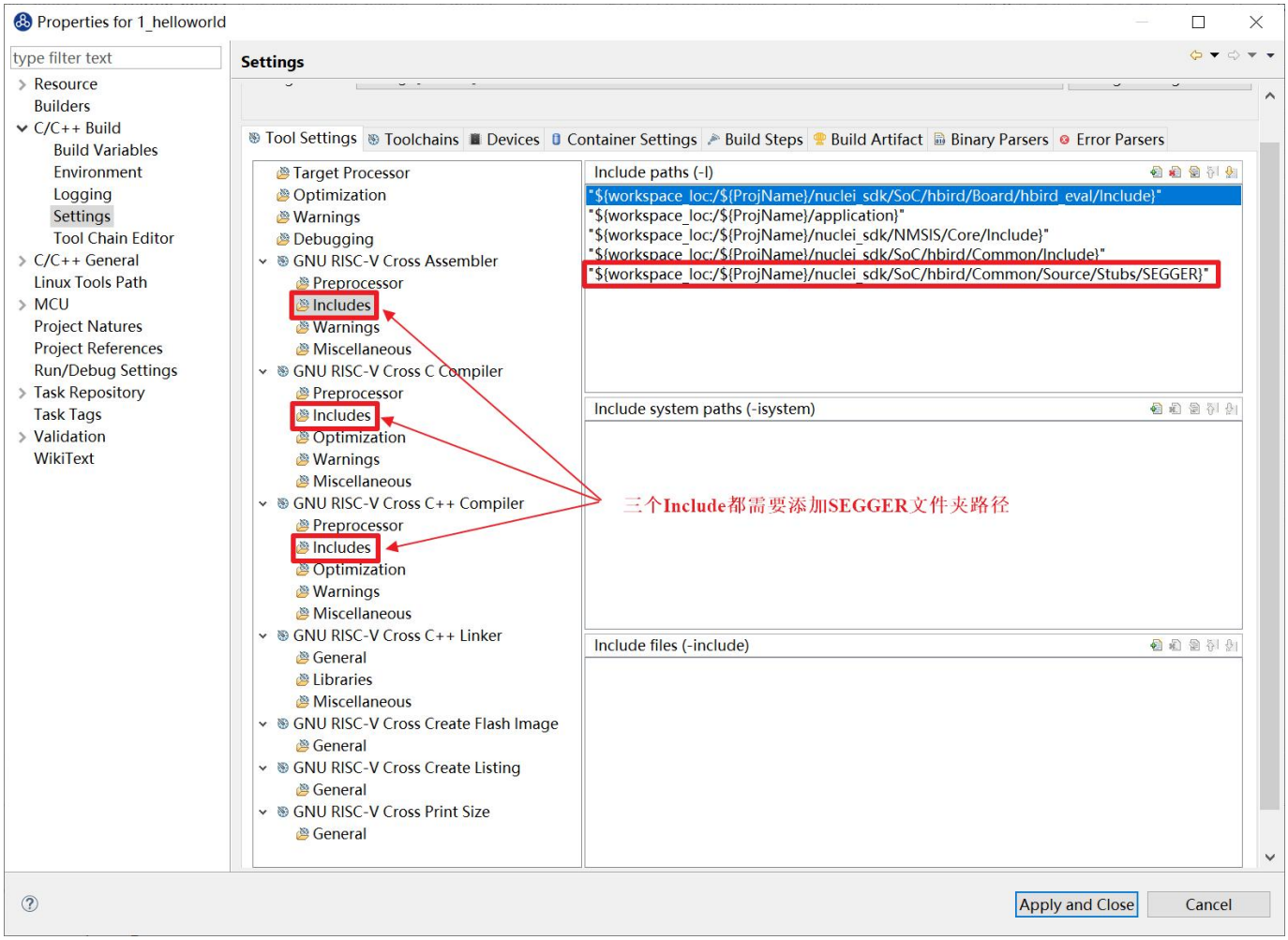


图 7-32 添加 SEGGER 文件夹路径至 Include 中

接下来移除原有的 write 函数。如图 7-29，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。

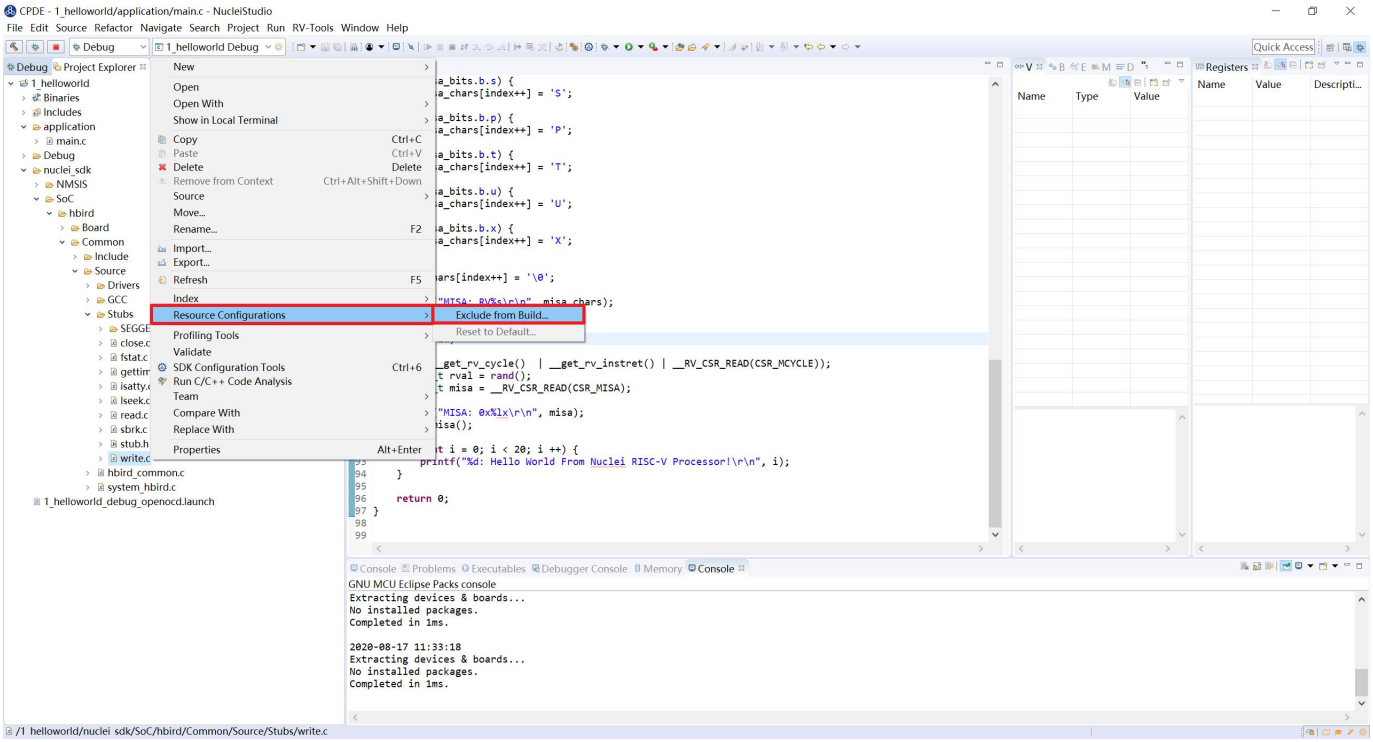


图 7- 33 打开 Exclude from Build

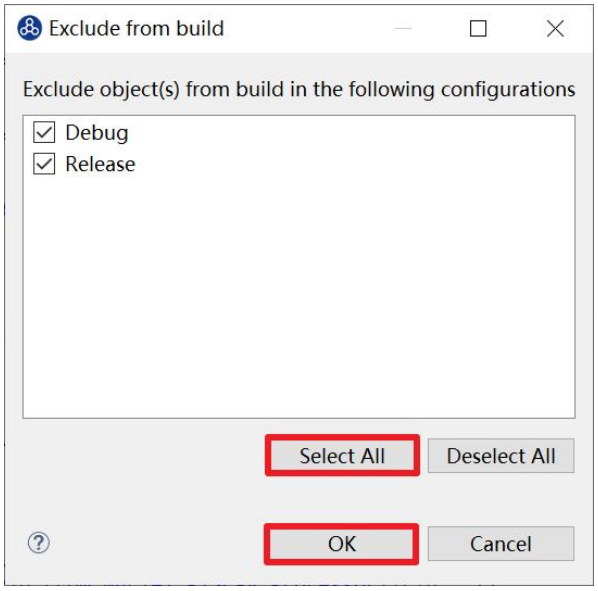


图 7- 34 移除 write.c 文件

7.3.2. Debug Configuration

7.3.2.1 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。

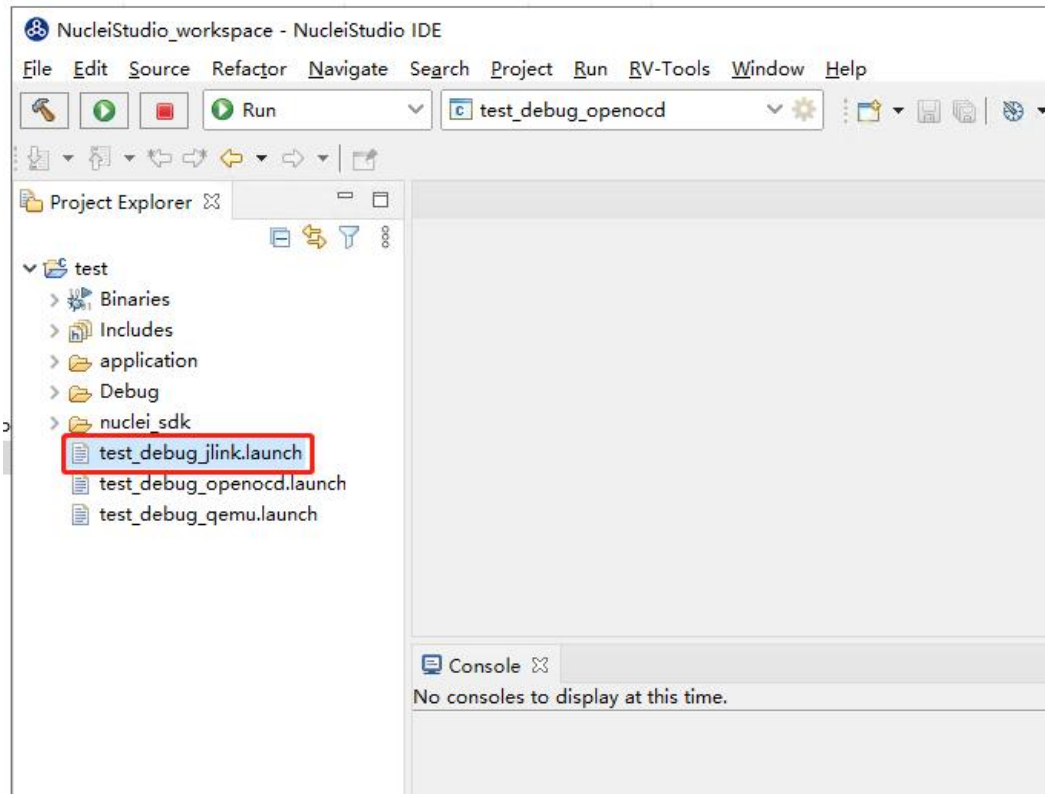


图 7-35 Launch 文件

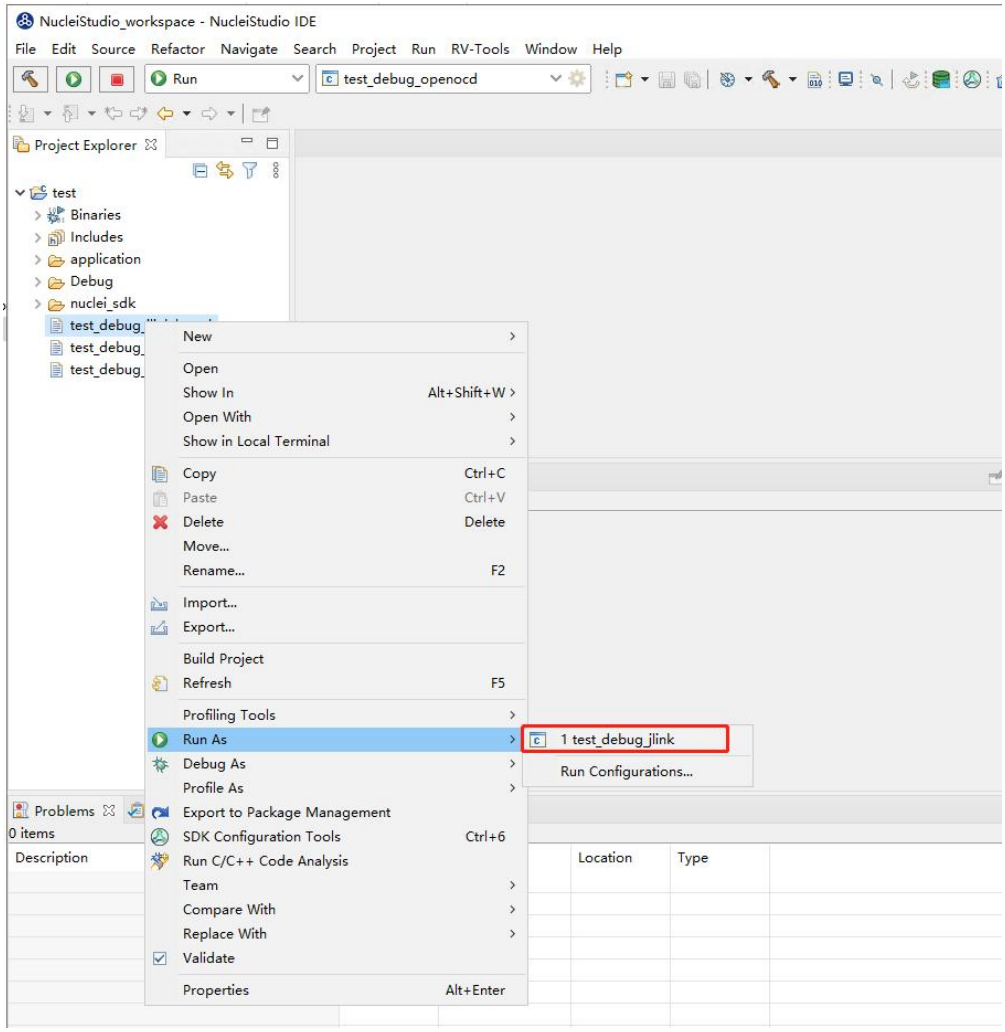


图 7-36 Run as/Debug as

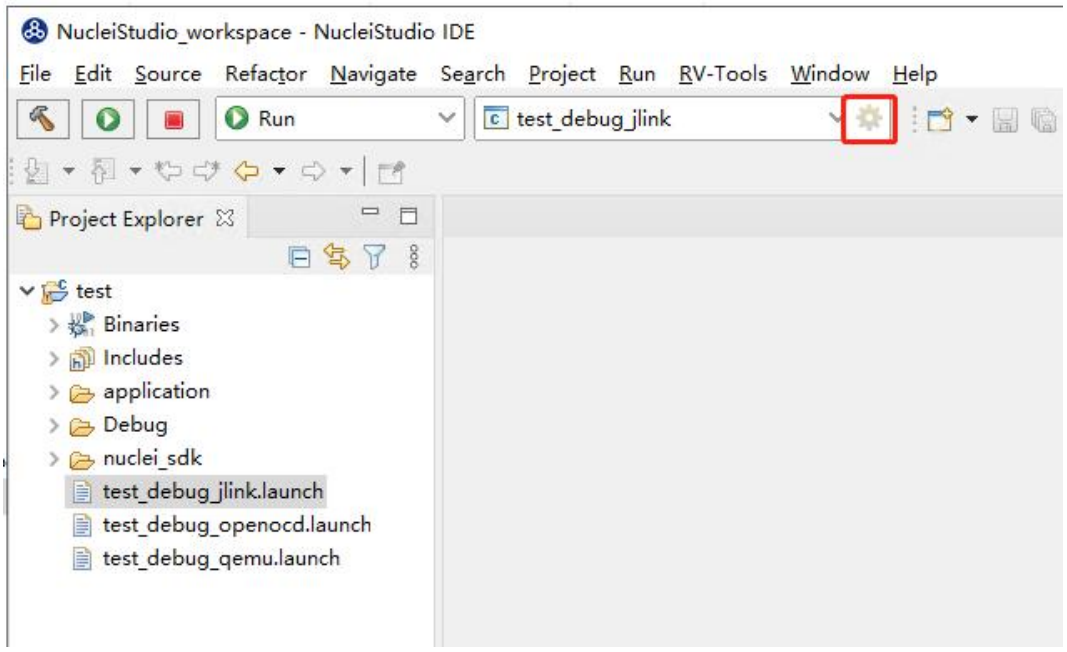


图 7-37 在 Launch Bar 中查看 Debug Configurations

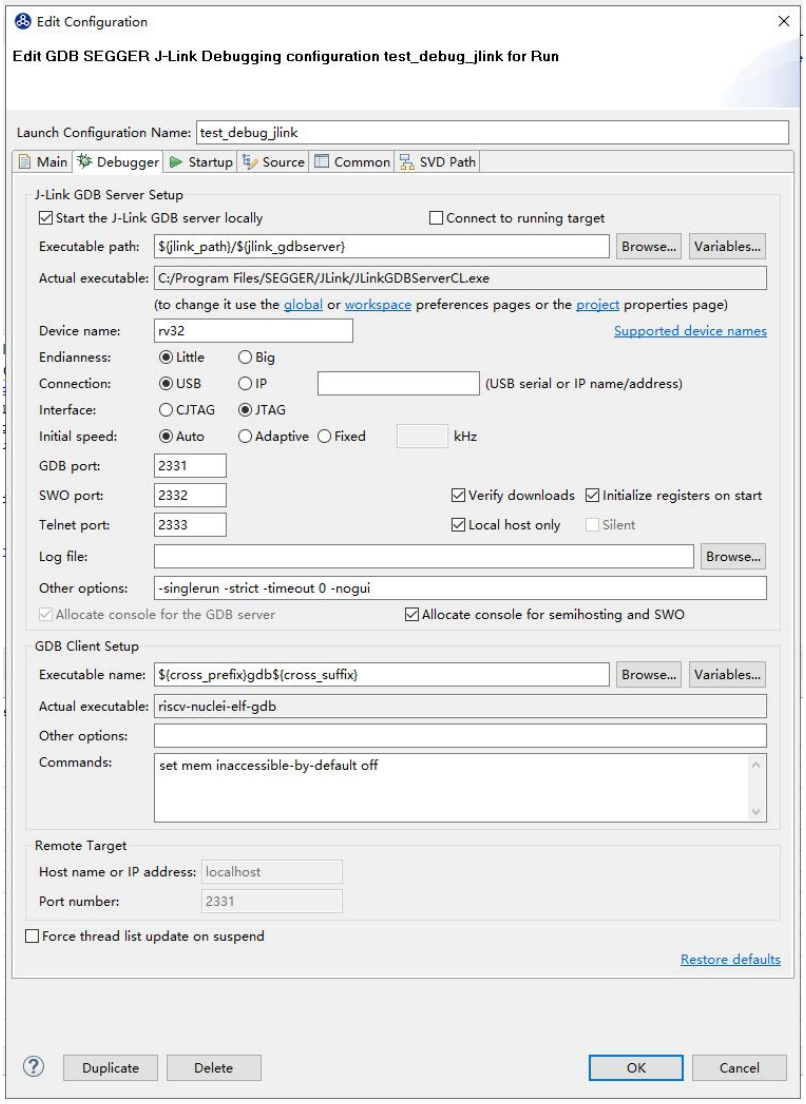


图 7-38 Debug Configurations

7.3.2.2 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”。

■如图 7-35，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”，如图 7-36。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。

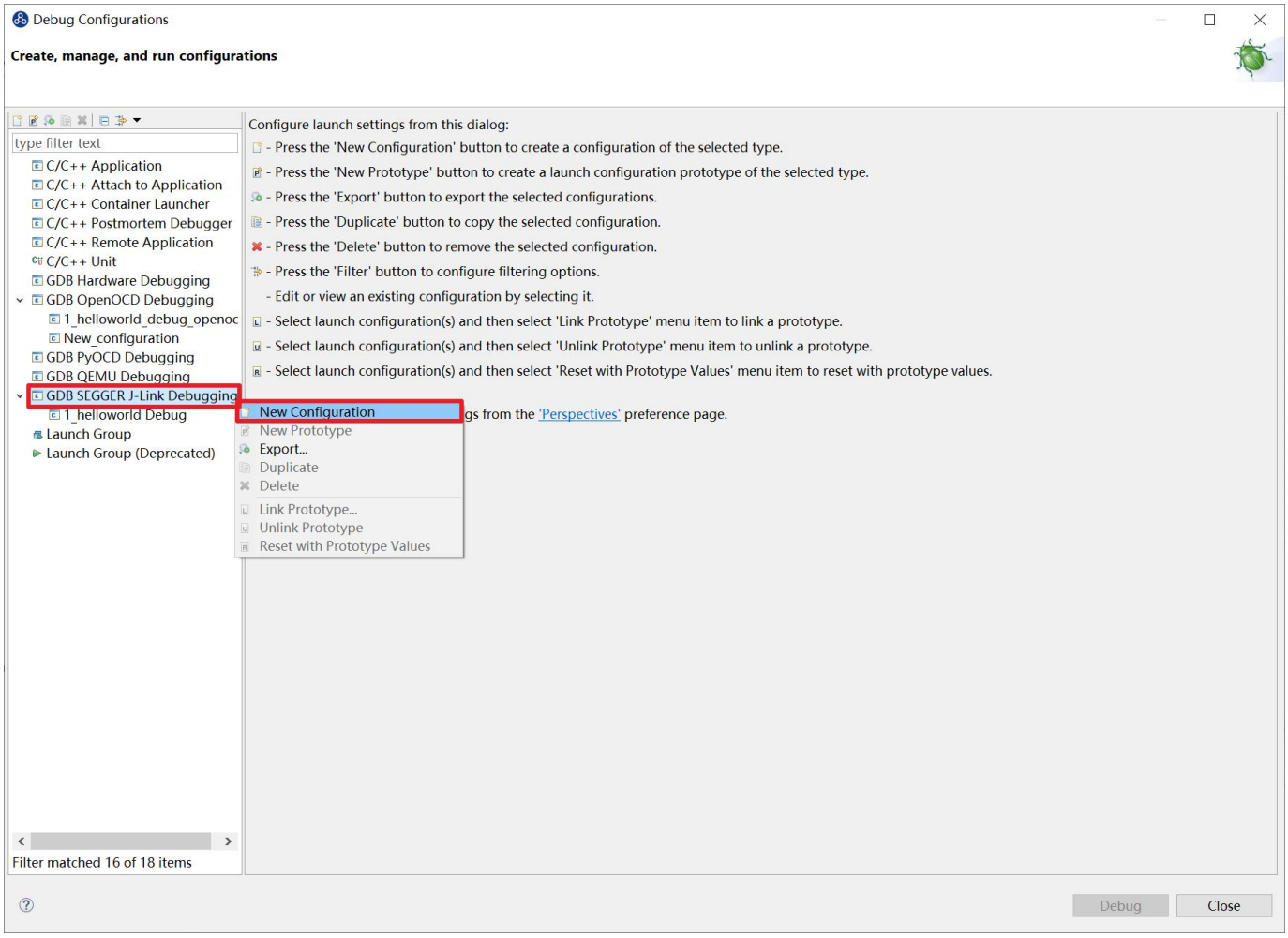


图 7-39 新建 J-Link Debug Configuration

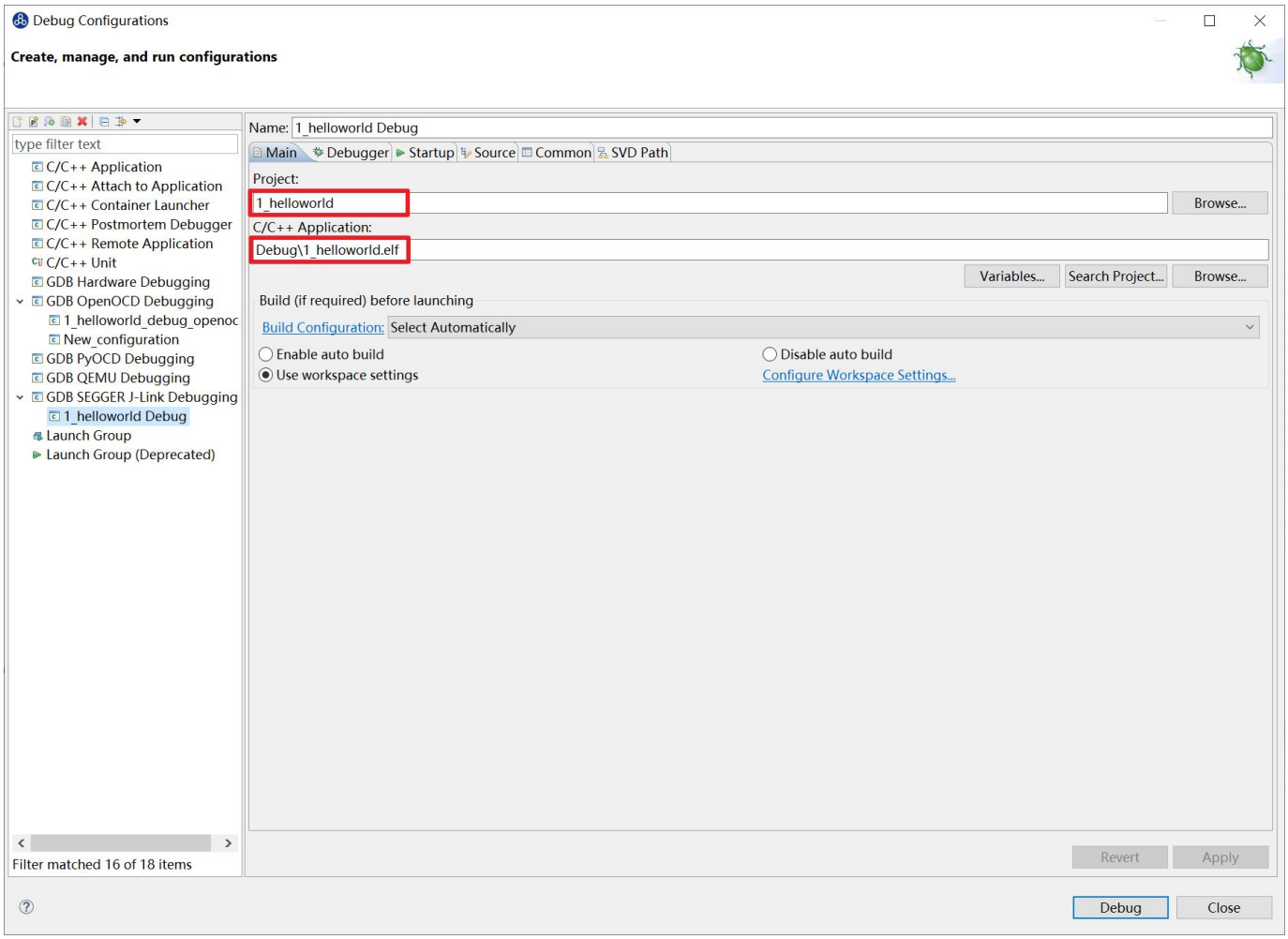


图 7-40 确保 Project 和 C/C++ Application 与工程一致

打开 Debugger 栏目，如图 7-377。

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。

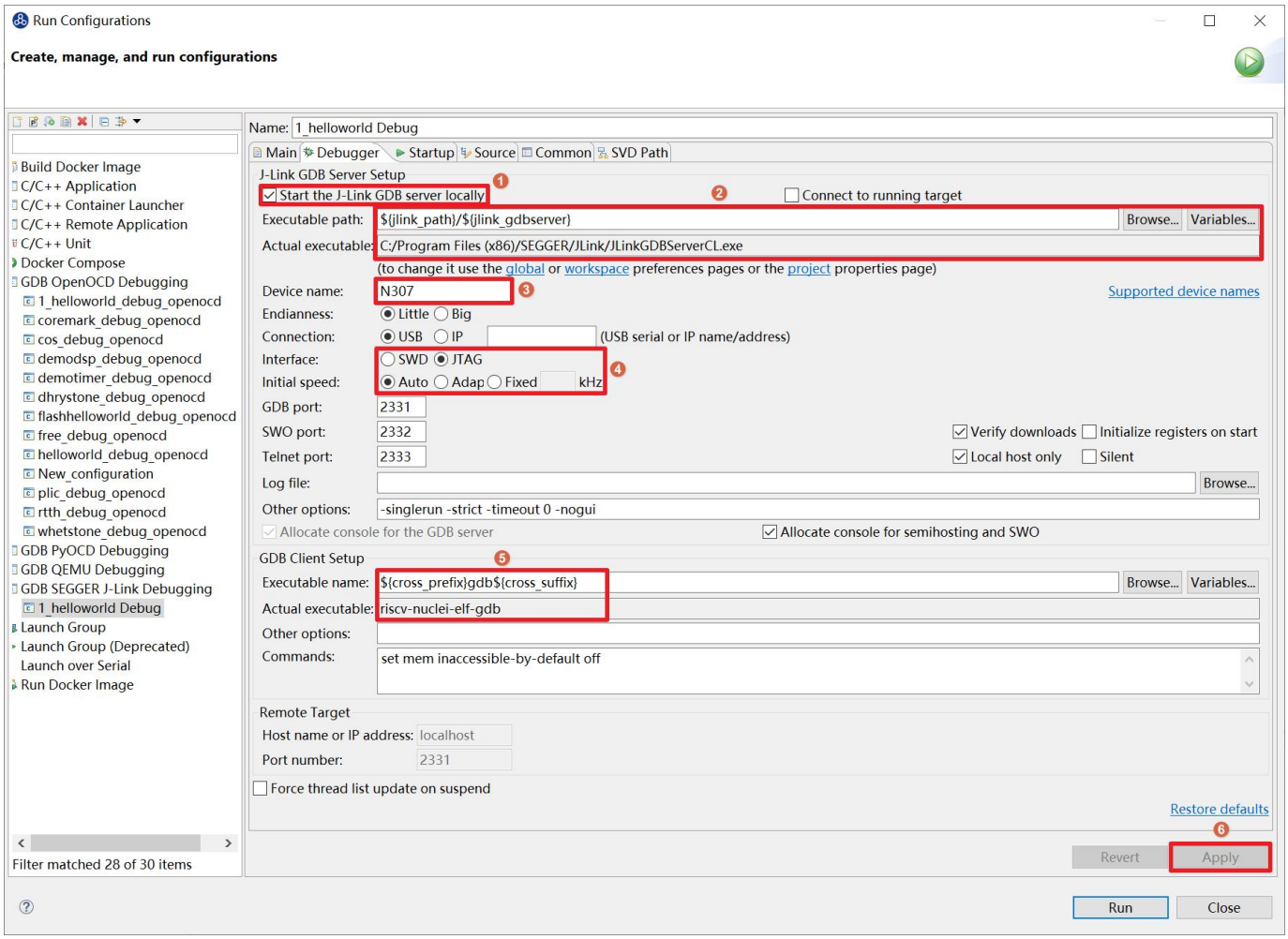


图 7-41 设置 Debugger 栏目内容

打开“Startup”栏目，如图 7-388，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。

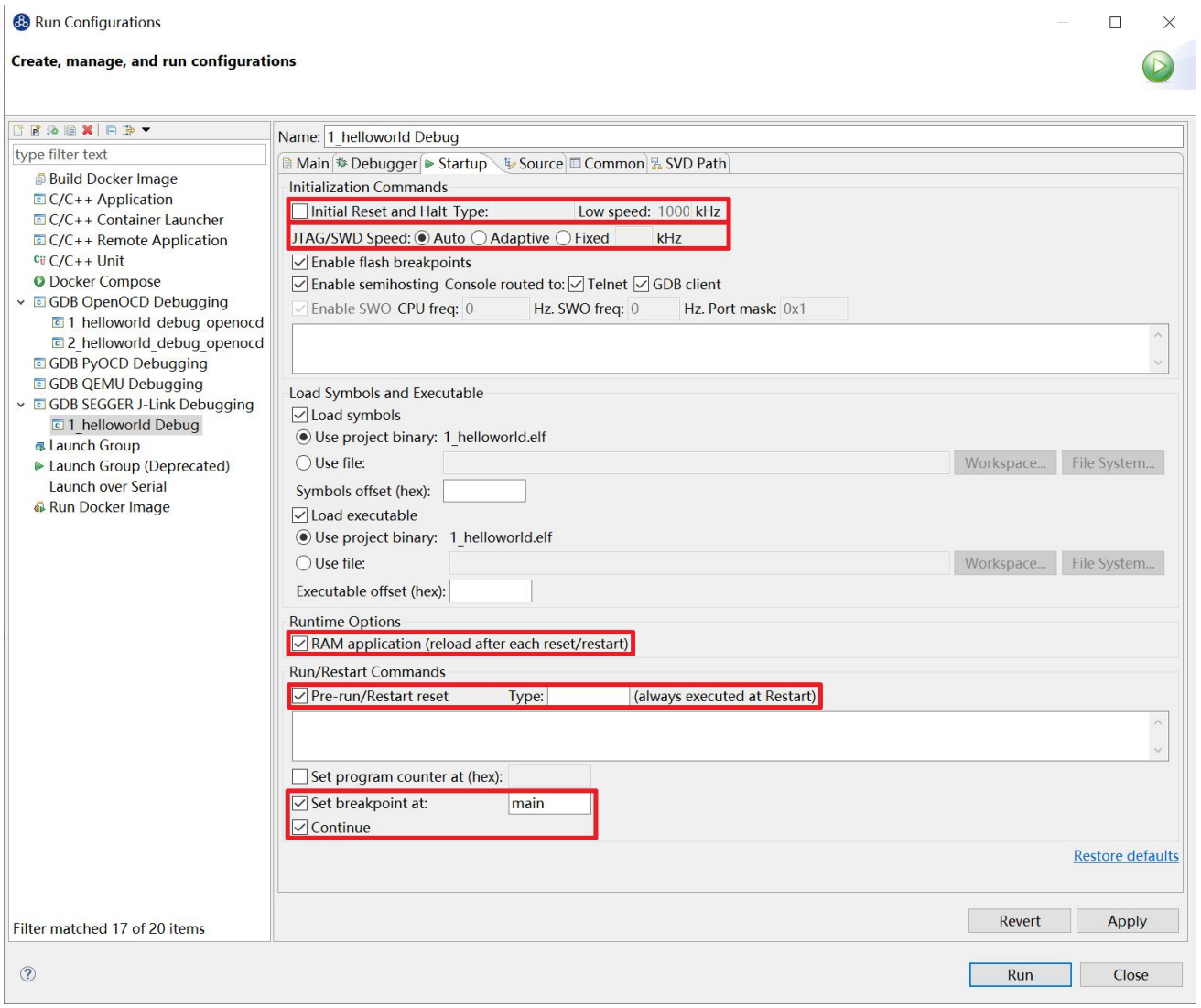


图 7-42 设置 Startup 栏目内容

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。

7.3.3.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。如图 7-39，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，最后实物连接如图 7-40。

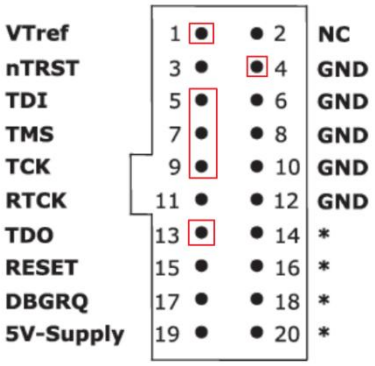


图 7-43J-Link 引脚图

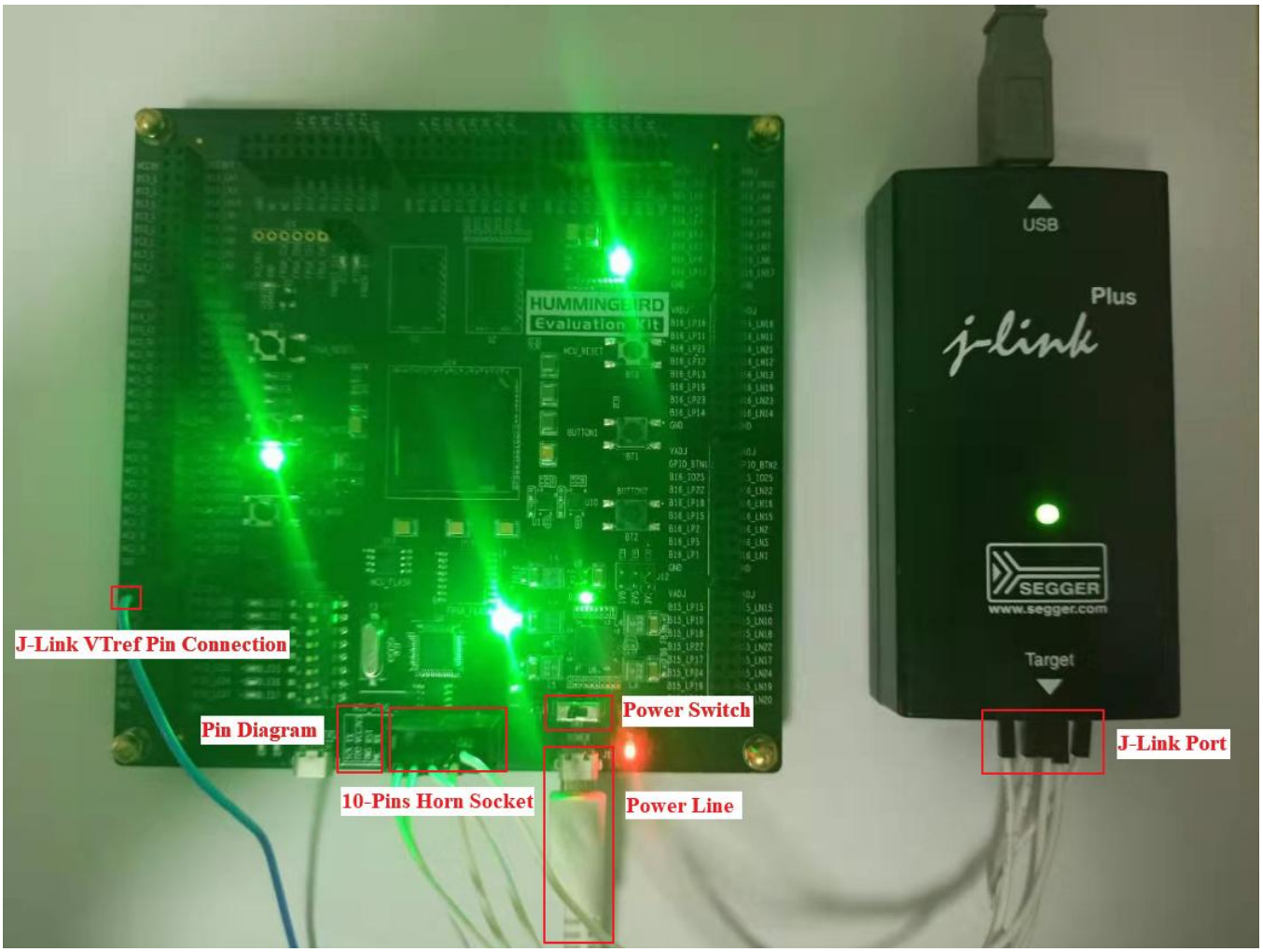


图 7- 44J-Link 实物连接图

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。如图 7-41，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。

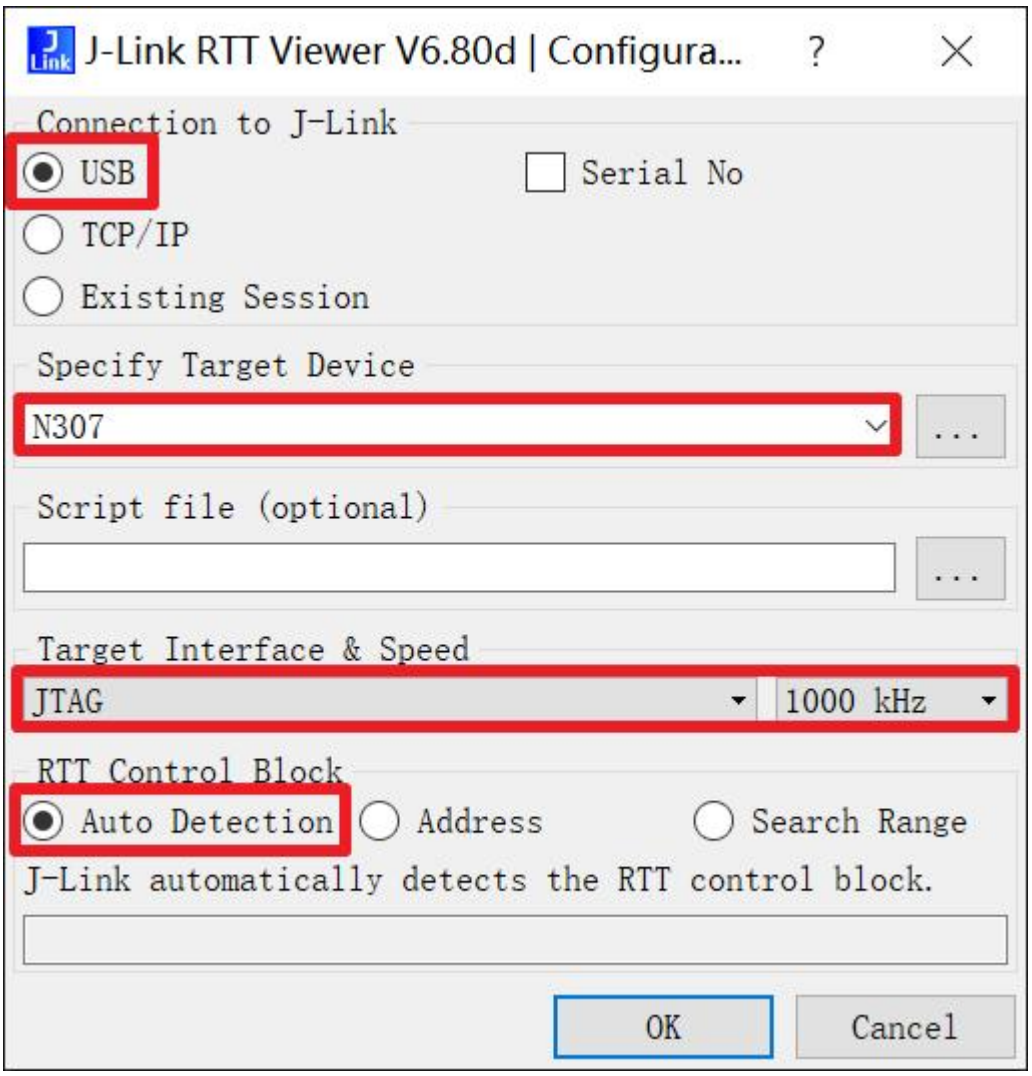


图 7-45 设置 RTT Viewer

图 7-42 中下拉框选中 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。

如果下载成功，则如图 7-43 所示，并且会启动调试界面。

■如图 7-43 的 1 号标注位置，这里功能包括单步，运行，汇编级调试等。

■如图 7-43 的 2 号标注位置，这个箭头表示当前程序运行位置。

■如图 7-43 的 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 7-43 的 4 号标注位置，这里可以切换编辑模式和调试模式。

■如图 7-43 的 5 号标注位置，这里是函数内变量显示的位置。

■如图 7-43 的 6 号标注位置，这里是查看寄存器数值的位置。

■如图 7-43 的 7 号标注位置，点击这里红色按钮可以退出调试模式。

■如图 7-43 的 8 号标注位置下方，这里可以使用 GDB 控制台指令进行调试。

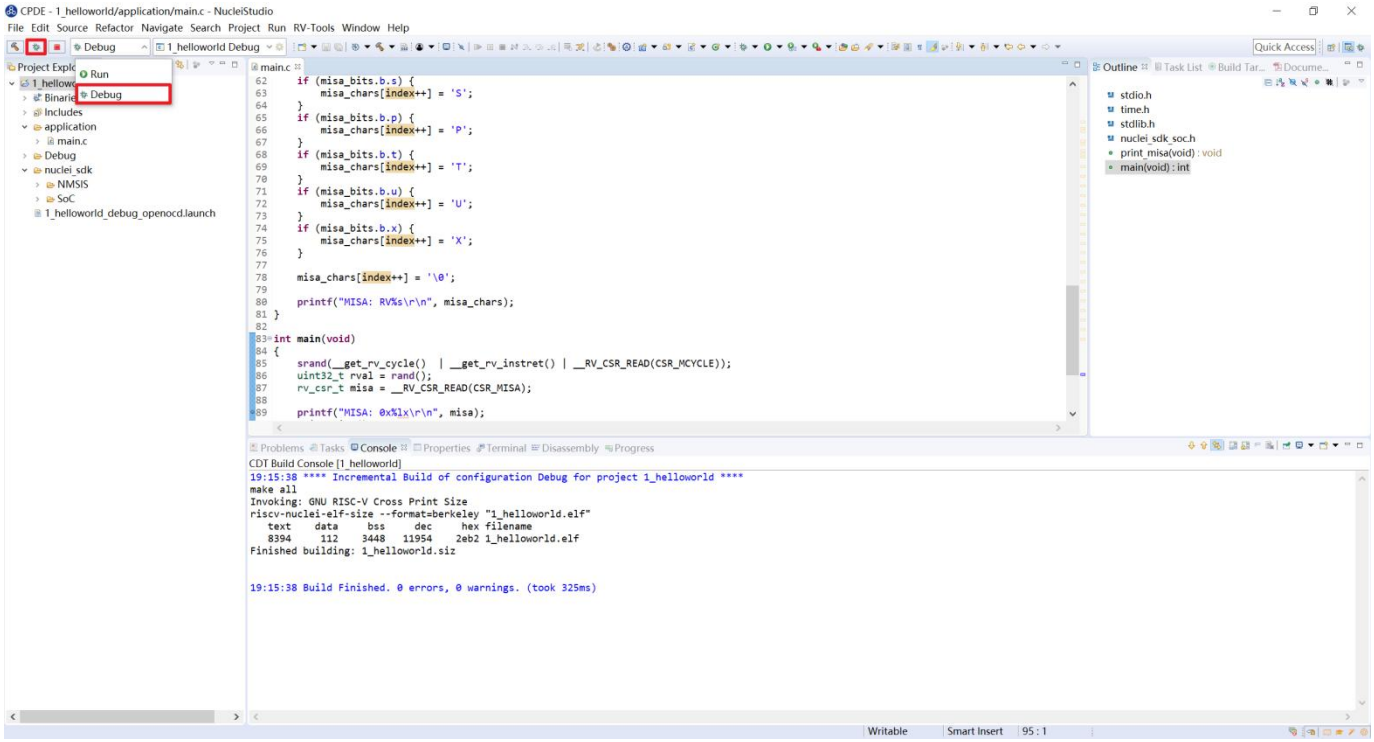


图 7-46 切换至下载模式

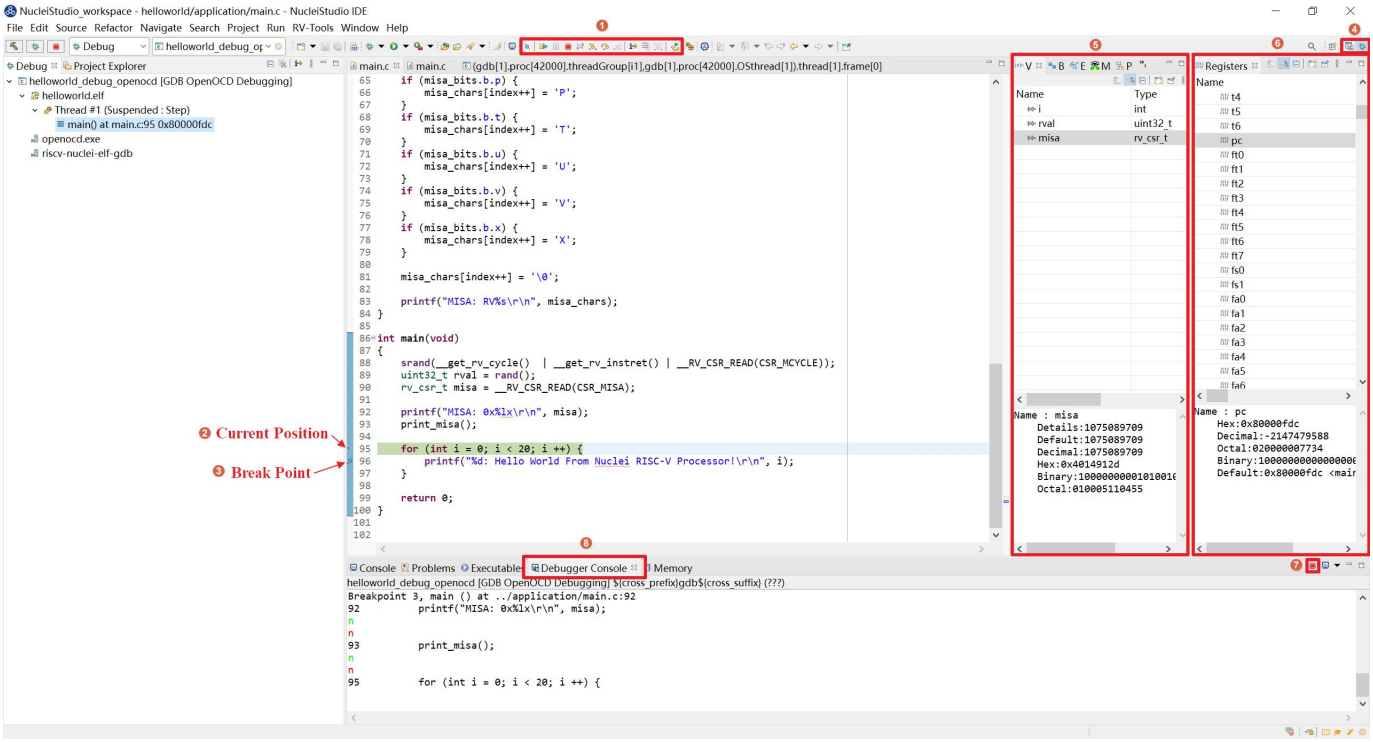


图 7-47 下载完成后进入调试页面

7.3.4.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。如图 7-44，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。如图 7-45，可以看到 RTT Viewer 正确打印出 helloworld 等信息。如果需要断开连接，如图 7-46，在 console 栏目下点击红色按钮即可断开连接。

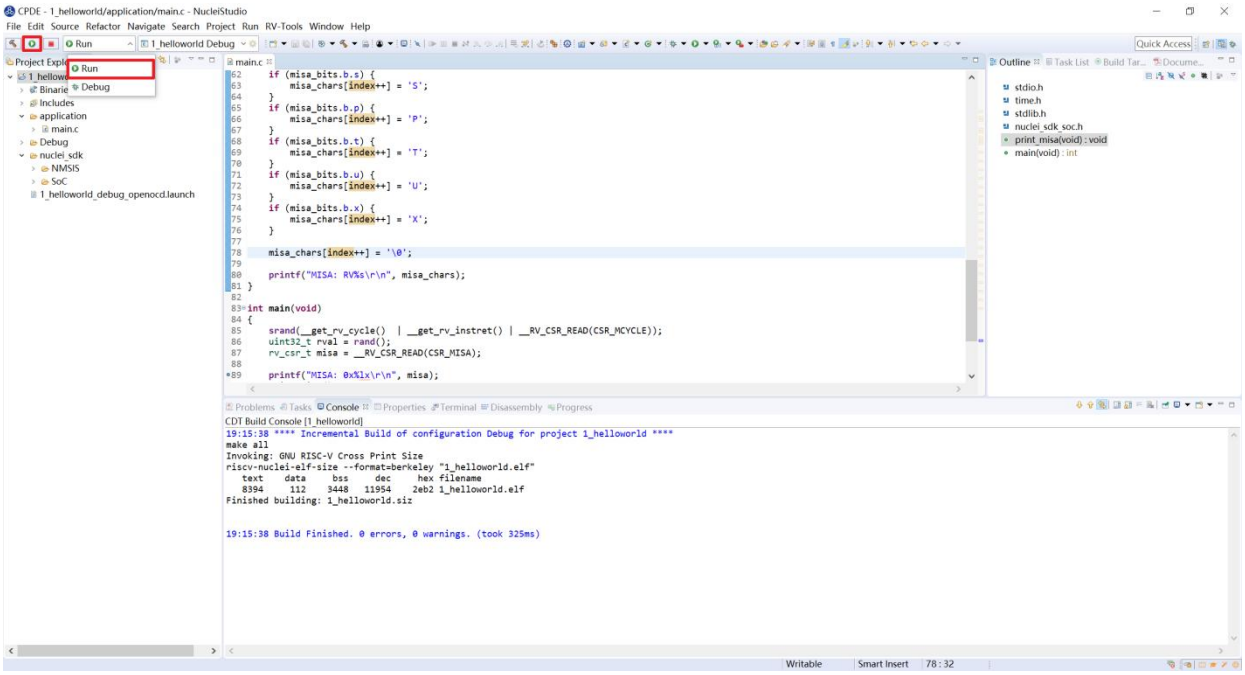


图 7- 48 下载程序到开发板中运行

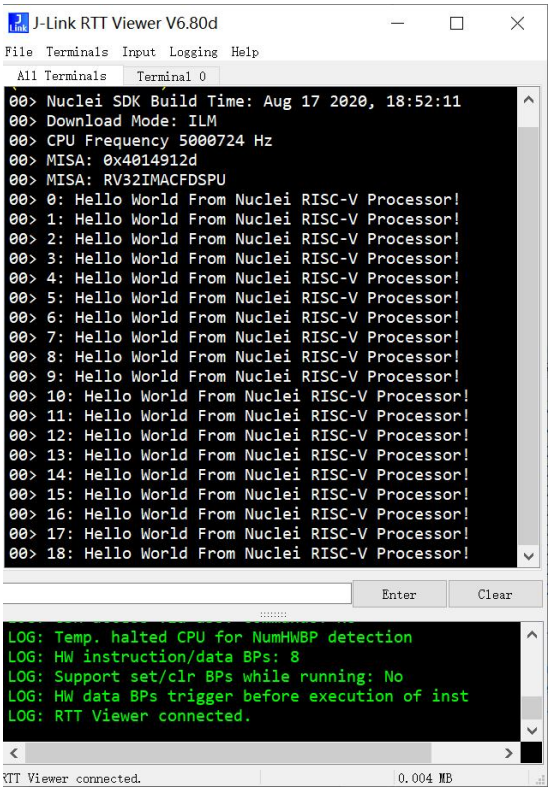


图 7- 49 打印输出 Hello World

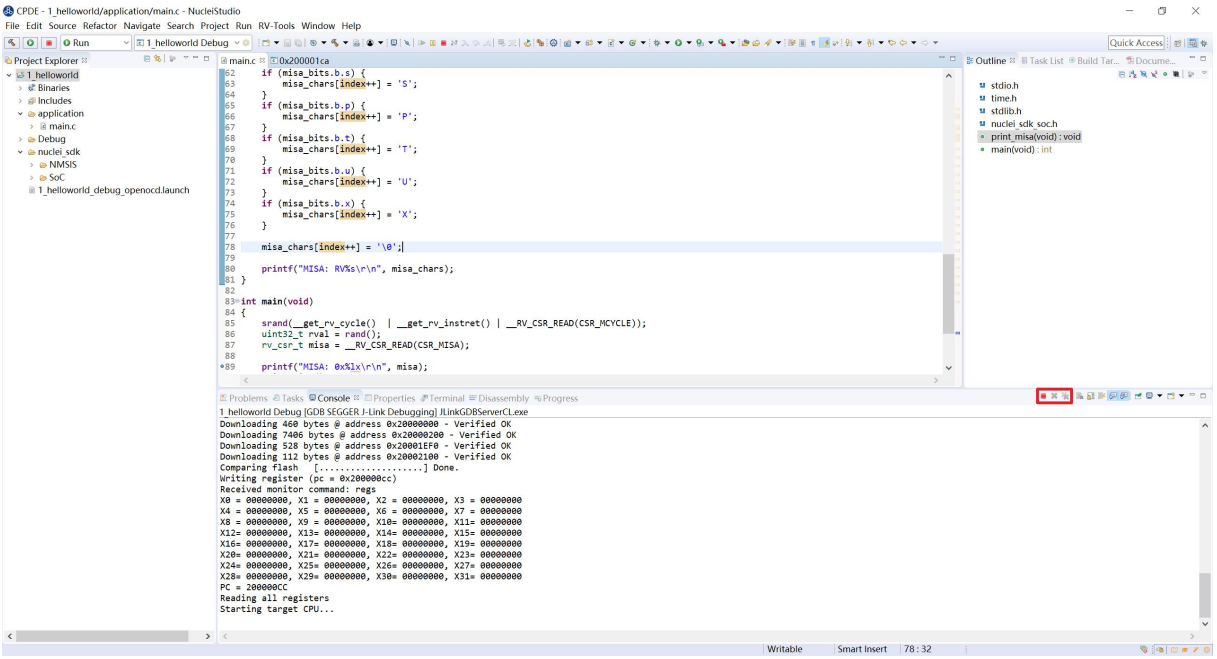


图 7- 50 下载完成单击红色按钮断开连接

7.4.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。

图 7- 51 Dlink 实物图

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：

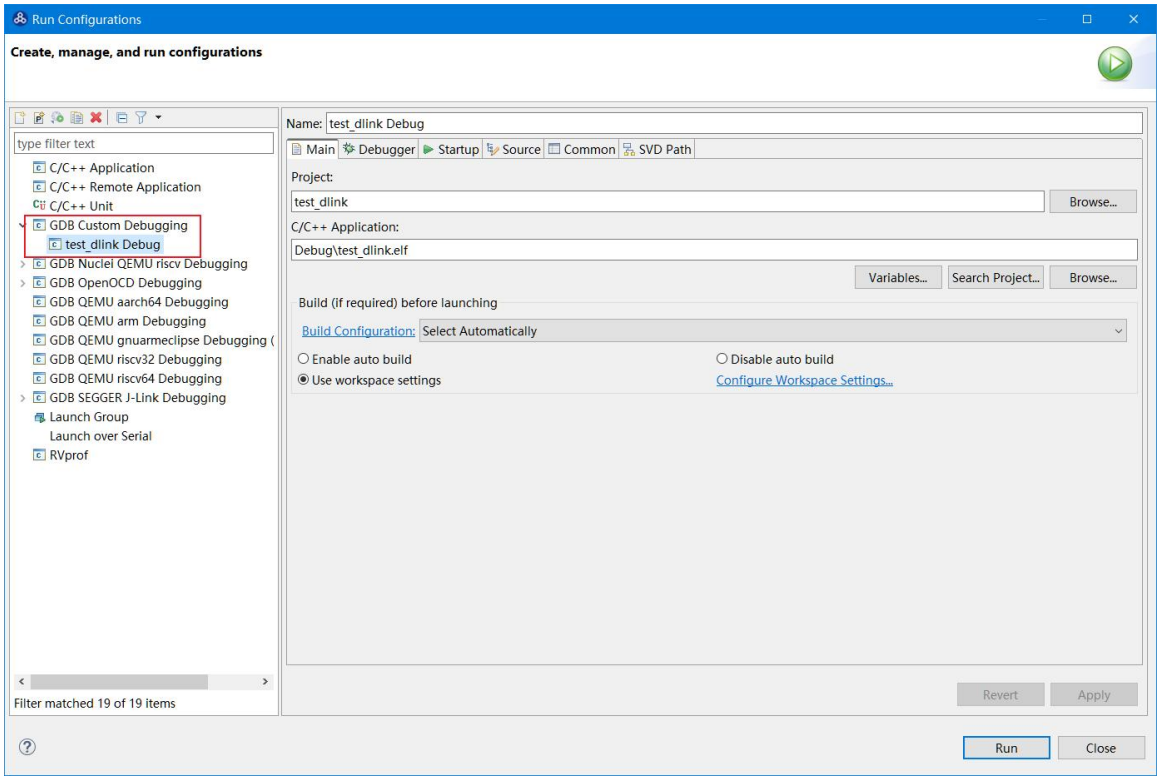


图 7- 52 Run Configurations 的配置页面

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

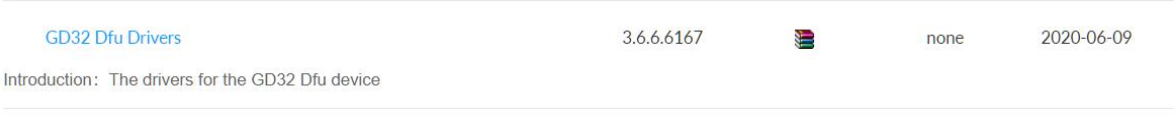


图 7- 53 下载 GD 32 Dfu Drivers

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

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 >。

■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。

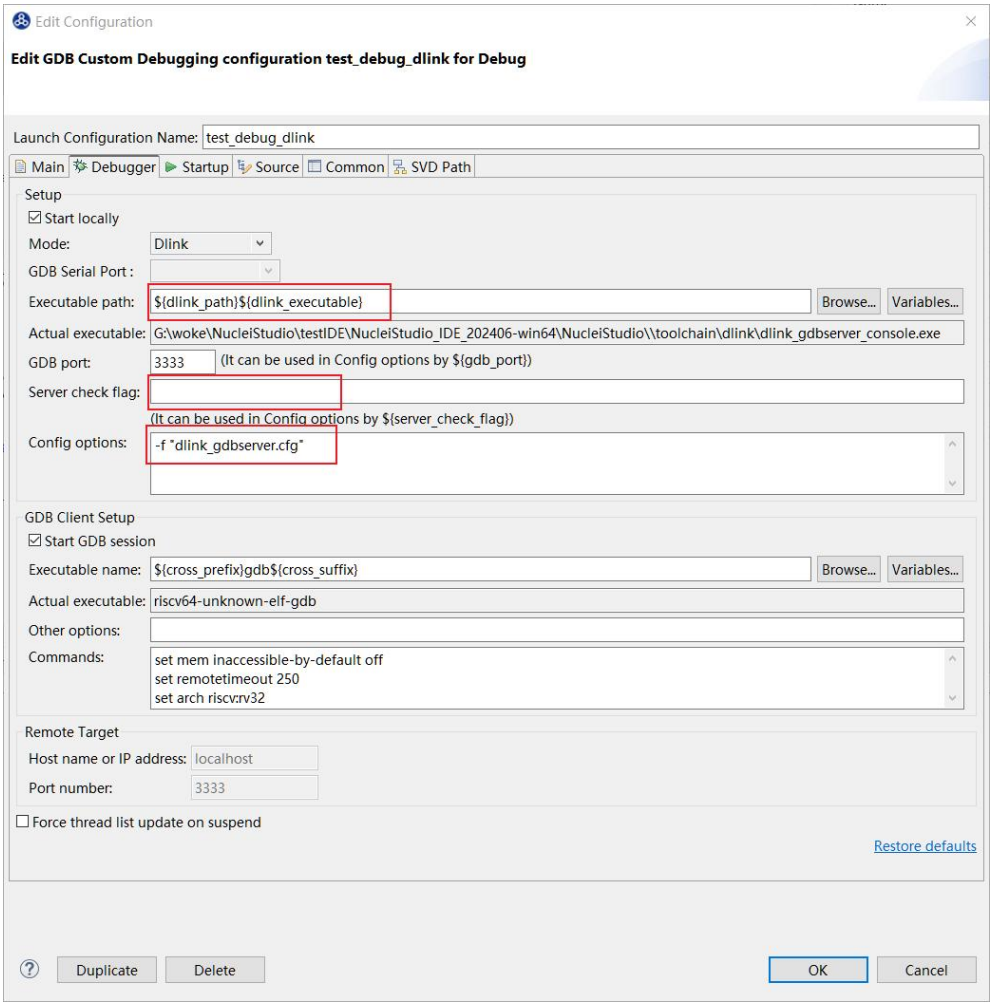


图 7- 54 Custom Debugging 配置页面

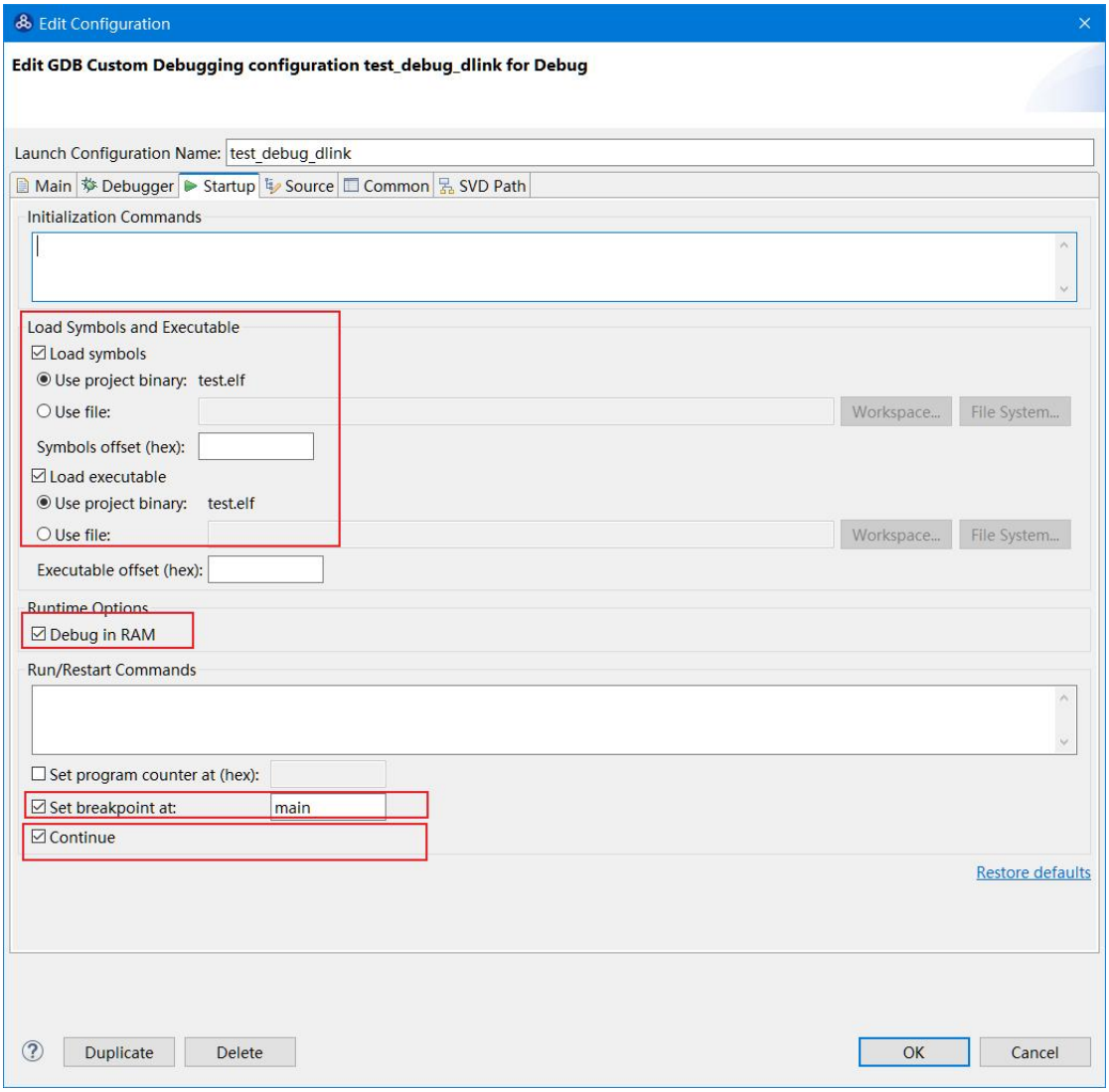


图 7- 55 Custom Debugging 配置页面

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。

图 7-56 Dlink 处理工作状态

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：

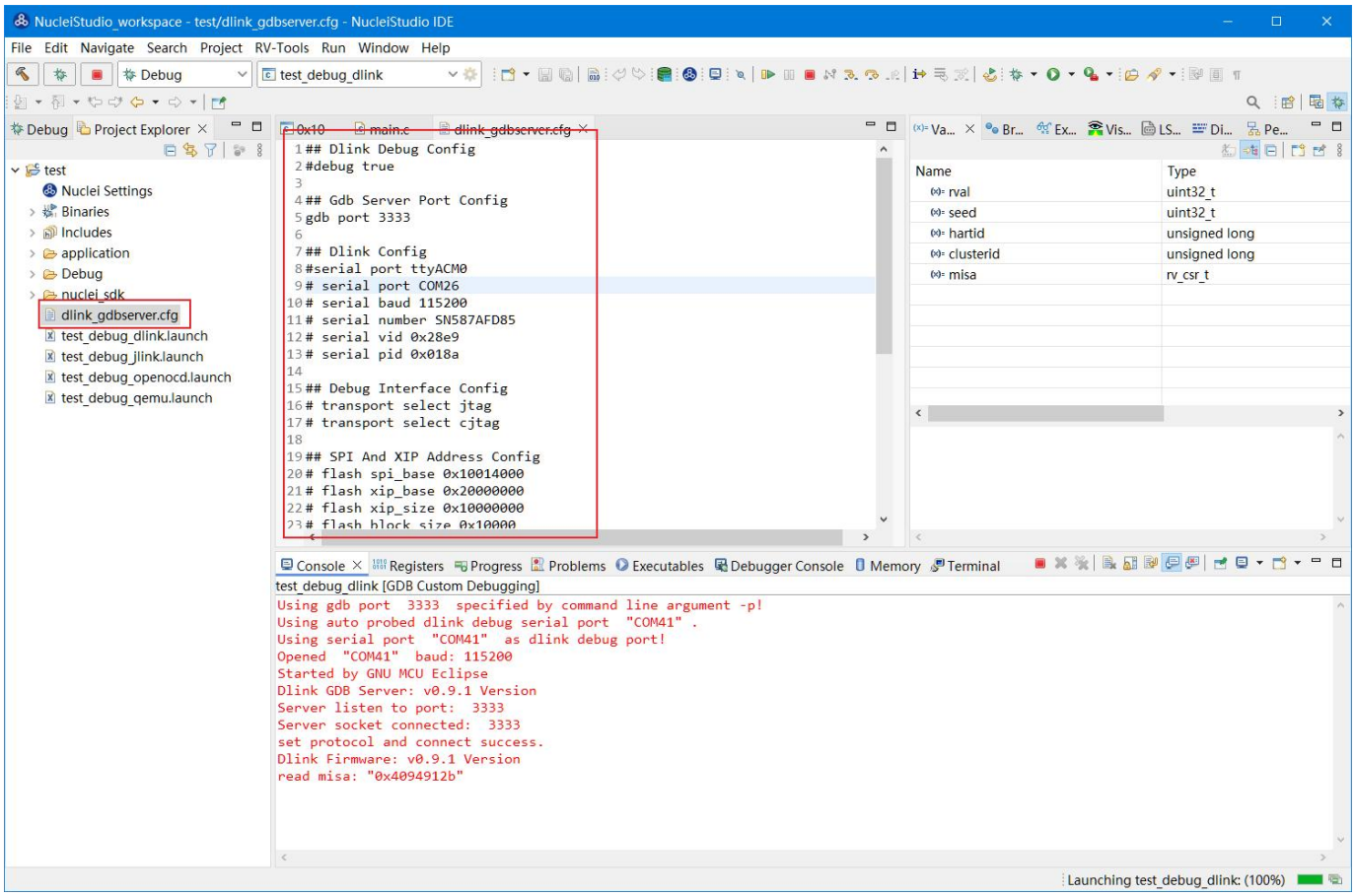


图 7-57 dlink_gdbserver.cfg 内容样例

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

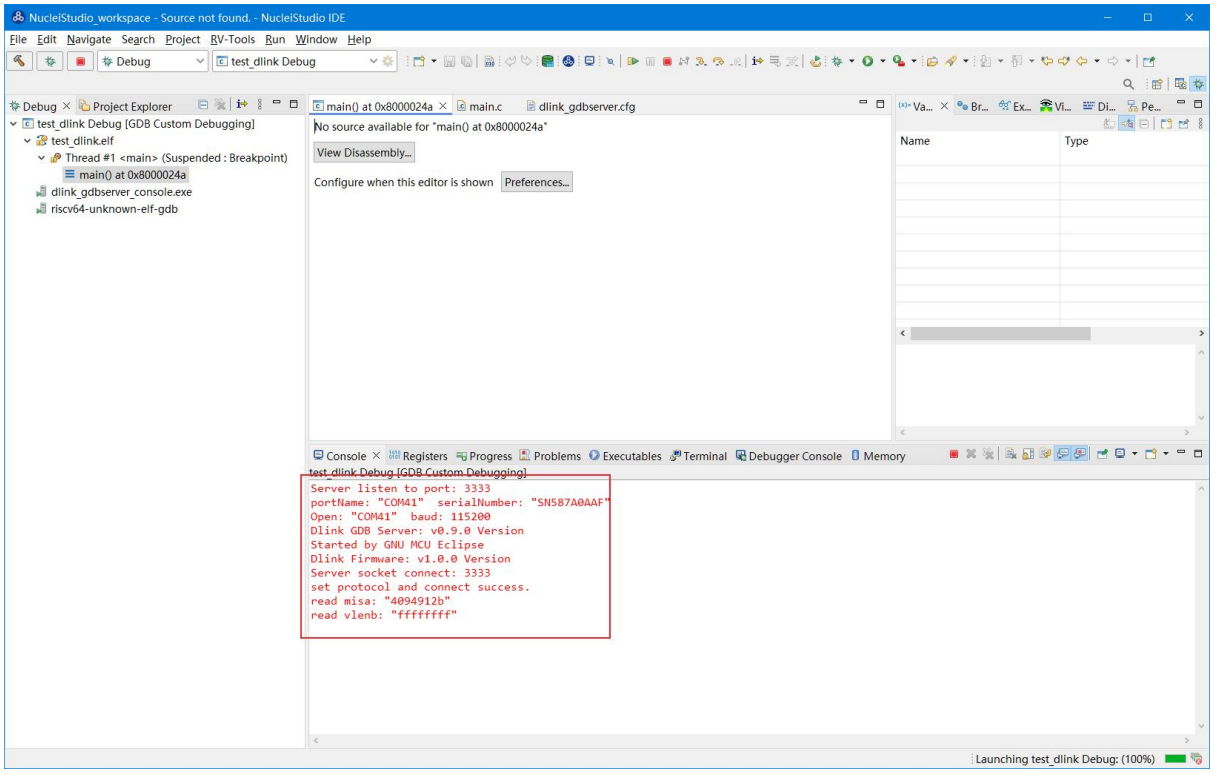


图 7- 58 Dlink 处理 debug 中

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。

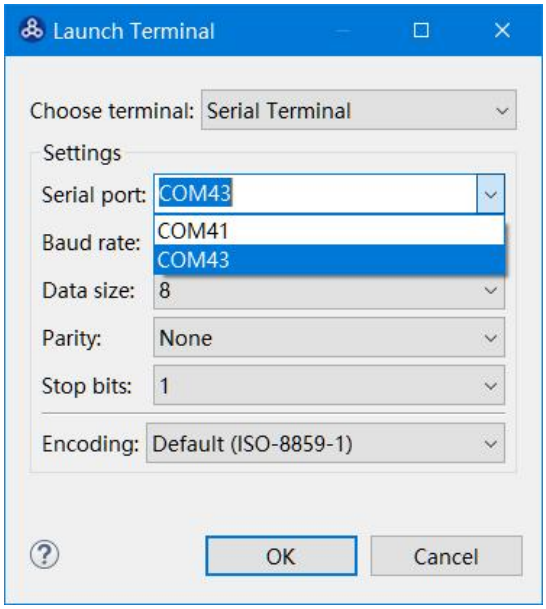


图 7- 59 联接串口输出

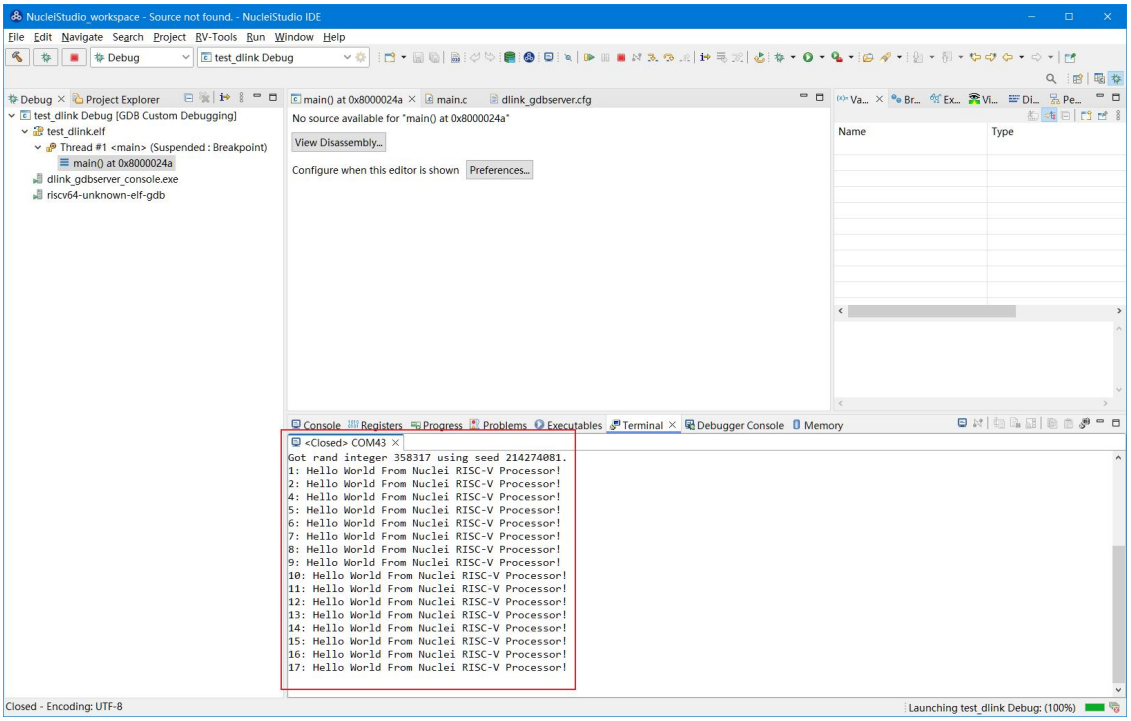


图 7- 60 Dlink 调试时串口输出

8. 导入旧版本 Nuclei Studio 创建的工程

8.1.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，可以阅读章节 8.2 内容），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。

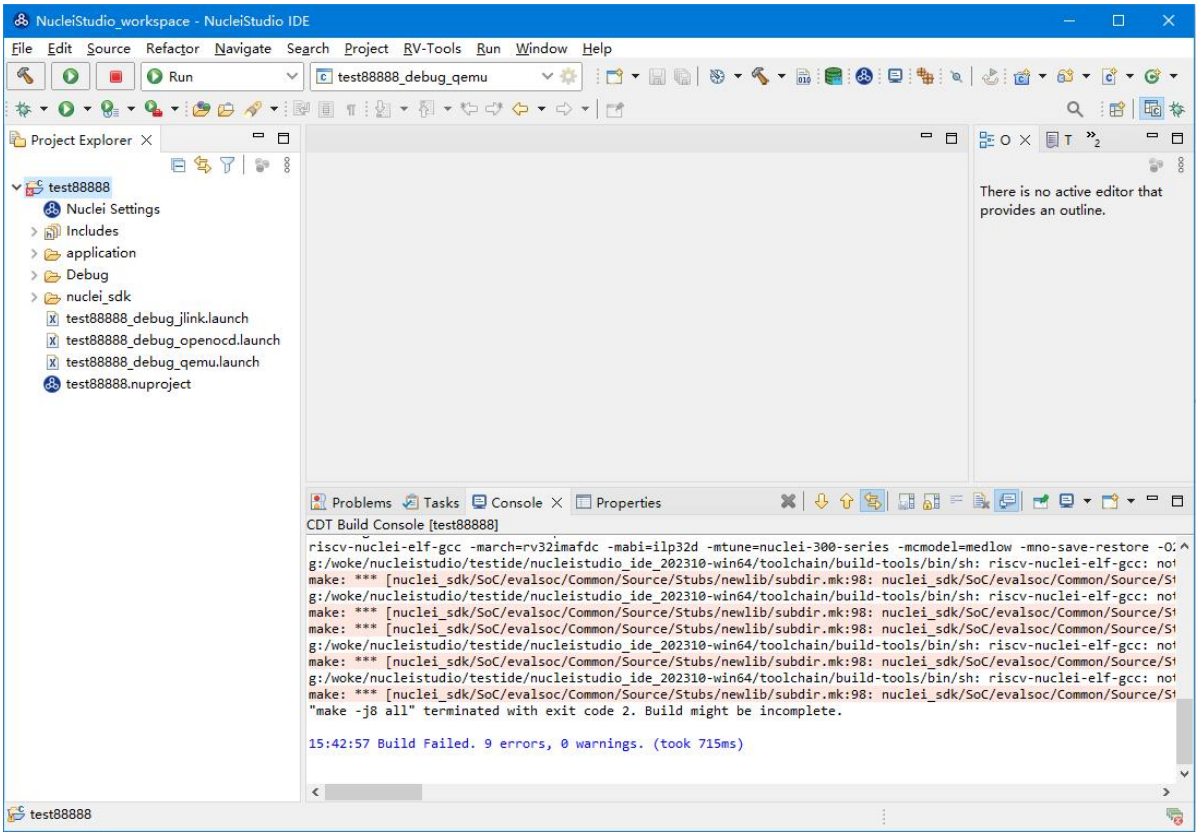


图 8-1 工程导入后编译报错

8.1.1.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。

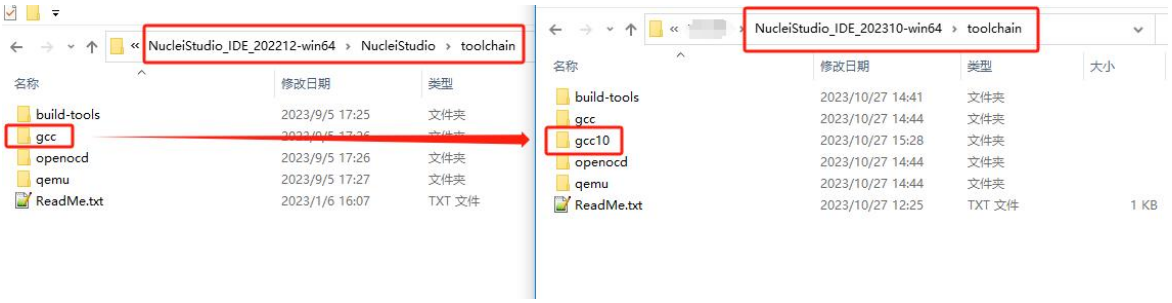


图 8- 2 手动导入 GCC 10

8.1.2.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。如图 8-3，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。

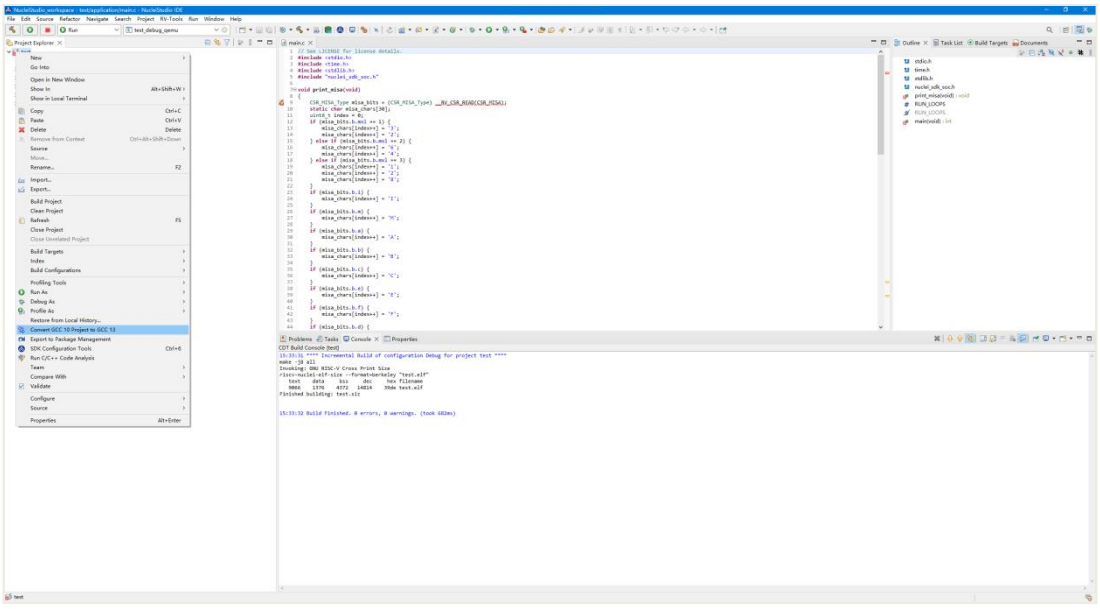


图 8- 3 Convert GCC 10 Project to GCC 13

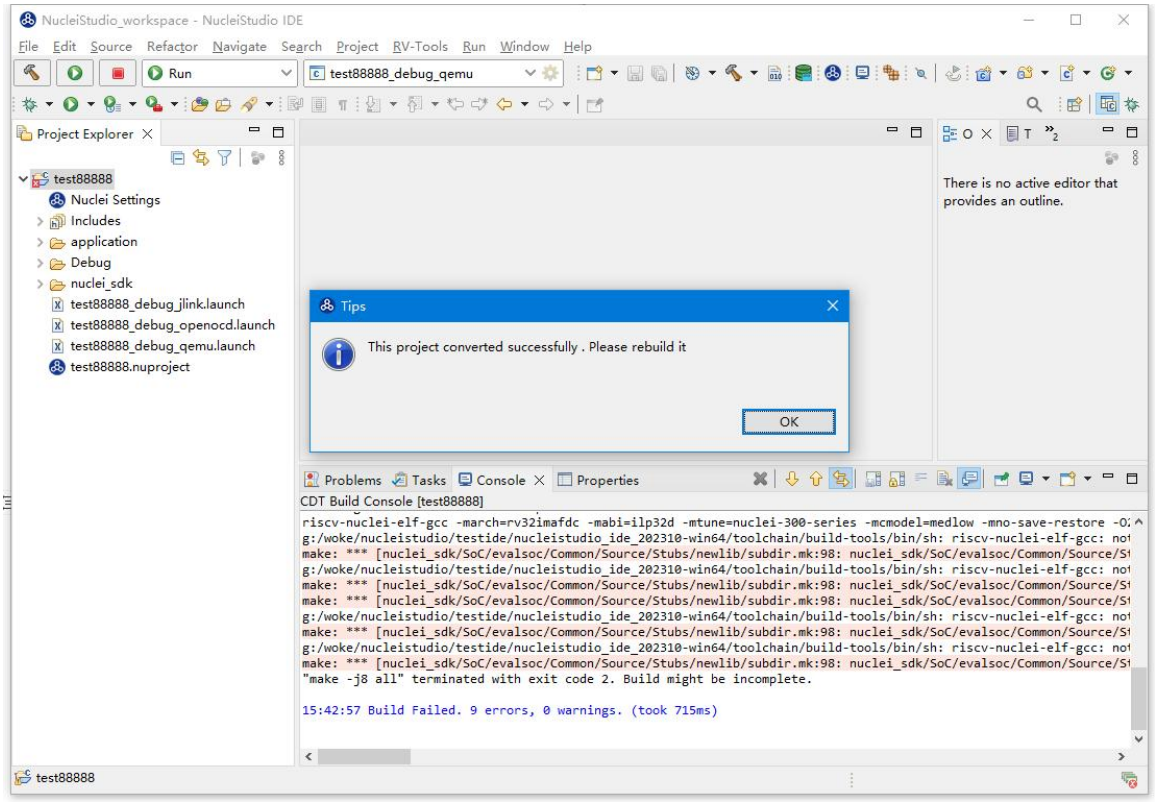


图 8- 4 Convert Project to GCC 13 OK

8.1.3.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。如图 8-5，打开转换工具，然后选择需要转换的工作，开始转换。工程转换完成后，页面会显示转换的结果。

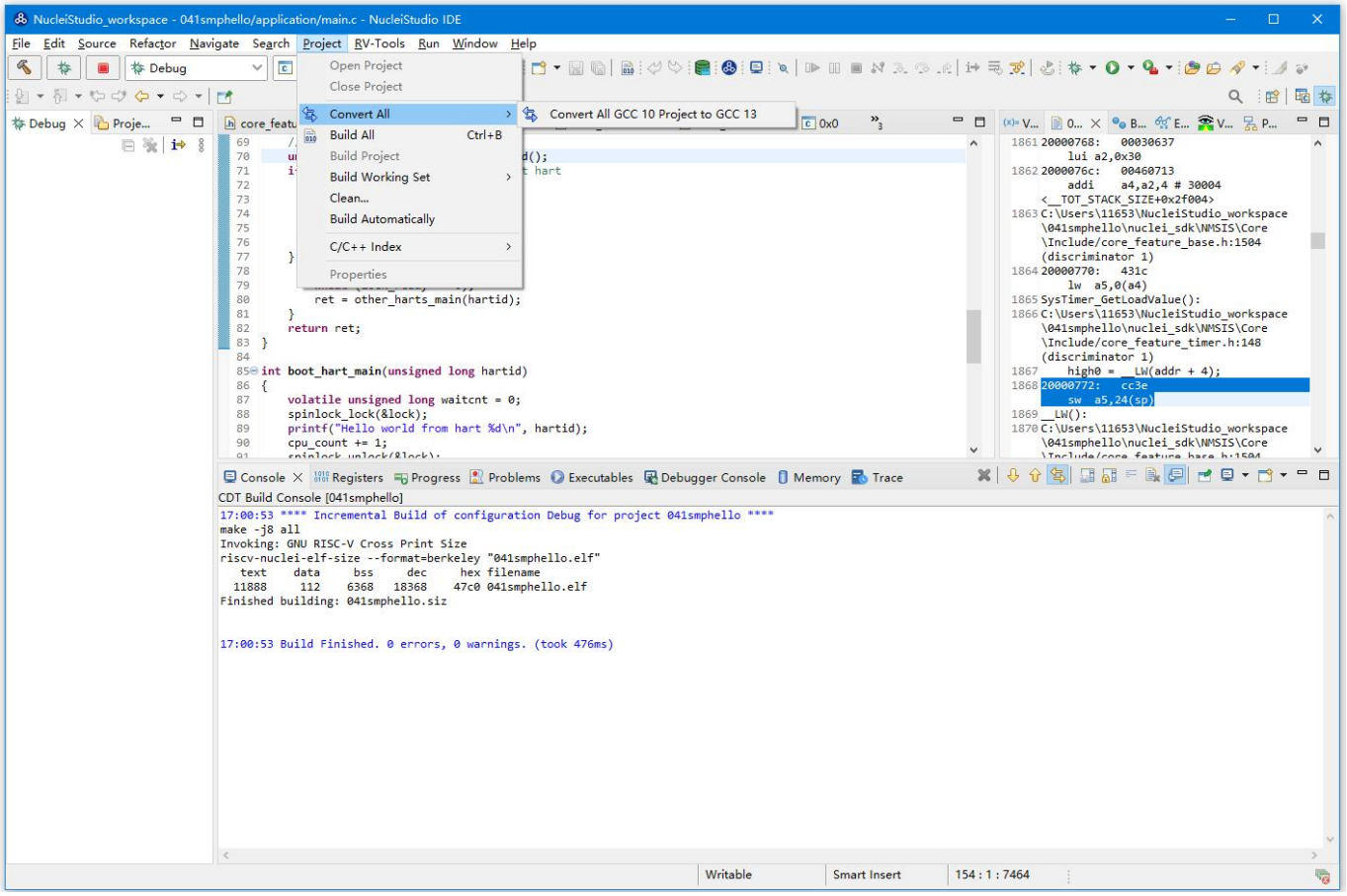


图 8- 5 打开 Convert All

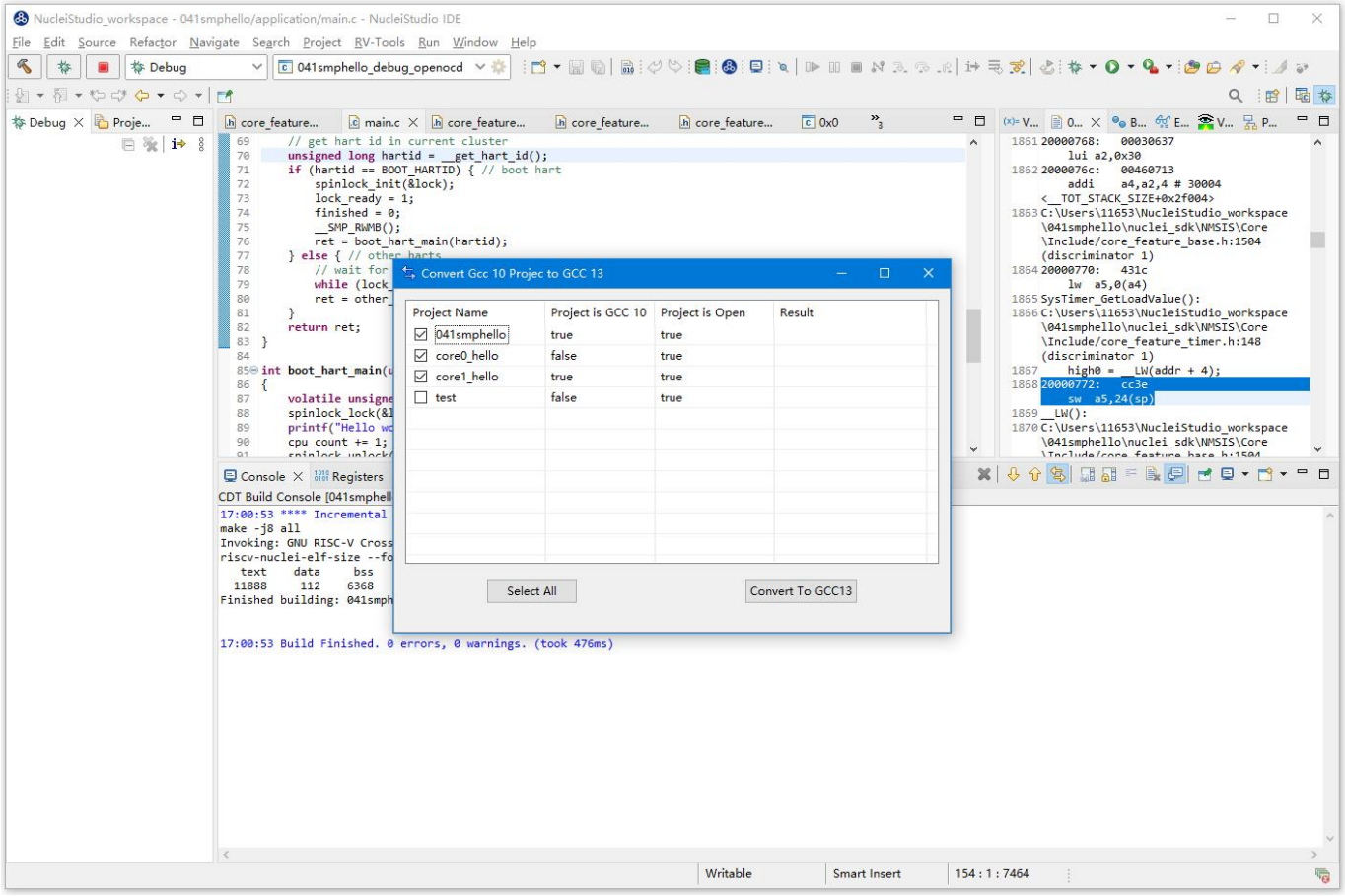


图 8-6 选择需要转换的工程

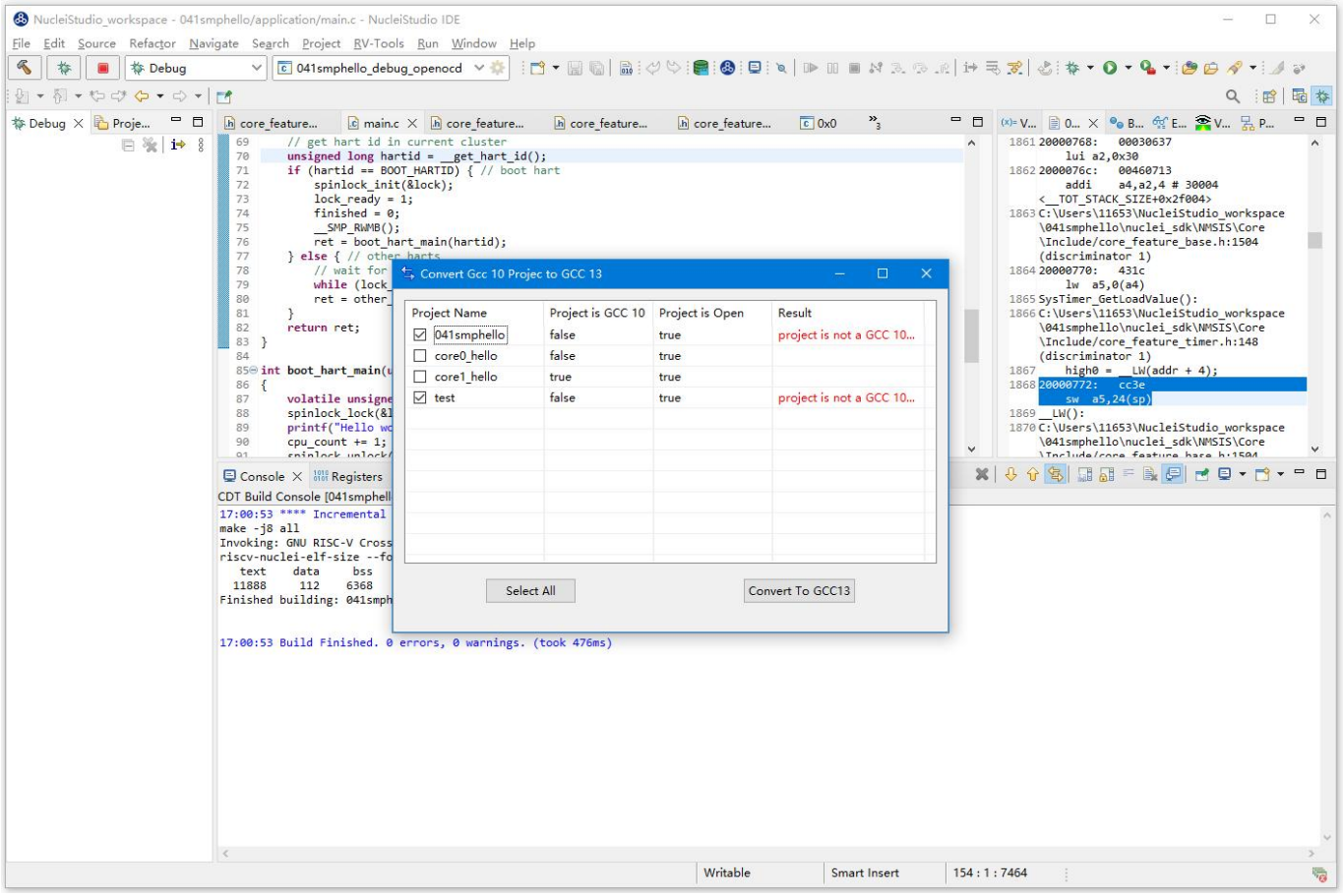


图 8-7 转换结果

8.2. 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。

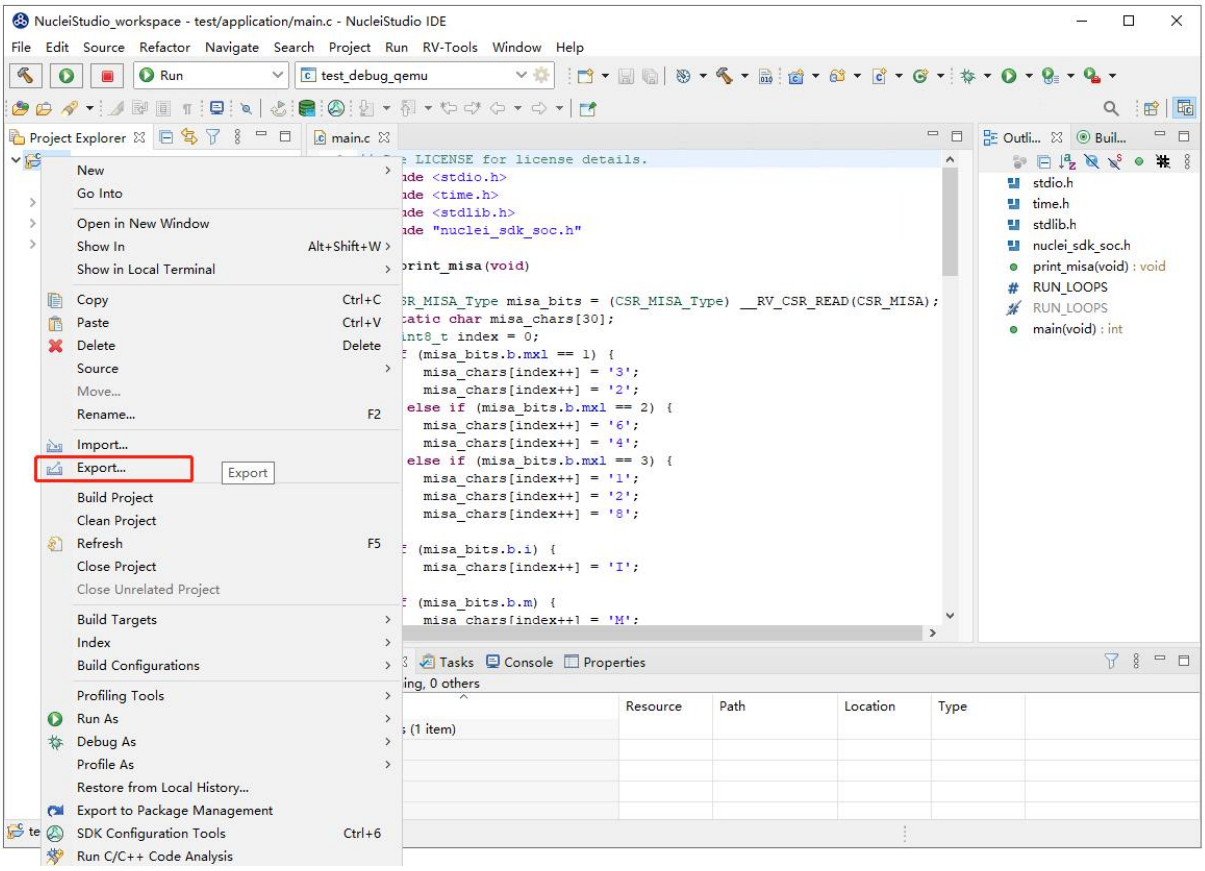


图 8-8 选择 Export 菜单

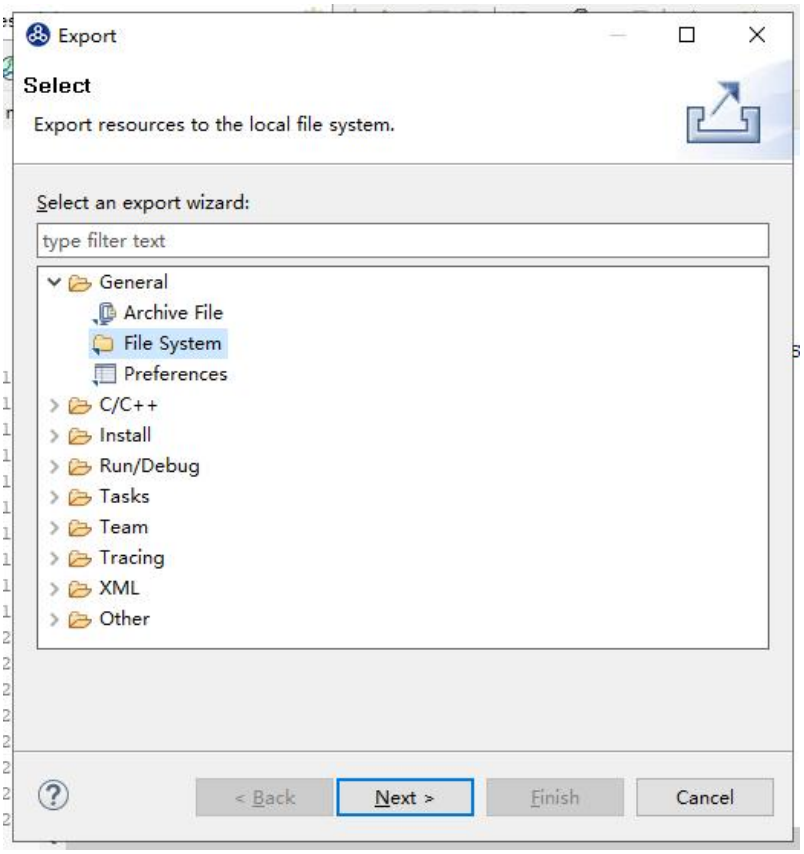


图 8-9 Export 工程

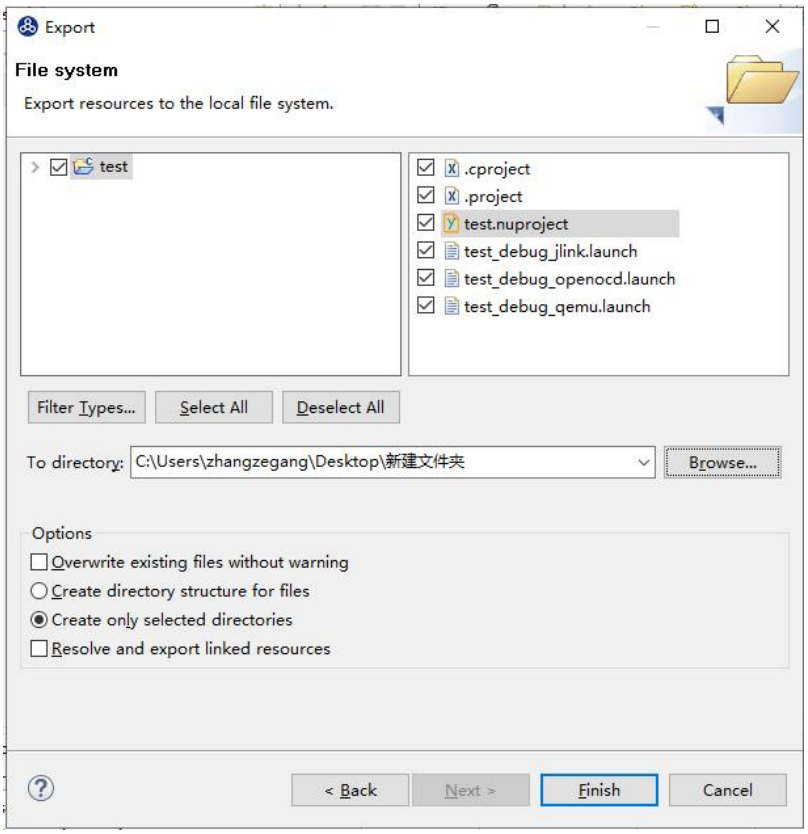


图 8-10 指定导出的目录及相应文件

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，具体的可以参考章节 5.2.2 内容。

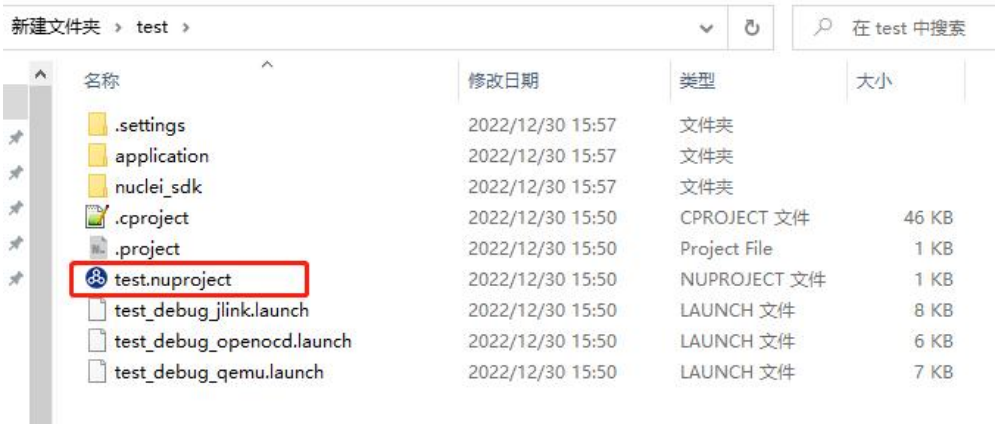


图 8-11 查看导出的工程

8.3. 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 GNU MCU RISC-V GCC (riscv-none-embed-gcc) 升级到 RISC-V Nuclei GCC (riscv-nuclei-elf-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)”，如所示。修改后点击 Apply 保存修改。

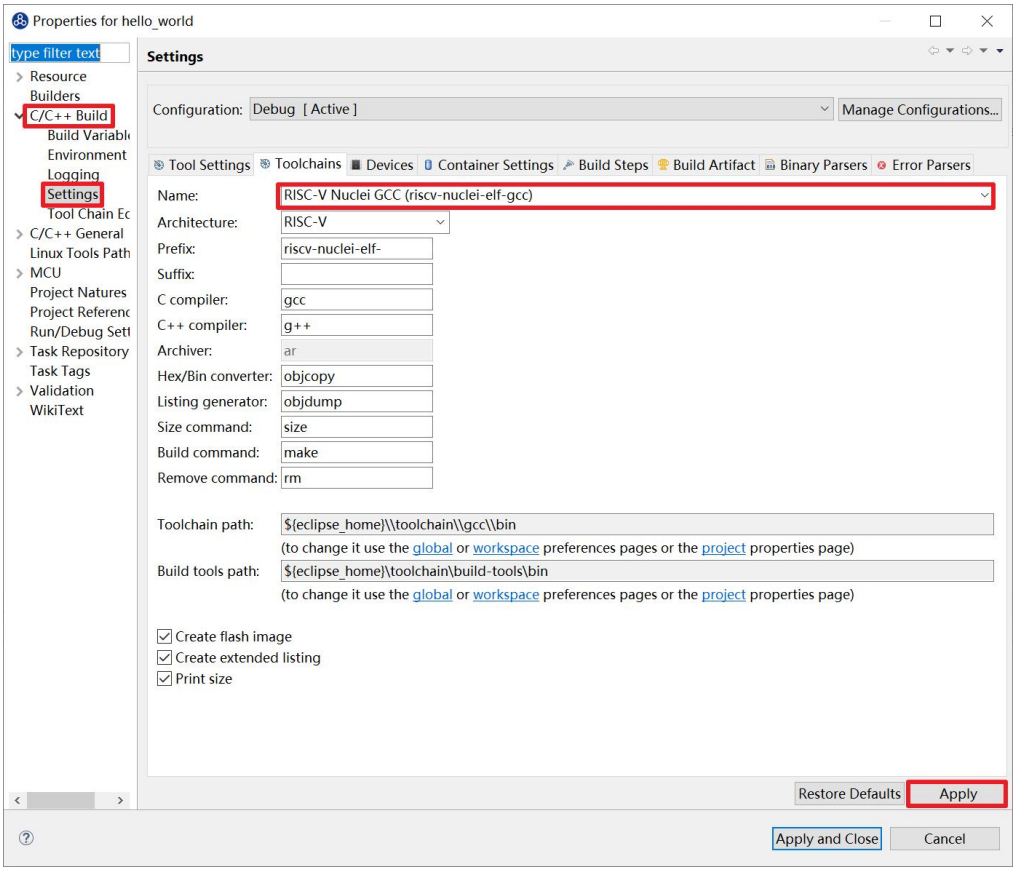


图 8- 12 修改 Nuclei Studio 工具链

■修改后右击工程选择“Clean Project”再选择“Build Project”即可。

9.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。

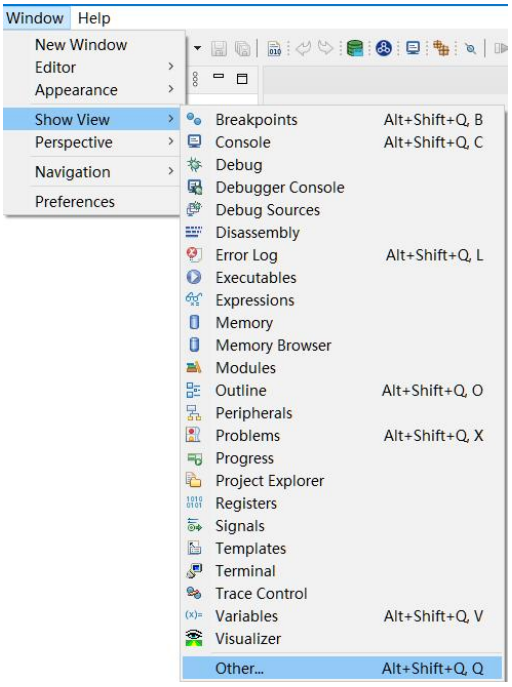


图 9- 1 打开 Show View

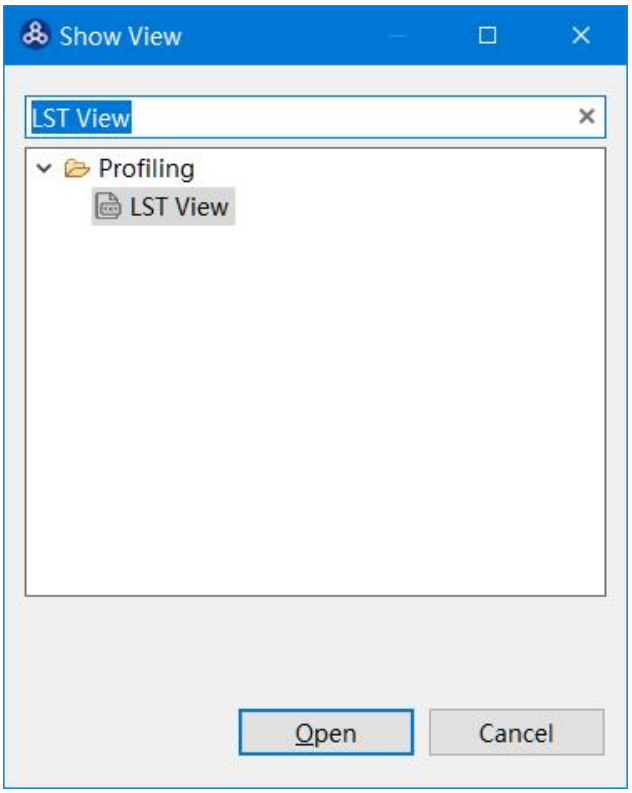


图 9- 2 打开 LST View

LST View 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。

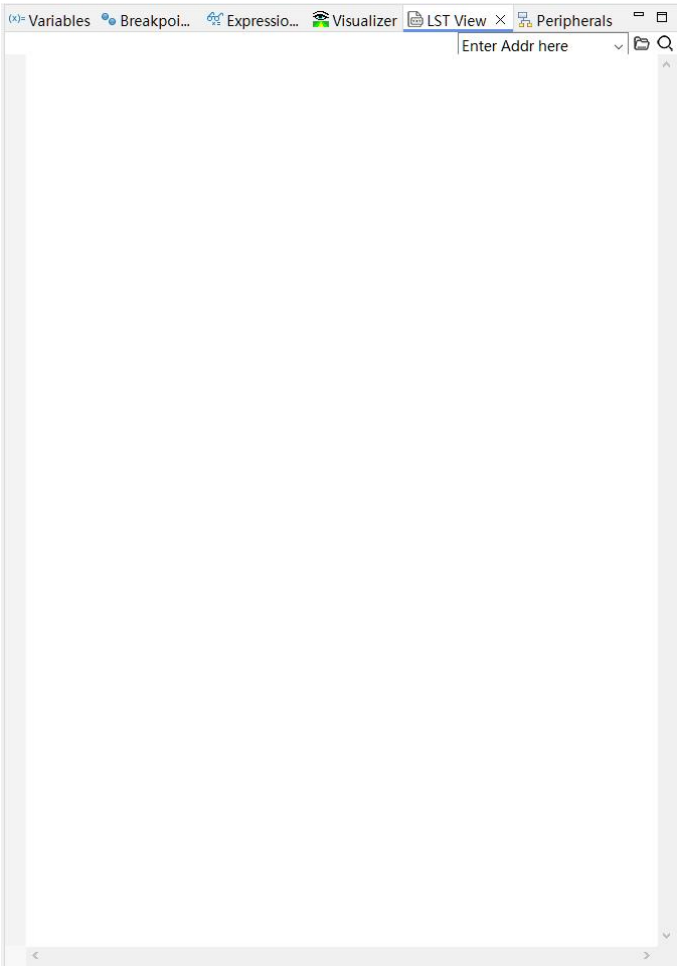


图 9- 3 LST View 上的菜单

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。

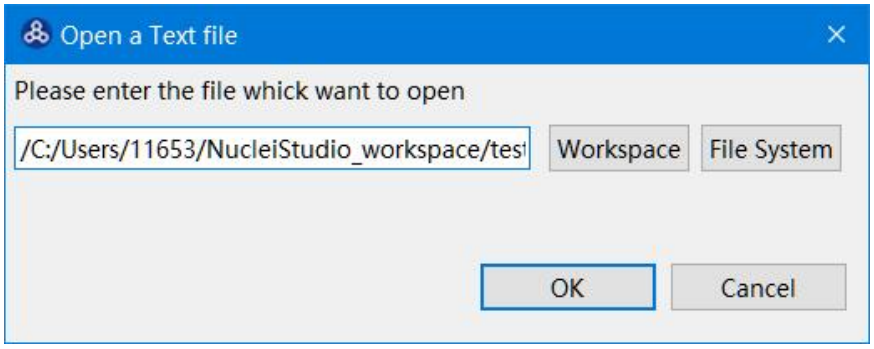


图 9-4 LST View 打开 lst 文件

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。

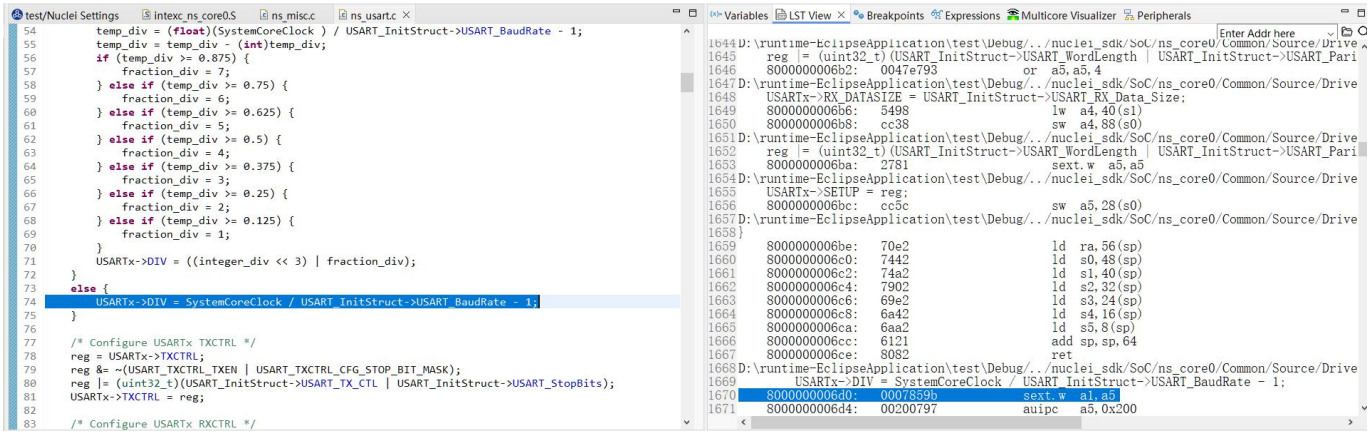


图 9-5 LST View 中 lst 与源码联动

10.Code Coverage 和 Profiling 功能

在 Nuclei Studio 2023.10 版以上版本中，集成了 Eclipse Linux Tools，并对 Eclipse Linux Tools 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 的功能做了进一步的优化和升级，使其更容使用。

关于 Eclipse Linux Tools 的详细参见

- Eclipse Linux Tools: Release notes

10.1.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 过带特定--coverage 编译选项编译指定源码文件，在实际开发板上运行，并配合 semihost 功能则可以收集需要的 coverage 文件(gcda/gcno 文件)， 则可以在 gcov 工具的配合下，以图形化的方式展示。

- .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 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。

需要注意的是进行 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))
- Gcov Data Files (Using the GNU Compiler Collection (GCC))
- Code Coverage for Embedded Target with Eclipse, gcc and gcov | MCU on Eclipse

10.2.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，并配合 semihost 功能则可 以收集需要的 gmon.out 文件，在 IDE 上以图形化的方式展示。

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。

在 NucleiSudio 2024.06 版中使用 Profiling 方法如下

关于 Profiling 的功能详细参见

- Introduction (GNU gprof)
- Using GNU Profiling (gprof) With ARM Cortex-M - DZone

10.3.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。

10.3.1.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。

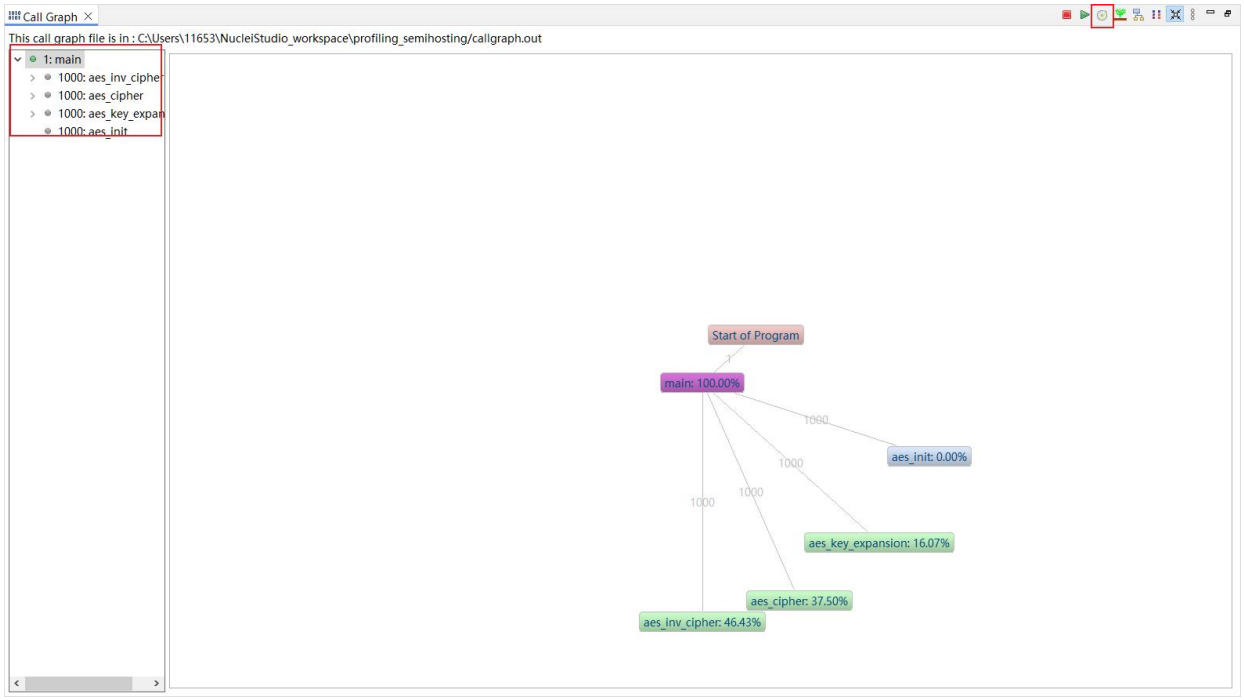


图 10- 1 Radial View

10.3.2.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。

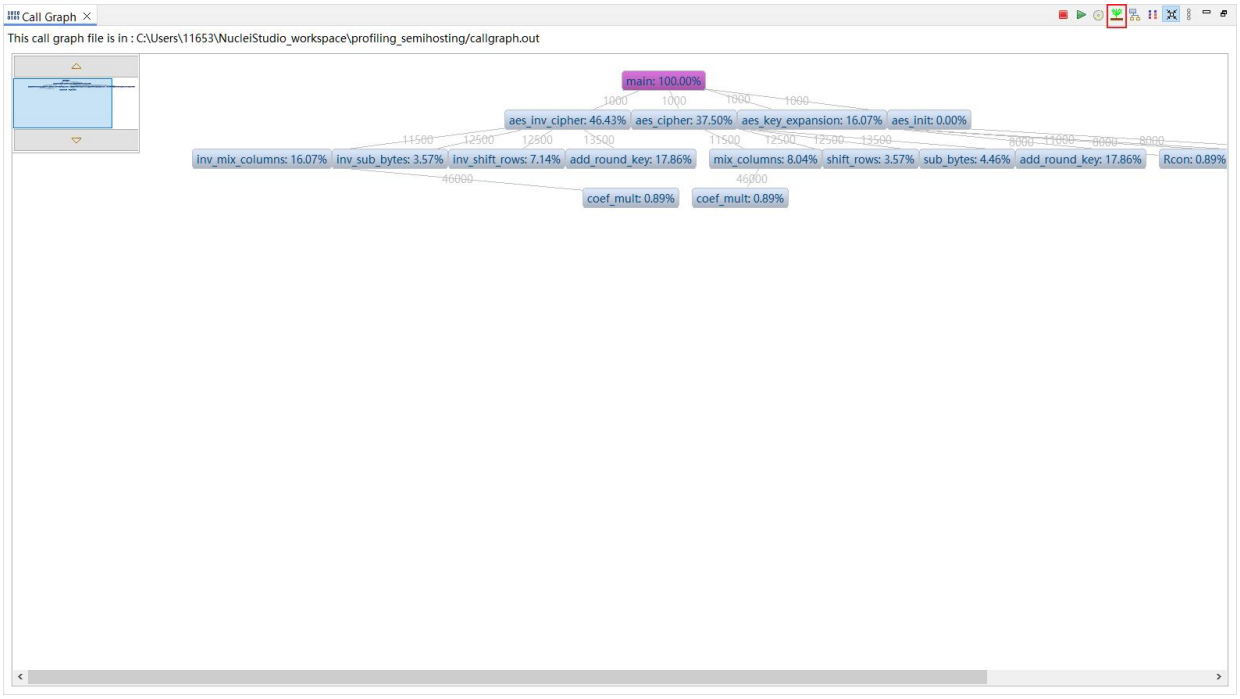


图 10- 2 Tree View

10.3.3.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。

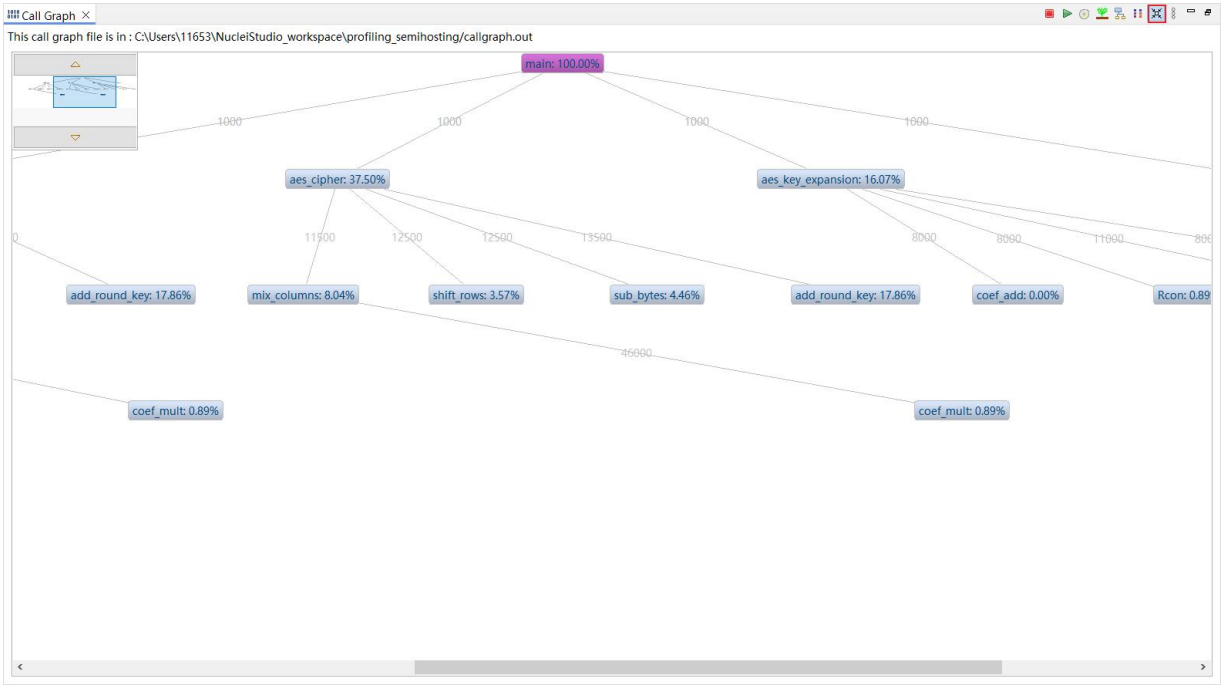


图 10- 3 Level View

10.3.4.Aggregate View

以方图的方式，非常直观的展示了程序的耗时关系。

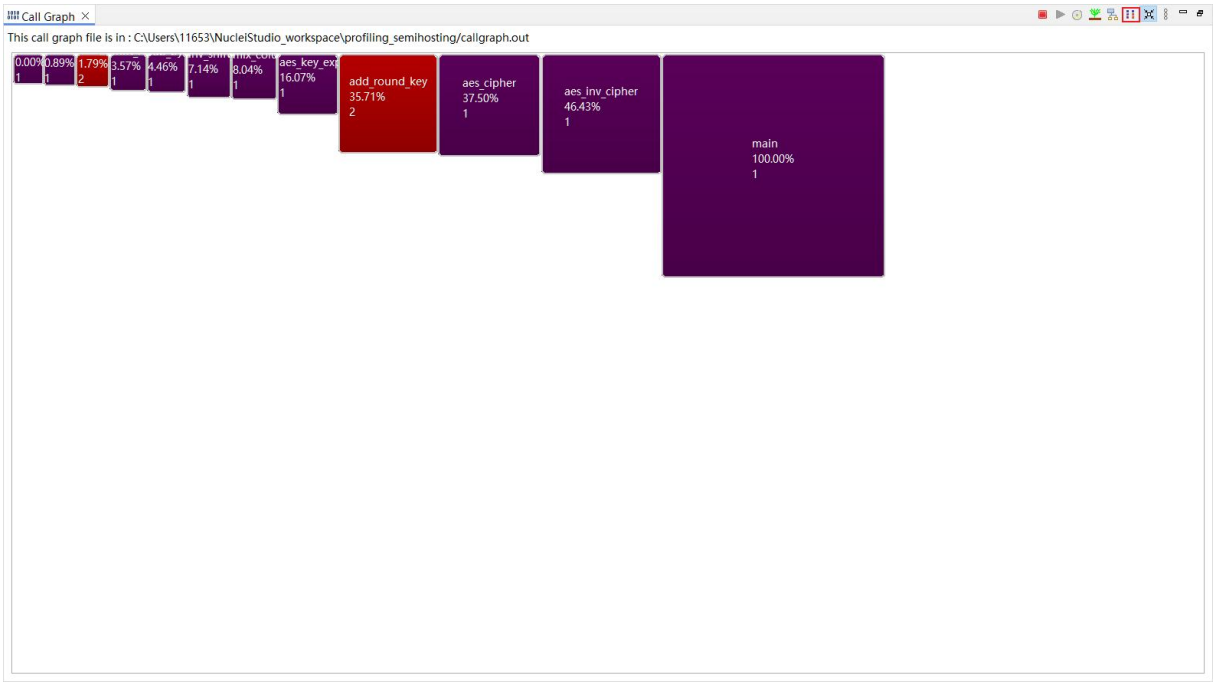


图 10- 4 Aggregate View

10.4.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。

10.4.1.通过串口使用

nuclei_sdk 0.6.0 及以上版本的 nulclei_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。

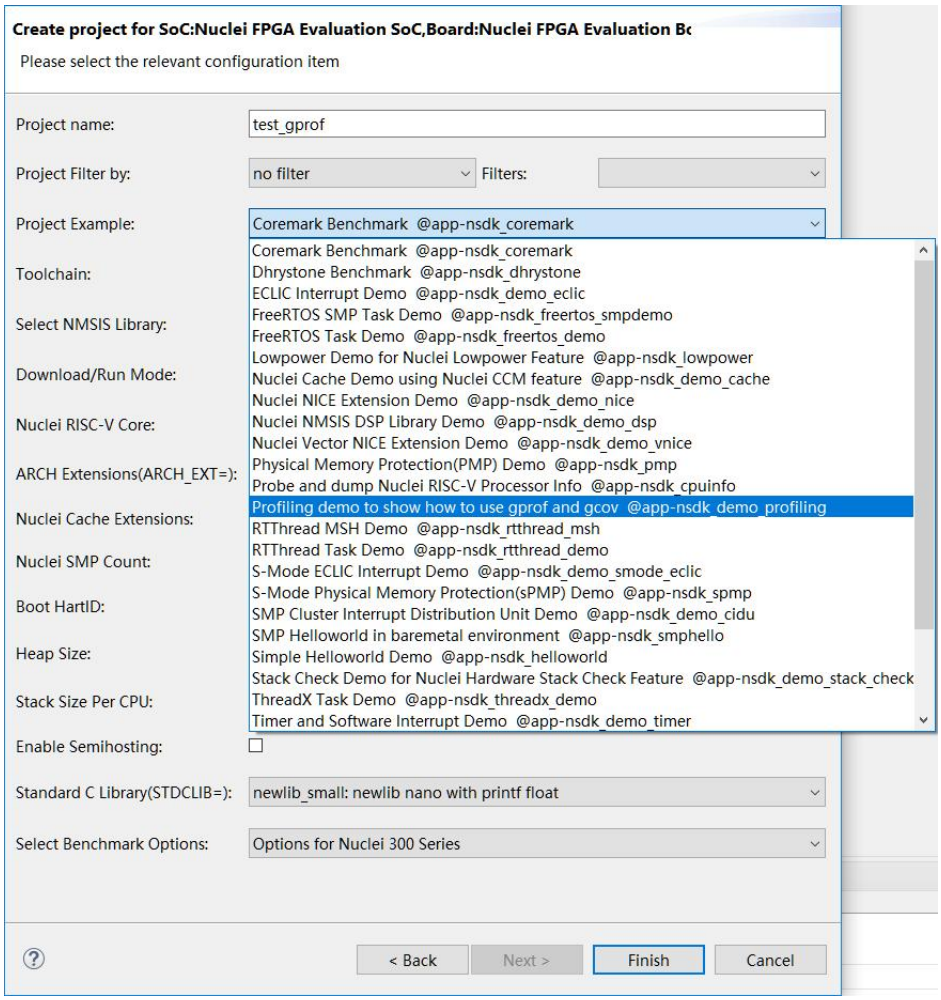


图 10- 5 创建测试工程

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 的编译选项，然后编译工程。

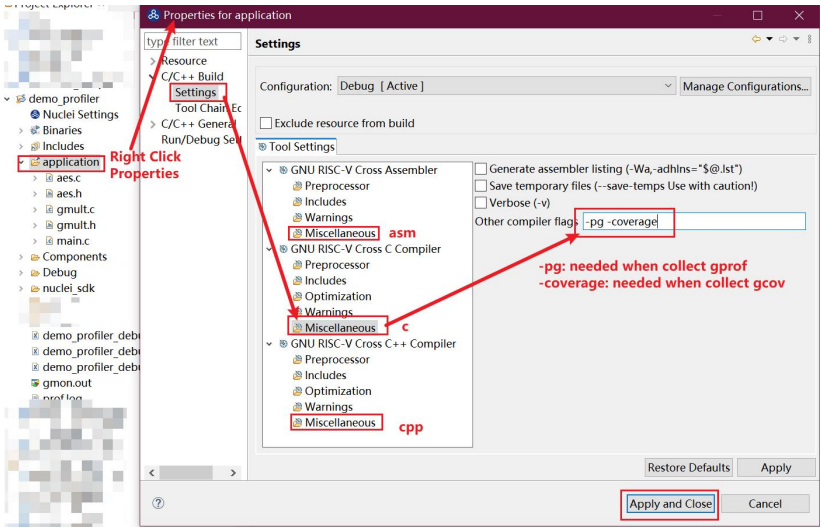


图 10- 6 设置-pg

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。

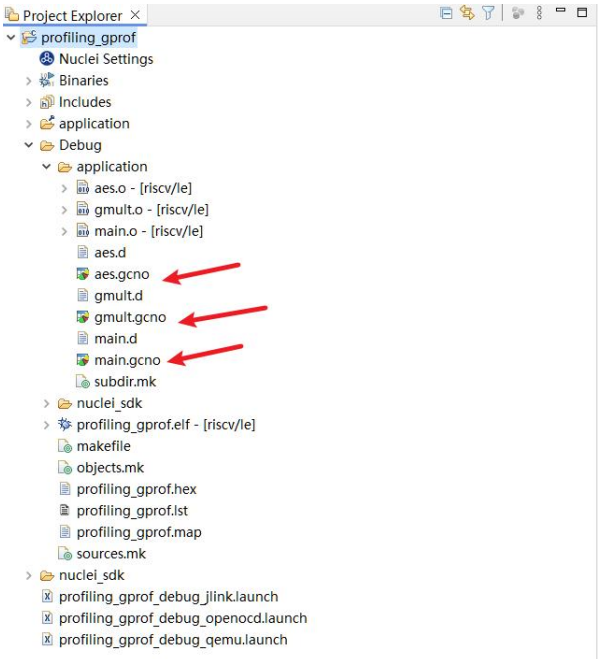


图 10- 7 编译测试工程

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。

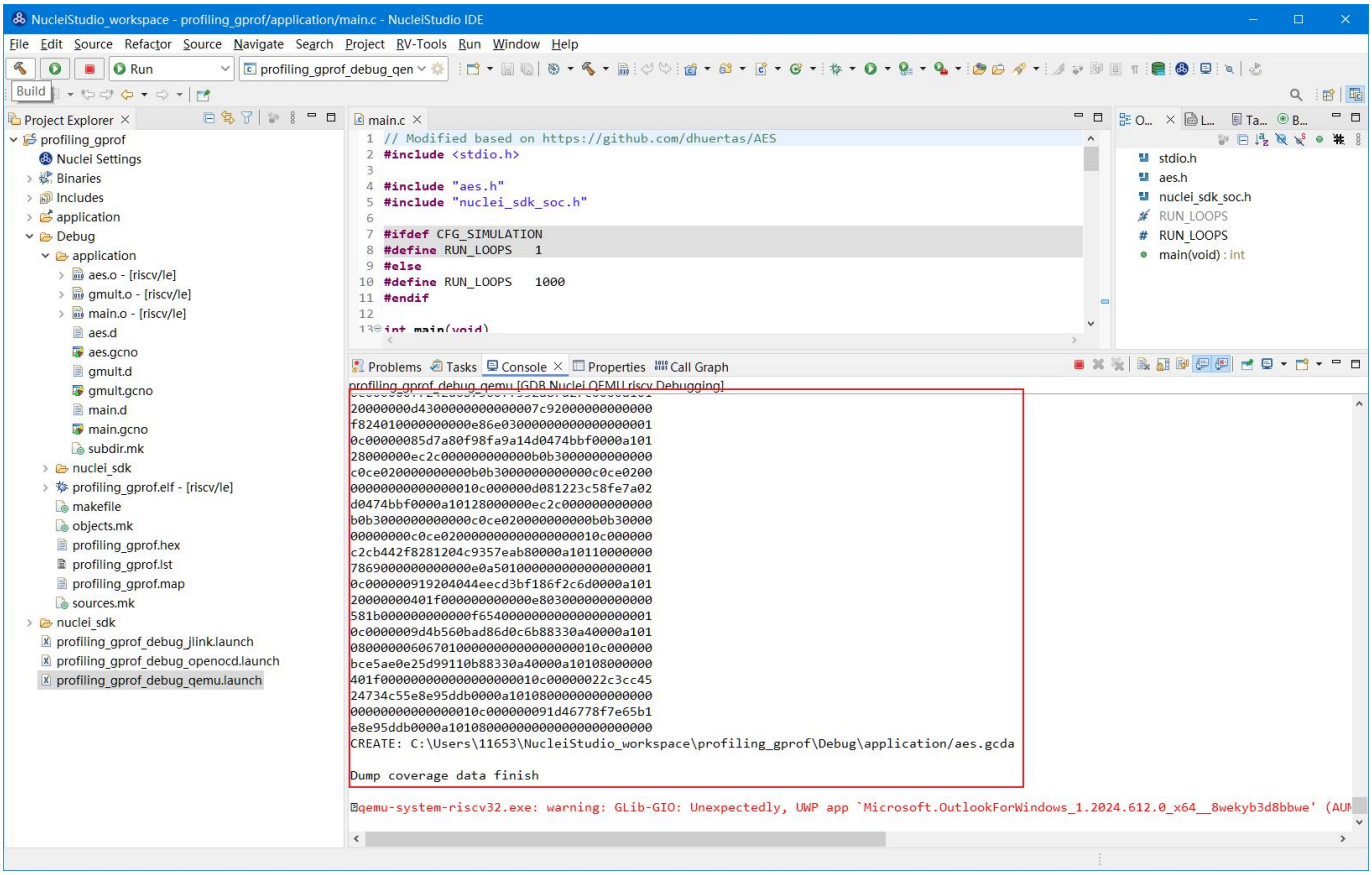


图 10-8 运行工程并查看 Profiling 信息

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，并将结果存别分存放在对应的文件中。

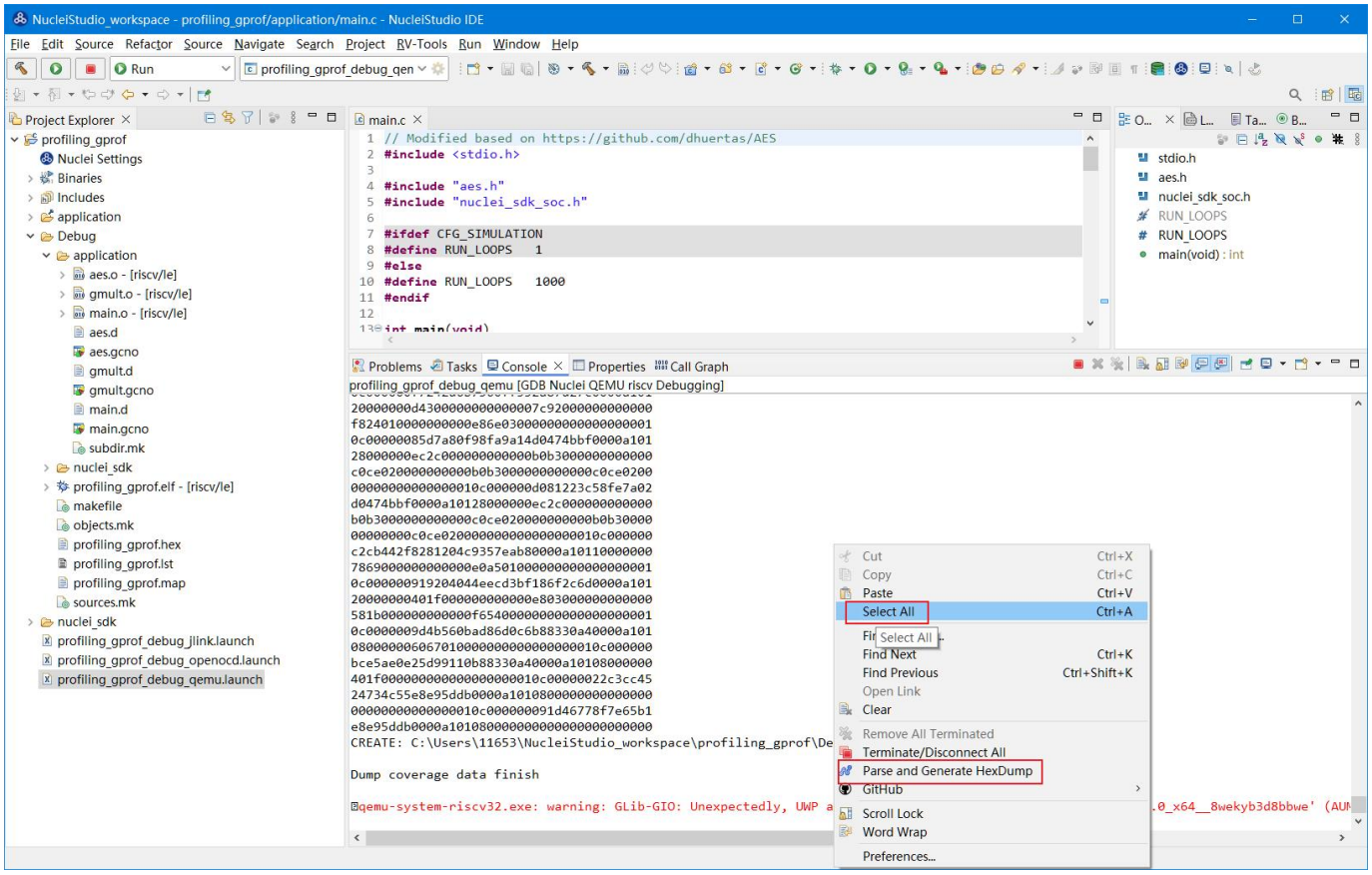


图 10- 9 Select All 并使用 Parse and Generate HexDump

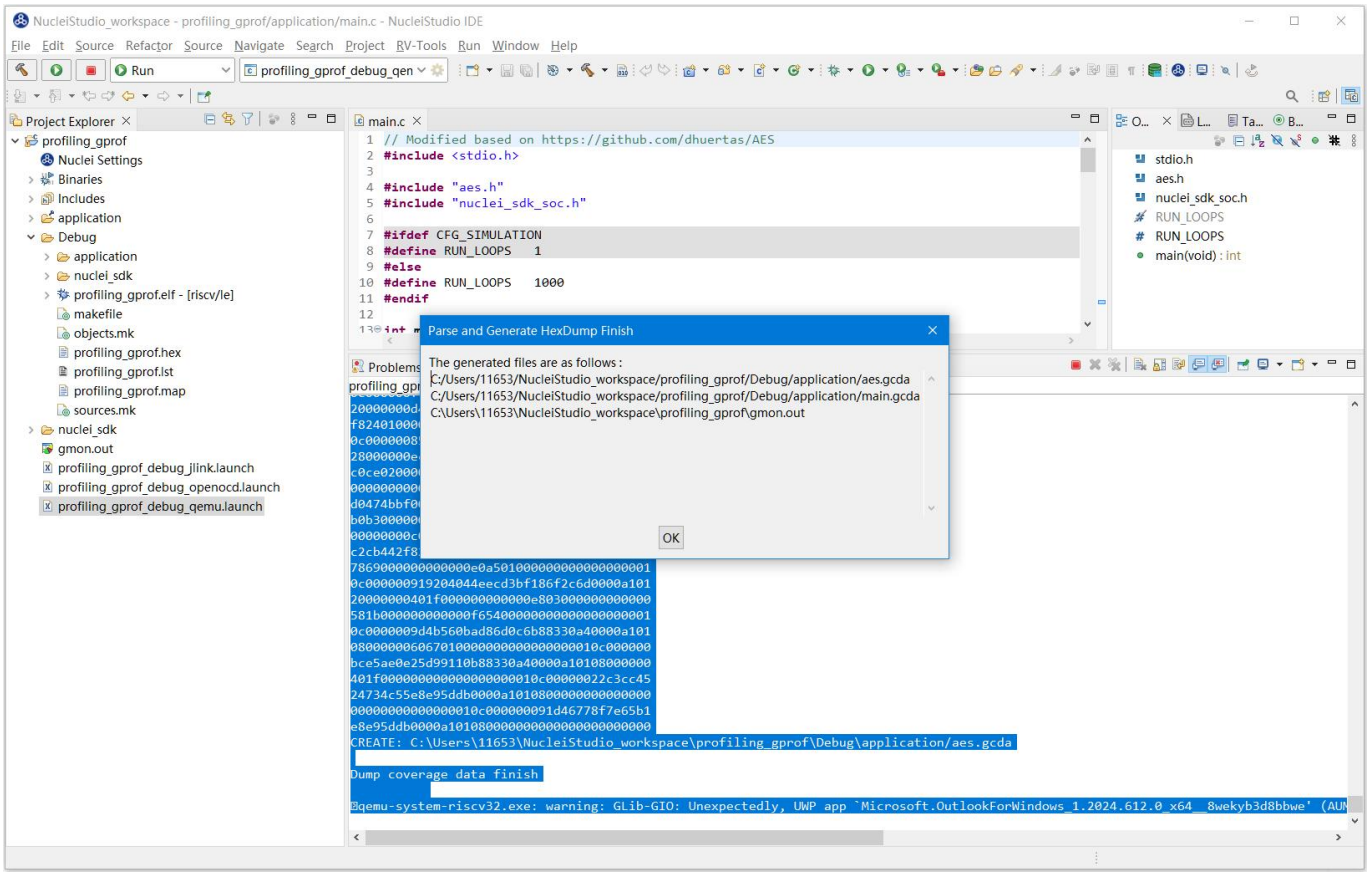


图 10- 10 解析 Profiling 信息并生成对应文件

再次查看工程的 Debug 目录，可以看到产生了对应的.gcda 文件。

图 10- 11 查看生成的对应文件

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，以及代码执行复盖比等数据。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。

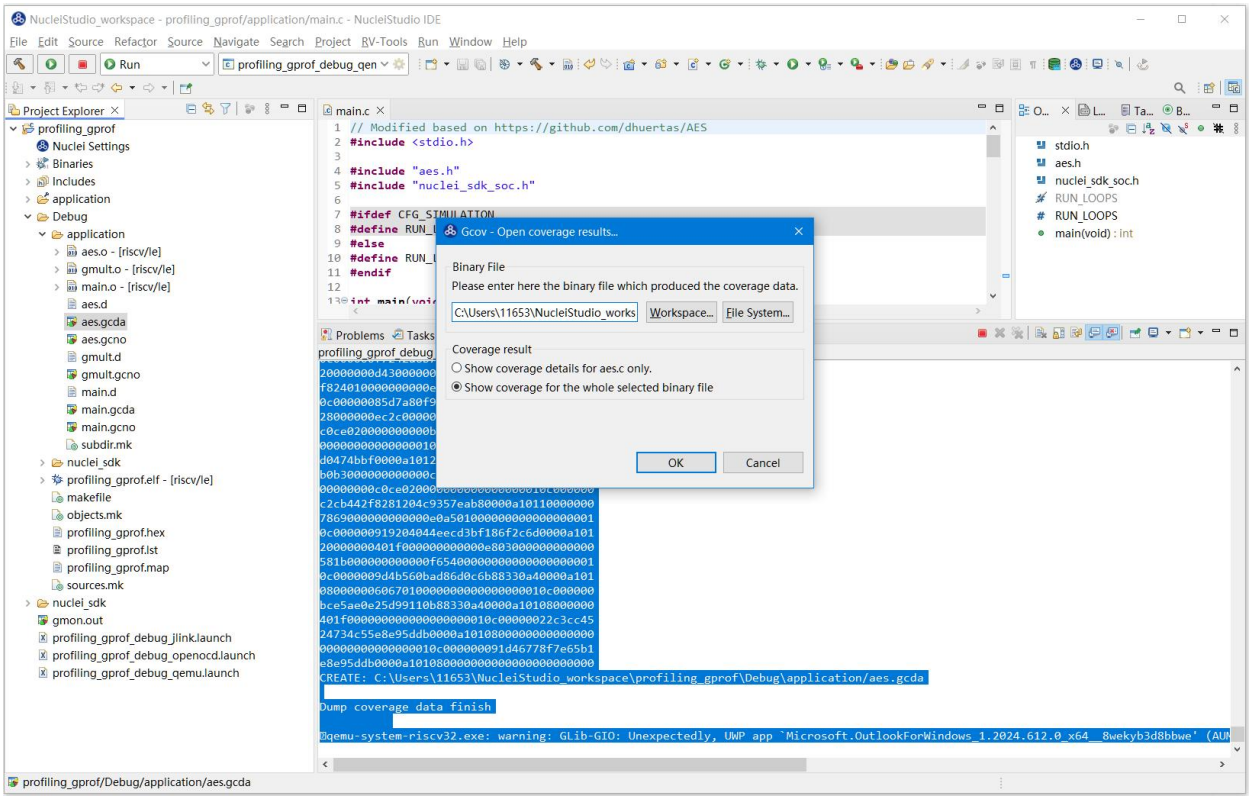


图 10- 12 打开.gcda 文件

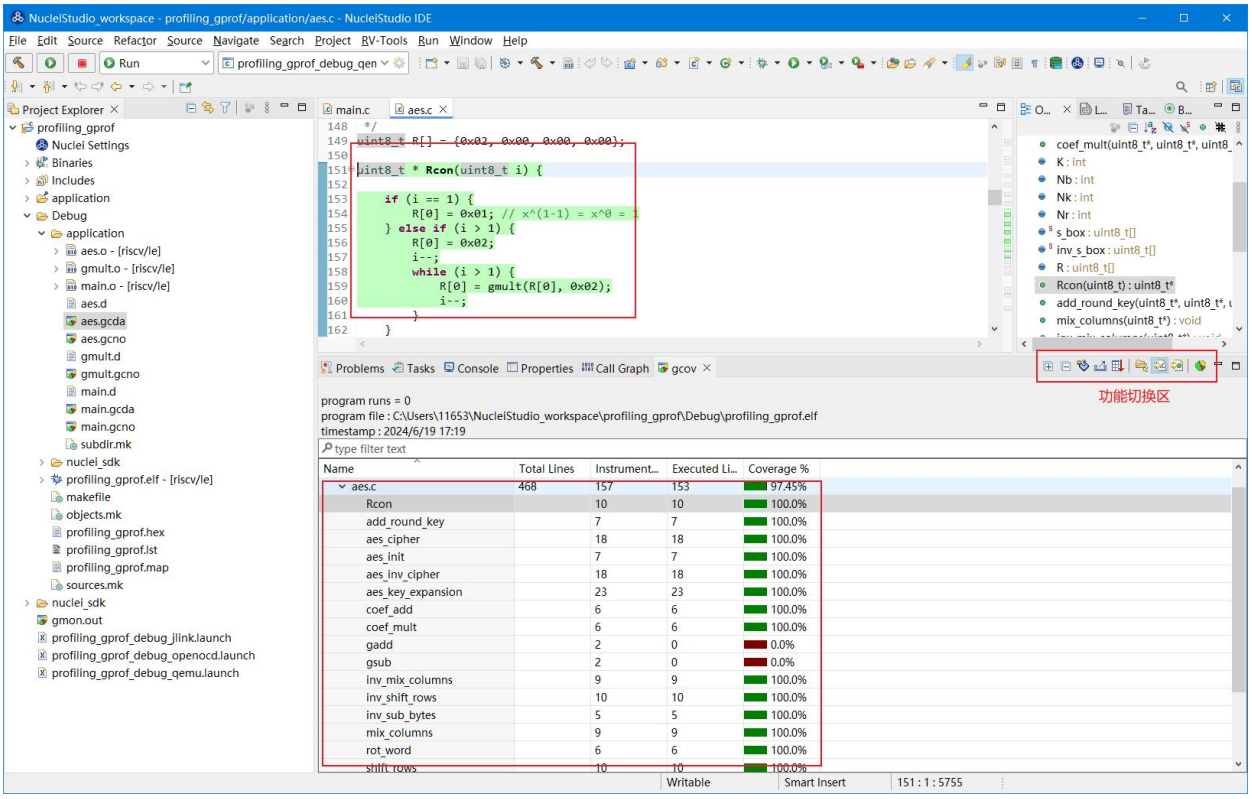


图 10- 13 打开 gcov 工具

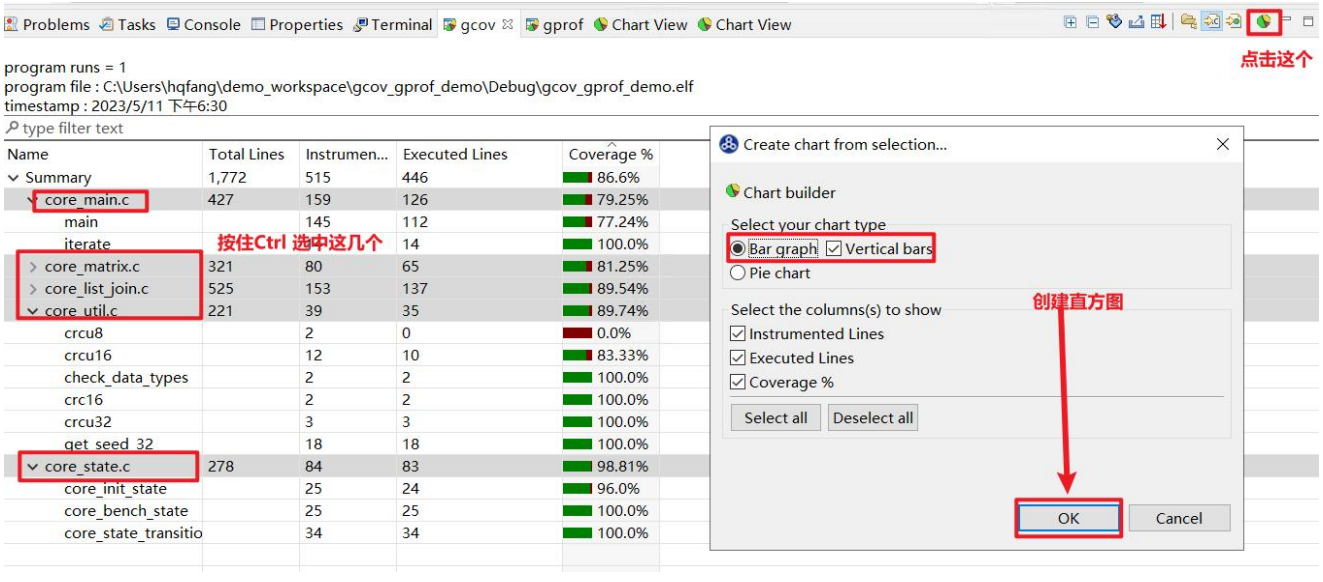


图 10- 14 在 Nuclei Studio 中查看 code coverage

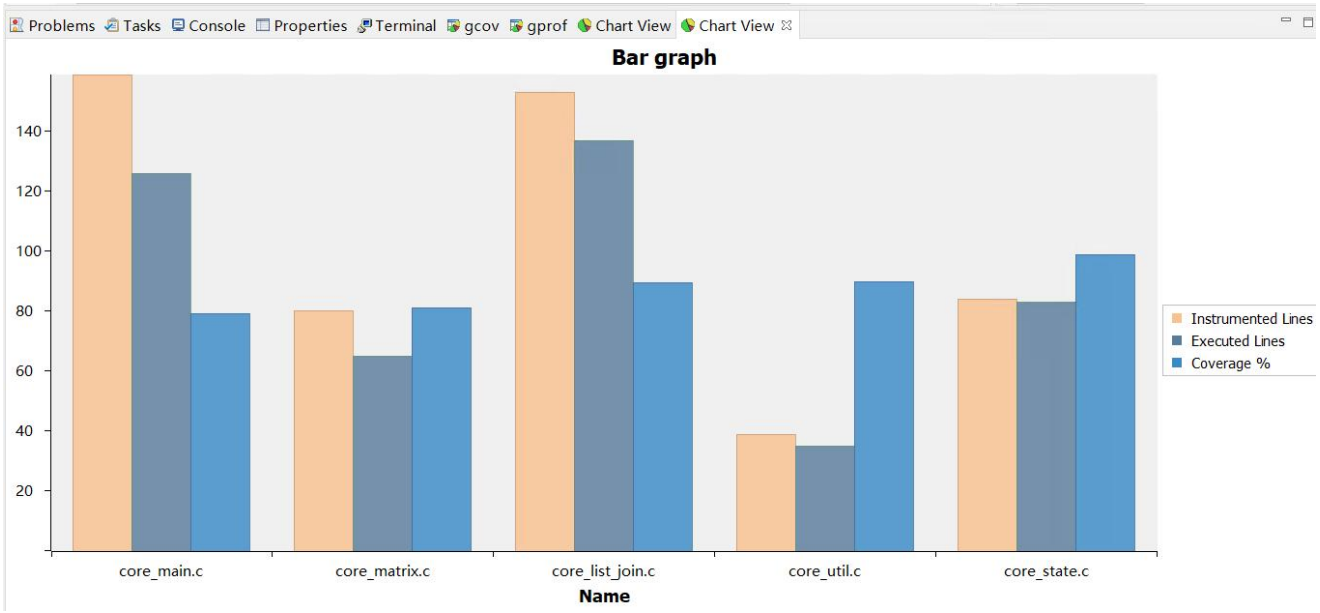


图 10- 15 在 Nuclei Studio 中查看 code coverage 直方图

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。

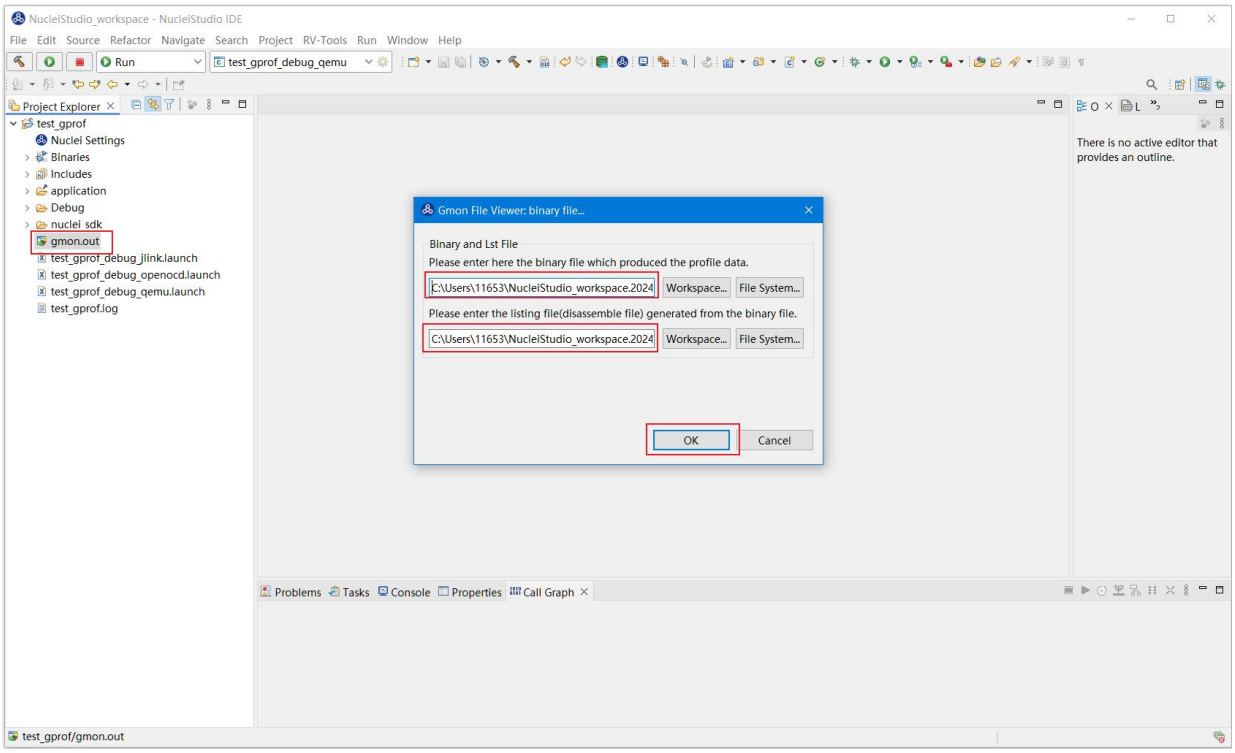


图 10- 16 在 Nuclei Studio 中查看 profiling 信息

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。

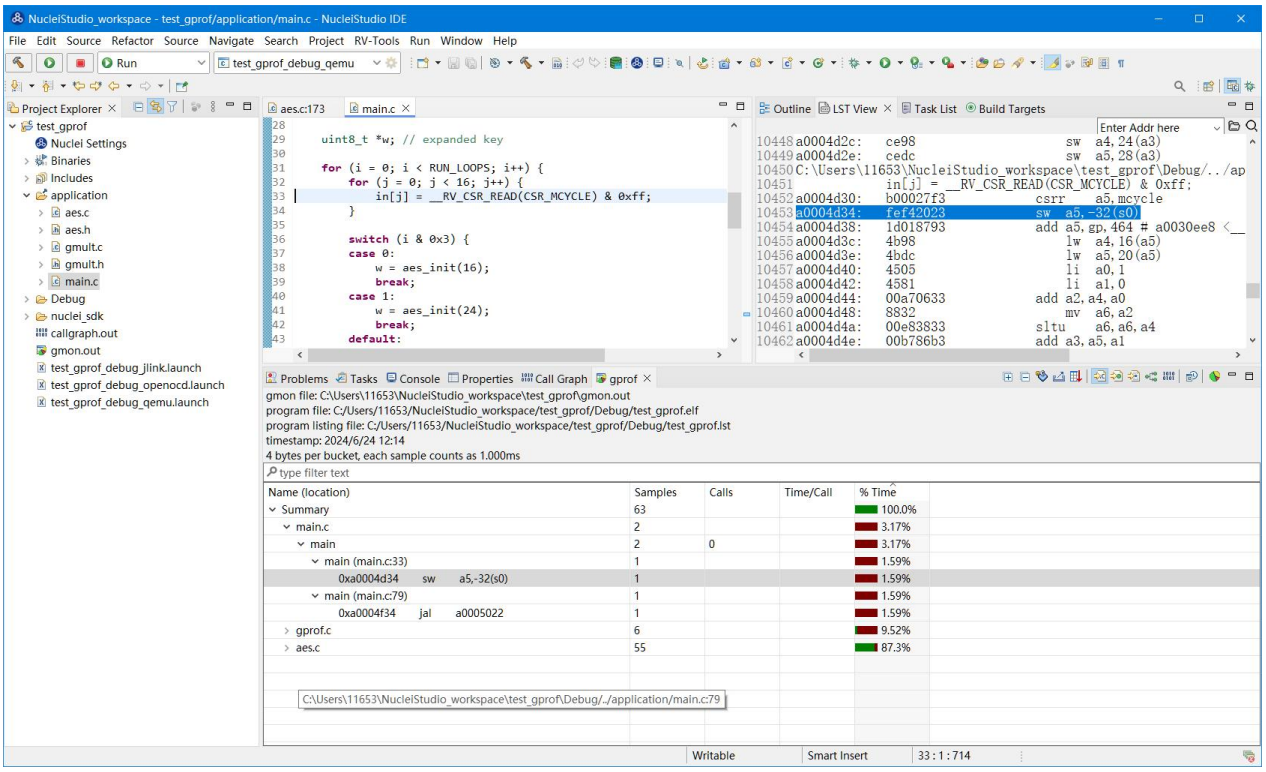


图 10- 17 在 Nuclei Studio 中查看 profiling 信息

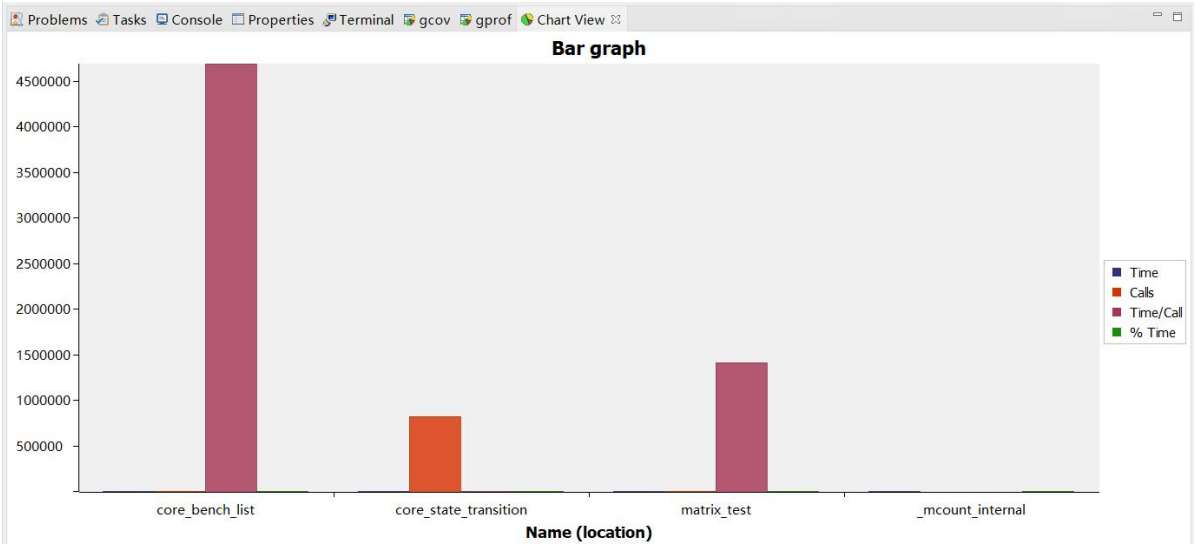


图 10- 18 在 Nuclei Studio 中查看 profiling 信息直方图

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 callgrph.out 文件。双击 callgrph.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 10.3 章节内容。

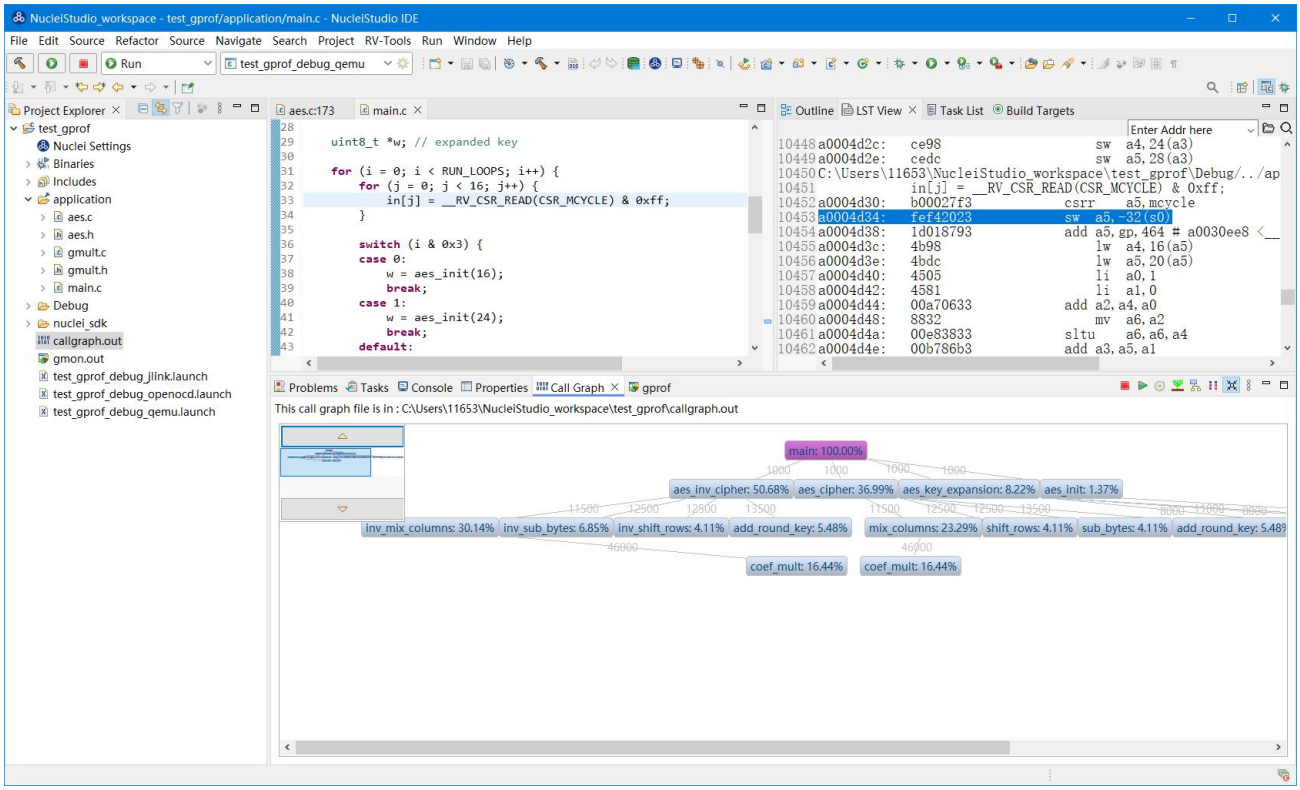


图 10- 19 在 Nuclei Studio 中查看 Call Graph

10.4.2.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。

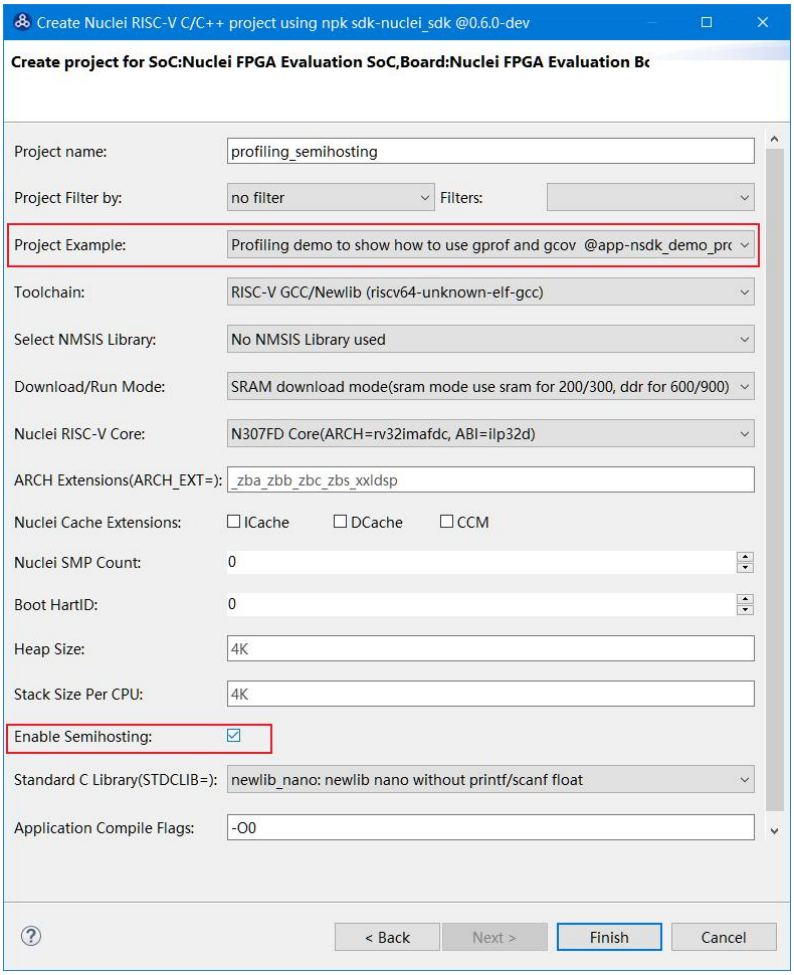


图 10- 20 创建一个启用 Semihosting 的工程

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。

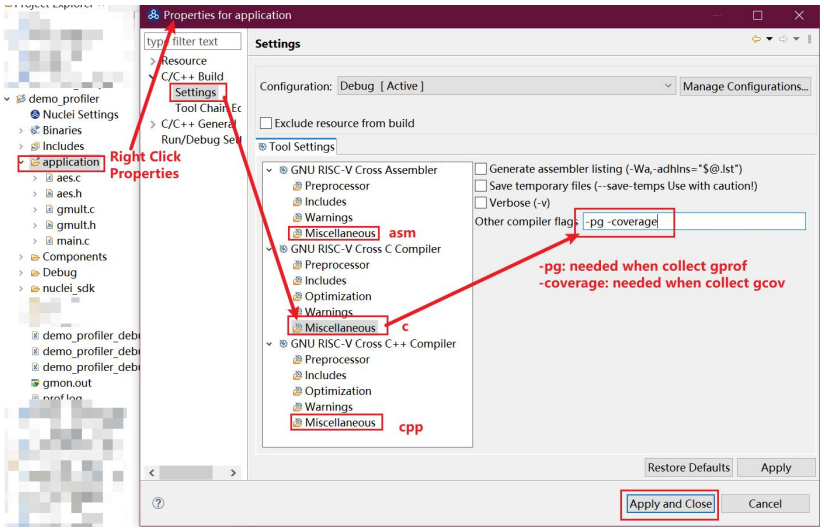


图 10- 21 设置-pg --coverage 属性

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。

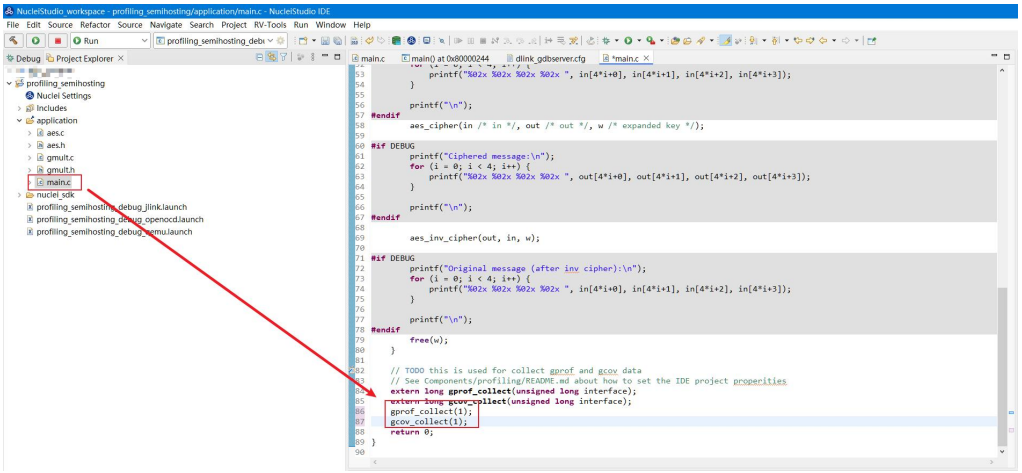


图 9- 5 修改参数以通过 Semihost 写入文件

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。

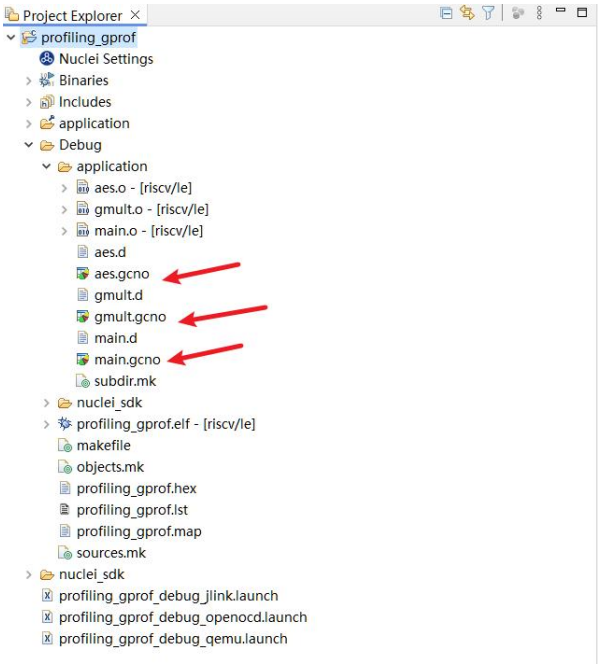


图 10- 23 编译测试工程

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。

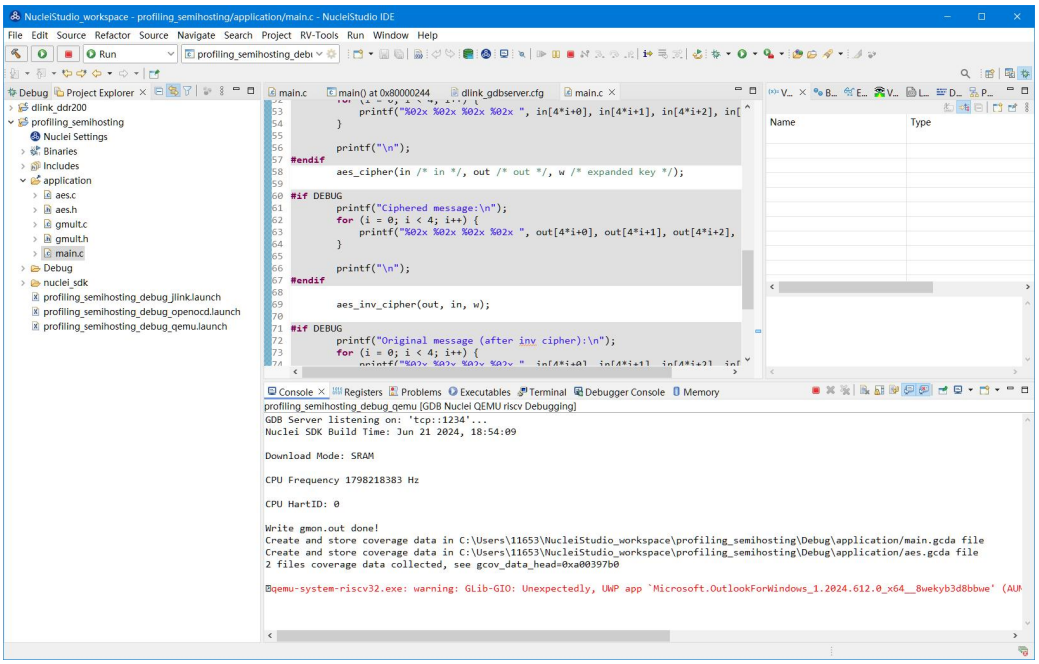


图 10- 24 运行工程

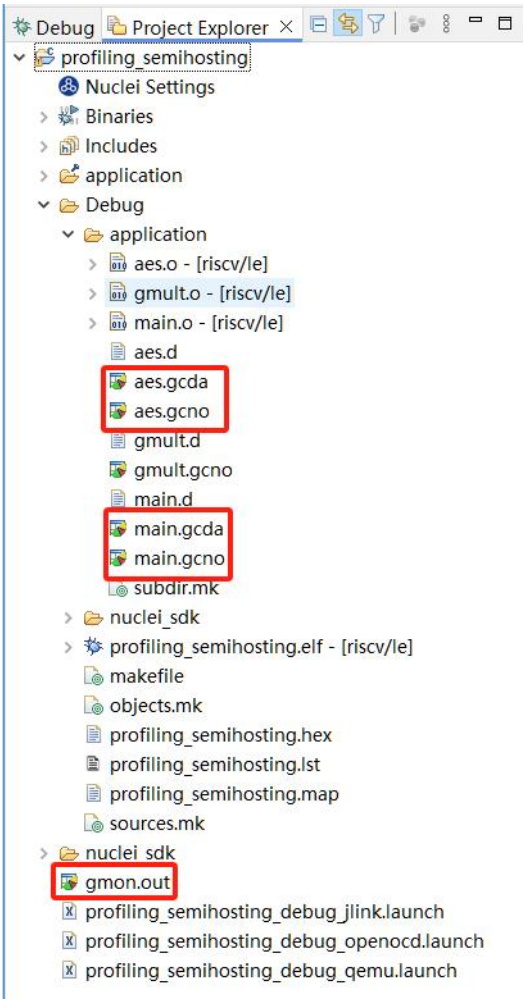


图 10- 25 查看生成的文件

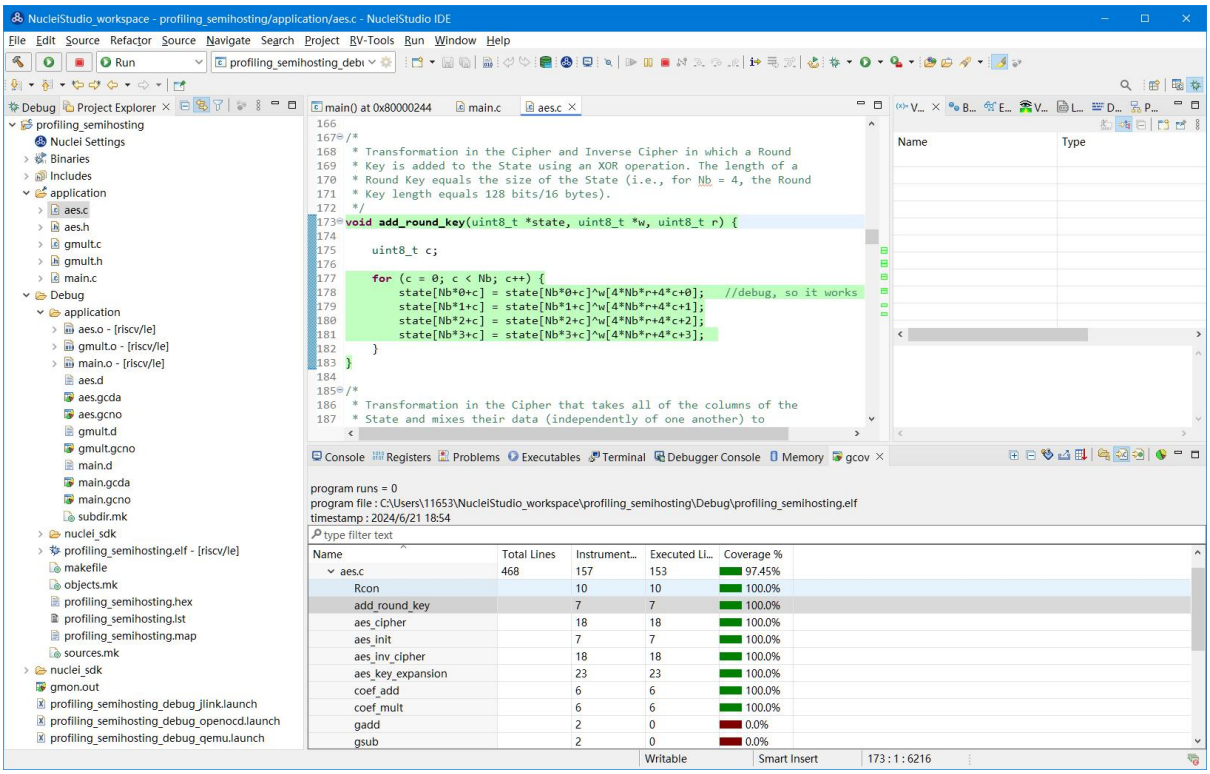


图 10- 26 在 Nuclei Studio 中查看 gcov

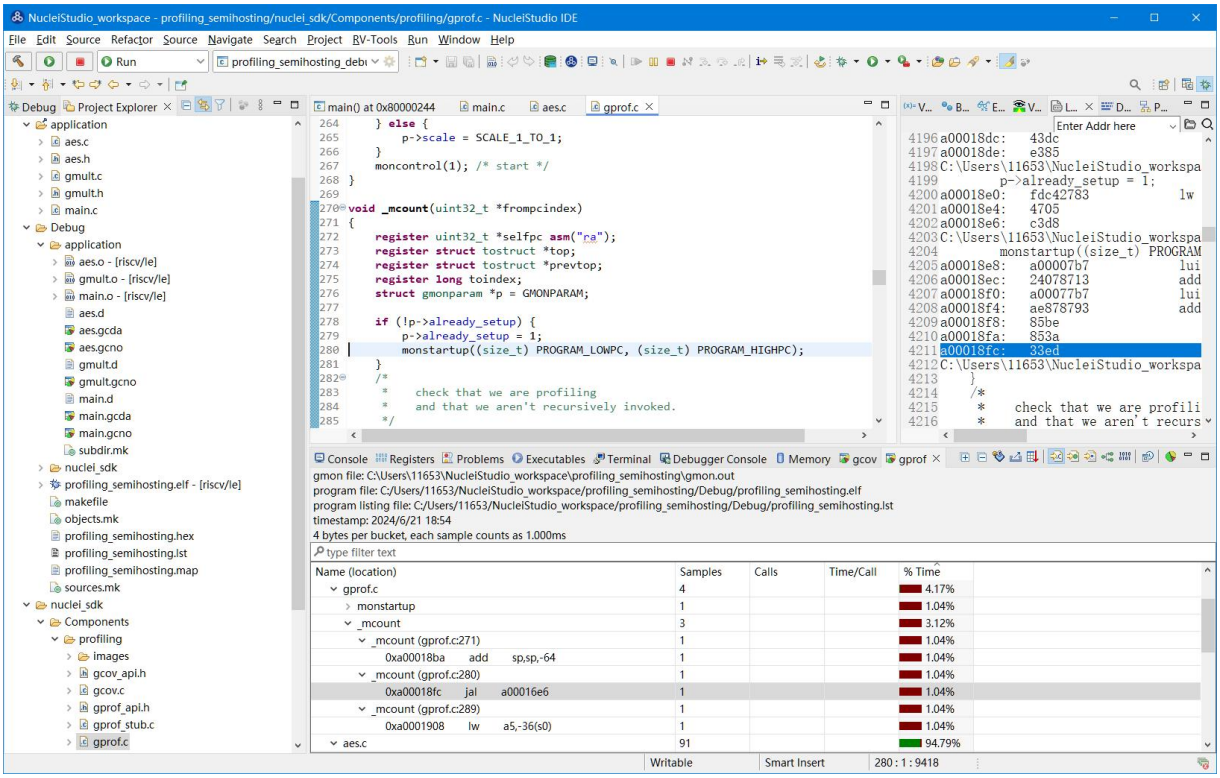


图 10- 27 在 Nuclei Studio 中查看 gprof

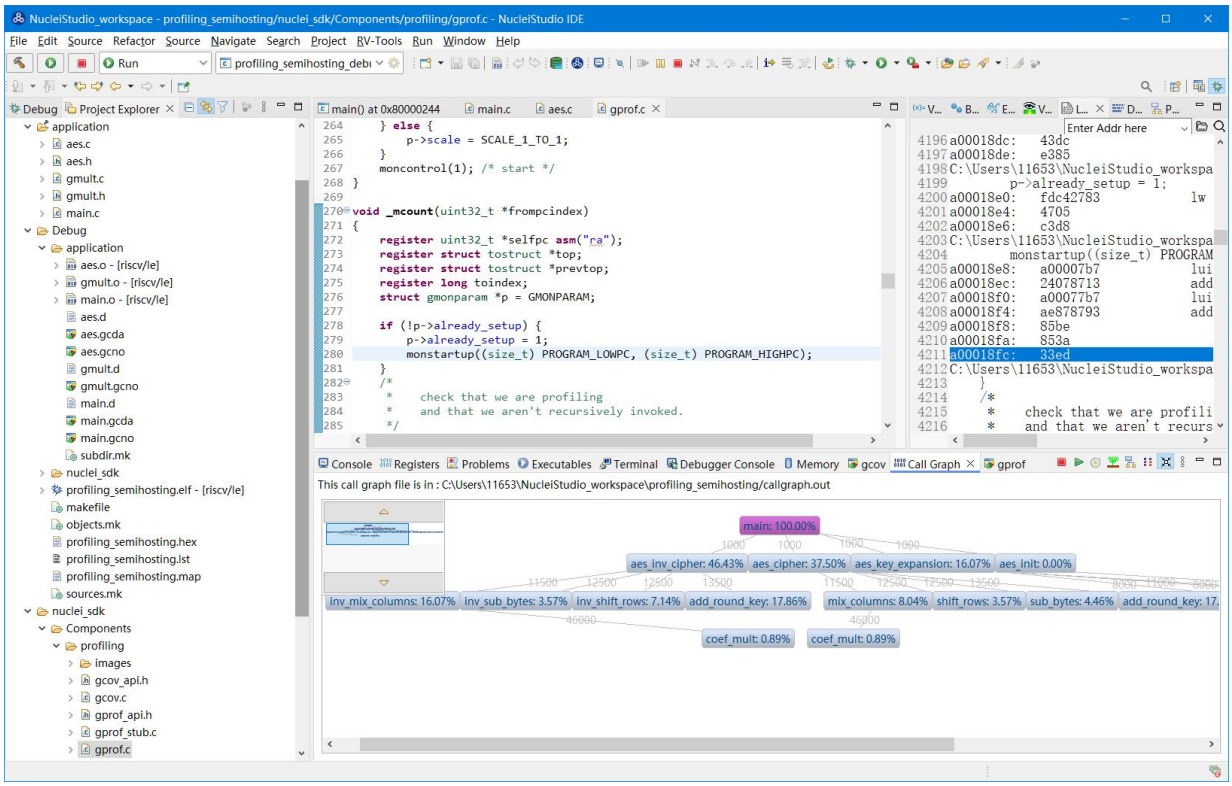


图 10- 28 在 Nuclei Studio 中查看 Call Graph

11.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，结合相对应的硬件和 Nuclei OpenOCD，用户在对工程进行 Debug 时，也可查看到 Trace 日志，并结合源码时行问题排查。

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

11.1.Trace 界面介绍

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，打击打开 Trace 菜单。

图 10- 1 在 Nuclei Studio 中 Trace 的视图

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 Module 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。

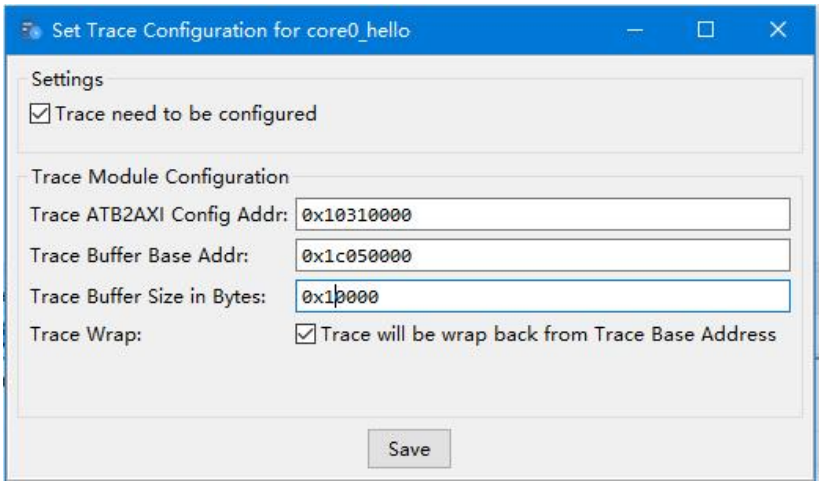


图 10- 2 Trace setting 的视图

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。

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

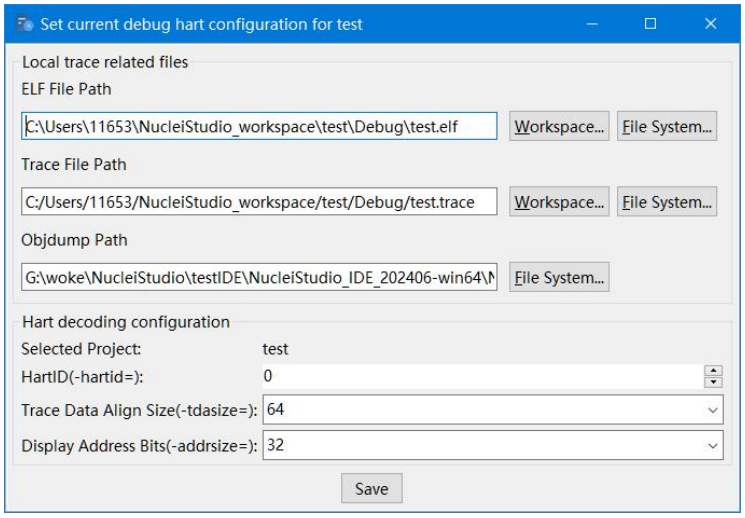


图 10- 2 Set Current Debug hart Configuration

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。

11.2.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。

11.2.1.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。

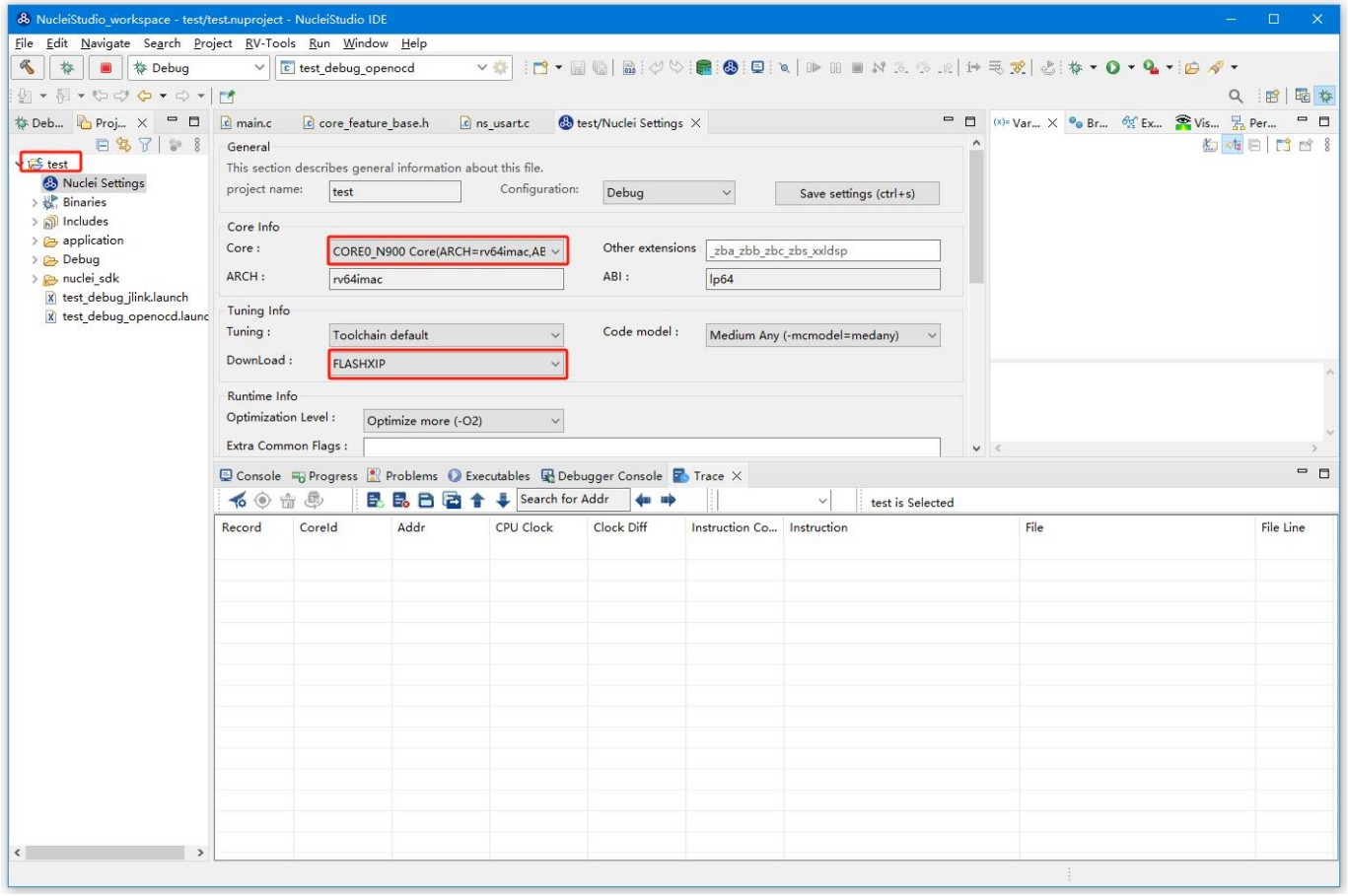


图 10- 3 N900 的单核应用 helloworld

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。

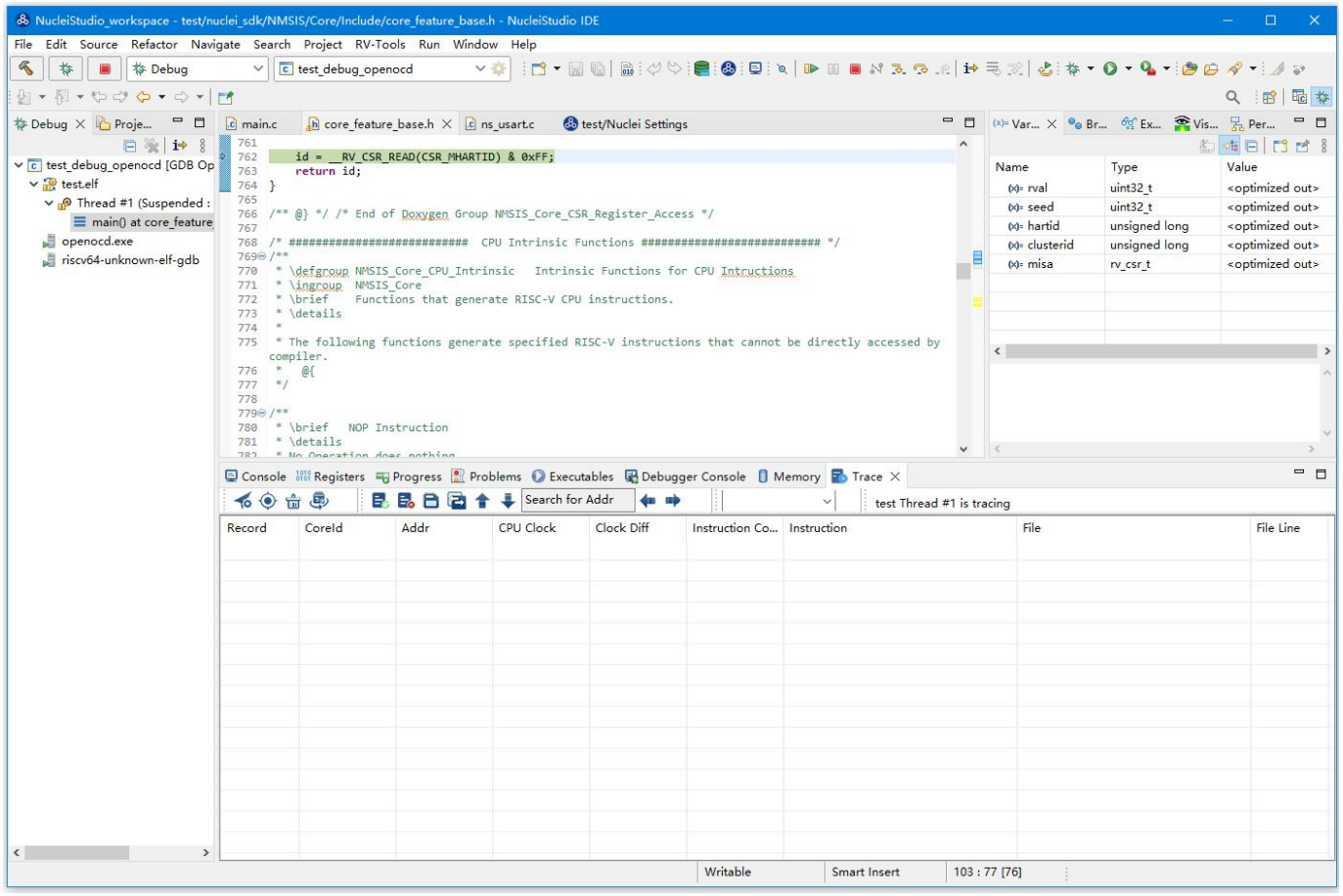


图 10- 4 helloworld 工程进入 Debug 模式

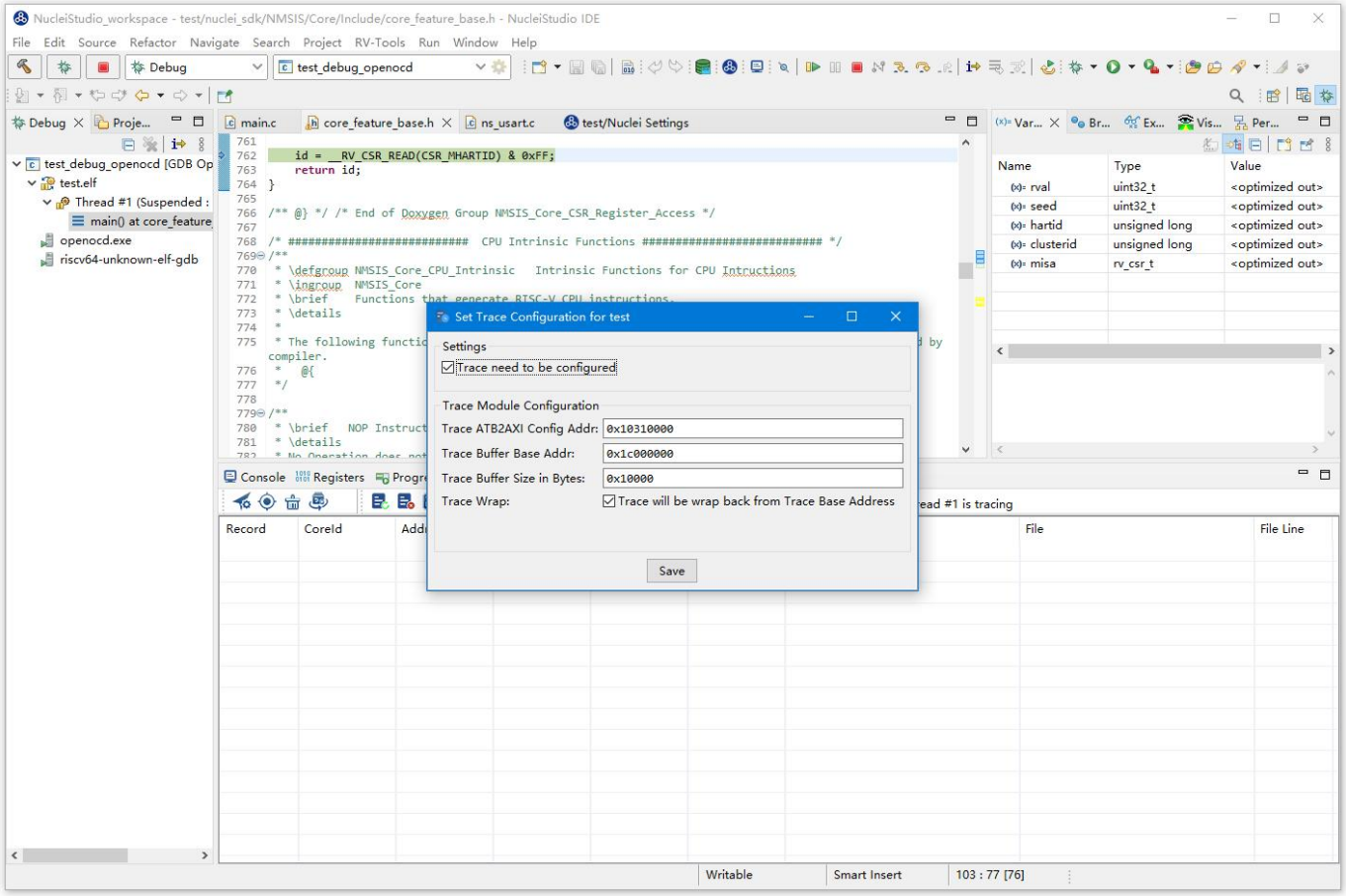


图 10- 5 设置 Trace Configuration

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。

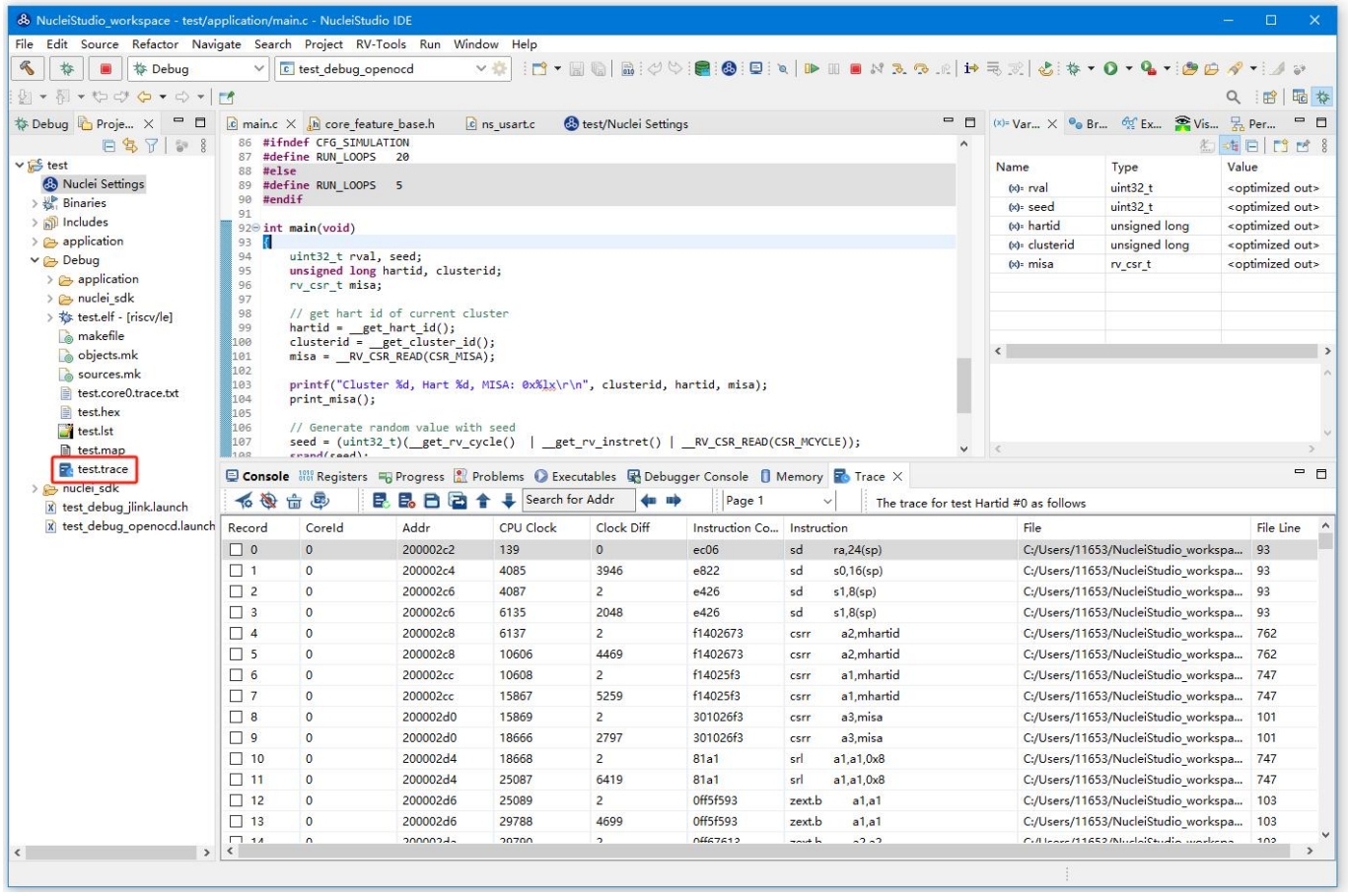


图 10- 6 dump 到的 trace 文件

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

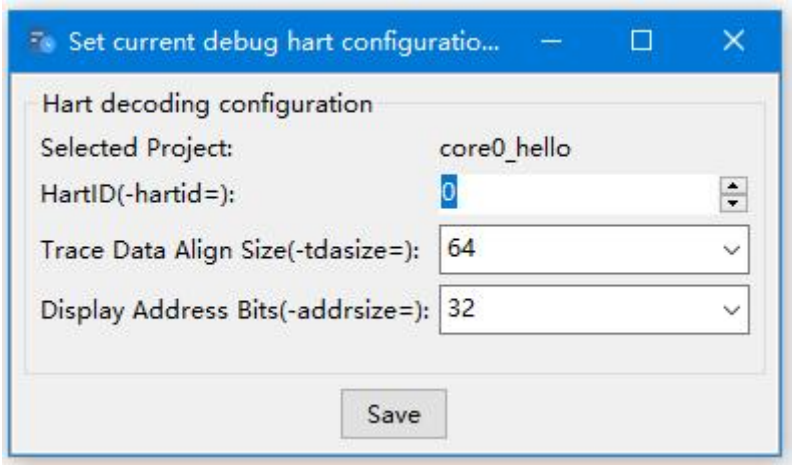


图 10- 7 Set current debug hart configuration

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）。

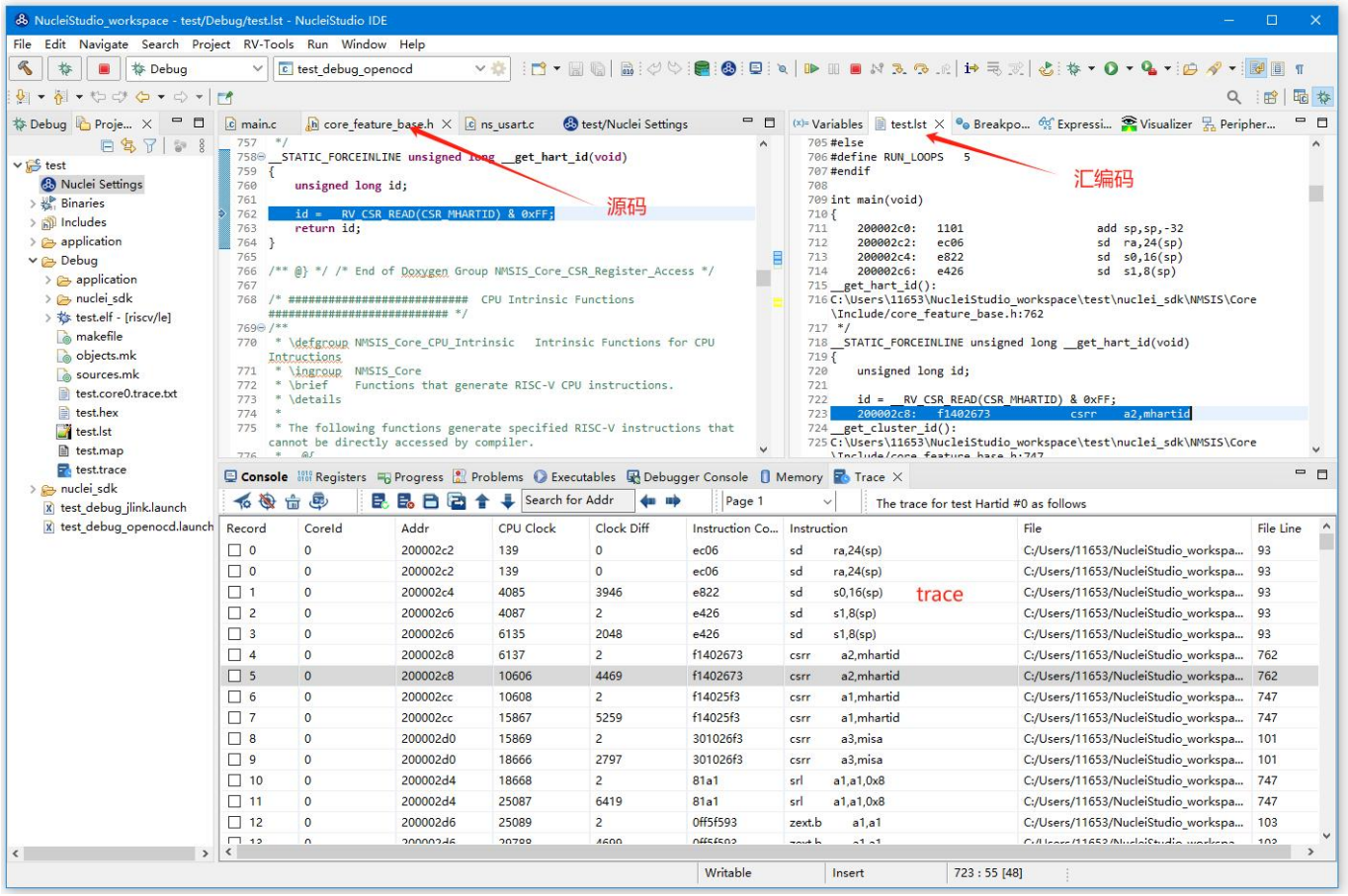


图 10- 8 trace 与源码、反汇编码联动

也可以双击“工程名.trace”文件，以文本的方式查看 trace 文件。

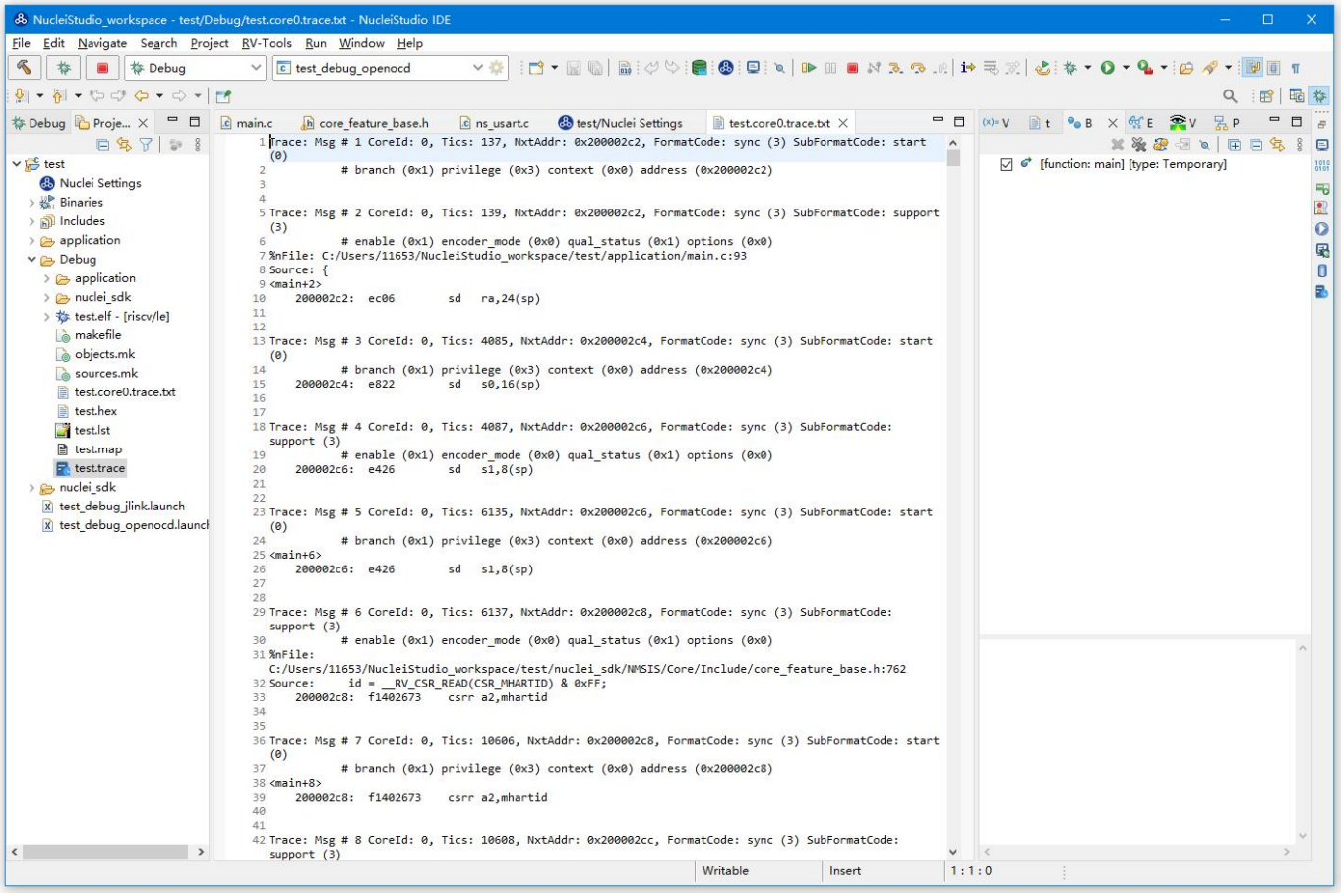


图 10- 9 以文本方式查看 trace

11.2.2.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。

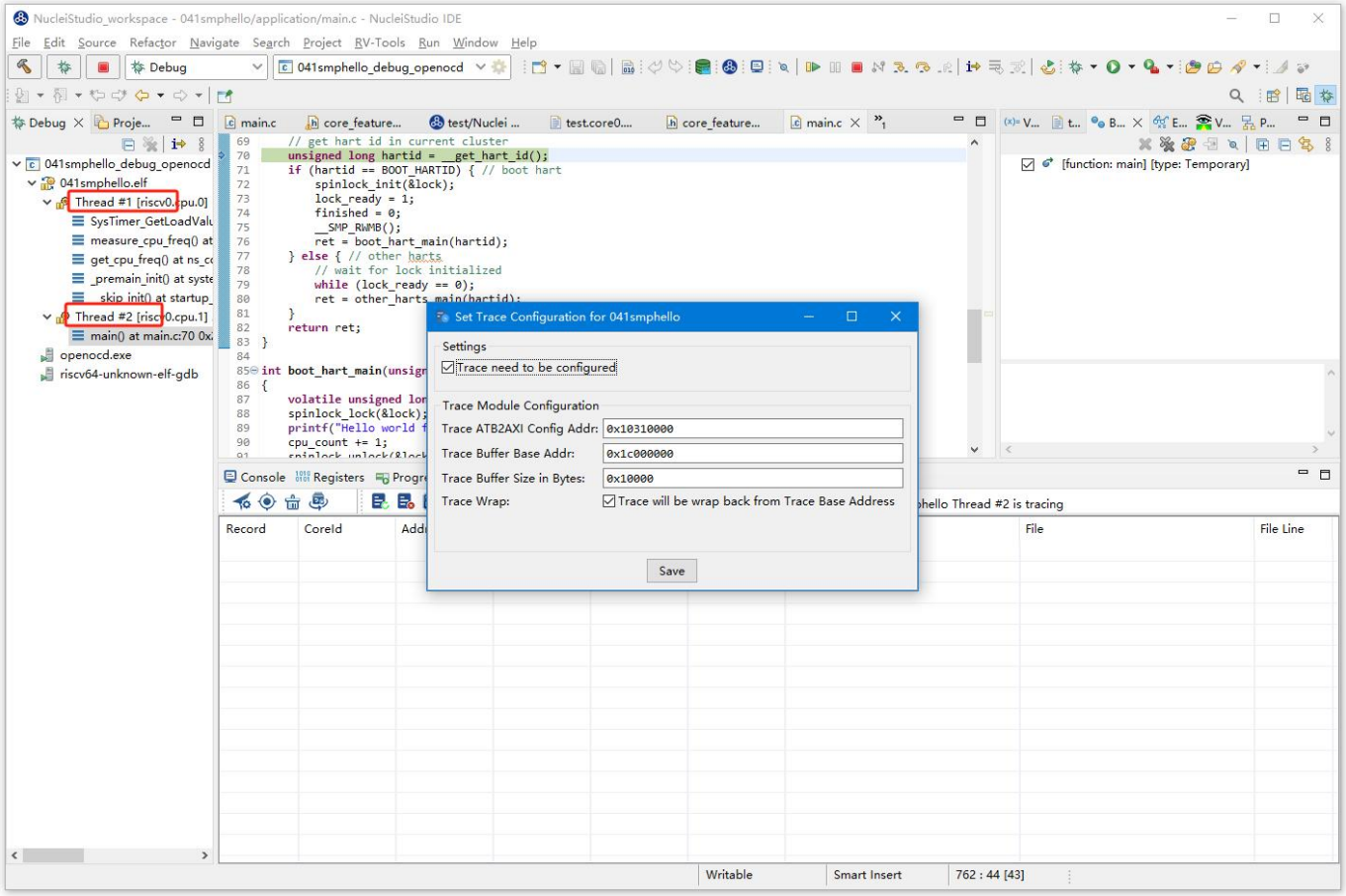


图 10- 10 SMP 多核应用中设置 Trace Configuration

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。

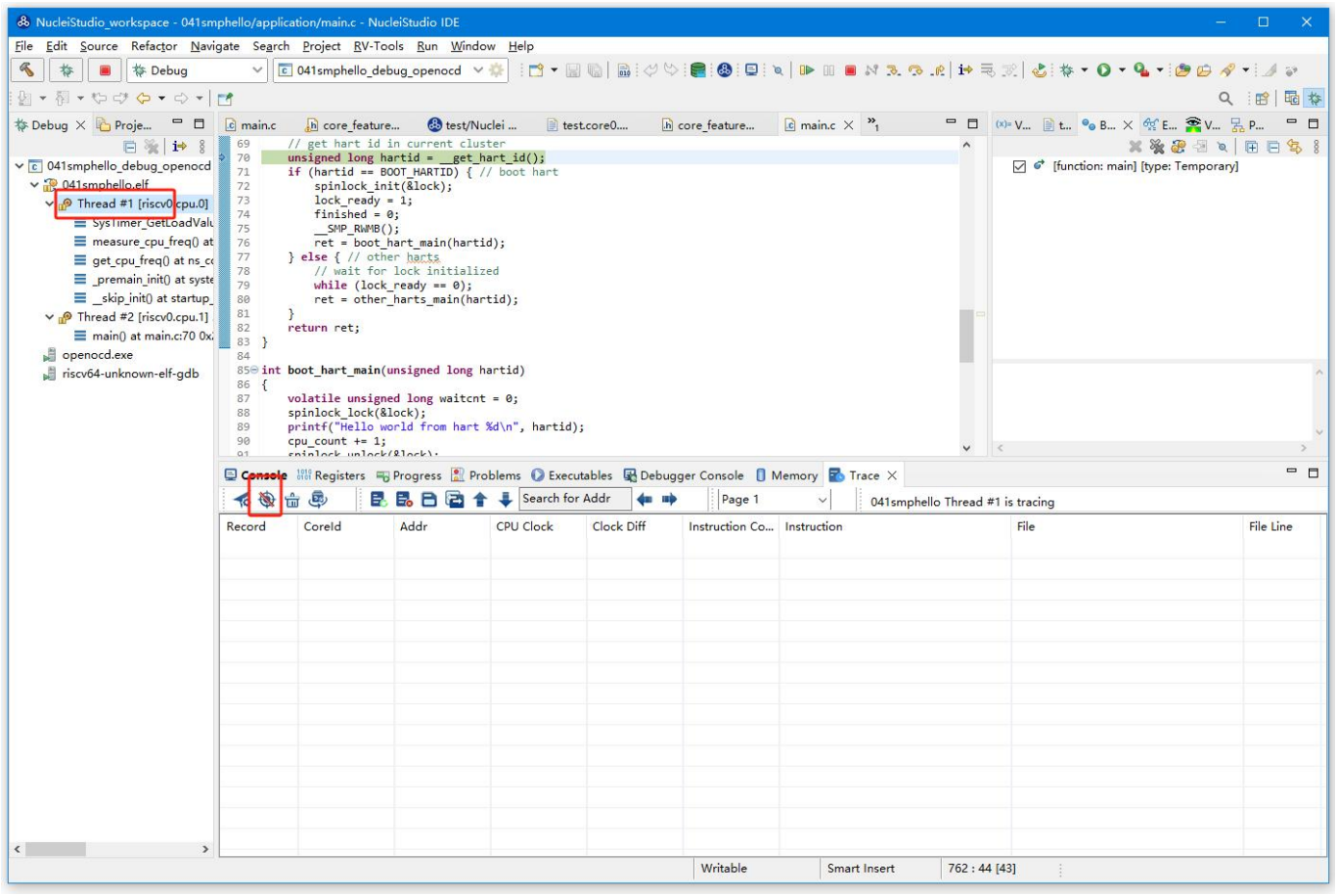


图 10- 11 SMP 多核应用中对 Thread #1 start trace

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。

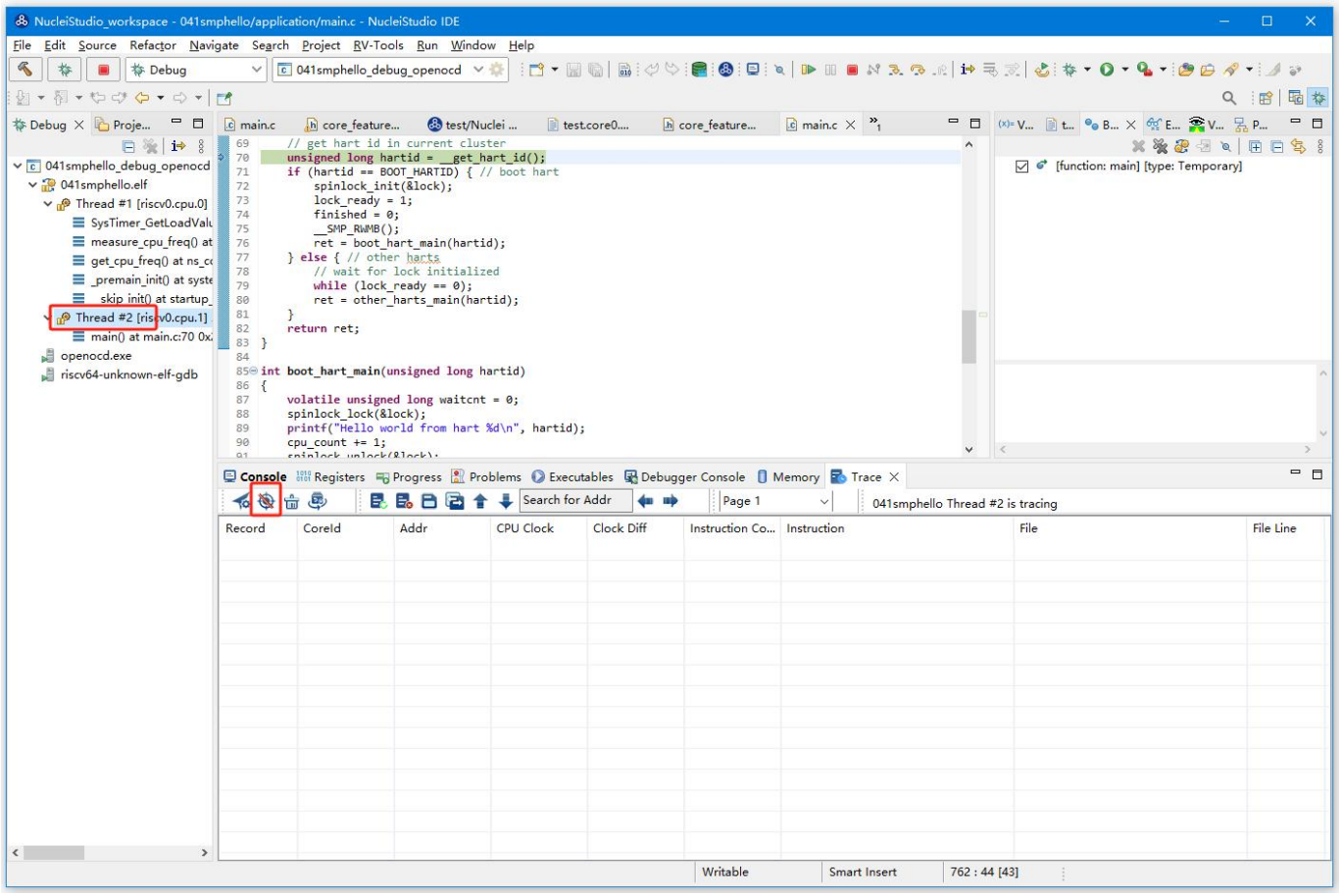


图 10- 12 SMP 多核应用中对 Thread #2 start trace

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 decoder 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。

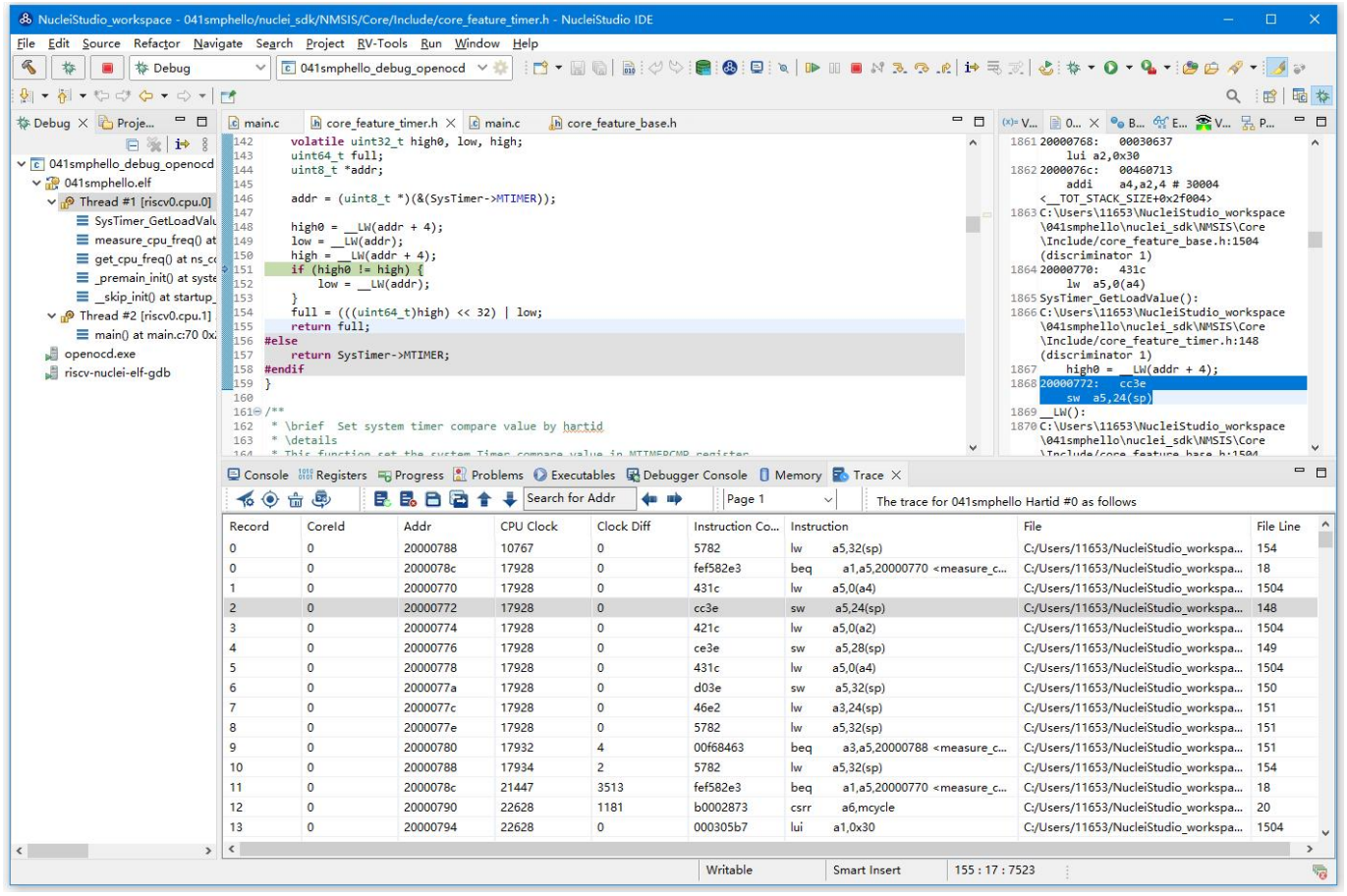


图 10- 13 SMP 多核应用中查看 HardID=0 的 trace

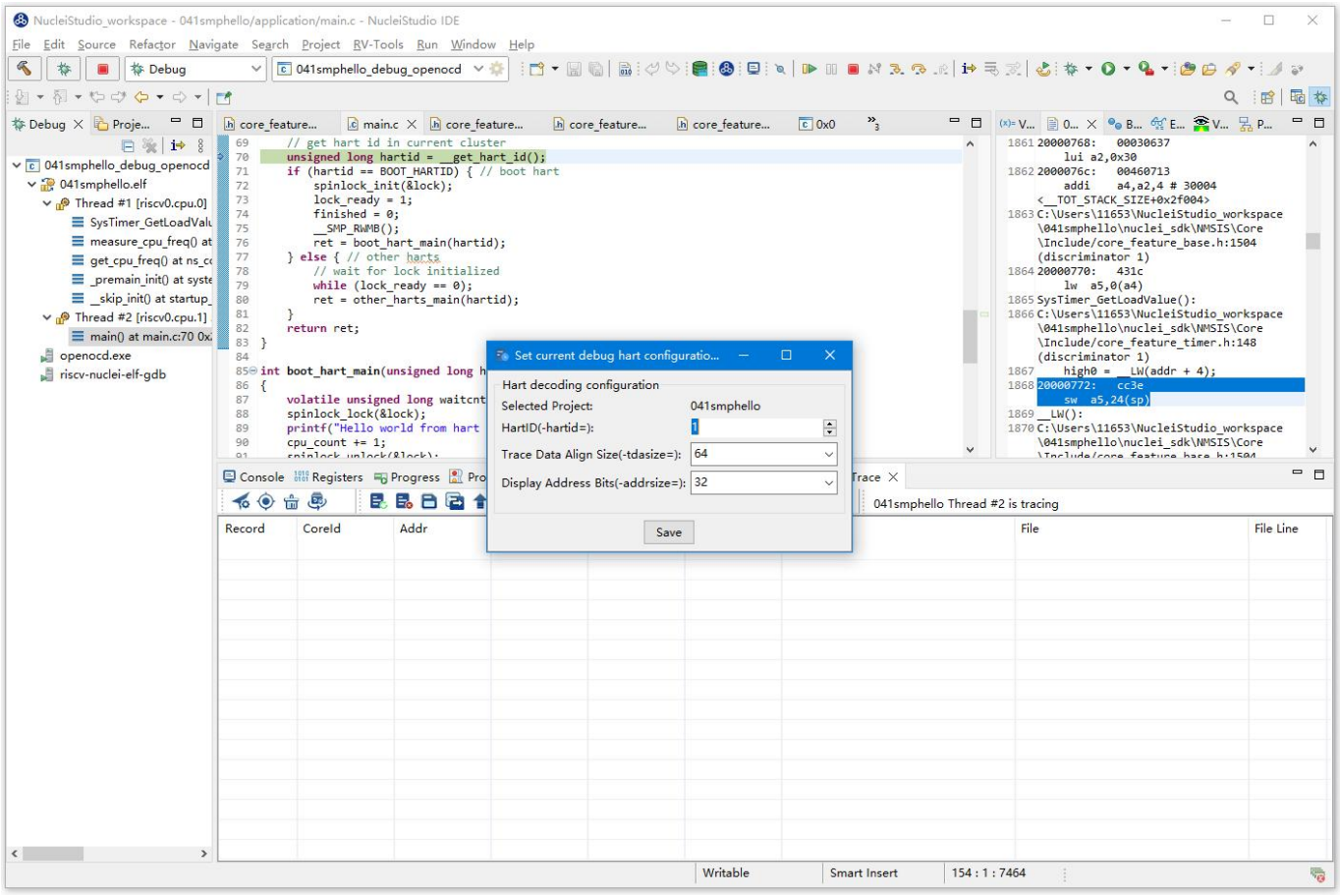


图 10- 14 多核应用中查看 HardID=1 的 trace

11.2.3.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。

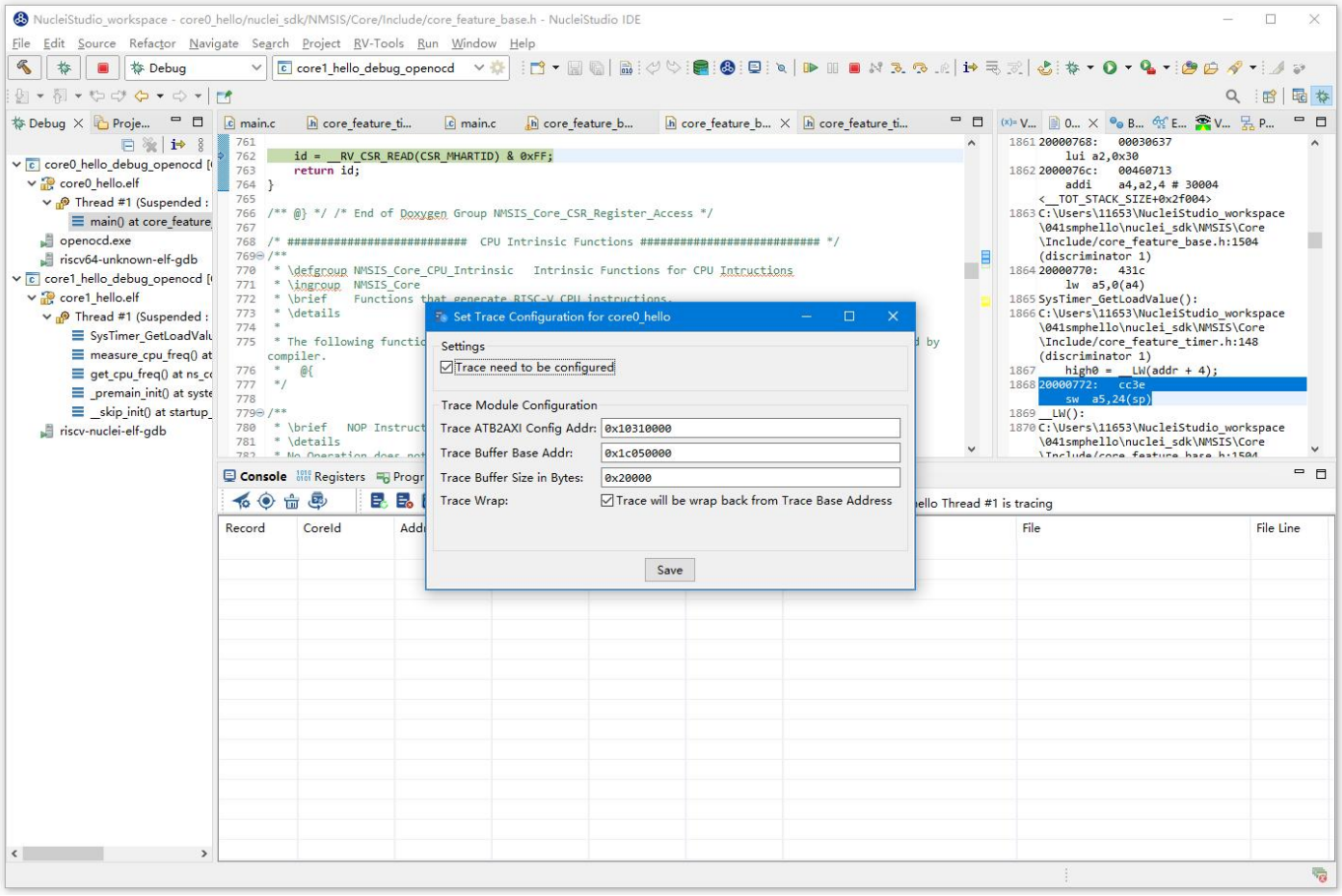


图 10- 15 AMP 多核应用中 Core 0 设置 trace 配置

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。

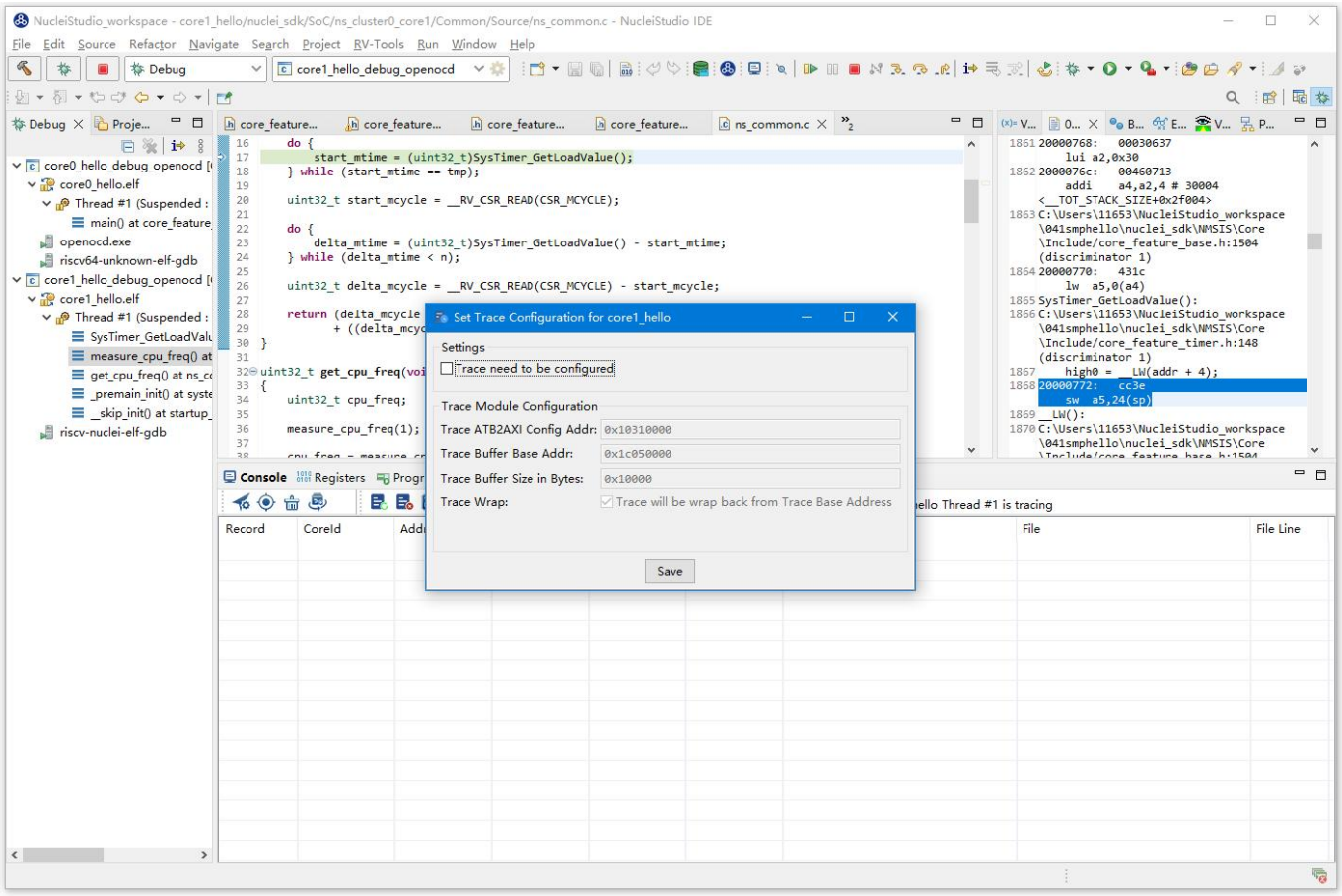


图 10- 16 AMP 多核应用中 Core 1 设置 trace 配置

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。

11.2.4.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。

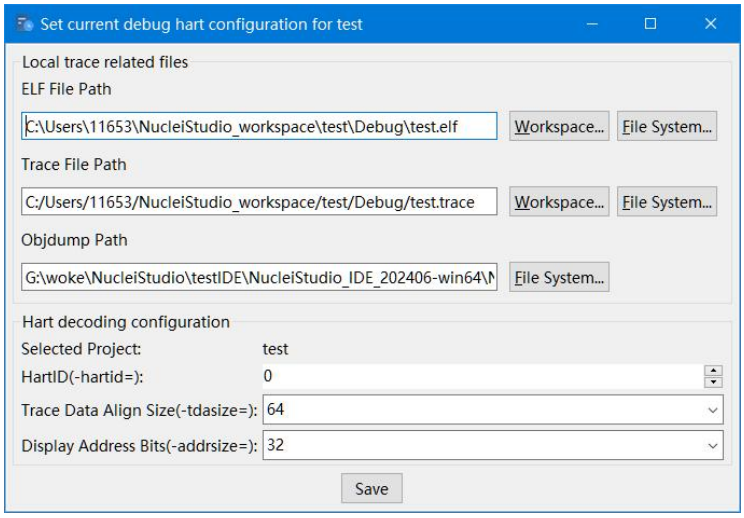


图 10- 2 Set Current Debug hart Configuration

12.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.trace 文件，然后 RVProf 工具，将.trace 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。

12.1.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。

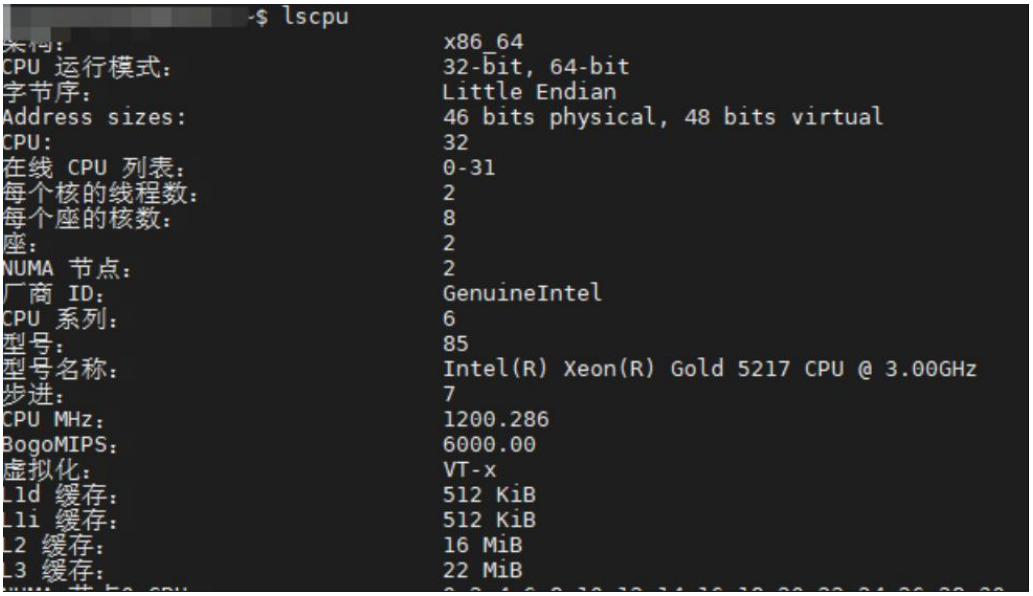


图 11- 1 rvprof 测试环境

12.1.1.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

12.1.2.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，因为测式 demo 是依赖于 nuclei_sdk，所以也要先安装 sdk-nuclei_sdk，具体如下：

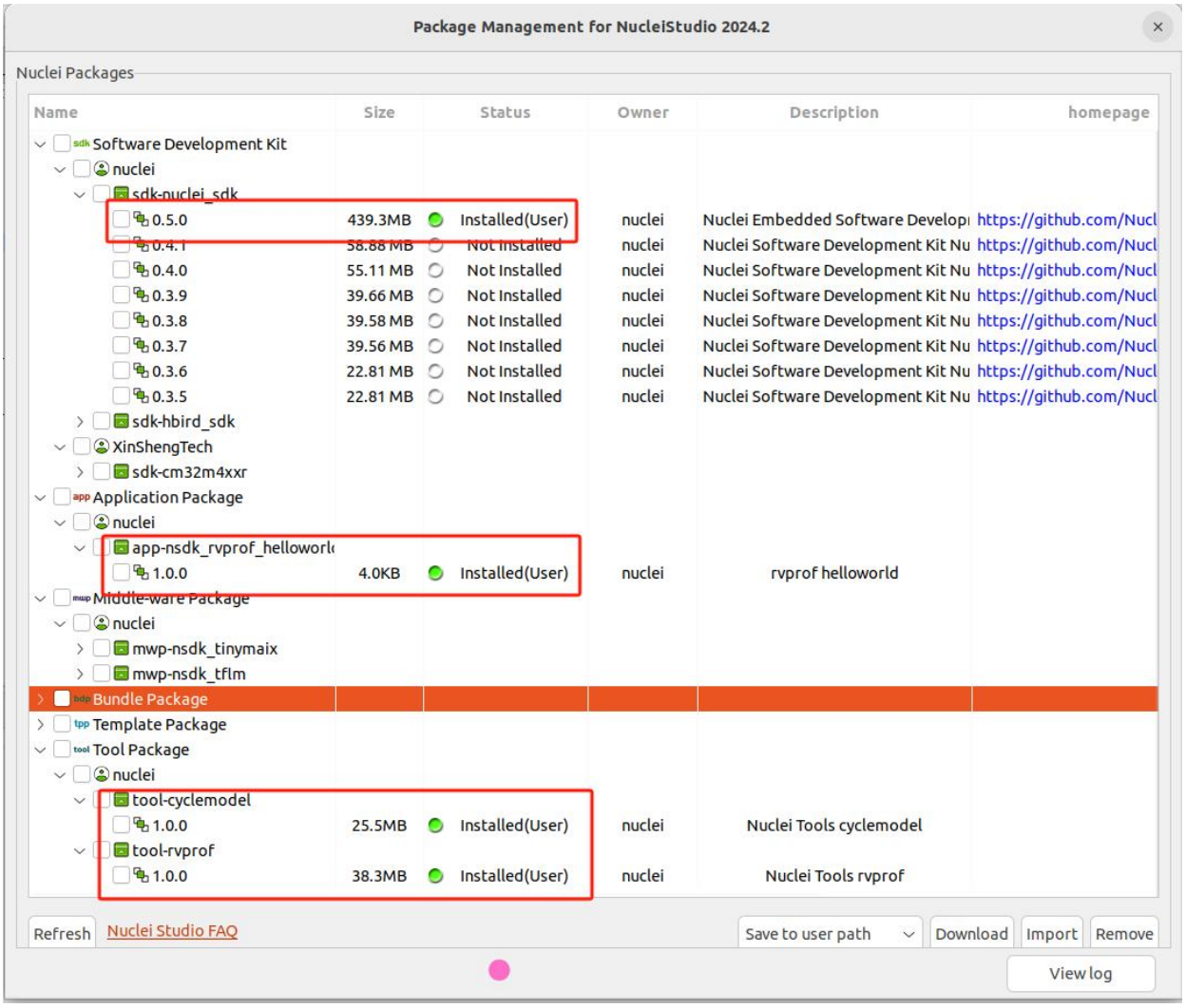


图 11- 2 安装 RVProf 测试包及依赖包

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）。

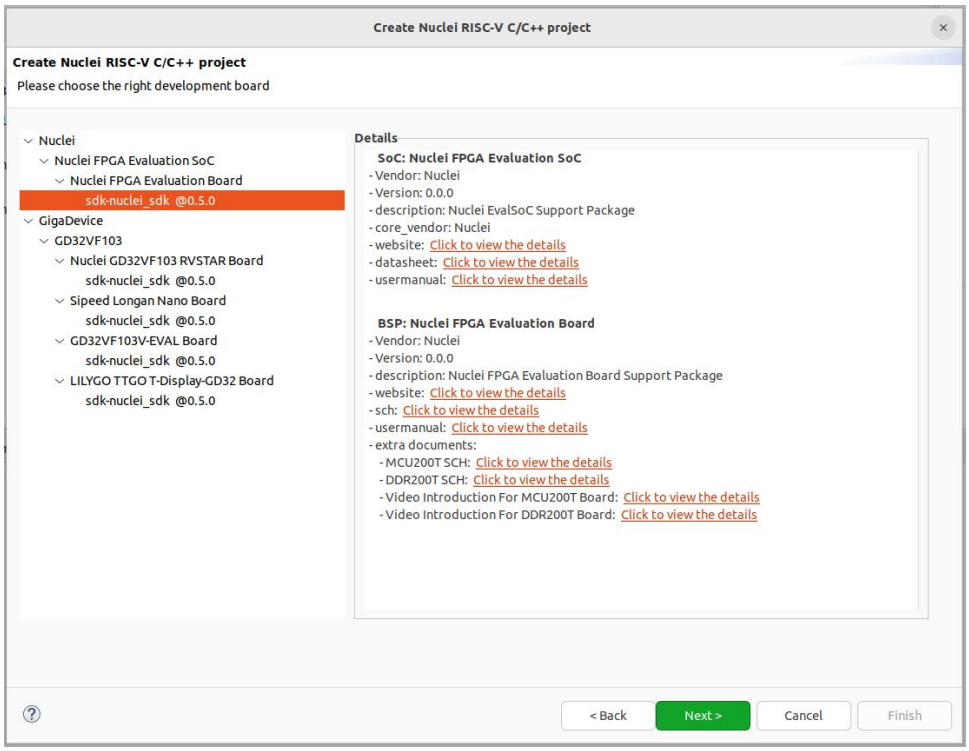


图 11- 3 选择最新的 nuclei_sdk

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，完工程的创建。

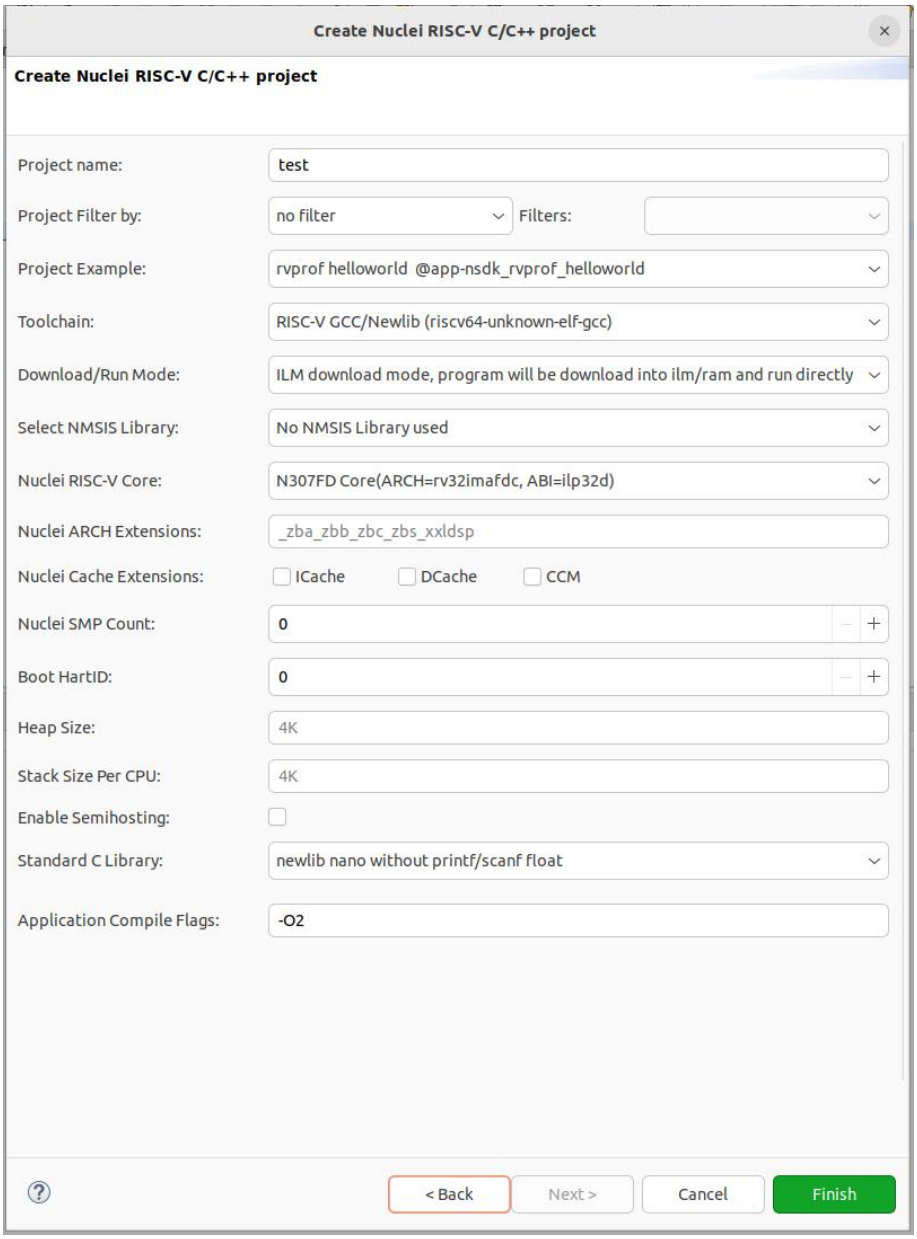


图 11- 4 创建 RVProf 测试工程

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时时间，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。

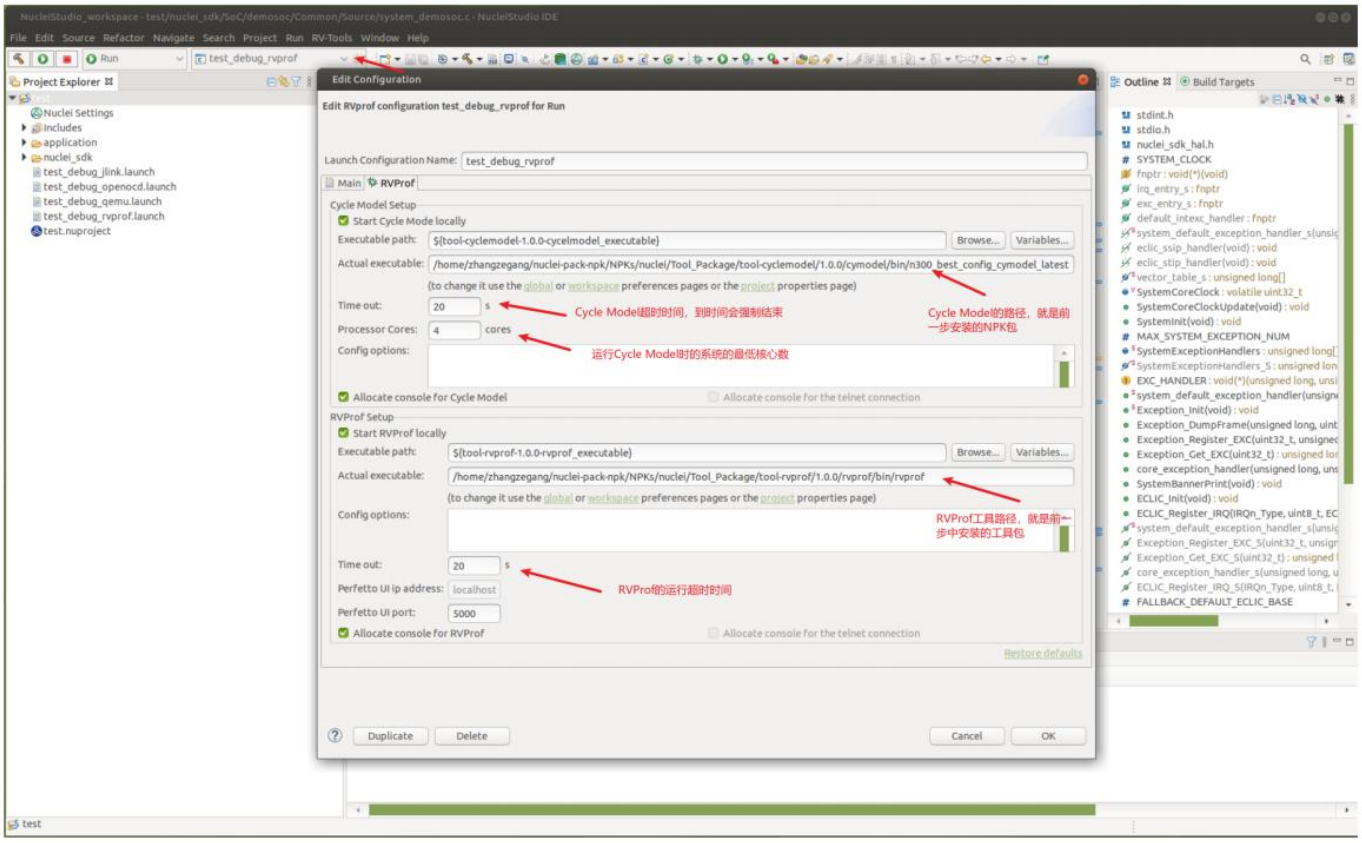


图 11- 5 RVProf 测试工程的 launch 文件

12.2.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

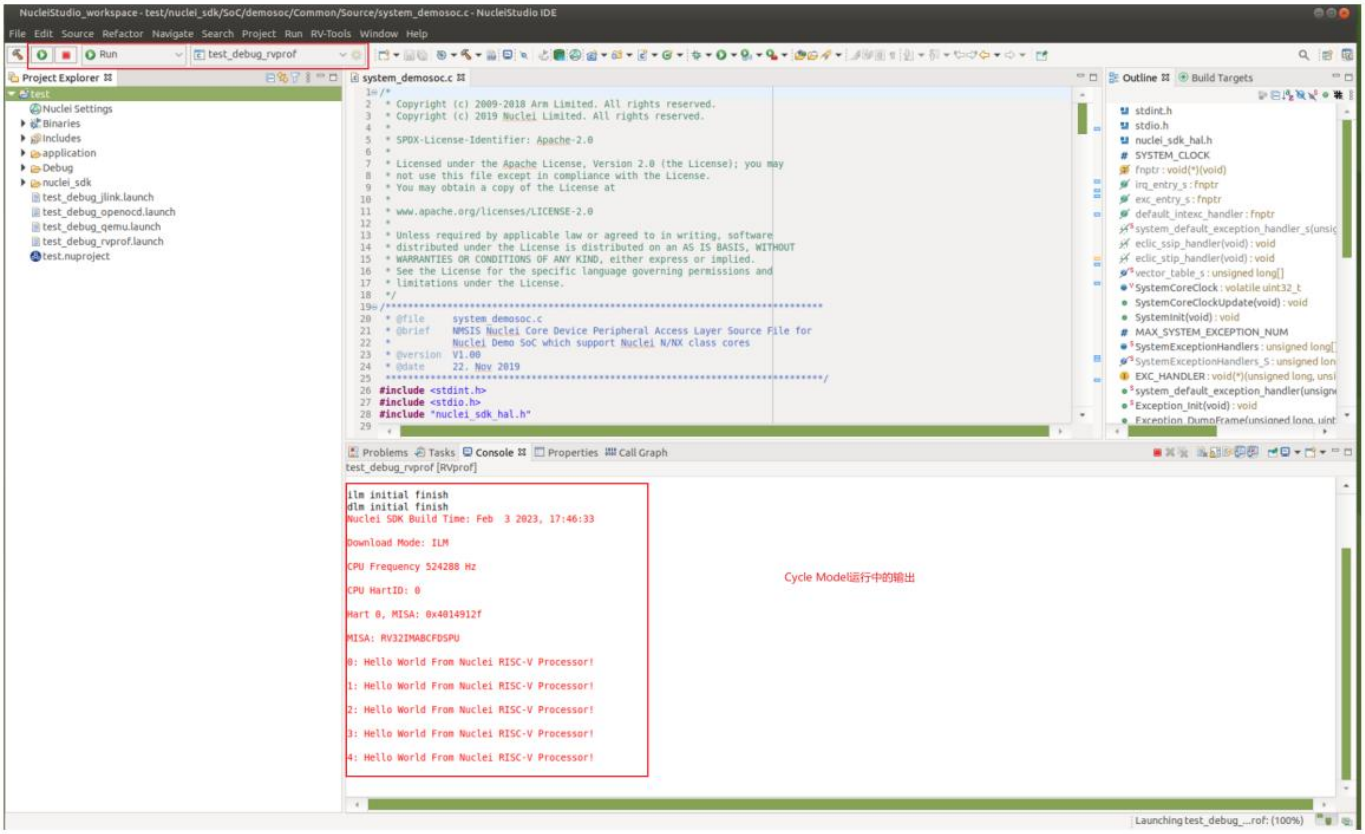


图 11- 6 Cycle Model 启动及 log 输出

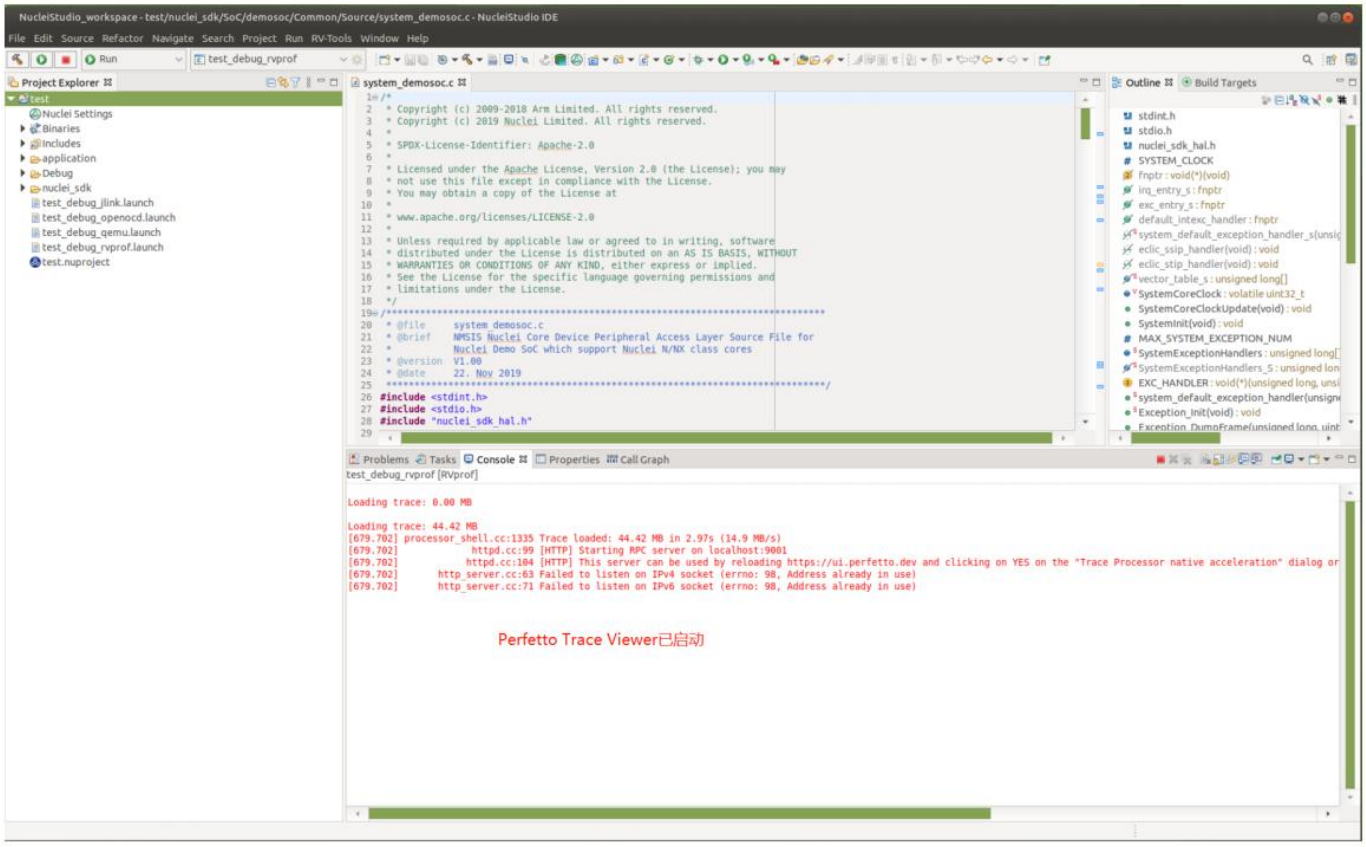


图 11- 7 perfetto 启动本地服务

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。

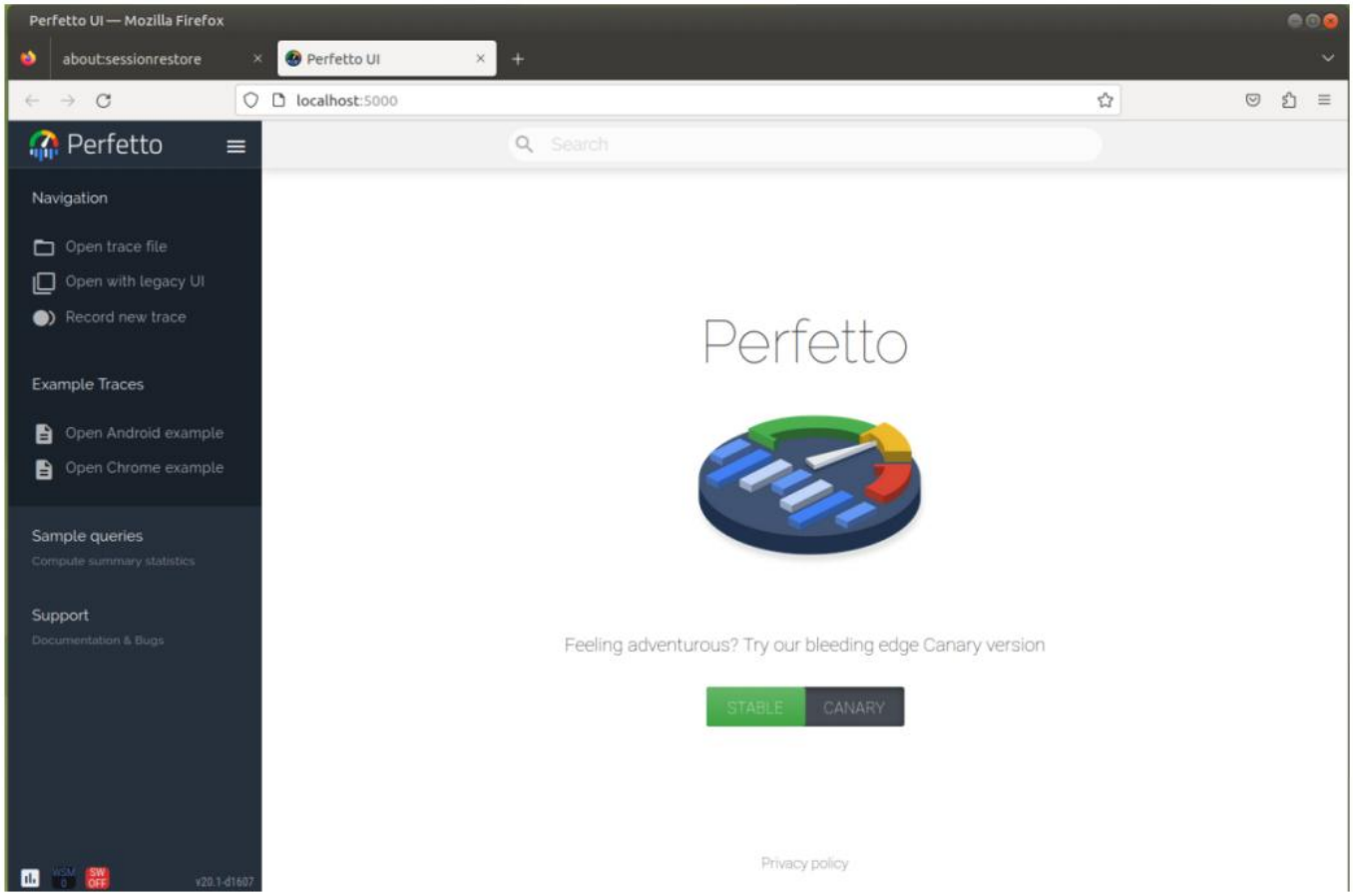


图 11- 8 打开 Perfetto Trace Viewer

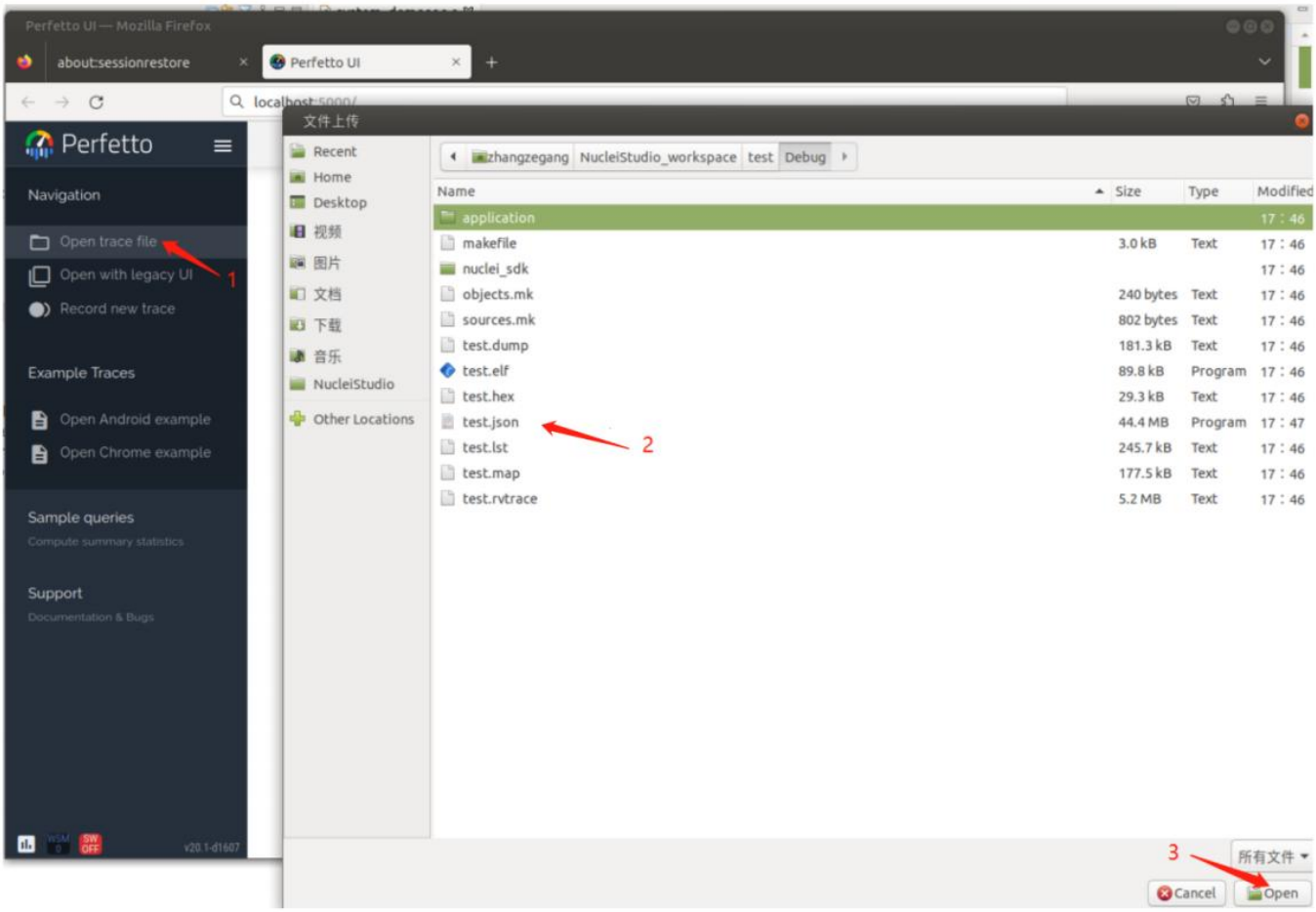


图 11- 9 手动载入 json 文件

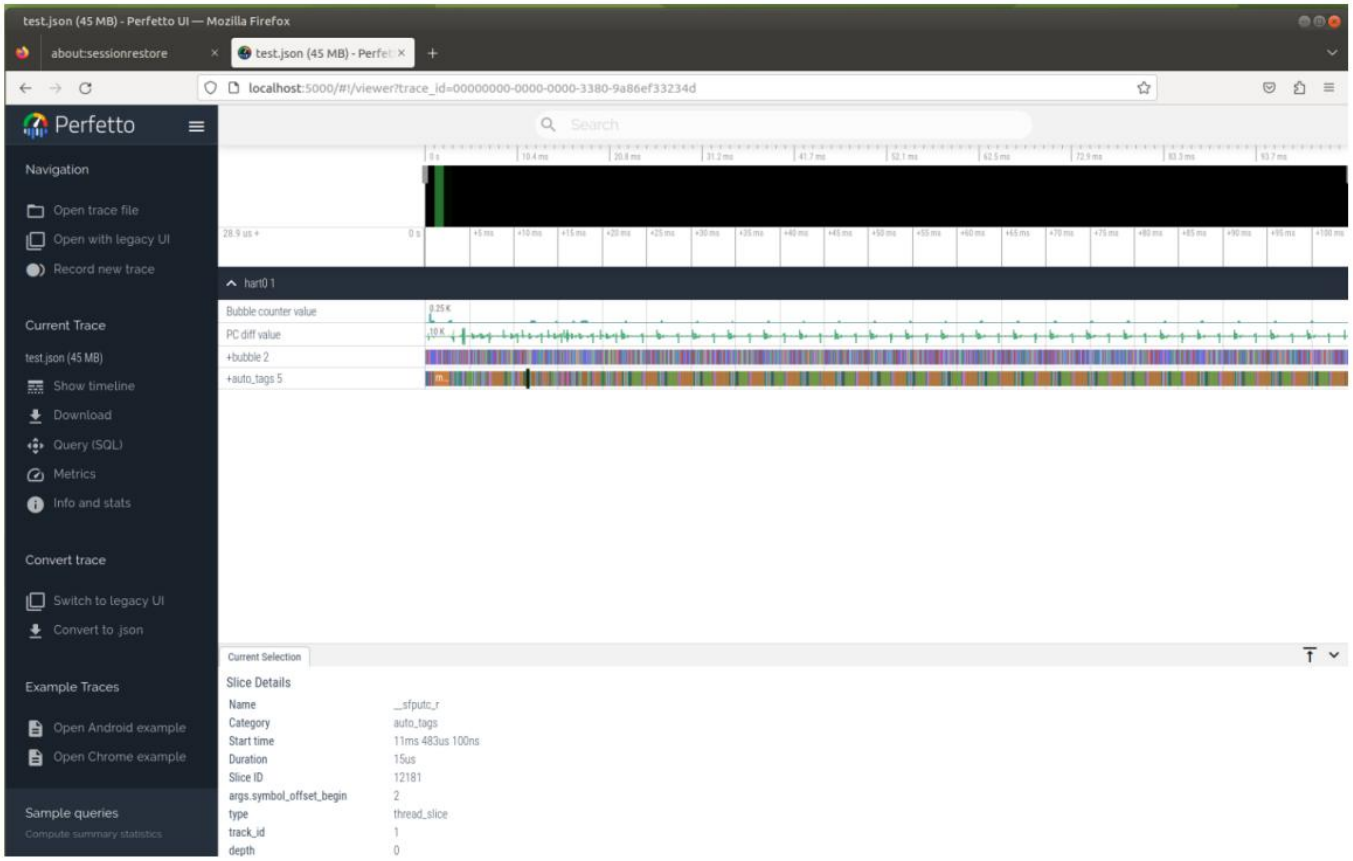


图 11- 10 rvprof trace 结果展示

13.使用 Nuclei Near Cycle Model 仿真性能分析

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 当前只有 Linux 版本，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。

13.1.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。

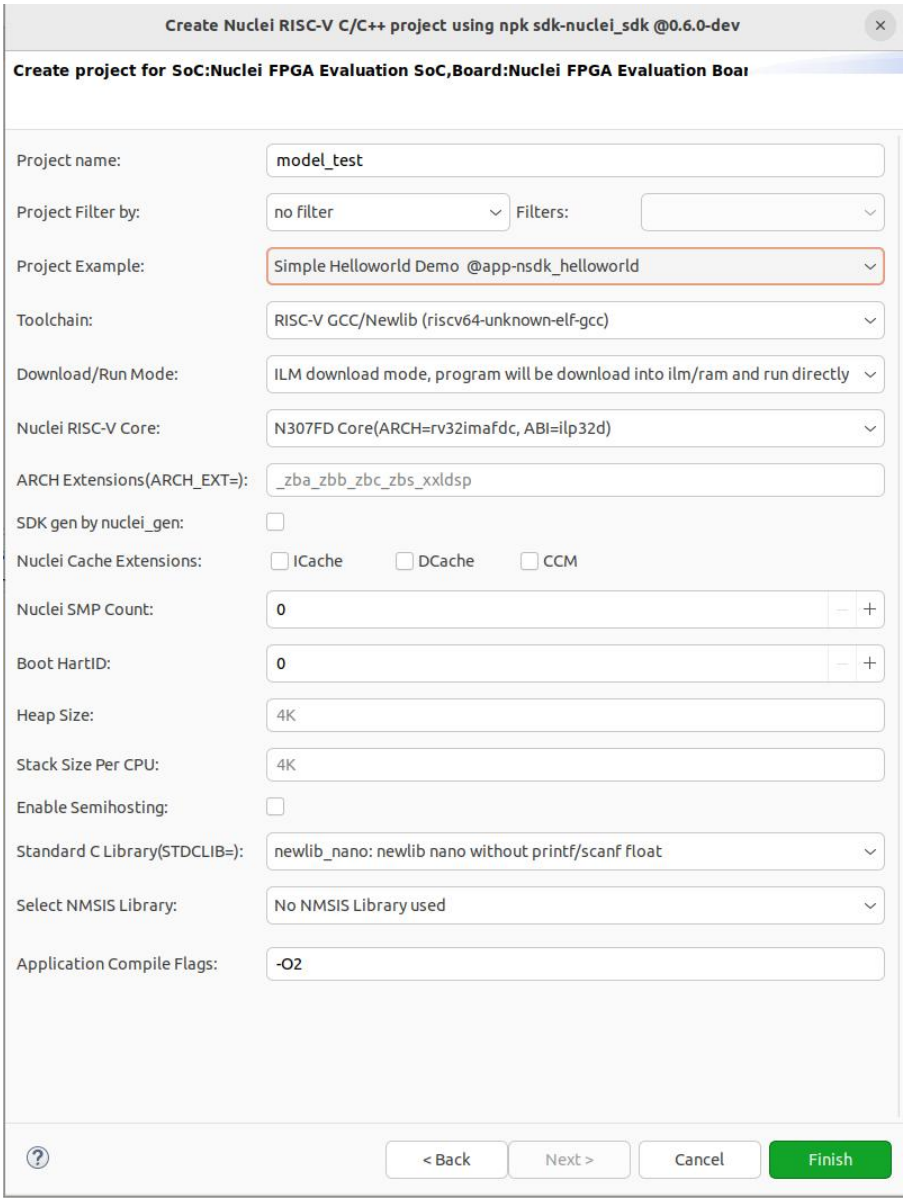


图 13-1 创建任意测试工程

Nuclei Near Cycle Model 采用 Nuclei Studio 中的 RVProf 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 RVProf 的配置，具体的配置及参数说明如下，其中在 Config options 中需要配置“--trace=1 --gprof=1 --logdir=Debug”,--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 文件存存放的路径为当前工程下的 Debug 目录。

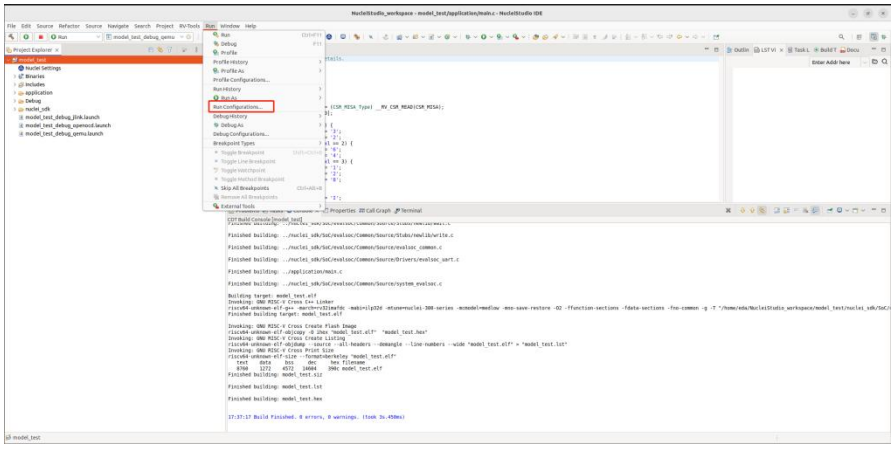


图 13- 2 打开 Run Configurations

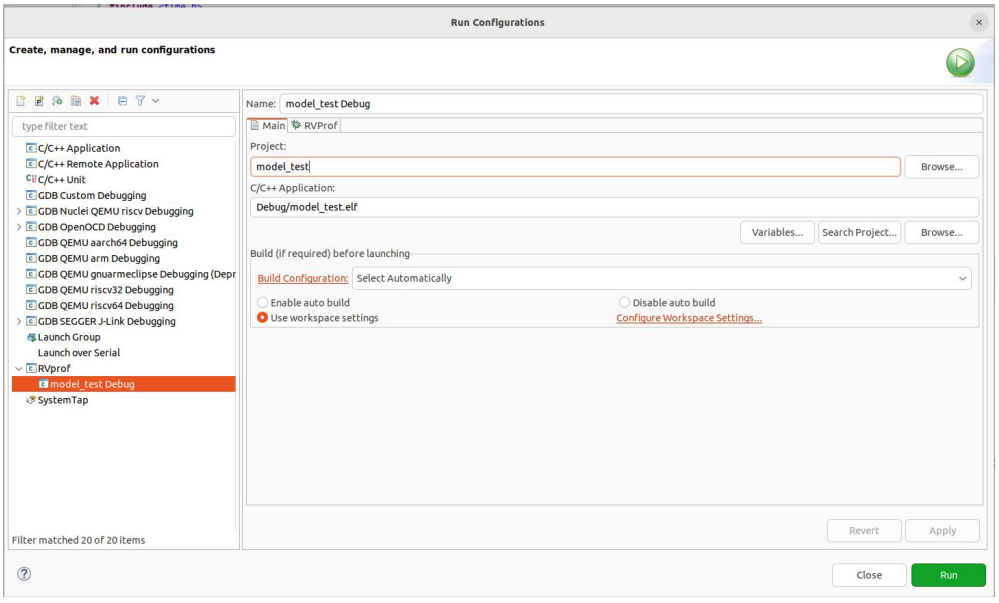


图 13- 3 配置 Run Configuration

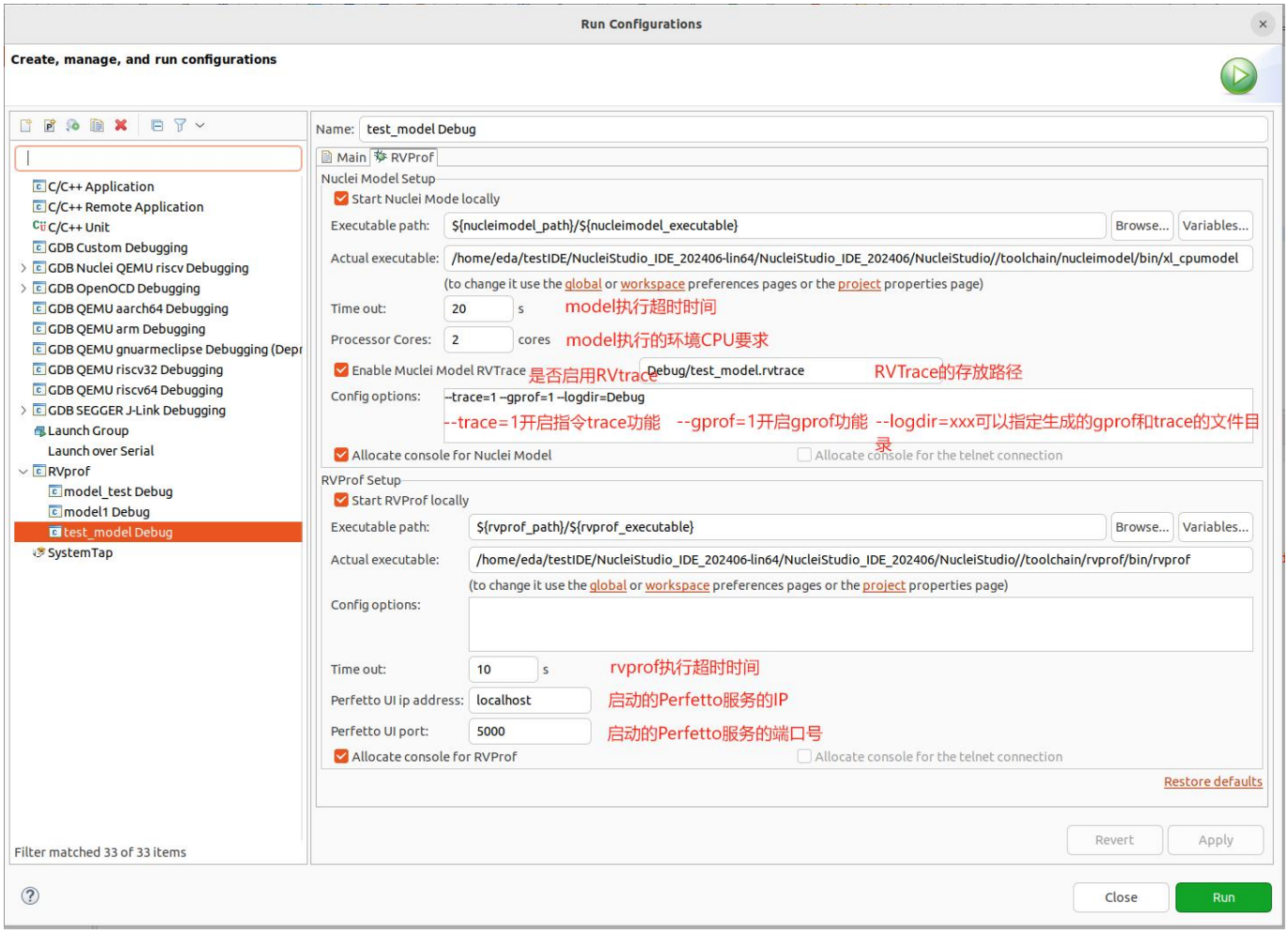


图 13- 4 配置 Run Configuration

13.2.运行工程并生成性能分析结果

点击运行（Run）工程，NucleiStudio 会依次调用 Nuclei Near Cycle Model 来仿真程序执行，并产生*.rvtrace 文件，再调用 rvprof 来解析*.rvtrace 文件并生成*.json 文件，最后启用一个 perfetto 服务来用来查看 rvprof 解析*.rvtrace 文件所产生的*.json 文件。具体可以参见第 12 章节内容。

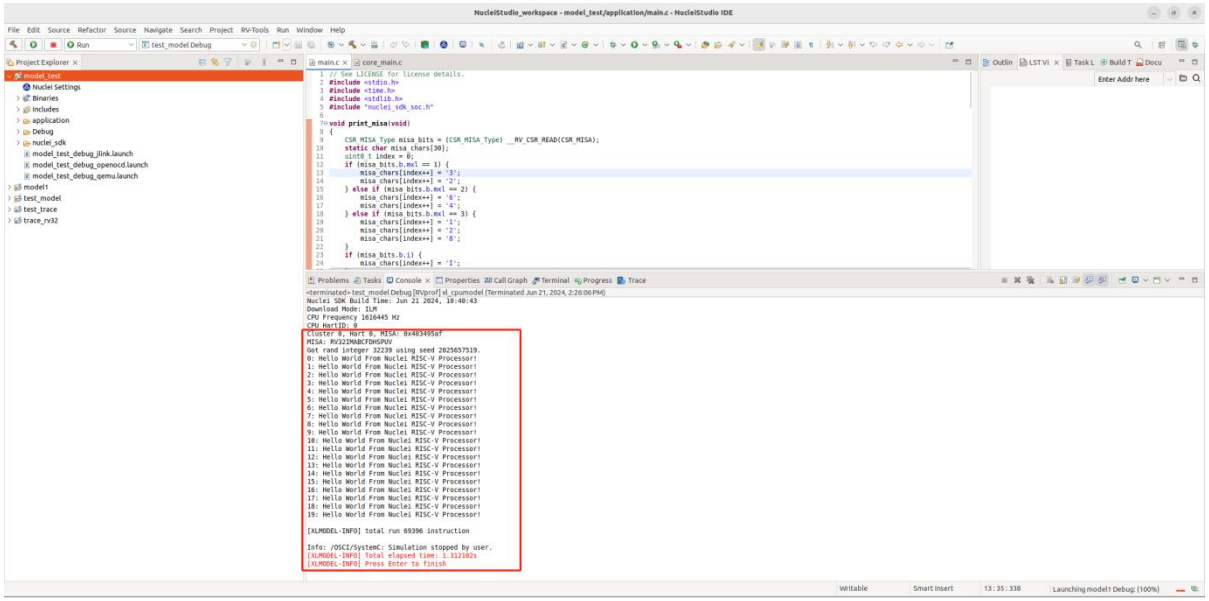


图 13- 5 运行程序

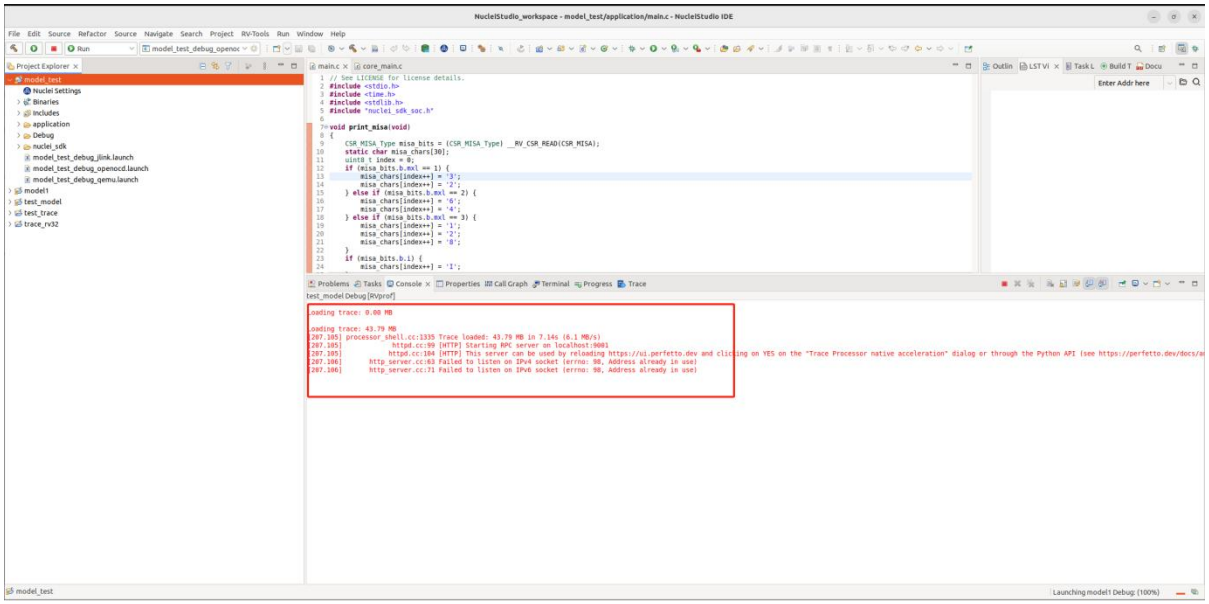


图 13- 6 程序在 Nuclei Near Cycle Model 中运行成功执行

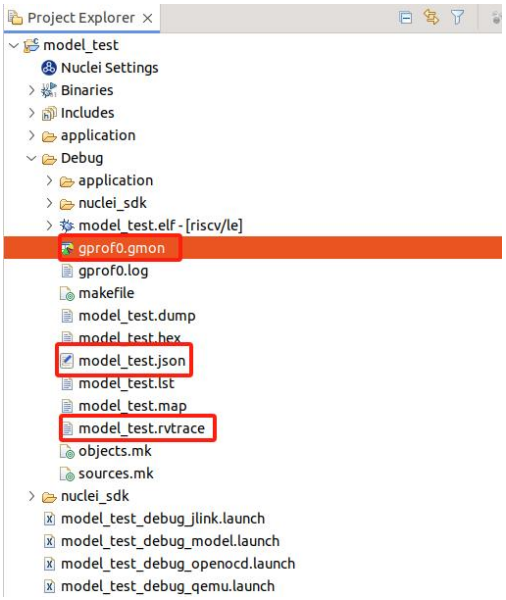


图 13- 7 查看生成的文件

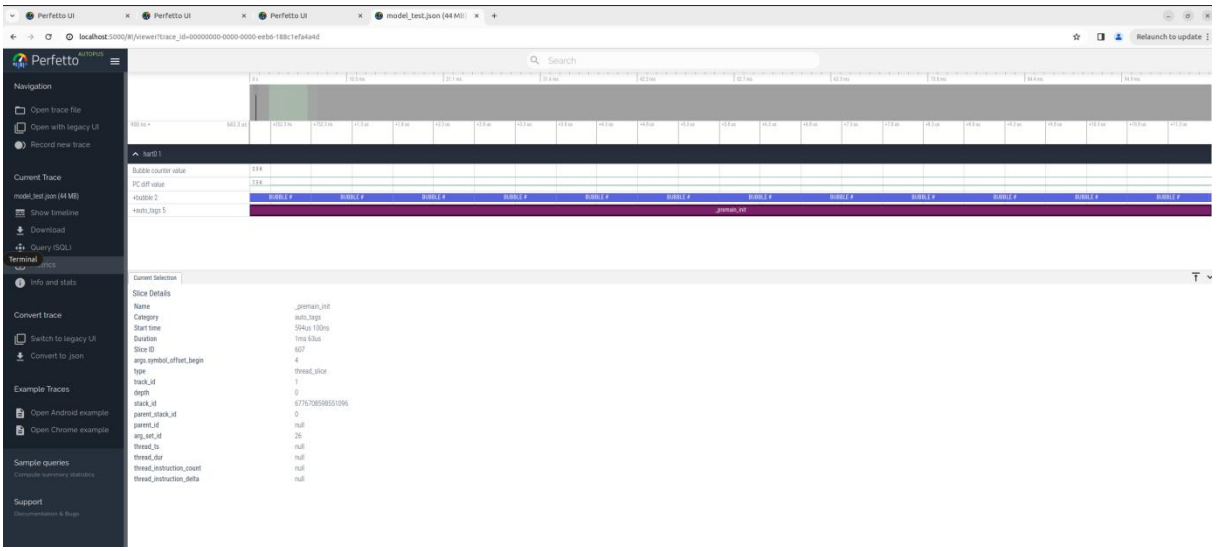


图 13- 8 通过 perfetto 查看*.rvtrace

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，具体可以参见第

10 章节内容。

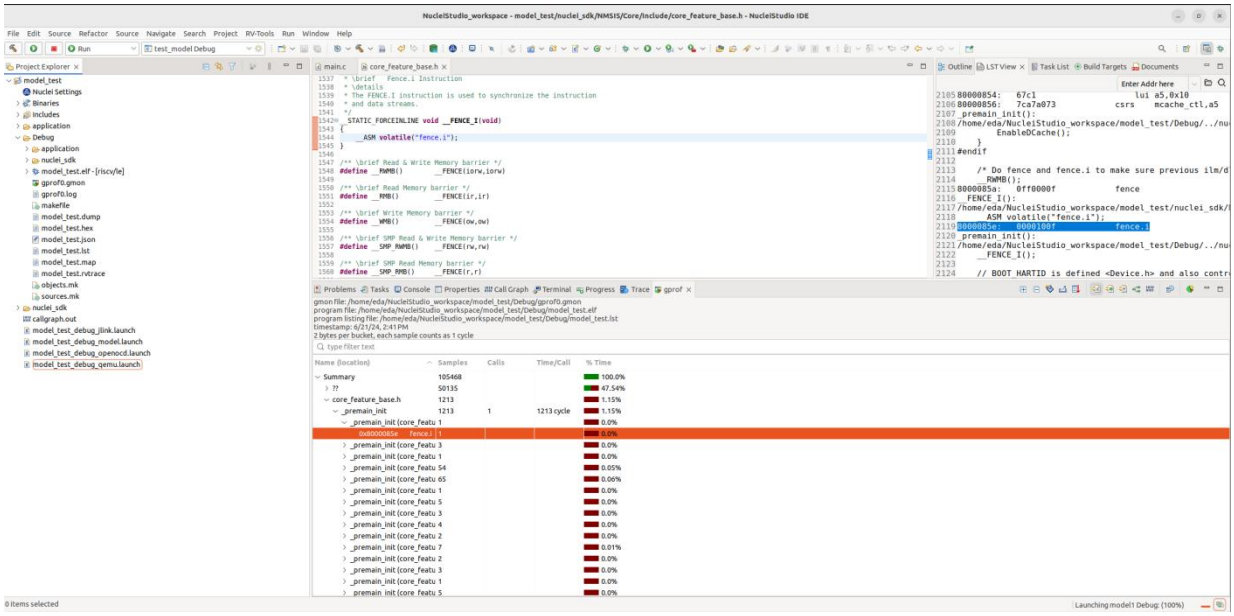


图 13- 9 通过 gprof 查看生成的*.gprof 文件

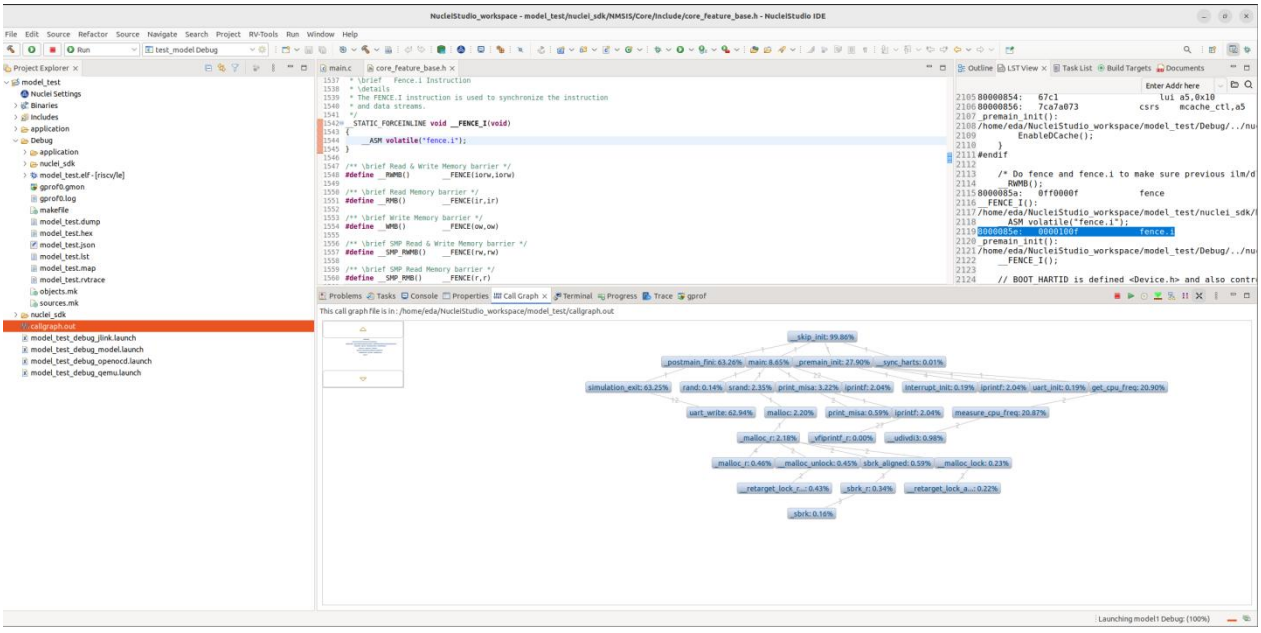
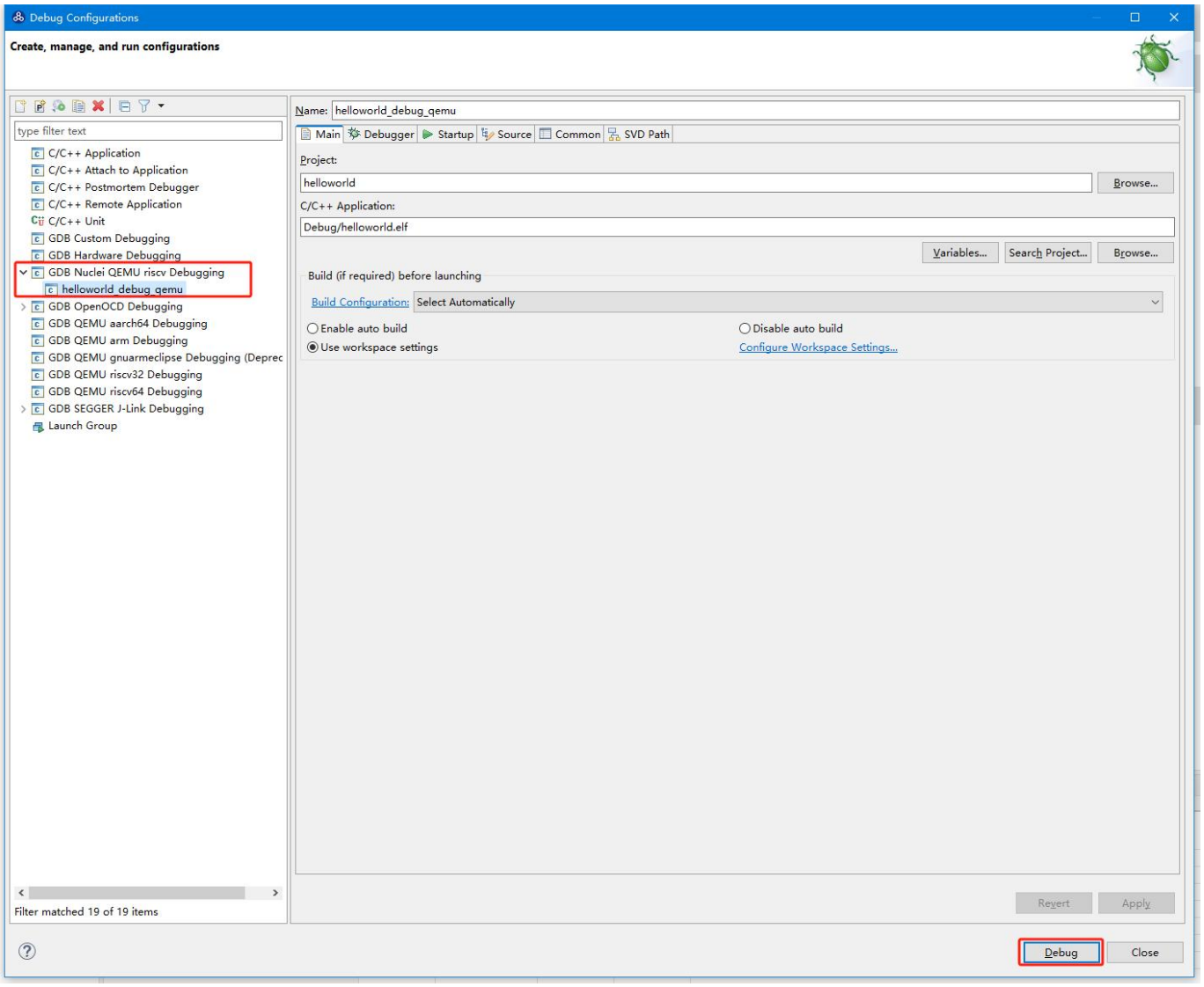


图 13- 10 查看 call graph 文件

14.使用 Nuclei QEMU 仿真调试

最新版本的 Nuclei Studio 整合了 Nuclei QEMU，支持芯来全系列 Core 的工程实现仿真调试功能，这里以 nuclei_evalsoc 工程为例。按照前文的内容新建一个 nuclei_evalsoc 工程，这里以 helloworld 工程为例，我们新建一个名为 helloworld 的工程。

右击新建的工程，选择“Debug As -> Debug Configurations”打开调试设置弹窗。如图 12-1，在弹窗中选择“GDB QEMU Debugging”下带有“qemu”后缀的设置选项。我们工程为“helloworld”，所以此处选择“helloworld_debug_qemu”，选中后点击右下方的“Debug”开始仿真调试。



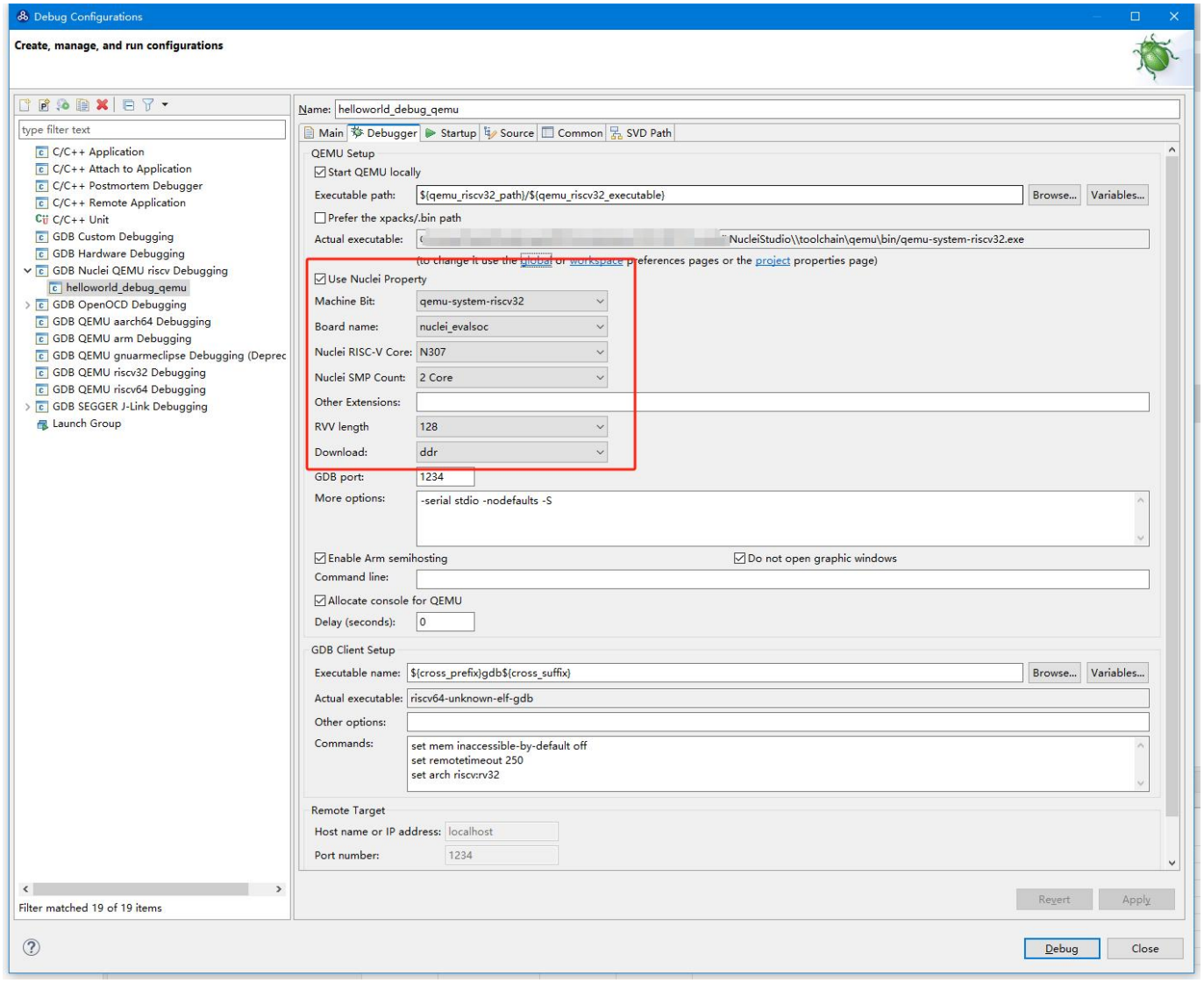


图 12-1 选择 QEMU 进行仿真调试

QEMU 的详细配置页，新增了 Machine Bit、Board name、Nuclei RISC-V Core、Download 选项，用户可以在这里对 QEMU 进行自定义配置，详细参见 Nuclei QEMU 用户手册。

■Machine Bit QEMU 可以仿真 32 位和 64 位两种设备

■Board name 针对不同的 Board 有不同的外设及配置

■Nuclei RISC-V Core QEMU 已经支持当前 nuclei 大多数的处理器型号，并会持续更新，支持更

多的最新处理器的型号。下面展示了支持的 CPU 型号的情况。

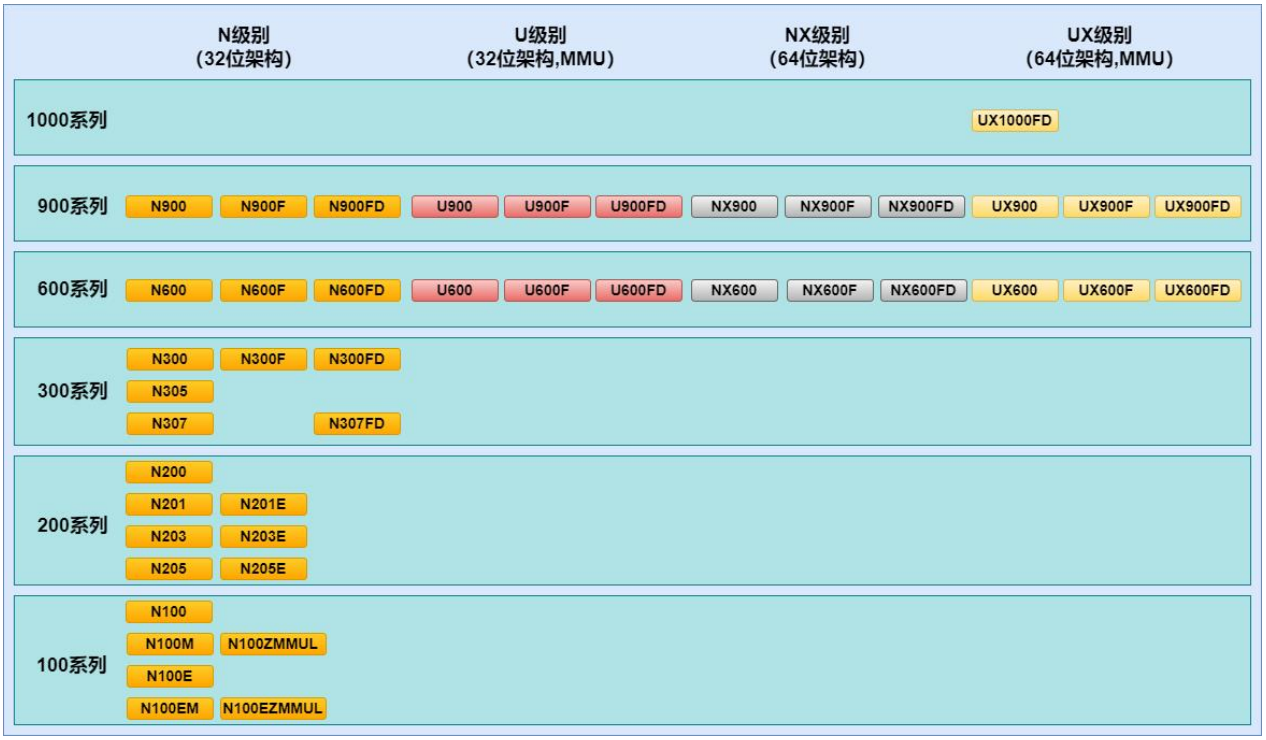


图 12- 2 QEMU 支持的 CPU 型号

■Download QEMU 仿真了不同的下载模式

备注：

1. Nuclei Studio 2023.10 版中所集成的 QEMU 对 rvstar 不再支持。

2. Nuclei Studio 2023.10 版不支持 QEMU semihosting 功能，如有必要，请在命令行下使用 qemu 的 semihosting 功能，2024.06 开始恢复支持。

3. Nuclei Studio 2023.10 版对之前版本创建的工程的 qemu 部分不兼容，需要修改，可以参考章节 8.1.2 内容

如图 12- 2 所示，仿真调试页面与一般的调试页面完全相同，其中，串口打印信息可直接在 Console 中查看。寄存器可能有部分未实现的会报错，属于正常现象。

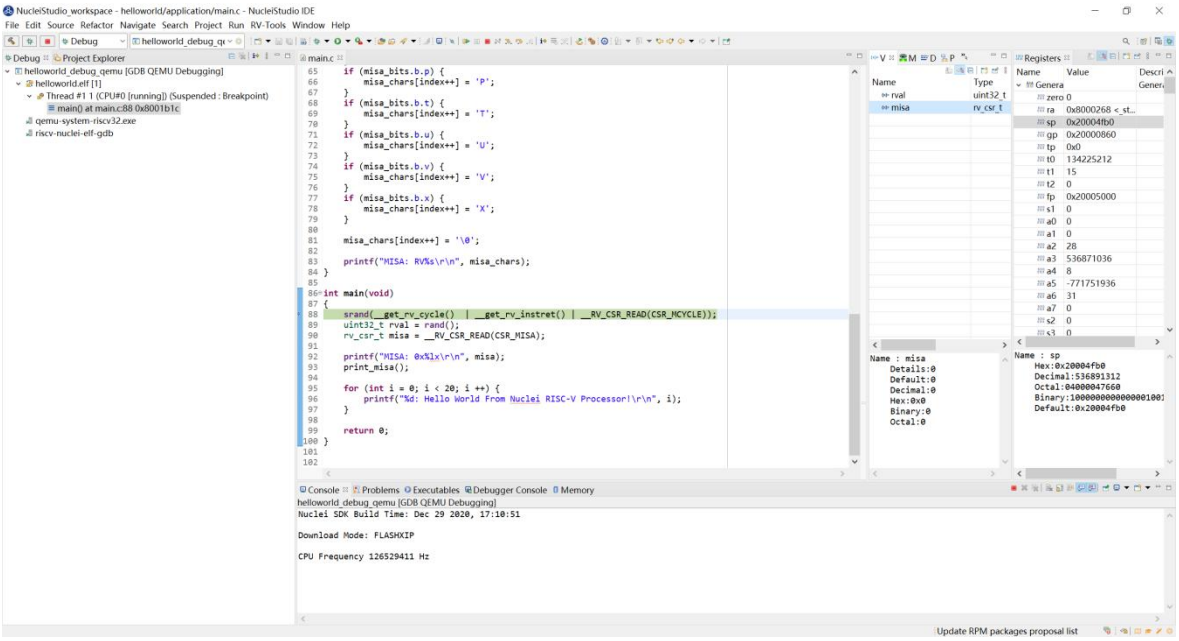


图 12- 3 QEMU 仿真调试界面

仿真运行流程与仿真调试相类似，右击 RVSTAR 工程选择“Run As -> Run Configurations”，弹窗中选择“GDB QEMU Debugging”下带有“qemu”后缀的设置选项。选中后点击“Run”，可以在控制台（Console 栏）看到运行的打印输出。

15.常见问题

15.1.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）

15.2.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。

15.3.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。

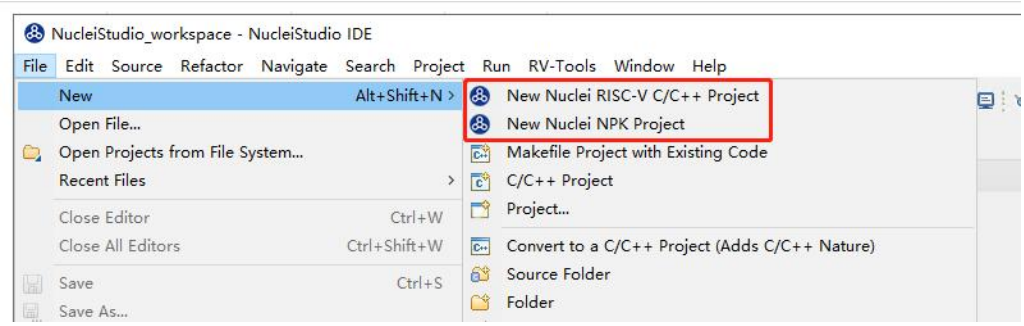


图 13-1 New Nuclei Risc-V C/C++ Project 菜单

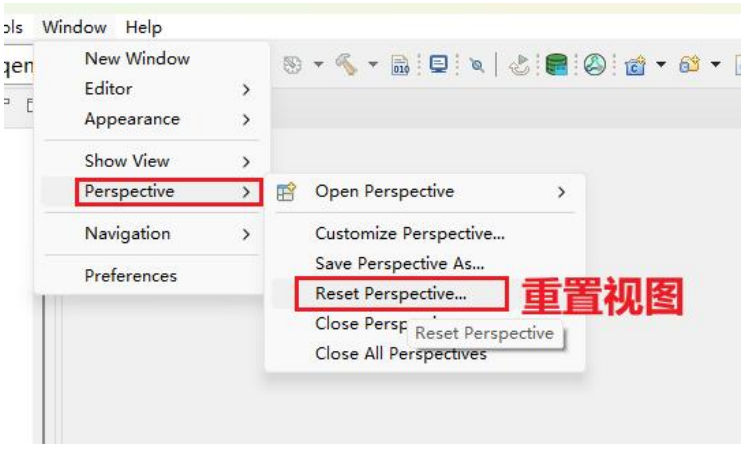


图 13-2 视图重置

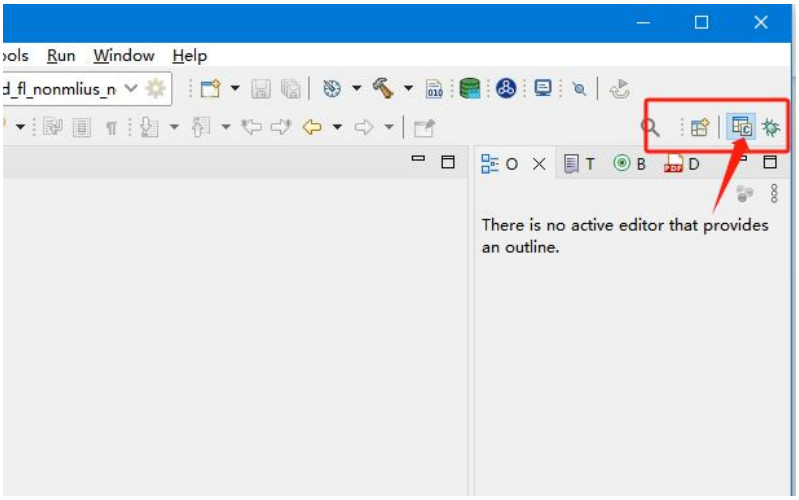


图 13-3 切换当前视图为 C/C++视图

15.4.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”，如图 13-4，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。

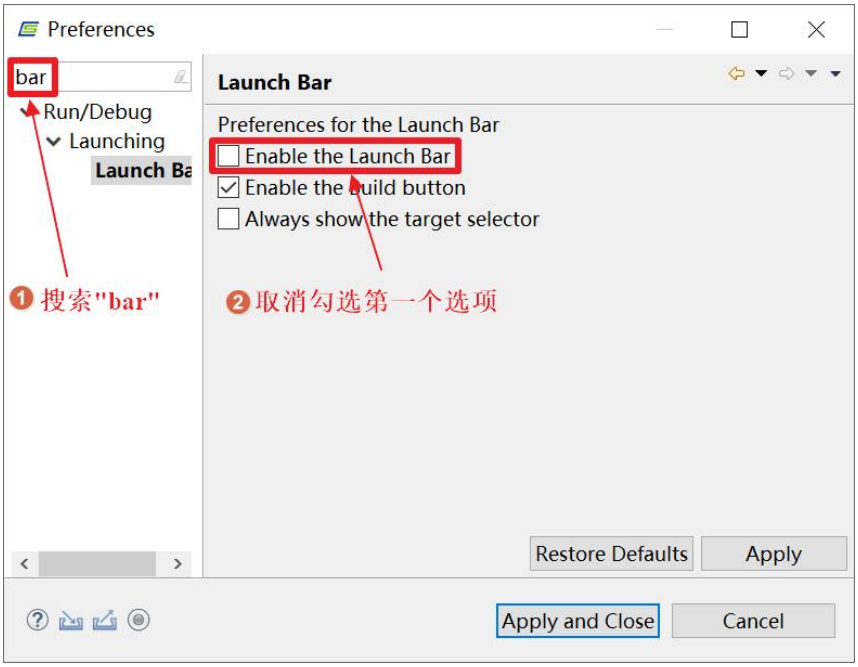


图 13- 4 关闭 Launch Bar

15.5.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：

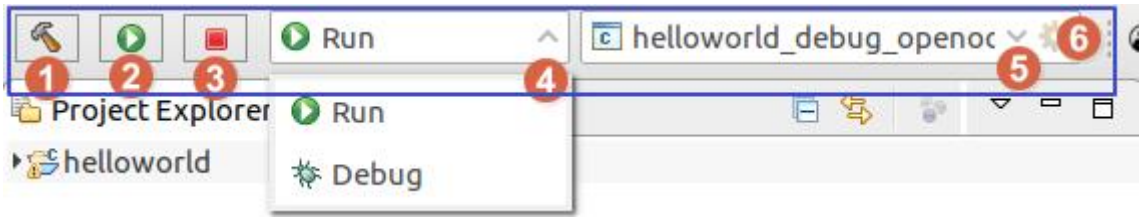


图 13- 5Launch Bar 功能介绍

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。

15.6.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，如图 10-6，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。

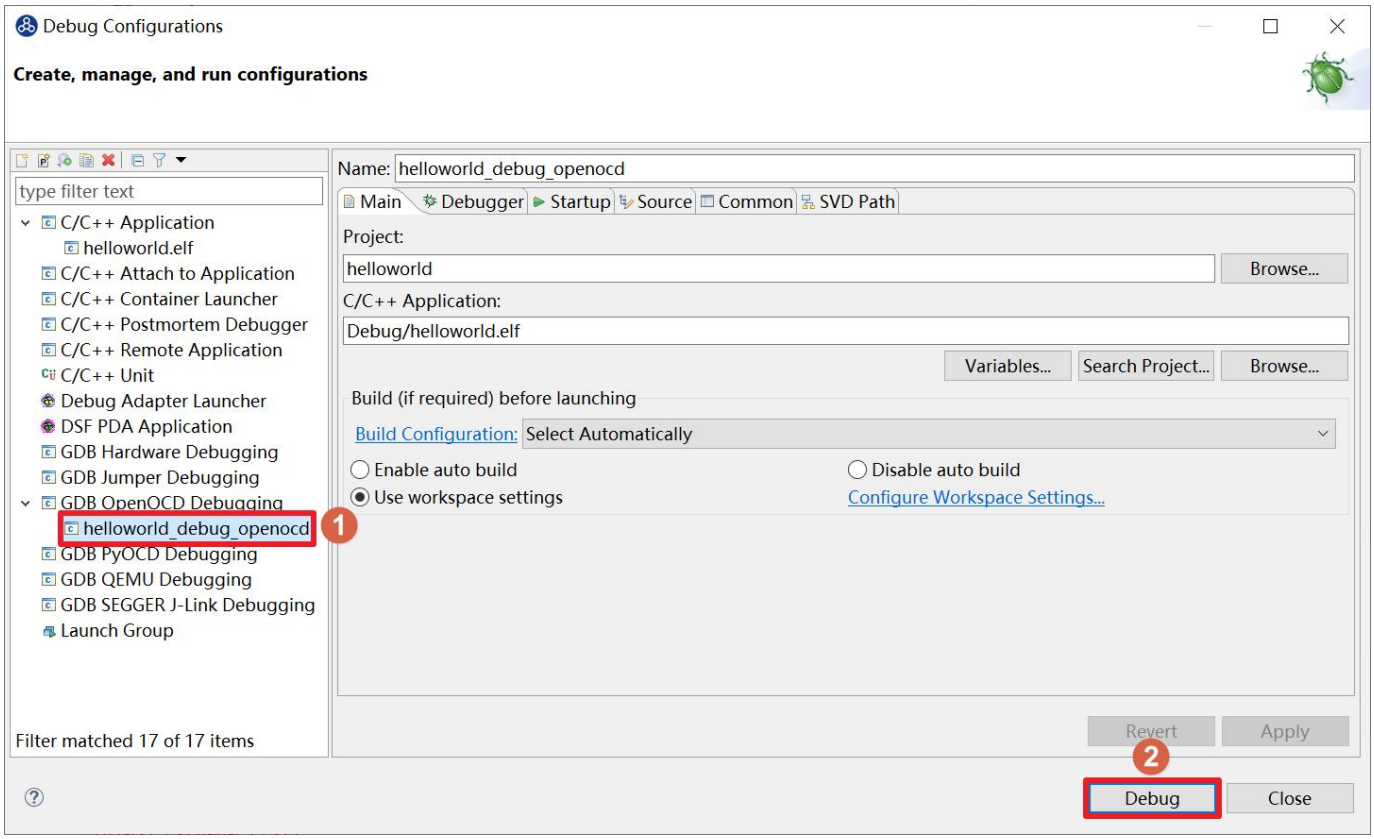


图 13- 6 不使用 Launch Bar 首次进行调试

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。

15.7.Debug 页面查看寄存器

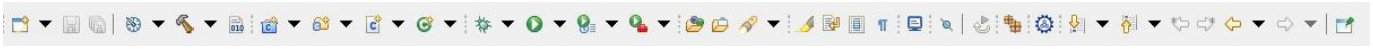
进入 Debug 模式后，可能有些页面并不能第一时间找到，这里以查看寄存器栏目为例。在菜单栏选择“Window -> Show View -> Registers”打开寄存器栏目，在 Register 页面可以通过 Ctrl – F 来查找寄存器。同样，如果想查看内存，可以选择“Window -> Show View -> Memory”，想打开调试控制台，可以选择“Window -> Show View -> Debugger Console”，其他页面此处不做过多介绍，请用户自行体验。

15.8.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。

15.9.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。

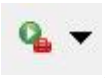
编译所有工程  ，一次性编译所有 Project Explorer 中的工程。

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。

15.10.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。如图 10-7，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）。

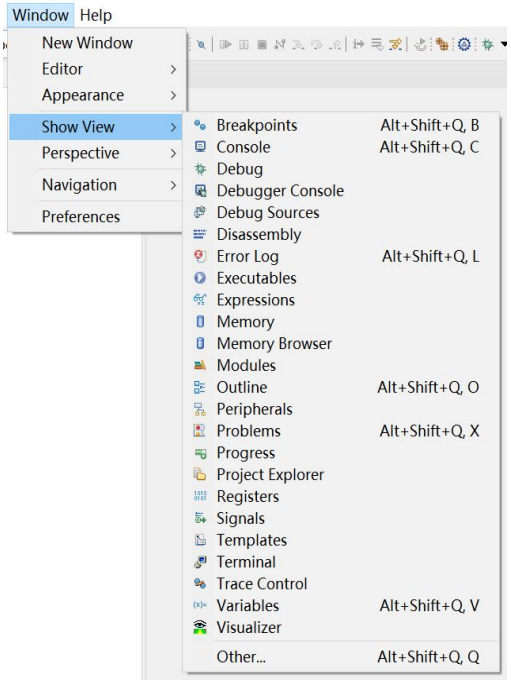


图 13-7 Debug 模式查看功能窗口

15.11.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。

15.12.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。如图 10-8，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。

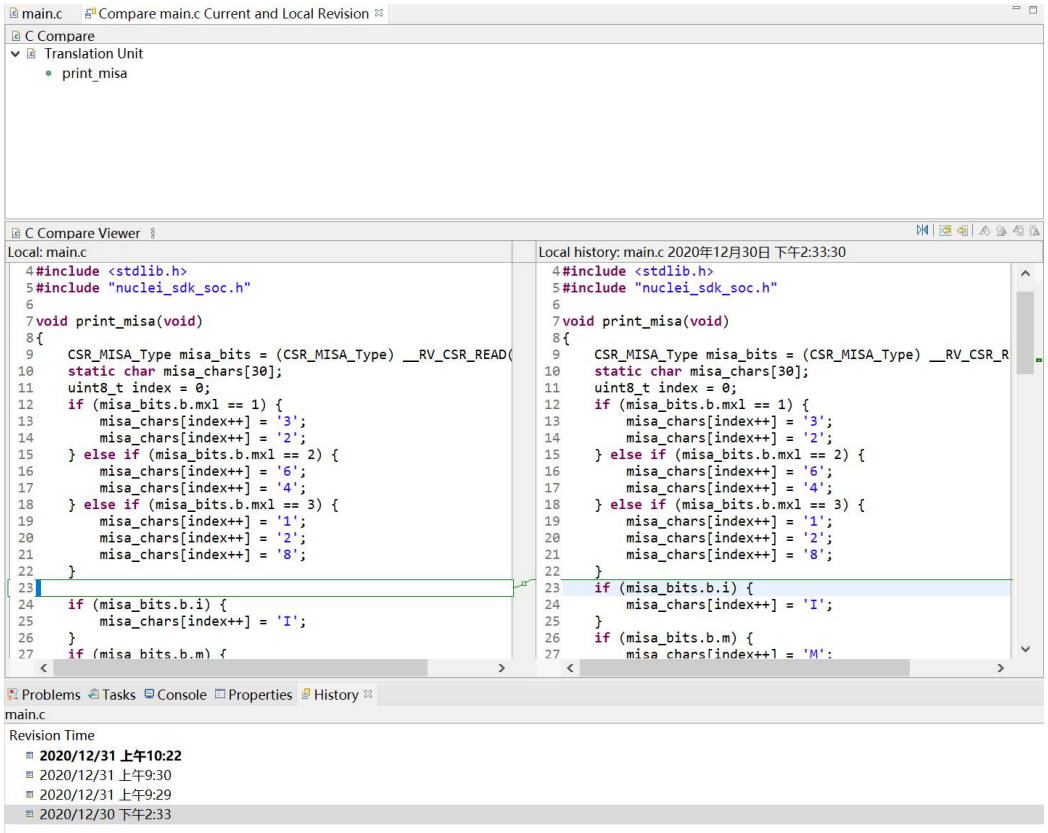


图 13-8 对比历史文件

15.13.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。

15.14.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。

15.15.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，如图 10-9，可点击红框中左右方向图标显示隐藏栏目。

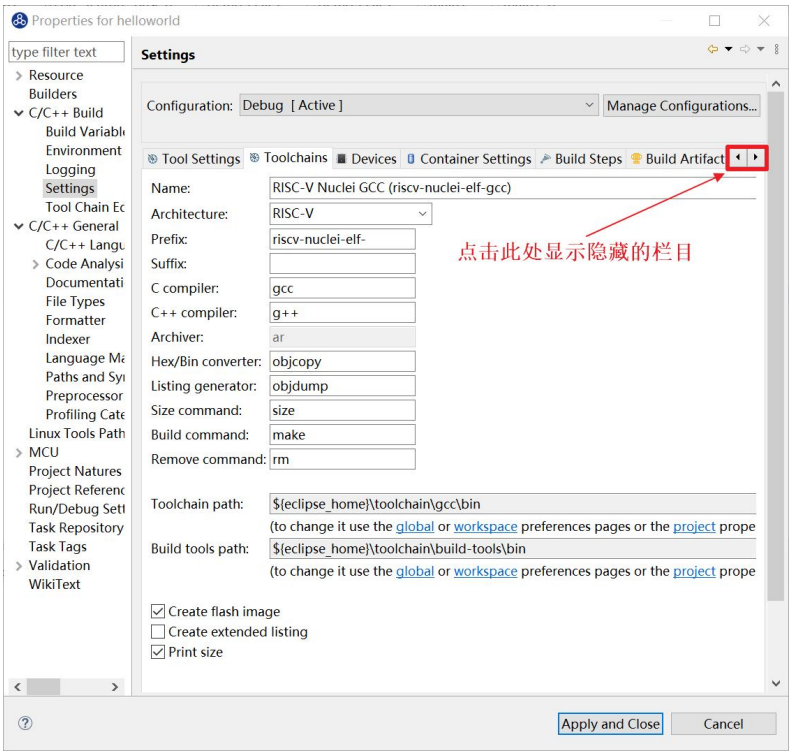


图 13-9 显示隐藏栏目

15.16.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。

15.17.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。

15.18.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，如图 10-10，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。

图 13- 10 设备管理器中显示两个串口

15.19.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:，如图

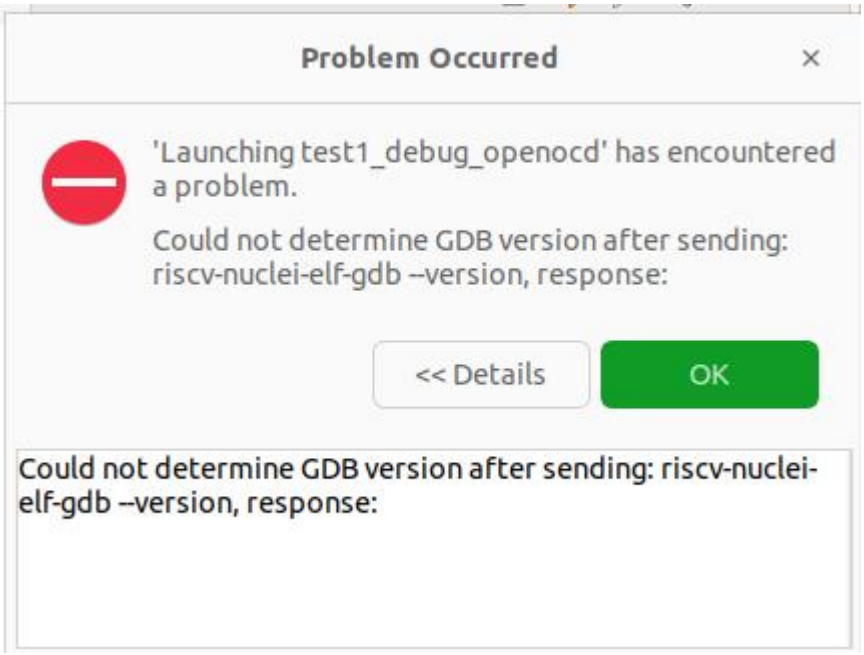


图 13-11 报错提示

riscv-nulcei-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-nulcei-elf-gdb)查看，依赖缺失

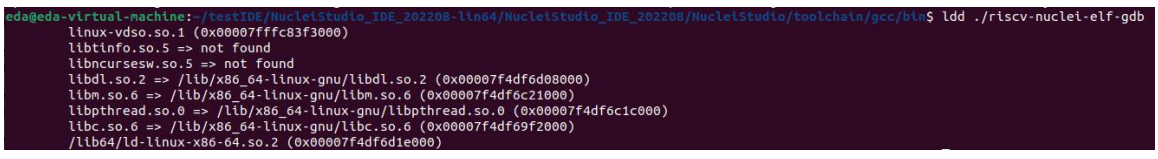


图 12-12 查看 gdb 缺失的依赖

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

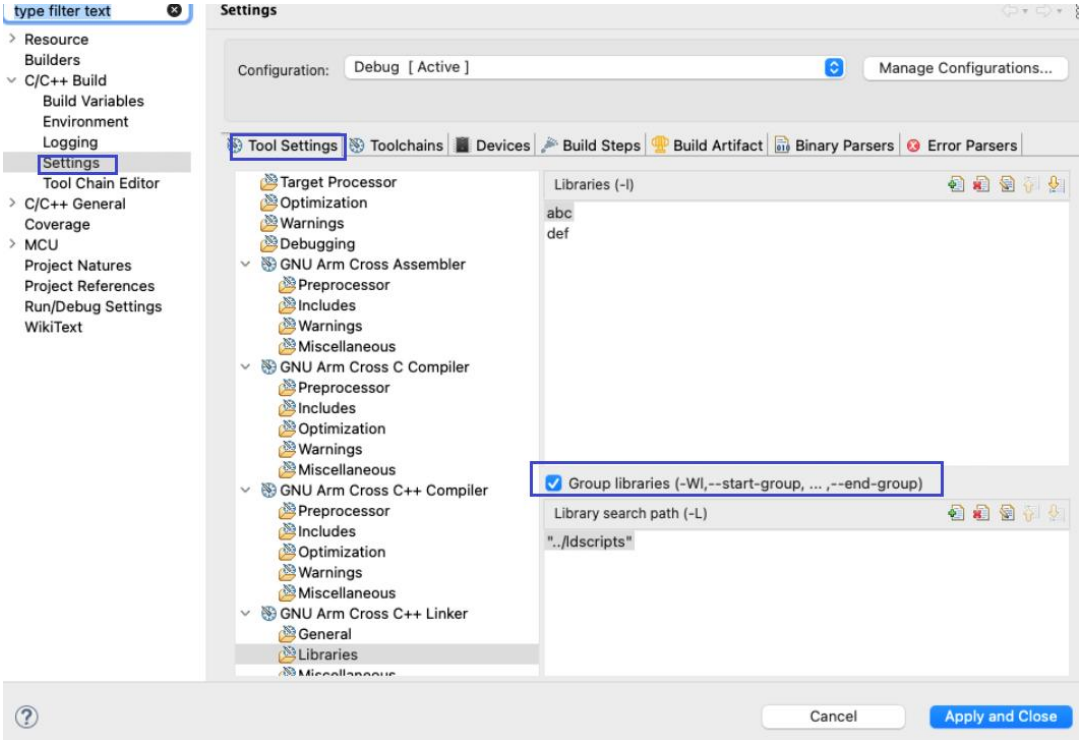
15.20.在 linux 下使用 QEMU 时报错

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**

15.21.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。

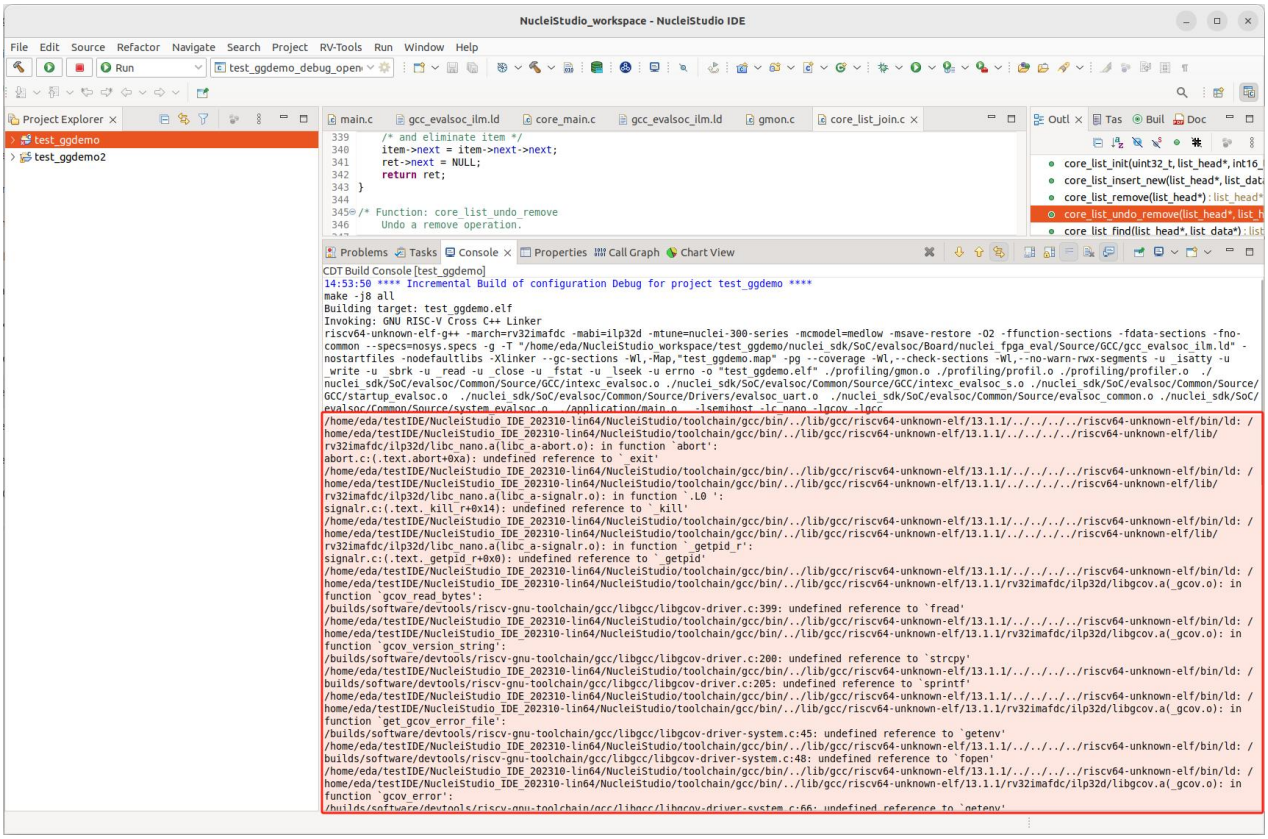


图 13-13 因为 Libraries 内部依赖而导致的编译报错

●通过调整 Libraries 的顺序或者添加多个链接来实现

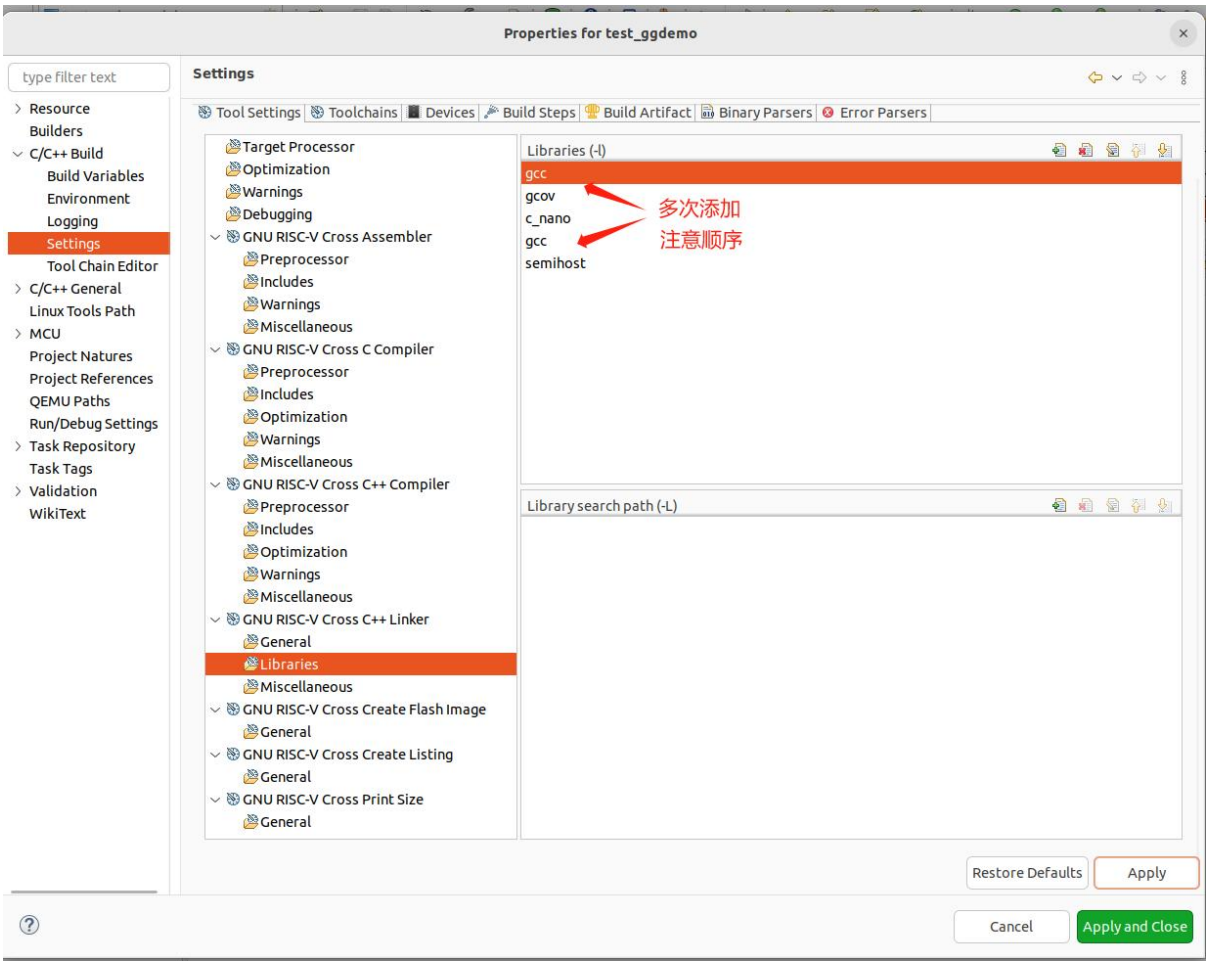


图 13- 14 整 Libraries 的顺序或者添加多个链接

●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。

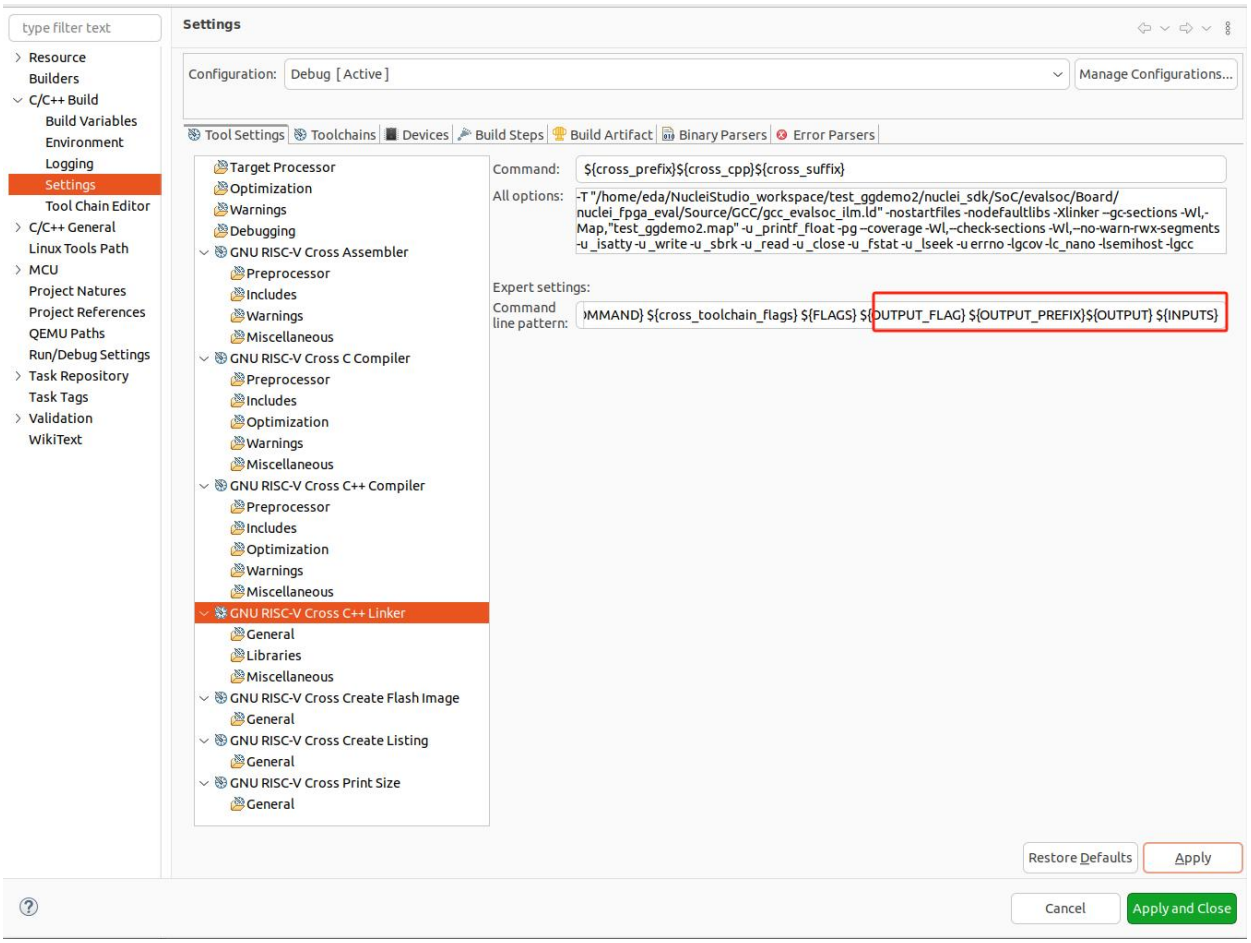


图 13- 15 修改 Command line pattern 内容

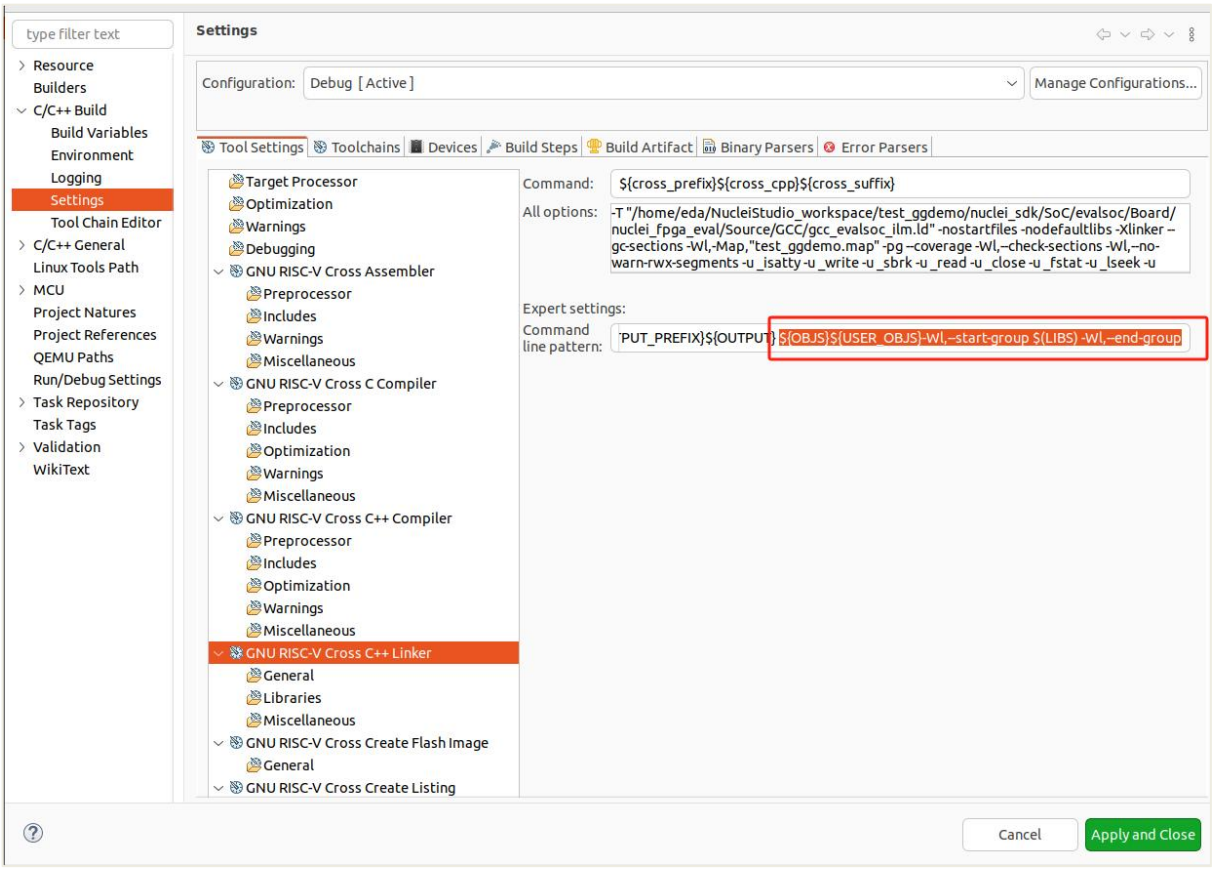


图 13- 16 修改 Command line paatern 内容

15.22.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。

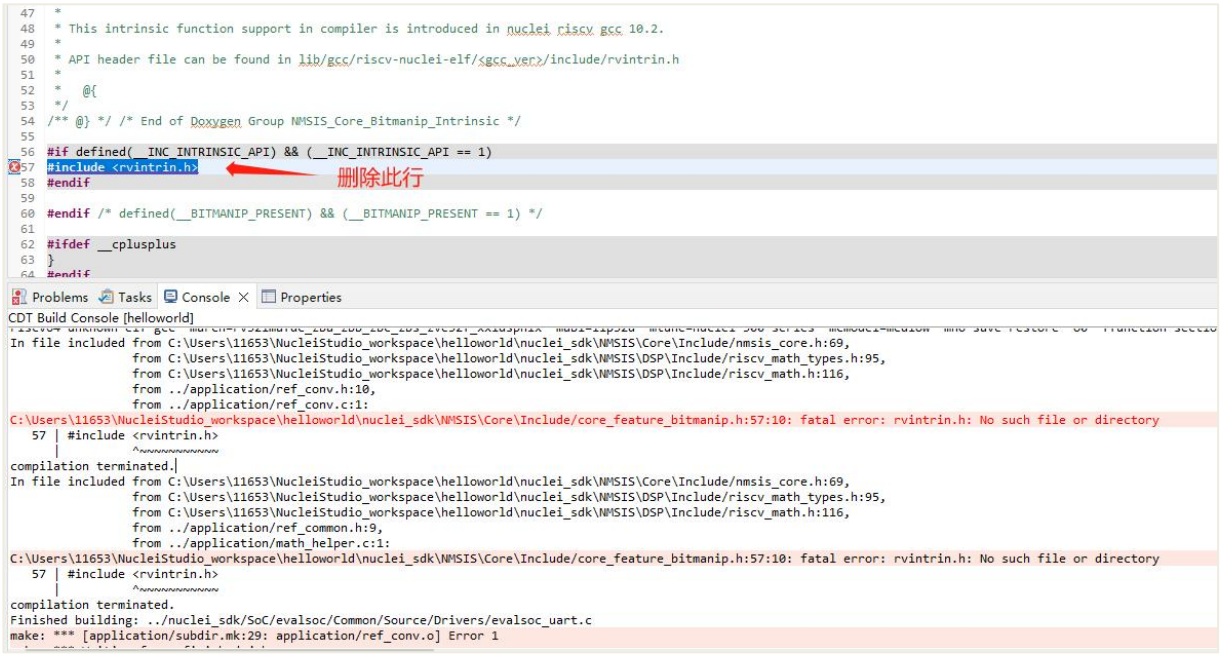


图 13- 17 fatal error: rvintrin.h: No such file or directory

15.23.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。

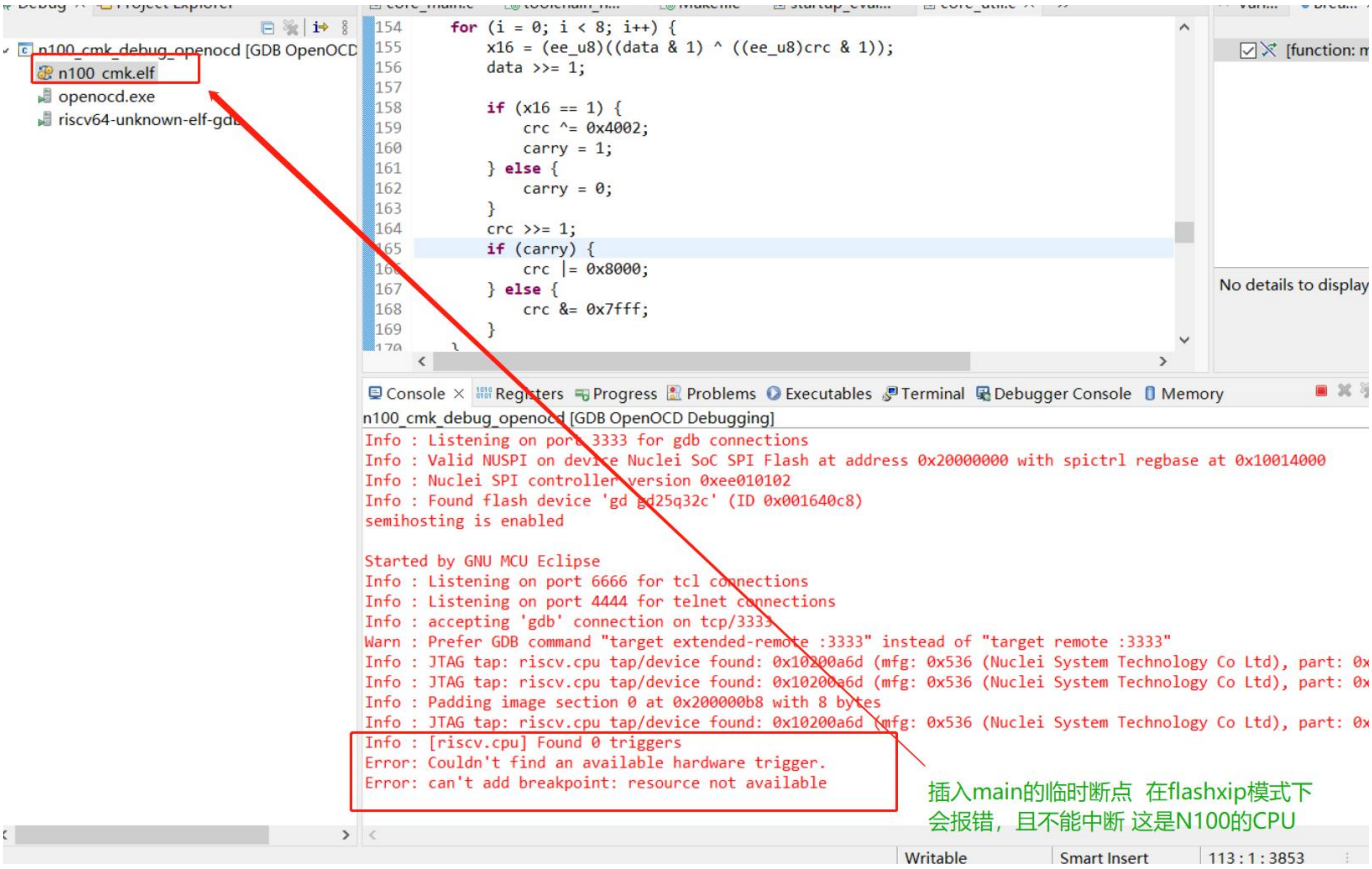


图 13- 18 Couldn't find an available hardware trigger.

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。

16.Nuclei Studio 升级

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。

16.1.GCC/OpenOCD 等工具链的安装

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\Nuclei Studio\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，如图 11-1 所示。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。

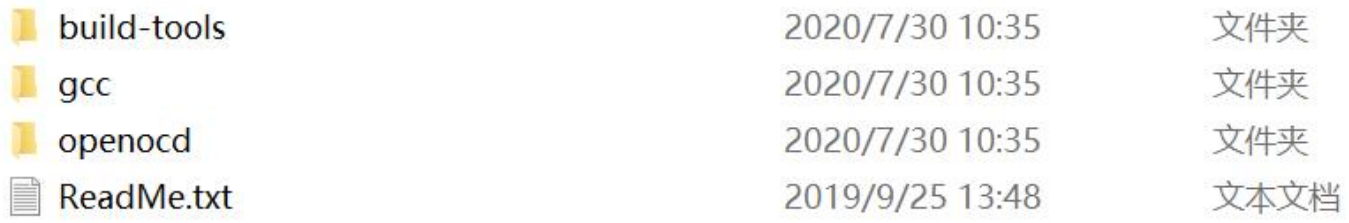


图 14-1 IDE 工具链路径

16.2.IDE Plugins 升级

Nuclei Studio 支持 IDE 内在线升级，步骤如下。

- 如图 11-2，打开安装软件弹窗，在菜单栏选择“Help → Install New Software”。
- 如图 11-3，点击“already installed”打开已安装界面。
- 如图 11-4，选择“Nuclei Studio”，点击“update”即可自动完成在线更新。

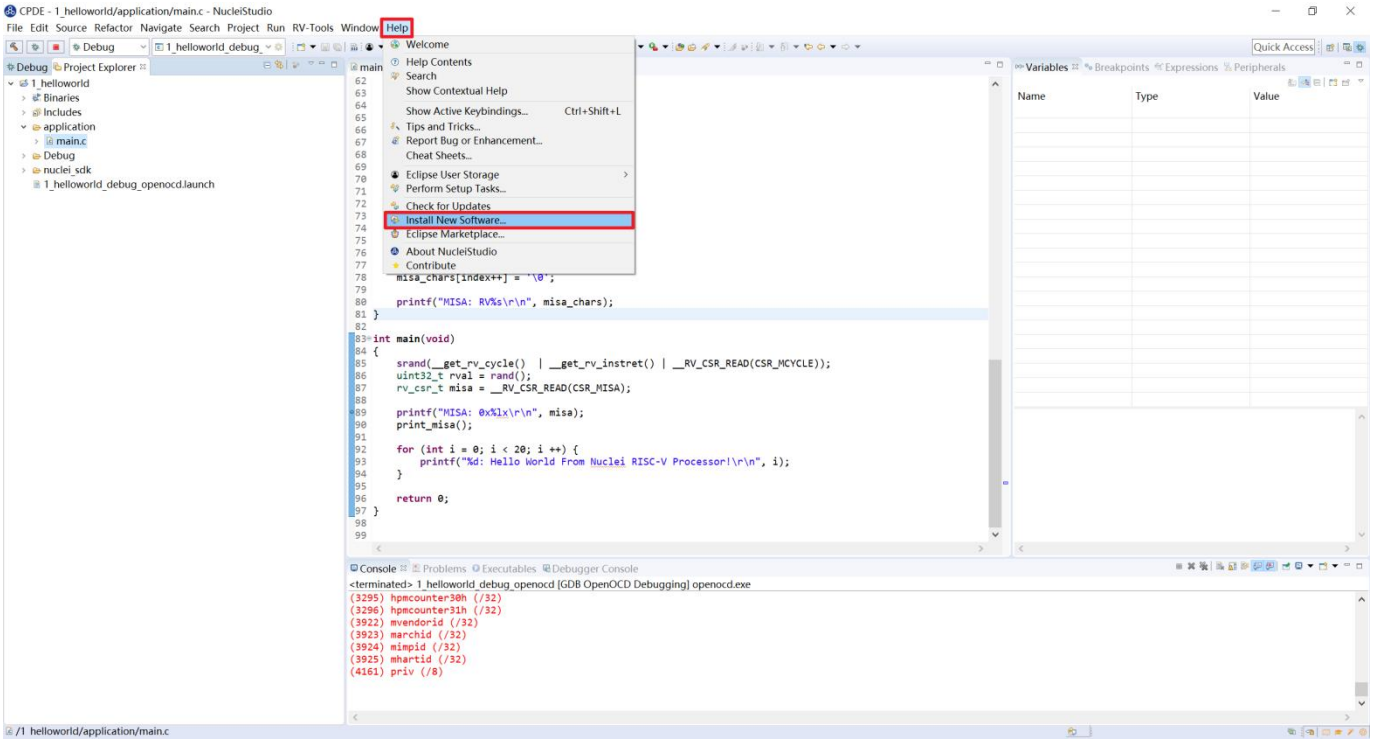


图 14- 2 打开安装软件弹窗

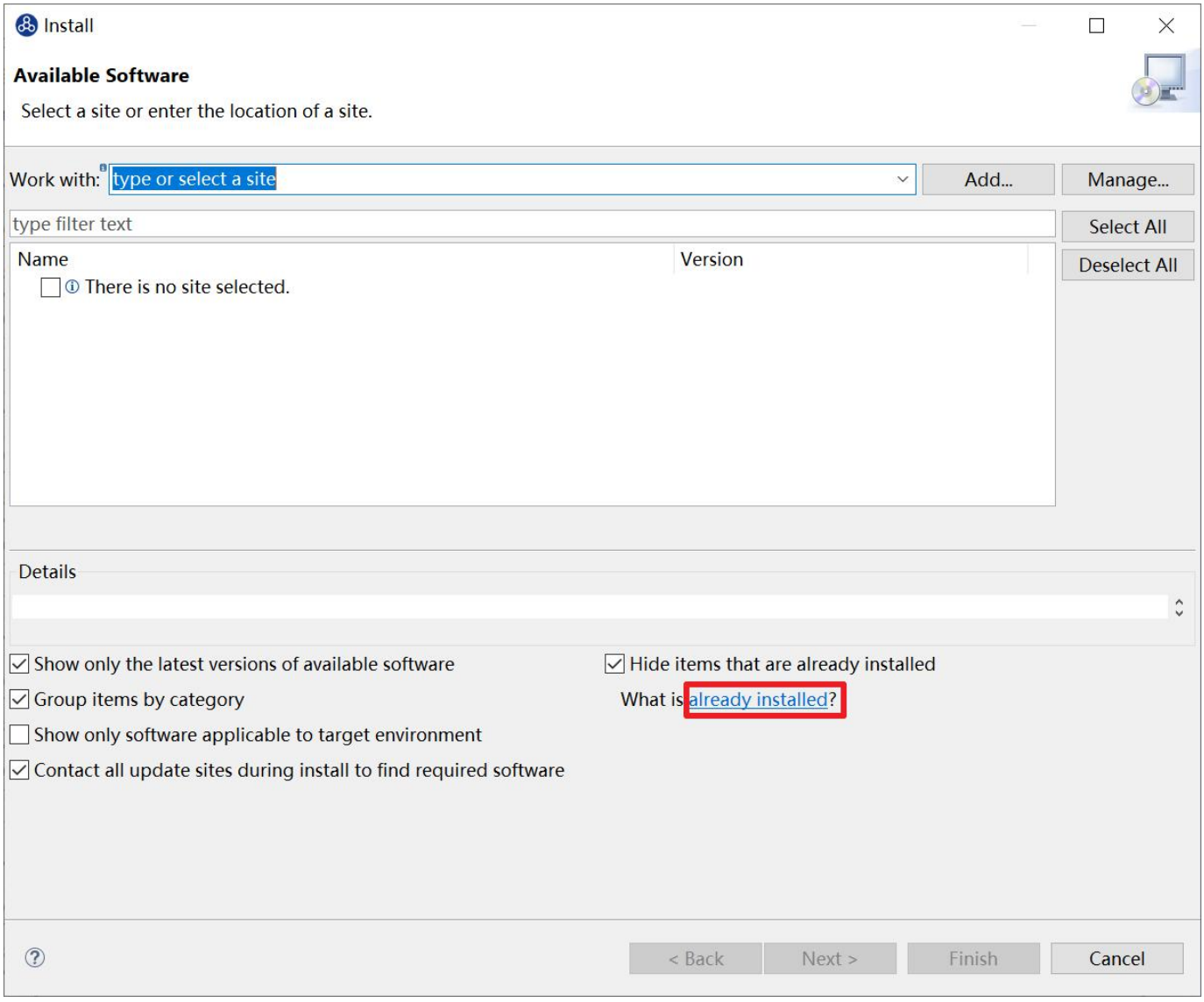


图 14-3 打开已安装界面

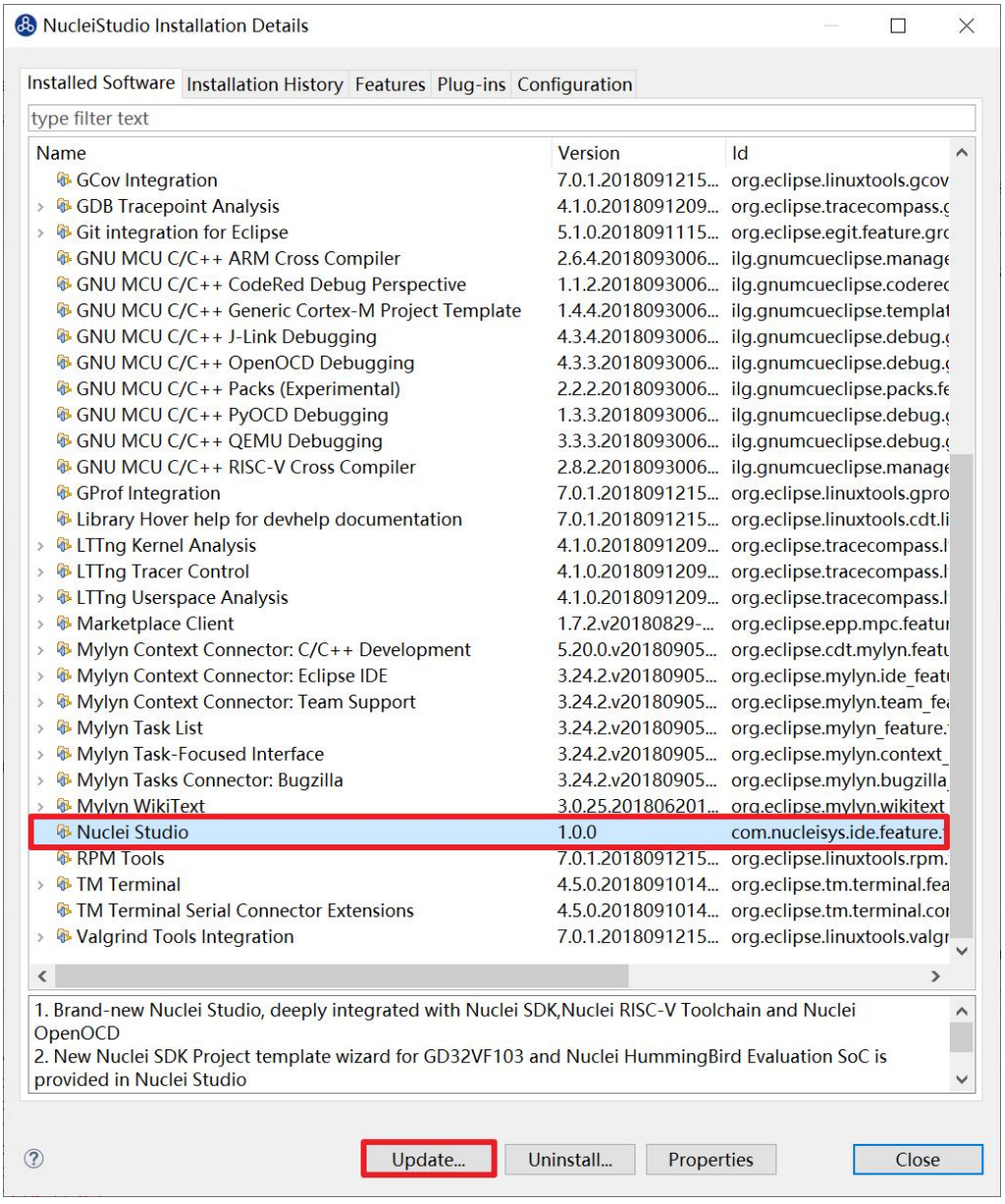


图 14- 4 选择“Nuclei Studio”更新

17.创建并贡献 Nuclei Studio 组件包

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 Nuclei Studio 组件包，并可以通过平台将自己的 Nuclei Studio 组件包贡献

出来，供其他开发者使用。具体可以参考在线文档

在线文档：https://www.rvmcu.com/Nuclei Studio-userguide-id-27.html

18.其他未注明或者遇到的版本问题

如本文档中有疏漏的地方，请关注 https://www.rvmcu.com/Nuclei Studio-faq.html 这里将列出不同版本后续遇到的常见问题。