Nuclei Studio Supply Documents

Table of contents

在编译工程时，使用了Pre-build Command/Post-build Command时报错

升级npk.yml以支持Nuclei Studio 2023.10

GCC13 auto generated RVV instructions when RVV enabled

更新 Nuclei Studio 2023.10 到最新修正版本

OpenOCD在操作容量大于16M-Byte的nor-flash时的问题

通过修改.cproject文件，升级工程工具链到GCC 13

在Nuclei Studio下用命令行编译工程

OpenOCD烧写程序时报错Error:Device ID 8xle2g8a6d is not known as FESPI capable

关于dhrystone在IDE上跑分和NSDK 0.5.0命令行跑分不一致的问题

Error: Couldn't find an available hardware trigger / Error: can't add breakpoint: resource not available

cannot find -lncrt_balanced: No such file or directory

UnsatisfiedLinkError of swt-win32-4965r8.dll on Windows 7

使用 Profiling 功能时可能遇到的一些问题

Nuclei Studio使用Profiling功能进行性能调优举例

通过Profiling展示Nuclei Model NICE/VNICE指令加速

Nuclei Model结合Nice Wizard快速验证NICE/VNICE指令加速

Flash Programming使用案例

Live Watch 功能的使用

在llvm中新增自定义汇编指令教程

如何使用芯来提供的DebugMap寄存器分析错误现场

在binutils中新增自定义汇编指令教程

OpenOCD 中 Nuclei 交叉触发功能使用指南

OpenOCD对FreeRTOS的调试支持使用指南

如何同时使用多个蜂鸟调试器进行调试

Nuclei SDK基于evalsoc快速适配customsoc

在链接脚本中使用OVERLAY命令

Nuclei Studio Supply Documents¶

This repository is utilized for providing supply documents, user guides, wikis, and facilitating discussions related to Nuclei Studio.

> **Note**
>
> - The latest version of Nuclei Studio IDE is 2025.02, which can be found in https://github.com/Nuclei-Software/nuclei-studio/releases/tag/2025.02
> - In Ubuntu 20.04, you must install `libncursesw5 libtinfo5 libfdt1 libpixman-1-0 libpng16-16 libasound2 libglib2.0-0` to make riscv64-unknown-elf-gdb and qemu able to run.

PDF Version can be found here: https://doc.nucleisys.com/nuclei_studio_supply/pdf/nuclei_studio_supply.pdf

- Nuclei Studio IDE Documentation: https://doc.nucleisys.com/nuclei_tools/ide/index.html
- Nuclei Tools(Toolchain/OpenOCD/Qemu/Model) Documentation: https://doc.nucleisys.com/nuclei_tools/
- Nuclei Studio NPK Introduction:
- https://github.com/Nuclei-Software/nuclei-sdk/wiki/Nuclei-Studio-NPK-Introduction
- https://doc.nucleisys.com/nuclei_tools/ide/npkoverview.html

Please create new doc based on Doc Template

Click this link to see online version.

> 如果您在文档中发现任何拼写错误或不完善之处，我们欢迎您提交Pull Request或Issue，以协助我们进行改进！
>
> If you come across any spelling errors or areas that need improvement in the document, feel free to submit a Pull Request or Issue to help us enhance it!

Documents¶

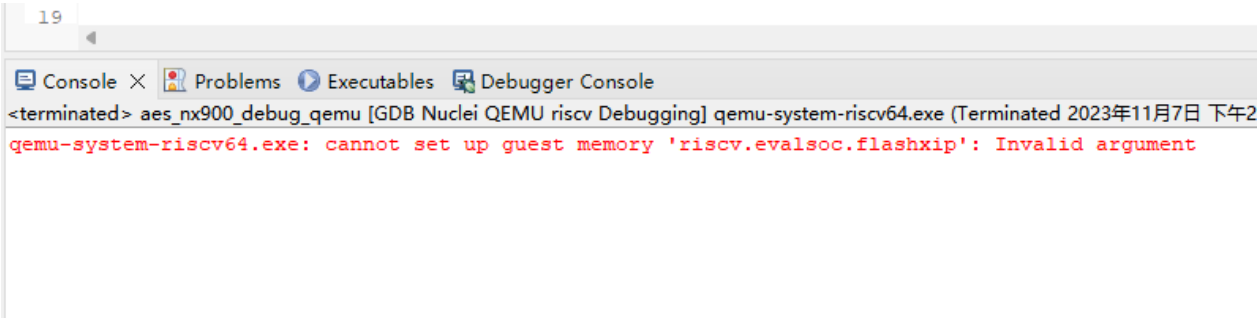
> Generated by `python3 update.py` @ 2025-09-28 02:56:44

因内存不足，导致在Nuclei Studio中启动qemu失败¶

问题说明¶

在实际开发中发现，因电脑同时运行了很多的进程或者电脑本身的系统内存不足，致使在Nuclei Studio中，使用qemu进行程序调试时，可能出现如下报错:

```
qemu-system-riscv64.exe: cannot set up quest memory
'riscv.evalsoc.flashxip' Invalid argument
```



解决方案¶

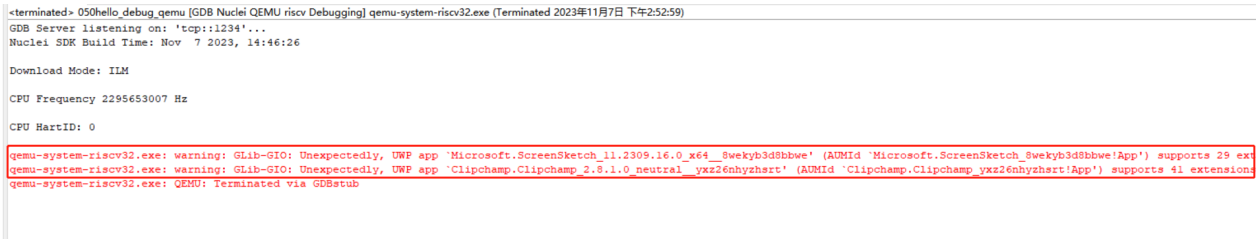
一般可以通过关闭某些应用，释放一部分内存以供qemu使用，即可解决些问题。

windows 11下使用Nuclei Studio进行qemu调试程序时报错¶

问题说明¶

windows 11下使用Nuclei Studio开发时，当使用qemu调试程序时,会有报错如下，是因为在windows 11下缺少相关依赖，但一般不影响qemu的正确使用，可以忽略此错误。

```
qemu-system-riscv32.exe: warning: GLib-GIO: Unexpectedly, UWP app
`Microsoft.ScreenSketch_11.2309.16.0_x64__8wekyb3d8bbwe' (AUMId
`Microsoft.ScreenSketch_8wekyb3d8bbwe!App') supports 29 extensions
but has no verbs
qemu-system-riscv32.exe: warning: GLib-GIO: Unexpectedly, UWP app
`Clipchamp.Clipchamp_2.8.1.0_neutral__yxz26nhyzhsrt' (AUMId
`Clipchamp.Clipchamp_yxz26nhyzhsrt!App') supports 41 extensions but
has no verbs
```



解决方案¶

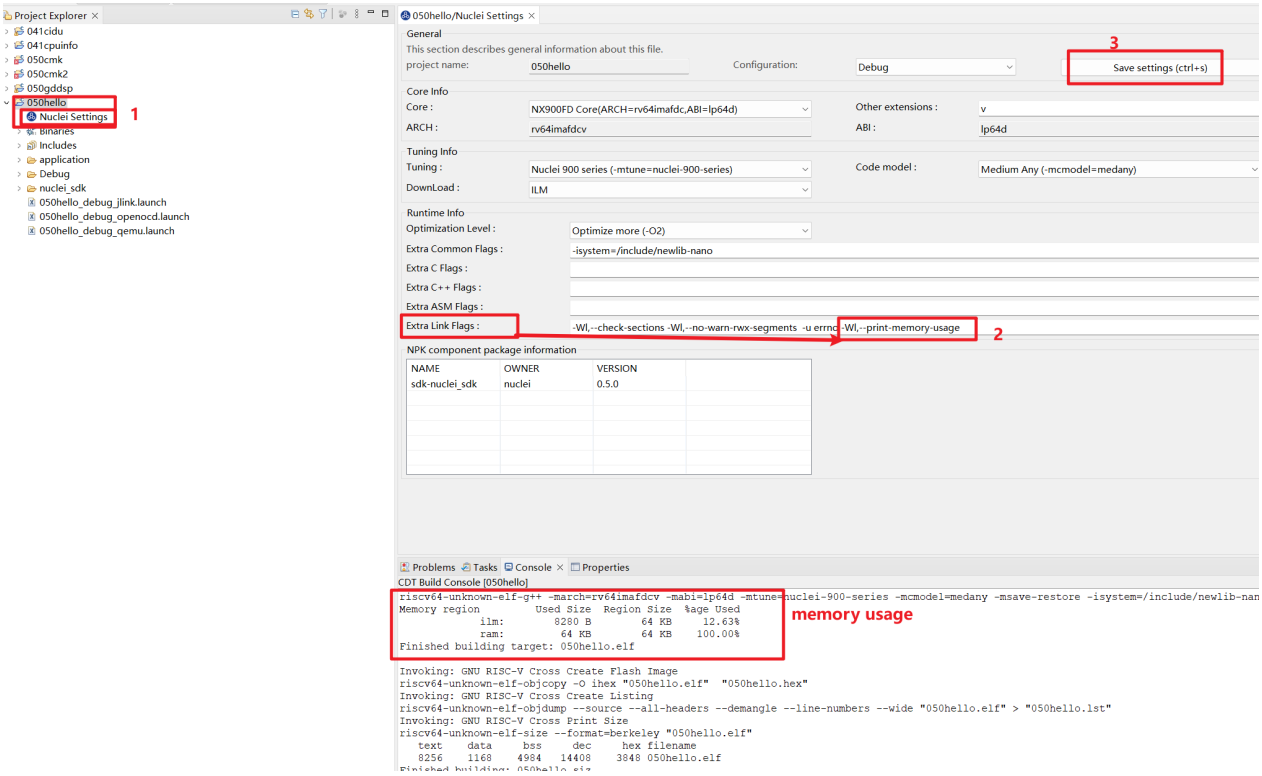
这个是Windows 10/11上存在的系统库匹配问题，不影响Qemu正常使用，可以忽略。

How to print memory usage in Nuclei Studio¶

问题说明¶

In order to print memory usage when compile an application, you can do it like this:

Click Nuclei Settings in selected project, and pass extra `-Wl,--print-memory-usage` in Extra Link Flags, and save settings, and then build this project, you will be able to see memory usage.




```
Building target: 050hello.elf
Invoking: GNU RISC-V Cross C++ Linker
... ...
Memory region         Used Size  Region Size  %age Used
             ilm:        8280 B        64 KB     12.63%
             ram:          64 KB        64 KB    100.00%
Finished building target: 050hello.elf
```

Why the ram usage here is 100% used?

For Nuclei SDK or NMSIS template linker script, the stack is placed at the bottom of ram memory, so the ram usage is 100%.

解决方案¶

Add extra link option `-Wl,--print-memory-usage` will solve this issue.

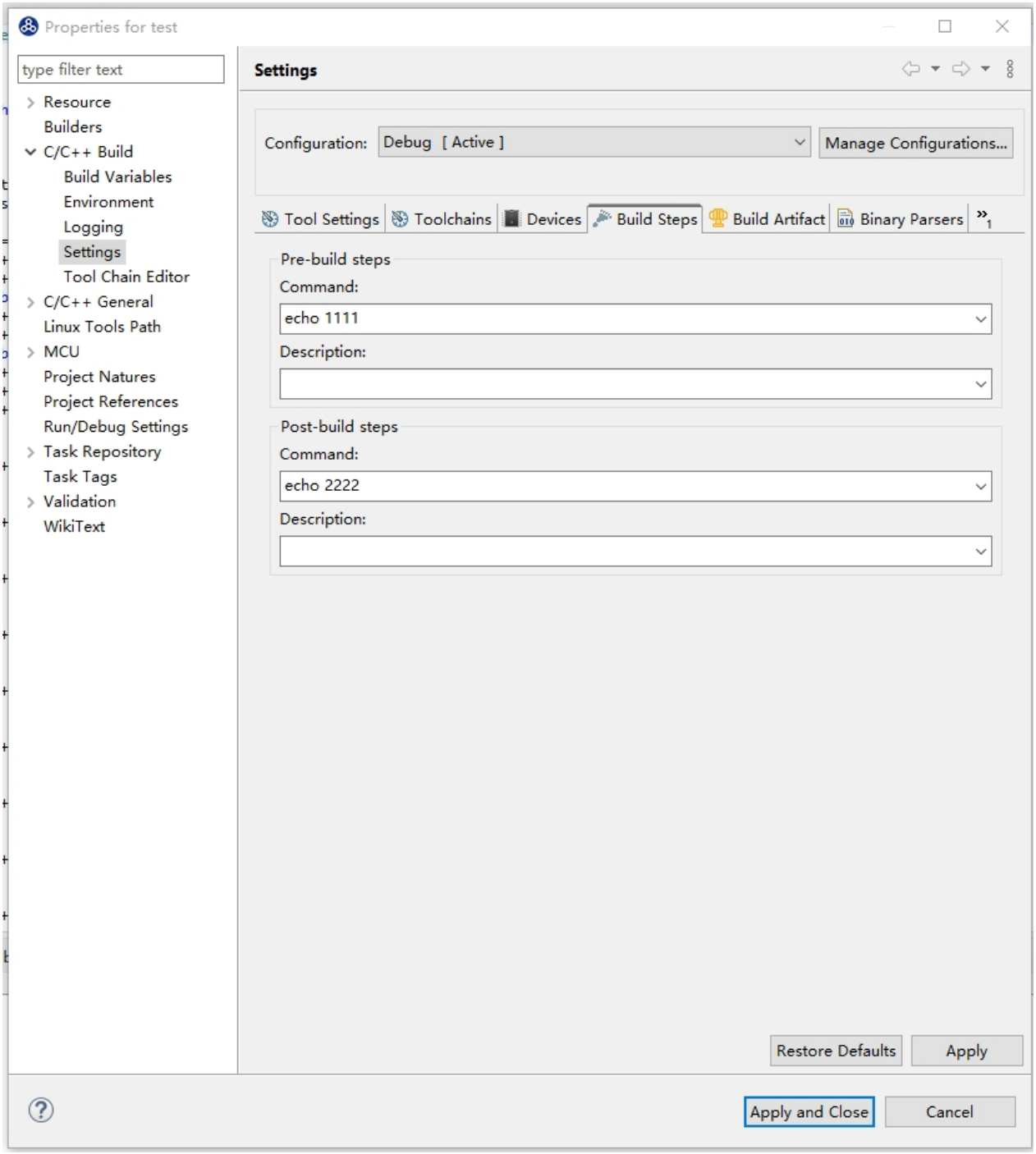
在编译工程时，使用了Pre-build Command/Post-build Command时报错¶

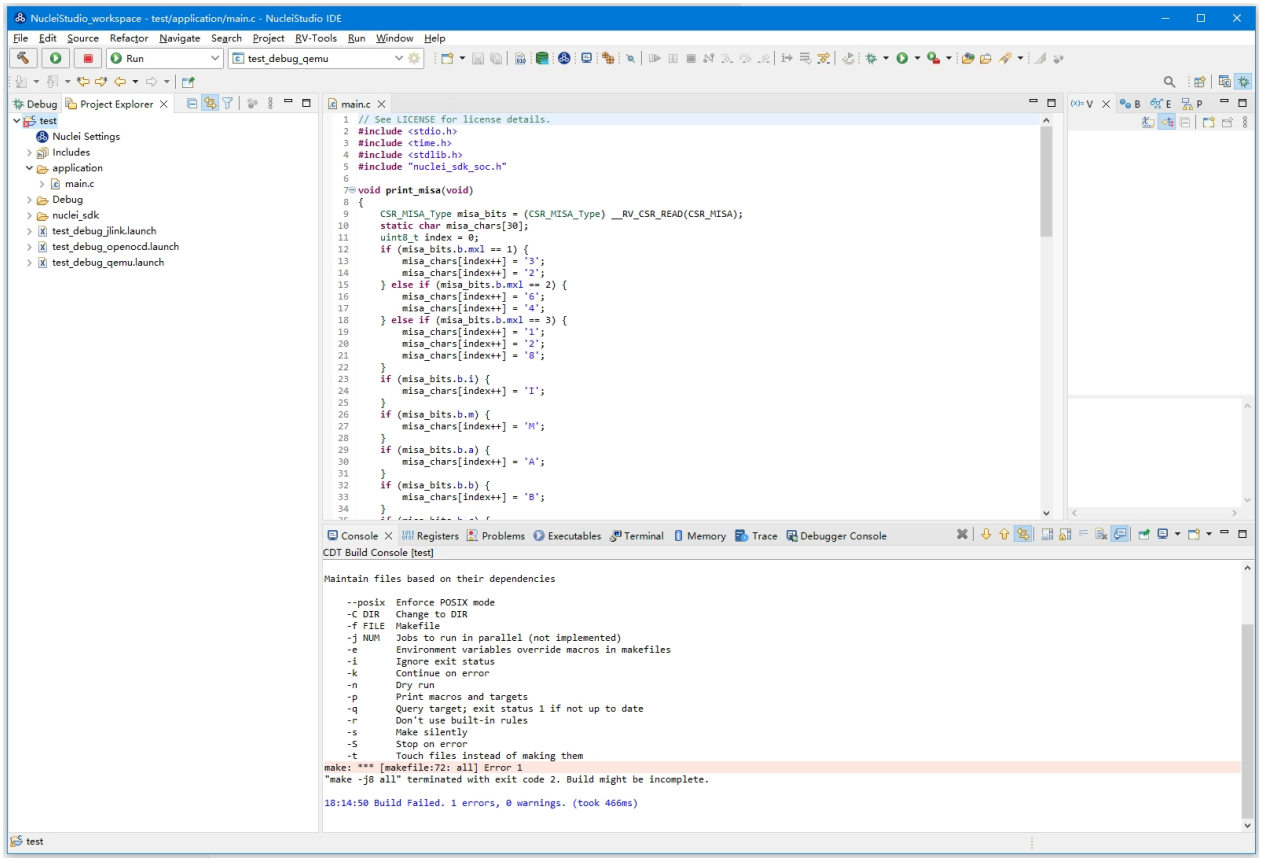
问题说明¶

在 Nuclei Studio 2023.10.17上传的2023.10 更正版本中修正，参见本文

参见 eclipse-embed-cdt/eclipse-plugins#597

在Nuclei Studio 2023.10版本中，如果在工程编译中需要使用到Pre-build Command/Post-build Command，因Nuclei Studio中集成的build-tools为v4.4.0版本，而上游CDT中在处理Pre-build Command/Post-build Command的方法，在build-tools v4.4.0无法正常使用，所以会出现报错问题。





解决方案¶

遇到这种情况时，可以下载 https://www.nucleisys.com/upload/files/toochain/build-tools/build-tools_202002.zip ，并替换工具链中的build-tools，问题可以得到解决。

升级到最新的 2024.06 和 2025.02军不存在这个问题。

```
NucleiStudio\toolchain\build-tools
```

升级npk.yml以支持Nuclei Studio 2023.10¶

在Nuclei Studio 2023.10中，一个重要变更，是支持GCC 13, 所以之前发布的NPK Package也需要做对应的变更，以更好的适用于Nuclei Studio 2023.10，其中有以下几个变更点。

> 需要注意新版的npk.yml 不再支持以前 2022.12版本的IDE

npk.yml中的工具链升级¶

在npk中，我们定义了buildconfig来自定义工程build时的各种参数，Nuclei Studio通过type标识使用的是那一种toolchain，如gcc、clang等， 通过 type->toolchain_name & cross_prefix 来标识使用的toolchain里面具体的那个发行版本。升级SDK以支持GCC 13，对比以下两个例子不难看出，只需要修改 toolchain_name: RISC-V GCC/Newlib 和 cross_prefix: riscv64-unknown-elf- ，就可以使SDK支持在创建工程时，可以选择GCC 13工具链。

以下内容是支持gcc 10 的buildconfig配置（为了方便举例，隐藏了部分参数，具体参数根据实际情况定义）。

```
## Build Configuration
buildconfig:
  - type: gcc
    description: Nuclei GNU Toolchain
    cross_prefix: riscv-nuclei-elf- # optional
    common_flags: # flags need to be combined together across all
packages
    ldflags:
    cflags:
    asmflags:
    cxxflags:
    common_defines:
    prebuild_steps: # could be override by app/bsp type
      command:
      description:
    postbuild_steps: # could be override by app/bsp type
      command:
      description:
```

下以内容，是支持GCC 13和Clang的buildconfig配置（为了方便举例，隐藏了部分参数，具体参数根据实际情况定义）。

```
## Build Configuration
buildconfig:
  - type: gcc
    description: Nuclei GNU Toolchain
    # 升级到GCC13时，这里进行如下两行的改变
    # 且针对所有npk.yml的文件只要包含buildconfig的都需要进行修改，不仅仅限于ssp/bsp类型，还包括bsp/app/mwp/osp/sdk类型
    toolchain_name: RISC-V GCC/Newlib
    cross_prefix: riscv64-unknown-elf- # optional
    common_flags: # flags need to be combined together across all packages
    ldflags:
    cflags:
    asmflags:
    cxxflags:
    common_defines:
    prebuild_steps: # could be override by app/bsp type
      command:
      description:
    postbuild_steps: # could be override by app/bsp type
      command:
      description:
  - type: clang
    description: Nuclei LLVM Toolchain
    toolchain_name: RISC-V Clang/Newlib
    cross_prefix: riscv64-unknown-elf- # optional
    common_flags: # flags need to be combined together across all packages
    ldflags:
    cflags:
    asmflags:
    cxxflags:
    common_defines:
    prebuild_steps: # could be override by app/bsp type
      command:
      description:
    postbuild_steps: # could be override by app/bsp type
      command:
      description:
```

除标准的IMAFDC之外的扩展(ARCHEXT)的升级¶

以下示例以Nuclei SDK 0.5.0的evalsoc的npk.yml升级举例， 仅考虑GCC的支持，如果需要考虑CLANG的支持，请参见 SDK中evalsoc的npk.yml的详细变更

在GCC 13中，对RISC-V 指令扩展使用有了很大的变更，具体内容可以查看Nuclei Studio用户手册2.1.4章内容和Nuclei SDK中ARCH_EXT说明。

- Nuclei Studio用户手册
- ARCH_EXT说明

升级npk.yml时，如果SDK中使用到了RISC-V 除了标准的IMAFDC之外指令扩展，例如B/P/K/V，也需要升级对应的配置。

在NPK中，RISC-V 指令扩展以是`-march=xxx`的方式传递给Nuclei Studio，Nuclei Studio接收到相关配置，就会存储并应用到编译的过程中。 以Nuclei SDK中的npk.yml为例，通过下面这段配置我们就可以得到`-march=`的值，不难看出与RISC-V指令扩展相关的是NPK中的变量nuclei_archext。

```
##  （为了方便举例，隐藏了部分参数，具体参数根据实际情况定义）
## Build Configuration
buildconfig:
  - type: gcc
    description: Nuclei RISC-V GNU Toolchain #must
    cross_prefix: riscv-nuclei-elf- # optional
    common_flags: # flags need to be combined together across all
packages
      # 这里 -march 传递的值 就是 nuclei_core.arch 和 nuclei_archext 两
个变量拼接而来
      # 例如 nuclei_core.arch设置为rv32imafdc, nuclei_archext设置为
_zba_zbb_zbc_zbs_xxldspn1x,
      # 那么 传递的就是 -march=rv32imafdc_zba_zbb_zbc_zbs_xxldspn1x
      # 如果你的 march是已知和确定的，这里直接就可以给定 -march/-mabi的选
项，无需通过 configuration字段来进行传递
      - flags: -march=${nuclei_core.arch}$(join($
{nuclei_archext},'')) -mabi=${nuclei_core.abi}
    ldflags:
    cflags:
    asmflags:
    cxxflags:
    common_defines:
    prebuild_steps: # could be override by app/bsp type
      command:
      description:
    postbuild_steps: # could be override by app/bsp type
      command:
      description:
```

在旧版的SDK中，nuclei_archext定义的是一个multicheckbox，用户可以自己选择，而在新版的SDK中nuclei_archext定义的是一个text输入框， 这样用户可以更灵活的使用RISC-V 指令扩展，如果在某些工程或场景下，想要预设一些RISC-V 指令扩展，建议给一个默认值就可以了，可以参考下代的示例代码。

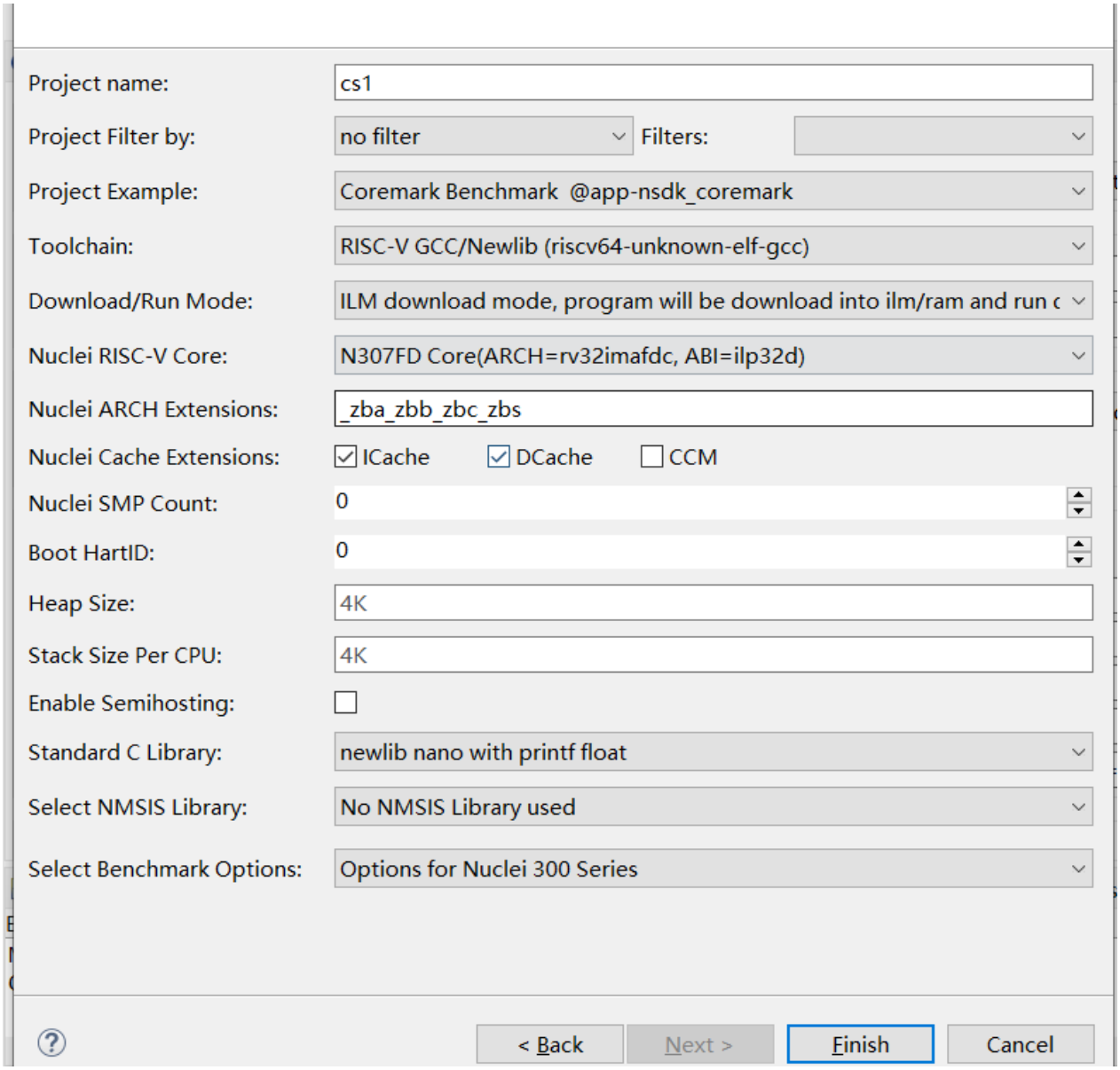
- 用于支持Nuclei RISC-V Toolchain 2022.12的写法

```
## 旧版的SDK中，nuclei_archext定义的是一个multicheckbox
##（为了方便举例，隐藏了部分参数，具体参数根据实际情况定义）
nuclei_archext:
  default_value: []
  type: multicheckbox
  global: true
  description: Nuclei ARCH Extensions
  choices:
    - name: b
      description: Bitmanip Extension
    - name: p
      description: Packed SIMD Extension
    - name: v
      description: Vector Extension
```

- 用于支持Nuclei RISC-V Toolchain 2023.10的写法

```
  ## 新版的SDK中nuclei_archext定义的是一个text输入框
  ## Package Configurations
  configuration:
  nuclei_archext:
      default_value: "_zba_zbb_zbc_zbs"
      type: text
      global: true
      # hints and tips are introduced in Nuclei Studio 2023.10
      # used to show tool tips and input hints
      tips: "Possible other ISA extensions, seperated by
underscores, like '_zba_zbb_zbc_zbs_xxldspn1x'"
      hints: "_zba_zbb_zbc_zbs_xxldspn1x"
      description: Nuclei ARCH Extensions
```

最终显示创建项目的时候显示效果如下



libncrt的升级¶

libncrt较之前也有了些许变化，在NPK中使用libncrt之前，新旧版SDK中都是一样的在conifguration中定义了一个变量`stdclib`， 它的值是一个下拉框，可以选择不同的值。不同点是在得到`stdclib`后，在`common_flags`或者其它地方使用`stdclib`时略有不同。

关于`stdclib`的一些说明，可以参见 这里

```
## 定义stdclib变量
##（为了方便举例，隐藏了部分参数，具体参数根据实际情况定义）
## Package Configurations
configuration:
  stdclib:
    default_value: newlib_nano
    type: choice
```

```
    global: true
    description: Standard C Library
    choices:
      - name: newlib_full
        description: newlib with full feature
      - name: newlib_fast
        description: newlib nano with printf/scanf float
      - name: newlib_small
        description: newlib nano with printf float
      - name: newlib_nano
        description: newlib nano without printf/scanf float
      - name: libncrt_fast
        description: nuclei c runtime library, optimized for speed
      - name: libncrt_balanced
        description: nuclei c runtime library, balanced, full
feature
      - name: libncrt_small
        description: nuclei c runtime library, optimized for size,
full feature
      - name: libncrt_nano
        description: nuclei c runtime library, optimized for size,
no float support
      - name: libncrt_pico
        description: nuclei c runtime library, optimized for size,
no long/long long support
      - name: nostd
        description: no std c library will be used, and don't
search the standard system directories for header files
      - name: nospec
        description: no std c library will be used, not pass any --
specs options
```

在新版的SDK中，如果使用`--specs=libncrt_xxx.specs` 或者链接库里面包含 `-lncrt_xxx`（表示采用libncrt c库）， 则需变更为 `-lncrt_xxx -lfileops_uart -lheapops_basic`，这也是旧SDK变更为支持GCC 13的新SDK的原则。

下面配置为在旧版SDK中的npk变量stdclib,当变量stdclib以libncrt开头时，会直接定义一个`--specs=${stdclib}.specs`， 按照上面我们说的原则，这里应该变成设置 `-l$(subst(${stdclib},lib,)) -lfileops_uart -lheapops_basic`，所以在新版SDK中的写法就变成了下面的配置方式。

```
## 在旧版SDK中使用stdclib变量
##（为了方便举例，隐藏了部分参数，具体参数根据实际情况定义）
## Build Configuration
buildconfig:
```

```
  - type: gcc
    description: Nuclei GNU Toolchain
    cross_prefix: riscv-nuclei-elf- # optional
    common_flags: # flags need to be combined together across all
packages
      - flags: --specs=${stdclib}.specs
        condition: $( startswith(${stdclib}, "libncrt") )
    ldflags:
    cflags:
    asmflags:
    cxxflags:
    common_defines:
    prebuild_steps: # could be override by app/bsp type
      command:
      description:
    postbuild_steps: # could be override by app/bsp type
      command:
      description:
```

转变为

```
## 在新版SDK中使用stdclib变量
## (为了方便举例，隐藏了部分参数，具体参数根据实际情况定义)
## Build Configuration
buildconfig:
  - type: gcc
    description: Nuclei GNU Toolchain
    toolchain_name: RISC-V GCC/Newlib
    cross_prefix: riscv64-unknown-elf- # optional
    common_flags: # flags need to be combined together across all
packages
      - flags: --specs=${stdclib}.specs
        condition: $( startswith(${stdclib}, "libncrt") )
    ldflags:
      - flags: -l$(subst(${stdclib},lib,)) -lheapops_basic -
lfileops_uart
        condition: $( startswith(${stdclib}, "libncrt") )
    cflags:
    asmflags:
    cxxflags:
    common_defines:
    prebuild_steps: # could be override by app/bsp type
      command:
      description:
    postbuild_steps: # could be override by app/bsp type
```

```
      command:
      description:
```

Link Warning的消除¶

在Nuclei Studio 2023.10中集成的GCC 13,在使用过程中会有warning，链接选项增加一个`-Wl,--no-warn-rwx-segments`可以隐藏warning。

具体可以参考以下配置（为了方便举例，隐藏了部分参数，具体参数根据实际情况定义）

```
## Build Configuration
buildconfig:
  - type: gcc
    description: Nuclei GNU Toolchain
    toolchain_name: RISC-V GCC/Newlib
    cross_prefix: riscv64-unknown-elf- # optional
    common_flags: # flags need to be combined together across all
packages
    ldflags:
       # 用于消除gcc13链接阶段的warning
       - flags: -Wl,--no-warn-rwx-segments
    cflags:
    asmflags:
    cxxflags:
    common_defines:
    prebuild_steps: # could be override by app/bsp type
      command:
      description:
    postbuild_steps: # could be override by app/bsp type
      command:
      description:
```

关于Nuclei SDK 0.5.0 npk.yml 详细变更¶

关于支持Nuclei Studio + Nuclei RISC-V Toolchain 2023.10的npk.yml变更，可以参考nuclei-sdk 0.5.0的变更。

- gd32vf103的变化 `git diff 0.4.1..0.5.0 SoC/gd32vf103/***npk.yml`
- evalsoc的变化: `git diff 0.4.1..0.5.0 SoC/evalsoc/***npk.yml`
- NMSIS的变化: `git diff 0.4.1..0.5.0 NMSIS/***npk.yml`

- application的变化: `git diff 0.4.1..0.5.0 application/***npk.yml`
- RTOS的变化: `git diff 0.4.1..0.5.0 OS/***npk.yml`

执行查看代码变更命令方法如下

```
git clone https://github.com/Nuclei-Software/nuclei-sdk/
cd nuclei-sdk
git fetch --all
git diff 0.4.1..0.5.0 SoC/gd32vf103/***npk.yml
git diff 0.4.1..0.5.0 SoC/evalsoc/***npk.yml
git diff 0.4.1..0.5.0 NMSIS/***npk.yml
git diff 0.4.1..0.5.0 application/***npk.yml
git diff 0.4.1..0.5.0 OS/***npk.yml
```

GCC13 auto generated RVV instructions when RVV enabled¶

问题说明¶

If you are using `Nuclei SDK 0.5.0` with Nuclei RISC-V Toolchain 2023.10, and when compile some examples with RVV enabled, it may generate rvv instructions which called auto-vectorzation.

Take `application/baremetal/benchmark/dhrystone` for example:

```
cd application/baremetal/benchmark/dhrystone
# enable extra vector extension, which means the -march=rv64imafdcv
make CORE=nx900fd ARCH_EXT=v clean
make CORE=nx900fd ARCH_EXT=v dasm
```

Then if you check the `dhrystone.dasm`, you will be able to see rvv instructions:

解决方案¶

This auto generated instructions may affect your hardware performance, so if you want to disable it, you don't need to pass rvv extension when compile application.

```
$ cat dhrystone.dasm |grep vs
    800003e2:   cc3ff057                vsetivli        zero,
31,e8,m8,ta,ma
    800003f8:   02038427                vse8.v  v8,(t2)
    8000040c:   020b8027                vse8.v  v0,(s7)
    800004a2:   cc3ff057                vsetivli        zero,
31,e8,m8,ta,ma
    800004b2:   02098827                vse8.v  v16,(s3)
    80000524:   cc3ff057                vsetivli        zero,
31,e8,m8,ta,ma
    80000530:   02098c27                vse8.v  v24,(s3)
    80000df2:   cdb3f057                vsetivli        zero,
7,e64,m8,ta,ma
    80000dfa:   0204f427                vse64.v v8,(s1)
    80000e20:   cdb3f057                vsetivli        zero,
7,e64,m8,ta,ma
    80000e28:   02047027                vse64.v v0,(s0)
```

You can check https://gcc.gnu.org/bugzilla/show_bug.cgi?id=112537 for more details.

In gcc 14.x, if you want to disable the RISC-V RVV automatic vectorization, you can use the options `-fno-tree-loop-vectorize -fno-tree-slp-vectorize`.

In gcc 13.x, you need to pass `--param=riscv-autovec-preference=none`

更新 Nuclei Studio 2023.10 到最新修正版本¶

2023.11.06上传的Nuclei Studio 2023.10版本存在一些问题，我们进行了修正，并于2023.11.17 13:30替换线上2023.10版本。

问题描述¶

2023年11月06日发布的Nuclei Studio 2023.10版本中存在一些问题,影响用户使用:

- build tools的busybox存在问题导致make 带 pre- post- steps时编译出问题
- Nuclei Settings中corner cases在特定场景下会出错
- Nuclei Settings的打开方式影响工程中其他文件的打开方式
- 在QEMU中使用V扩展时，没有传入RVV length
- 修复打开一个全新的workspace，创建新的工程的时候，能够创建同名项目的问题，重开workspace即可解决这个问题

我们重新做了一些变更，以修复以上问题：

- 修改并发布Nuclei Studio Plugins 2.1.0， 上传到插件更新网站
- 修改并发布Windows build-tools 1.2，替换了线上的Windows Build Tools 2023.10
- 发布了新的Nuclei Studio 2023.10，替换了线上的Nuclei Studio 2023.10

升级Nuclei Studio 2023.10 到最新版本的方法¶

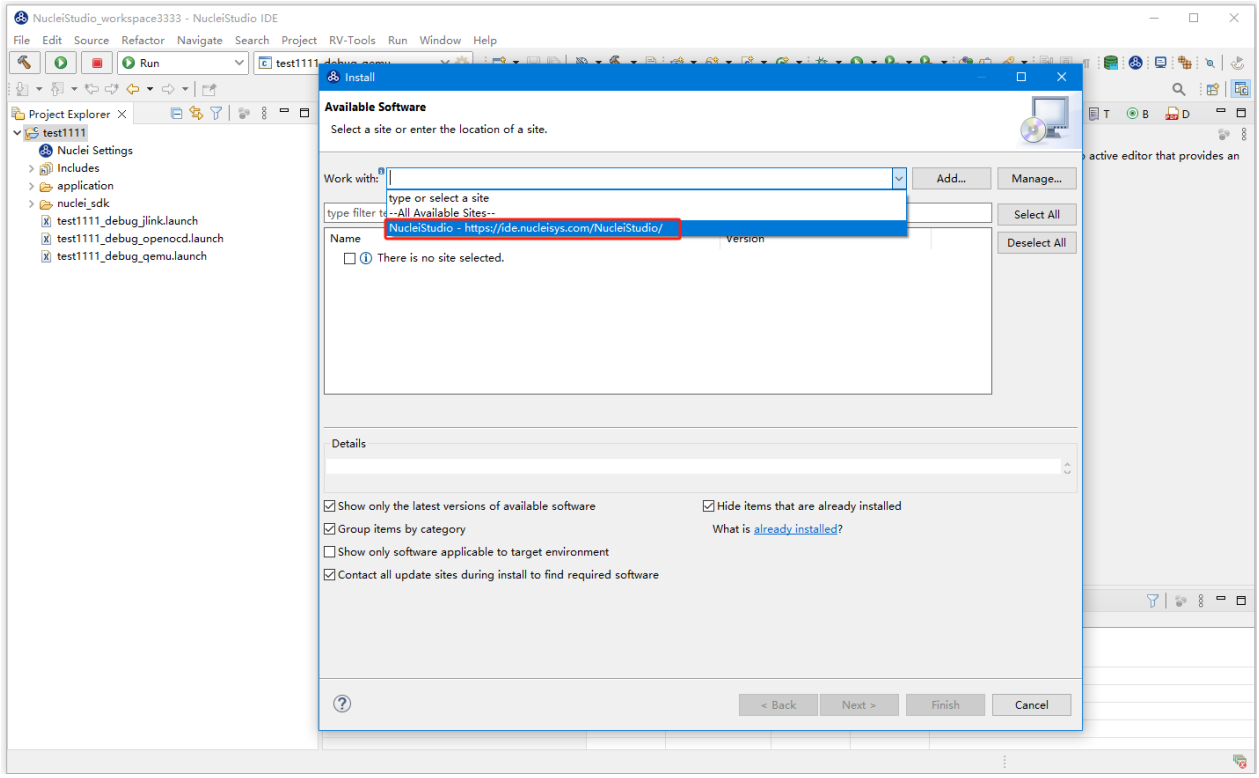
如果您的Nuclei Studio 2023.10，是在2023年11月18日之前下载，版本中存在的上述问题可能会引响您的使用体验， 您可以选择手动进行升级，也可以选择重官网上下载我们最新发布的版本。

对2023年11月18日之前下载了Nuclei Studio 2023.10进行升级¶

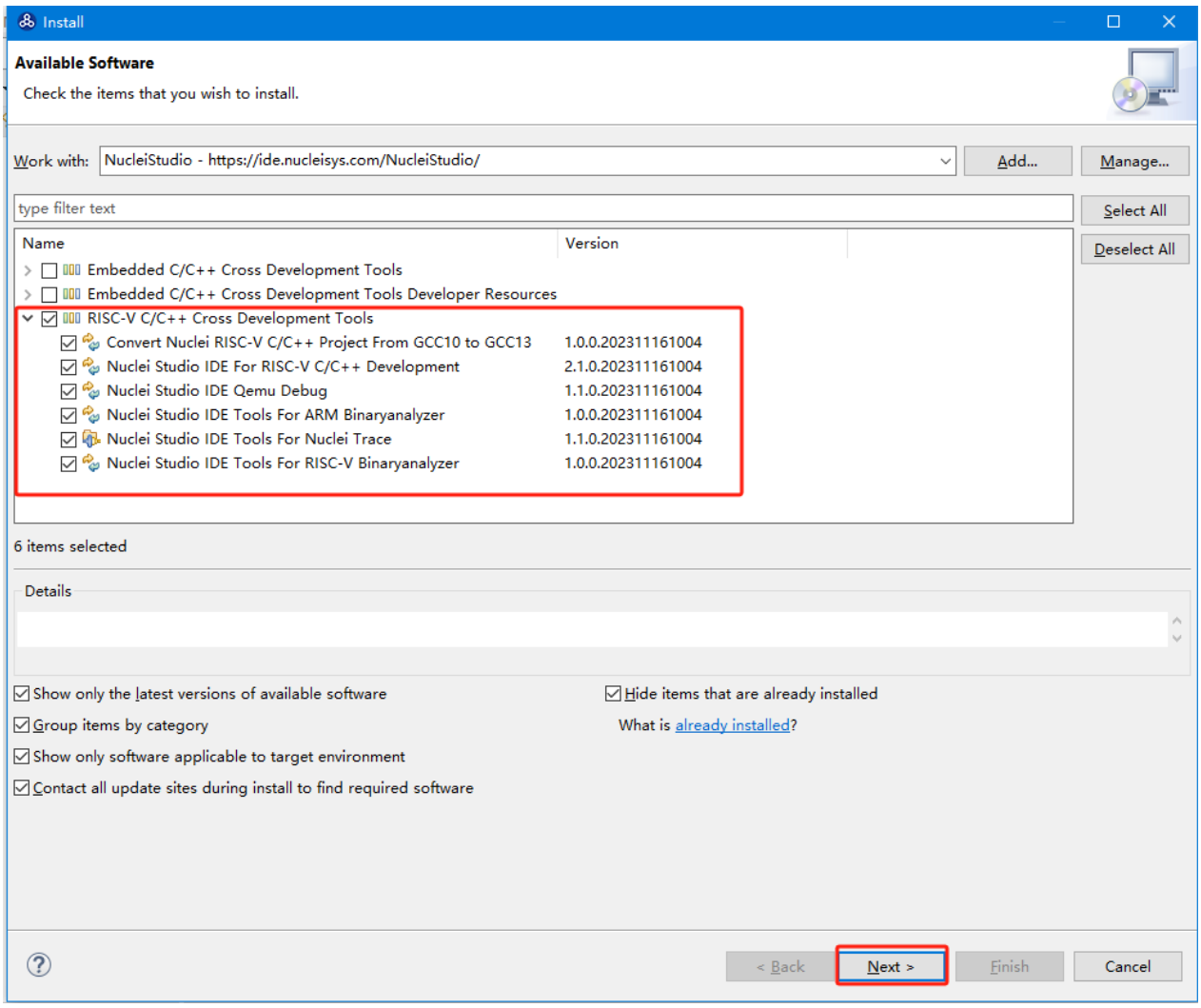
如果您是在2023年11月18日之前下载了Nuclei Studio 2023.10，可以通过以下方式更新您的Nuclei Studio 2023.10 到最新版本

1. 升级Nuclei Studio Plugins

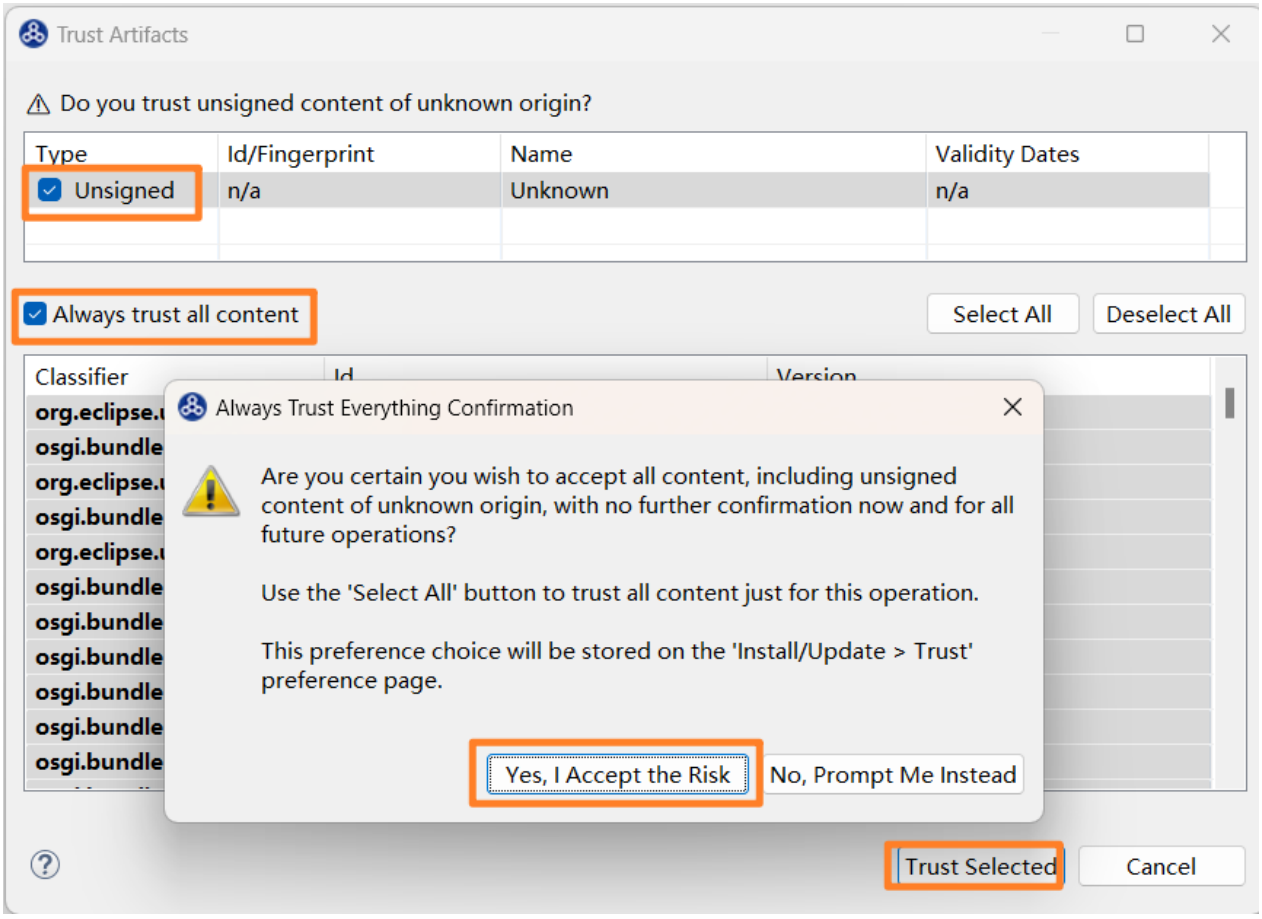
在Nuclei Studio菜单中找到Help->Install New Software, 然后在Install工具的`Work with`选中`NucleiStudio - https://ide.nucleisys.com/NucleiStudio/`,下面会列出所有待更新的插件。



在弹出的插件列表中选中需要升级的插件，我们选中`RISC-V C/C++ Cross Development Tools`, 然后Next。



在升级过程中，Nuclei Studio会询问Trust Artifacts时，操作如下图，选择Trust Selected, 然后升级完成，Nuclei Studio会重启。至此Nuclei Studio Plugins升级完成。



2. 升级build-tools

Linux版本不需要执行此步骤，只需要确保系统中装了`make`工具就行。

下载`build-tools-1.2`，并替换Nuclei Studio 2023.10中的`NucleiStudio\toolchain\build-tools`中内容。

关于这部分，可以查阅编译工程时，使用了Pre-build Command/Post-build Command时报错中的详细说明。

- build-tools-1.2下载

经此两步，完成了对Nuclei Studio 2023.10的升级。

从官网下载最新的版本¶

如果不想做手动升级工作，可以直接从我们的网站上下载最新的Nuclei Studio 2023.10。

- Windows版下载
- Linux版下载

参考资料¶

- Nuclei Studio FAQs
- Nuclei Studio/Tools 不断更新的补充文档
- Nuclei Studio Issues

OpenOCD在操作容量大于16M-Byte的nor-flash时的问题¶

问题说明¶

操作0 ~ 16M地址区间spi控制器需要发送三个字节的地址信息，称为3byte地址模式；操作16M ~ 2G地址区间spi控制器则需要发送四个字节的地址信息，称为4byte地址模式；

nuspi控制器的普通spi和xip默认都是3byte地址模式

解决方案¶

我们在OpenOCD里开发了两组spi驱动分别是nuspi和custom，都可以支持3byte模式和4byte模式，其中nuspi可通过判断操作地址，自动切换模式

在OpenOCD里有很多种方式可以read/verify flash内的数据，可以归结为两大类，一类是直接通过xip的方式读取flash数据，另一类则是通过调用驱动使用普通spi的方式读取flash数据。

因此，直接通过xip的方式读取flash数据时，就会有只能读到前面16M地址范围的限制，这样的命令有

- `flash verify_image filename [offset] [type]`
- `dump_image filename address size`
- gdb的x命令
- 等等 直接读取memory的命令

当然OpenOCD里面也存在一些读取flash的命令，会直接调用cfg文件注册的spi驱动，这样的命令有

- `flash read_bank num filename [offset [length]]`
- `flash verify_bank num filename [offset]`

通过修改.cproject文件，升级工程工具链到GCC 13¶

问题描述¶

> Nuclei Studio 2023.10的IDE进行了一次大版本的升级，其中自带的工具链从gcc10升级到了gcc13，并且工具链的前缀也发生了变化。 参见 https://github.com/Nuclei-Software/nuclei-studio/releases/tag/2023.10

虽然我们在2023.10的IDE中提供了右键选中工程一键升级的工具（参见IDE的手册第8章节），但是这个只能一个工程一个工程的转换，对于有大量工程需要批量转换的项目而言不太友好，因此我们这里列出来如果写脚本进行工程的转换升级，则可以参考如下的思路进行转换。

以下变更仅针对Nuclei Studio 2023.10之前版本创建的gcc10的工程，进行升级变更，如果需要批量变更，编写脚本的时候应先检查工程是否是riscv gcc10的工程。

修改toolchain相关配置¶

在Nuclei Studio 2023.10之前的版本中使用的gcc是做了许多个性化的变更，需要Nuclei Studio 2023.10版中使用的gcc,继承了官方版本的特性和一些命名方式，在工程中的`.cproject`文件中，主要是要修改以下几个值。其中`ilg.gnumcueclipse.managedbuild.cross.riscv.option.toolchain.name`的值是`RISC-V Nuclei GCC`、`ilg.gnumcueclipse.managedbuild.cross.riscv.option.toolchain.id`的值是`3901352267`、`ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.prefix`的值是`riscv-nuclei-elf-`,则说明工程在创建时所使用的是GCC 10。如果需要使工程支持GCC 13,需要进行如下变更：

- toolchain.name的值 从RISC-V Nuclei GCC变更为RISC-V GCC/Newlib
- toolchain.id的值 从3901352267变更为2262347901
- command.prefix的值 从riscv-nuclei-elf-变更为riscv64-unknown-elf-

变更前`.cproject`文件的内容

```
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.toolchain.name.
129748485"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.toolchain.name"
value="RISC-V Nuclei GCC" valueType="string"/>
```

```
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.toolchain.id.
1143901706"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.toolchain.id"
value="3901352267" valueType="string"/>
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.prefix.
1270840820"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.prefix"
value="riscv-nuclei-elf-" valueType="string"/>
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.c.
718590769"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.c"
value="gcc" valueType="string"/>
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.cpp.
243660928"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.cpp"
value="g++" valueType="string"/>
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.ar.
416250093"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.ar"
value="ar" valueType="string"/>
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.objcopy.
741068581"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.objcopy"
value="objcopy" valueType="string"/>
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.objdump.
1474975752"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.objdump"
value="objdump" valueType="string"/>
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.size.
2085350427"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.size"
value="size" valueType="string"/>
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.make.
1355881376"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.make"
value="make" valueType="string"/>
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.rm.
```

```
1330665916"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.rm"
value="rm" valueType="string"/>
```

变更后.cproject文件的内容

```
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.toolchain.name.
129748485"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.toolchain.name"
value="RISC-V GCC/Newlib" valueType="string"/>
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.toolchain.id.
1143901706"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.toolchain.id"
value="2262347901" valueType="string"/>
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.prefix.
1270840820"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.prefix"
value="riscv64-unknown-elf-" valueType="string"/>
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.c.
718590769"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.c"
value="gcc" valueType="string"/>
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.cpp.
243660928"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.cpp"
value="g++" valueType="string"/>
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.ar.
416250093"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.ar"
value="ar" valueType="string"/>
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.objcopy.
741068581"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.objcopy"
value="objcopy" valueType="string"/>
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.objdump.
1474975752"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.objdump"
value="objdump" valueType="string"/>
```

```
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.size.
2085350427"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.size"
value="size" valueType="string"/>
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.make.
1355881376"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.make"
value="make" valueType="string"/>
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.rm.
1330665916"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.command.rm"
value="rm" valueType="string"/>
```

修改RISC-V扩展相关配置¶

在Nuclei Studio 2023.10之前的版创建的工程中，RISC-V扩展是存放在四个单独的boolean类型的值中，而在Nuclei Studio 2023.10创建的工程中， 改为一个string类型的值中,所以我在要在工程的`.cproject`文件中找到四个旧的值，并按规则转换成为新的RISC-V扩展的字符串， 存放到`ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.isa.extensions`中，同时将旧的四个单独的boolean类型的值置空或者删除。

```
# 四个单独的boolean类型的值
ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.isa.extens
ion.rvb
ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.isa.extens
ion.rvk
ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.isa.extens
ion.dsp
ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.isa.extens
ion.vector

# 一个string类型的值
ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.isa.extens
ions
```

1. 首先，根据`ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.isa.base`确认工程对应的arch是rv32/rv64

2. 其次，根据`ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.isa.fp`确认是否带f/d
3. 最后，根据对应转换规则转换出正确的RISC-V扩展字符串

转换规则(特别说明，p的值需要接在RISC-V扩展字符串的最后)：

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

例如，现在有一个N307FD的工程，它的`arch=rv32imafdcbpv`(gcc10)，可以知道它是一个rv32,带fd并且使用了bpv扩展，那么根据转换规则，转换出来的RISC-V扩展字符串为`_zba_zbb_zbc_zbs_zve32f_xxldspn1x`。

变更前`.cproject`文件的内容

```
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.isa.base.
489743203"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.isa.base"
value="ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.arch.rv32i"
valueType="enumerated"/>
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.isa.fp.
1936924005"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.isa.fp"
value="ilg.gnumcueclipse.managedbuild.cross.riscv.option.isa.fp.double"
valueType="enumerated"/>
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.isa.extension.rvb.
168405526"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.isa.extens
value="true" valueType="boolean"/>
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.isa.extension.dsp.
565204765"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.isa.extens
value="true" valueType="boolean"/>
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.isa.extension.vect
1142078455"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.isa.extens
value="true" valueType="boolean"/>
```

变更后.cproject文件的内容

```
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.isa.extensions.
1832321358"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.isa.extens
value="_zba_zbb_zbc_zbs_zve32f_xxldspn1x" valueType="string"/>
```

修改libncrt C库相关配置¶

在Nuclei Studio 2023.10之前的版创建的工程中,使用libncrt C库时，会在工程中包含一个--specs=libncrt_xxx.specs或者链接库 里面包含 -lncrt_xxx，而在Nuclei Studio 2023.10创建的工程中，如果使用了libncrt C库，需要将--specs=libncrt_xxx.specs的方式变更为-lncrt_xxx， 然后额外需要链接的时候补上 -lncrt_small -lheapops_basic -lfileops_uart, 通用的target编译选项需要补上 -isystem=/include/libncrt

举例如下： * -lncrt_small -> -lncrt_small -lheapops_basic -lfileops_uart * --specs=libncrt_small.specs -> -lncrt_small -lheapops_basic -lfileops_uart

1. 在.cproject文件中确认否存存在--specs=libncrt_xxx.specs，如果存在，则表示这个是一个使用了libncrt的工程，则可以进行后续的步骤
2. 如果--specs=libncrt_xxx.specs存在，先将其删除
3. 如果-lm存在，则先将其删除
4. 查找ilg.gnumcueclipse.managedbuild.cross.riscv.option.c.linker.libs或者ilg.gnumcueclipse.managedbuild.cross.riscv.option.cpp.linker.libs中是否存m，如果存在则先将删除
5. 查找ilg.gnumcueclipse.managedbuild.cross.riscv.option.c.linker.libs或者ilg.gnumcueclipse.managedbuild.cross.riscv.option.cpp.linker.libs中是否存ncrt_xxx
6. 根据上面的结果，在ilg.gnumcueclipse.managedbuild.cross.riscv.option.c.linker.libs或者ilg.gnumcueclipse.managedbuild.cross.riscv.option.cpp.linker.libs中补充对应的值
    - --specs=libncrt_xxx.specs存在，添加ncrt_xxx；或者ncrt_xxx存在。需要额外添加heapops_basic和fileops_uart
7. 在ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.other中补上 -isystem=/include/libncrt

```
# --specs=libncrt_xxx1.specs可能存在于以下string类型的值
ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.other
ilg.gnumcueclipse.managedbuild.cross.riscv.option.optimization.othe
r
ilg.gnumcueclipse.managedbuild.cross.riscv.option.warnings.other
ilg.gnumcueclipse.managedbuild.cross.riscv.option.debugging.other
ilg.gnumcueclipse.managedbuild.cross.riscv.option.assembler.otherwa
rnings
ilg.gnumcueclipse.managedbuild.cross.riscv.option.assembler.other
ilg.gnumcueclipse.managedbuild.cross.riscv.option.c.compiler.othero
ptimizations
ilg.gnumcueclipse.managedbuild.cross.riscv.option.c.compiler.otherw
arnings
ilg.gnumcueclipse.managedbuild.cross.riscv.option.c.compiler.other
ilg.gnumcueclipse.managedbuild.cross.riscv.option.c.linker.other
ilg.gnumcueclipse.managedbuild.cross.riscv.option.cpp.compiler.othe
roptimizations
ilg.gnumcueclipse.managedbuild.cross.riscv.option.cpp.compiler.othe
rwarnings
ilg.gnumcueclipse.managedbuild.cross.riscv.option.cpp.compiler.othe
r
ilg.gnumcueclipse.managedbuild.cross.riscv.option.cpp.linker.other
ilg.gnumcueclipse.managedbuild.cross.riscv.option.createflash.other
ilg.gnumcueclipse.managedbuild.cross.riscv.option.createlisting.oth
er
ilg.gnumcueclipse.managedbuild.cross.riscv.option.printsize.other
```

举例，工程中用到了`--specs=libncrt_balanced.specs`

变更前`.cproject`文件的内容

```
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.other.
1735566114"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.other"
value=" " valueType="string"/>
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.optimization.other.
443378574"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.optimization.othe
value="--specs=libncrt_balanced.specs" valueType="string"/>
```

变更后`.cproject`文件的内容

```
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.other.
1735566114"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.other"
value="-isystem=/include/libncrt " valueType="string"/>
<option IS_BUILTIN_EMPTY="false" IS_VALUE_EMPTY="false"
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.cpp.linker.libs.
146128417"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.cpp.linker.libs"
valueType="libs">
    <listOptionValue builtIn="false" value="ncrt_balanced"/>
    <listOptionValue builtIn="false" value="fileops_uart"/>
    <listOptionValue builtIn="false" value="heapops_basic"/>
</option>
```

增加link warning消除的配置¶

在GCC 13使用过程中会产生很多的warning信息，可以在链接选项中额外增加`-Wl,--no-warn-rwx-segments`参数，用以关闭这些warning信息。

具体参见 https://sourceware.org/binutils/docs/ld/Options.html#index-_002d_002dwarn_002drwx_002dsegments

变更前`.cproject`文件的内容

```
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.cpp.linker.other.
1000044097"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.cpp.linker.other"
value="" valueType="string"/>
```

变更后`.cproject`文件的内容

```
<option
id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.cpp.linker.other.
1000044097"
superClass="ilg.gnumcueclipse.managedbuild.cross.riscv.option.cpp.linker.other"
value="-Wl,--no-warn-rwx-segments" valueType="string"/>
```

完成以上变更后，reload一下工程，工程就可以在Nuclei Studio 2023.10下正常编译、调试、运行了。

说明：

本文档中，所有引用的例子中关于.cproject文件，出现的类似 id="ilg.gnumcueclipse.managedbuild.cross.riscv.option.target.other.1735566114"中， 1735566114是一个Nuclei Studio生成的hash值，不同时间不同工程各不相同，且其不影响配置，如果能保持与原值相同的情况下，尽量保持相同。

在Nuclei Studio下用命令行编译工程¶

问题说明¶

很多客户咨询怎么在Nuclei Studio上使用IDE的无头Headless模式来构建和编译工程。

解决方案¶

> 所有以 `NucleiStudio.exe` 开头的命令行执行时，会有一个弹框显示执行日志，如果需屏蔽弹框，可以将命令改为 `eclipsec.exe`
>
> 以下文档是在2024.06版本的IDE中实测，作为补充说明。

因NucleiStudio 2024.06版运行在java 21的环境上，实际应用中很多用户的本地没有java 21环境，故在运行命令时发现在执行该命令时，因找不到对应的jre而报错。为解决上述问题，可以在本地机器上安装java 21的环境（如何安装用户可以自行搜索相关教程），也可以在命令行中通过 `-vm` 参数指定NucleiStudio 2024.06中自带的jre的路径。

```
NucleiStudio.exe -vm "<user_nucleistudio_path>/plugins/
org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v202
40426-1530/jre/bin" --launcher.suppressErrors -nosplash -
application org.eclipse.cdt.managedbuilder.core.headlessbuild -
data C:\NucleiStudio_workspace -cleanBuild test/Debug -Debug
```

提供一组批量导入工程并批量编译工程的命令

创建workspace并批量导入工程

```
NucleiStudio.exe -vm "<user_nucleistudio_path>/plugins/
org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v202
40426-1530/jre/bin"  --launcher.suppressErrors -noSplash -
application org.eclipse.cdt.managedbuilder.core.headlessbuild -
data $CI_PROJECT_DIR -importAll $CI_PROJECT_DIR
```

编译这组导入的工程

```
NucleiStudio.exe -vm "<user_nucleistudio_path>/plugins/
org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v202
40426-1530/jre/bin" --launcher.suppressErrors -noSplash -
```

```
application org.eclipse.cdt.managedbuilder.core.headlessbuild -
data $CI_PROJECT_DIR -build ${TARGET_PHASE}_Project/Debug
```

> 以下文档是在2023.10版本的IDE中实测，其他版本可能需要做一些调整适配才可以正常工作。

Nuclei Studio是图形化（GUI）的代码编写工具，但是在某些特定的场景下，用户需要通过命令行来快速编译工程， 在Nuclei Studio中，只需要一行命令就可以实现。下载好Nuclei Studio后，在Nuclei Studio的workspace已经创建好了需要编译的工程`test`, **同时Nuclei Studio已退出运行**, 执行以下命令就可以完成工程的编译。

> **提醒**: 请确保 NucleiStudio的PATH已经设置到系统中，这样 `NucleiStudio.exe`/`NucleiStudio` 才可以被执行。
>
> 下面以Windows系统举例

```
NucleiStudio.exe --launcher.suppressErrors -nosplash -application
org.eclipse.cdt.managedbuilder.core.headlessbuild -data C:
\NucleiStudio_workspace -cleanBuild test/Debug -Debug
```

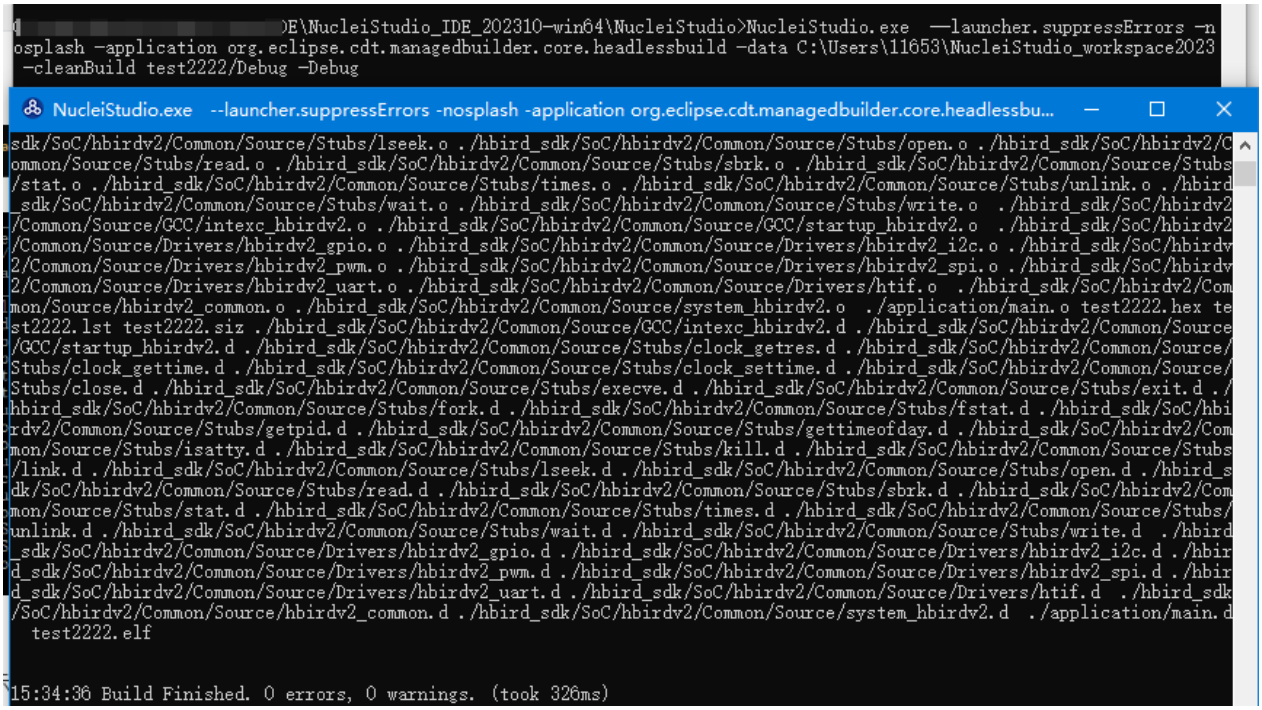
> `--launcher.suppressErrors` 用来屏蔽构建出错时，Eclipse会出错弹窗.

如果需要在2022.12版本的IDE上进行使用，则需要先设置好toolchain目录下`gcc/bin`和`build-tools/bin`的路径到系统PATH中，然后将`NucleiStudio.exe`换成`eclipsec.exe`

针对2022.12版本，命令举例如下:

```
# 这里请修改成自己的IDE路径
set NSIDE=D:\NucleiStudio_IDE_202212-win64\NucleiStudio
# 必须设置好系统PATH
set PATH=%NSIDE%\toolchain\gcc\bin;%NSIDE%\toolchain\build-
tools\bin;%PATH%
# 注意NucleiStudio.exe换成了eclipsec.exe
%NSIDE%\eclipsec.exe --launcher.suppressErrors -nosplash -
application org.eclipse.cdt.managedbuilder.core.headlessbuild -
data C:\NucleiStudio_workspace -cleanBuild test/Debug
```

> 这个2023.10版本的举例的命令 会 弹出一个额外的命令行窗口进行输出。



- `NucleiStudio.exe`：该参数是Nuclei Studio的启动应用，在Nuclei Studio的安装目录下。
- `--launcher.suppressErrors`：该参数是用于抑制Nuclei Studio启动时的错误信息。
- `-nosplash`：该参数用于关闭启动时的 Splash 屏幕。这意味着在启动 Eclipse 时不会显示一个短暂的加载屏幕。
- `-application`：该参数用于指定要运行的应用程序。在这里，`org.eclipse.cdt.managedbuilder.core.headlessbuild` 是指 Headless 构建应用程序。该应用程序用于执行构建操作，而不需要图形用户界面（GUI）。
- `-data`：该参数用于指定工作区路径。它告诉 Nuclei Studio 将数据存储在哪里，例如工作空间、项目和文件。
- `-build`：该参数用于指定需要编译的工程，`test/Debug`，表示的是编译test工程中的Debug配置； 一般Nuclei Studio创建的工程有Debug、Release两套配置，如果不指定配置，这个默认会编译出Debug、Release， 可以看到编译后工程目录下有Debug、Release两个目录。

```
├─.settings
├─application
├─Debug
│  ├─application
│  └─nuclei_sdk
├─nuclei_sdk
└─Release
```

```
        ├─application
        └─nuclei_sdk
```

- -cleanBuild：该参数与-build类似，只是在编译之前，会清空清理工作空间。建议使用-cleanBuild。
- -Debug：该参数用于指定编译过程是Debug模式，在编译时会输出详细的编译过程日志。如果不带此参数，命令将静默执行，没有任何输出。

以下为上面举例命令的输出内容，以供参考

```
17:00:17 **** Clean-only build of configuration Debug for project
test ****
make -j8 clean
rm -rf ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
chown.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
clock_getres.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
clock_gettime.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/clock_settime.o ./nuclei_sdk/SoC/evalsoc/Common/Source/
Stubs/newlib/close.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/environ.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/errno.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
execve.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
exit.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
fork.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
fstat.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
getpid.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
gettimeofday.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
isatty.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
kill.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
link.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
lseek.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
open.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
read.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
readlink.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
sbrk.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
stat.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
symlink.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
times.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
unlink.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
wait.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
write.o  ./nuclei_sdk/SoC/evalsoc/Common/Source/GCC/
intexc_evalsoc.o ./nuclei_sdk/SoC/evalsoc/Common/Source/GCC/
intexc_evalsoc_s.o ./nuclei_sdk/SoC/evalsoc/Common/Source/GCC/
startup_evalsoc.o  ./nuclei_sdk/SoC/evalsoc/Common/Source/Drivers/
evalsoc_uart.o  ./nuclei_sdk/SoC/evalsoc/Common/Source/
evalsoc_common.o ./nuclei_sdk/SoC/evalsoc/Common/Source/
```

```
system_evalsoc.o  ./application/main.o test.hex test.lst
test.siz ./nuclei_sdk/SoC/evalsoc/Common/Source/GCC/
intexc_evalsoc.d ./nuclei_sdk/SoC/evalsoc/Common/Source/GCC/
intexc_evalsoc_s.d ./nuclei_sdk/SoC/evalsoc/Common/Source/GCC/
startup_evalsoc.d ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/chown.d ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
clock_getres.d ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
clock_gettime.d ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/clock_settime.d ./nuclei_sdk/SoC/evalsoc/Common/Source/
Stubs/newlib/close.d ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/environ.d ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/errno.d ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
execve.d ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
exit.d ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
fork.d ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
fstat.d ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
getpid.d ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
gettimeofday.d ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
isatty.d ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
kill.d ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
link.d ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
lseek.d ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
open.d ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
read.d ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
readlink.d ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
sbrk.d ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
stat.d ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
symlink.d ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
times.d ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
unlink.d ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
wait.d ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
write.d  ./nuclei_sdk/SoC/evalsoc/Common/Source/Drivers/
evalsoc_uart.d  ./nuclei_sdk/SoC/evalsoc/Common/Source/
evalsoc_common.d ./nuclei_sdk/SoC/evalsoc/Common/Source/
system_evalsoc.d  ./application/main.d  test.elf


17:00:17 Build Finished. 0 errors, 0 warnings. (took 371ms)

17:00:18 **** Build of configuration Debug for project test ****
make -j8 all
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/chown.c
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/clock_getres.c
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/clock_gettime.c
```

```
Invoking: GNU RISC-V Cross C Compiler
Invoking: GNU RISC-V Cross C Compiler
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fp
ga_eval\Include" -I"C:\NucleiStudio_workspace\test\application" -
std=gnu11 -MMD -MP -MF"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/chown.d" -MT"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/chown.o" -c -o "nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/chown.o" "../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/chown.c"
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/clock_settime.c
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
clock_getres.d" -MT"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/clock_getres.o" -c -o "nuclei_sdk/SoC/evalsoc/Common/Source/
Stubs/newlib/clock_getres.o" "../nuclei_sdk/SoC/evalsoc/Common/
Source/Stubs/newlib/clock_getres.c"
Invoking: GNU RISC-V Cross C Compiler
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
```

```
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
clock_gettime.d" -MT"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/clock_gettime.o" -c -o "nuclei_sdk/SoC/evalsoc/Common/
Source/Stubs/newlib/clock_gettime.o" "../nuclei_sdk/SoC/evalsoc/
Common/Source/Stubs/newlib/clock_gettime.c"
Invoking: GNU RISC-V Cross C Compiler
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
clock_settime.d" -MT"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/clock_settime.o" -c -o "nuclei_sdk/SoC/evalsoc/Common/
Source/Stubs/newlib/clock_settime.o" "../nuclei_sdk/SoC/evalsoc/
Common/Source/Stubs/newlib/clock_settime.c"
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/close.c
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/environ.c
Invoking: GNU RISC-V Cross C Compiler
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/errno.c
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/execve.c
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
```

```
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/close.d" -
MT"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/close.o" -c -o
"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/close.o" "../
nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/close.c"
Invoking: GNU RISC-V Cross C Compiler
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/environ.d" -
MT"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/environ.o" -c
-o "nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/environ.o"
"../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/environ.c"
Invoking: GNU RISC-V Cross C Compiler
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/errno.d" -
MT"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/errno.o" -c -o
"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/errno.o" "../
nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/errno.c"
Invoking: GNU RISC-V Cross C Compiler
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
```

```
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/execve.d" -
MT"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/execve.o" -c -
o "nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/execve.o" "../
nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/execve.c"
Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/environ.c
Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/chown.c
Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/clock_getres.c
Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/clock_settime.c
Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/close.c
Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/clock_gettime.c
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/exit.c
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/fork.c
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/fstat.c
Invoking: GNU RISC-V Cross C Compiler
Invoking: GNU RISC-V Cross C Compiler
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/exit.d" -
MT"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/exit.o" -c -o
"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/exit.o" "../
nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/exit.c"
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/getpid.c
```

```
Invoking: GNU RISC-V Cross C Compiler
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/fork.d" -
MT"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/fork.o" -c -o
"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/fork.o" "../
nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/fork.c"
Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/execve.c
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/fstat.d" -
MT"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/fstat.o" -c -o
"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/fstat.o" "../
nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/fstat.c"
Invoking: GNU RISC-V Cross C Compiler

riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
```

```
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/getpid.d" -
MT"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/getpid.o" -c -
o "nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/getpid.o" "../
nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/getpid.c"
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/gettimeofday.c
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/isatty.c
Invoking: GNU RISC-V Cross C Compiler
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/kill.c
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
gettimeofday.d" -MT"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/gettimeofday.o" -c -o "nuclei_sdk/SoC/evalsoc/Common/Source/
Stubs/newlib/gettimeofday.o" "../nuclei_sdk/SoC/evalsoc/Common/
Source/Stubs/newlib/gettimeofday.c"
Invoking: GNU RISC-V Cross C Compiler
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/isatty.d" -
MT"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/isatty.o" -c -
o "nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/isatty.o" "../
nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/isatty.c"
```

```
Invoking: GNU RISC-V Cross C Compiler
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/kill.d" -
MT"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/kill.o" -c -o
"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/kill.o" "../
nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/kill.c"
Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/exit.c

Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/errno.c
Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/fork.c
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/link.c


Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/lseek.c
Invoking: GNU RISC-V Cross C Compiler
Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/gettimeofday.c
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/link.d" -
MT"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/link.o" -c -o
```

```
"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/link.o" "../
nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/link.c"
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/open.c
Invoking: GNU RISC-V Cross C Compiler
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/lseek.d" -
MT"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/lseek.o" -c -o
"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/lseek.o" "../
nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/lseek.c"

Invoking: GNU RISC-V Cross C Compiler
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/open.d" -
MT"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/open.o" -c -o
"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/open.o" "../
nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/open.c"
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/read.c
Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/kill.c
Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/getpid.c
Invoking: GNU RISC-V Cross C Compiler
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
```

```
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/read.d" -
MT"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/read.o" -c -o
"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/read.o" "../
nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/read.c"


Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/readlink.c
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/sbrk.c
Invoking: GNU RISC-V Cross C Compiler
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/readlink.d" -
MT"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/readlink.o" -c
-o "nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/readlink.o"
"../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/readlink.c"
Invoking: GNU RISC-V Cross C Compiler
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
```

```
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/sbrk.d" -
MT"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/sbrk.o" -c -o
"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/sbrk.o" "../
nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/sbrk.c"
Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/isatty.c

Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/fstat.c
Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/link.c
Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/lseek.c
Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/read.c



Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/stat.c

Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/symlink.c
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/times.c
Invoking: GNU RISC-V Cross C Compiler
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/unlink.c
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/stat.d" -
MT"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/stat.o" -c -o
"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/stat.o" "../
```

```
nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/stat.c"
Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/readlink.c
Invoking: GNU RISC-V Cross C Compiler
Invoking: GNU RISC-V Cross C Compiler
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/symlink.d" -
MT"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/symlink.o" -c
-o "nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/symlink.o"
"../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/symlink.c"
Invoking: GNU RISC-V Cross C Compiler
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/times.d" -
MT"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/times.o" -c -o
"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/times.o" "../
nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/times.c"
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
```

```
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/unlink.d" -
MT"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/unlink.o" -c -
o "nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/unlink.o" "../
nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/unlink.c"

Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/wait.c
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/write.c
Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/sbrk.c
Invoking: GNU RISC-V Cross C Compiler
Invoking: GNU RISC-V Cross C Compiler

riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/wait.d" -
MT"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/wait.o" -c -o
"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/wait.o" "../
nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/wait.c"
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/write.d" -
MT"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/write.o" -c -o
```

```
"nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/write.o" "../
nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/write.c"
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/GCC/
intexc_evalsoc.S
Invoking: GNU RISC-V Cross Assembler
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -x assembler-with-cpp -D__IDE_RV_CORE=n307fd -
DBOOT_HARTID=0 -DRUNMODE_IC_EN=0 -DRUNMODE_DC_EN=0 -
DRUNMODE_CCM_EN=0 -DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -
DDOWNLOAD_MODE_STRING=\"ILM\" -I"C:
\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -MMD -MP -
MF"nuclei_sdk/SoC/evalsoc/Common/Source/GCC/intexc_evalsoc.d" -
MT"nuclei_sdk/SoC/evalsoc/Common/Source/GCC/intexc_evalsoc.o" -c -o
"nuclei_sdk/SoC/evalsoc/Common/Source/GCC/intexc_evalsoc.o" "../
nuclei_sdk/SoC/evalsoc/Common/Source/GCC/intexc_evalsoc.S"
Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/unlink.c

Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/symlink.c
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/GCC/
intexc_evalsoc_s.S

Invoking: GNU RISC-V Cross Assembler
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/GCC/
startup_evalsoc.S
Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/GCC/
intexc_evalsoc.S
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -x assembler-with-cpp -D__IDE_RV_CORE=n307fd -
DBOOT_HARTID=0 -DRUNMODE_IC_EN=0 -DRUNMODE_DC_EN=0 -
DRUNMODE_CCM_EN=0 -DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -
DDOWNLOAD_MODE_STRING=\"ILM\" -I"C:
\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -MMD -MP -
MF"nuclei_sdk/SoC/evalsoc/Common/Source/GCC/intexc_evalsoc_s.d" -
```

```
MT"nuclei_sdk/SoC/evalsoc/Common/Source/GCC/intexc_evalsoc_s.o" -c
-o "nuclei_sdk/SoC/evalsoc/Common/Source/GCC/intexc_evalsoc_s.o"
"../nuclei_sdk/SoC/evalsoc/Common/Source/GCC/intexc_evalsoc_s.S"
Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/open.c
Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/wait.c
Invoking: GNU RISC-V Cross Assembler



riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -x assembler-with-cpp -D__IDE_RV_CORE=n307fd -
DBOOT_HARTID=0 -DRUNMODE_IC_EN=0 -DRUNMODE_DC_EN=0 -
DRUNMODE_CCM_EN=0 -DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -
DDOWNLOAD_MODE_STRING=\"ILM\" -I"C:
\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -MMD -MP -
MF"nuclei_sdk/SoC/evalsoc/Common/Source/GCC/startup_evalsoc.d" -
MT"nuclei_sdk/SoC/evalsoc/Common/Source/GCC/startup_evalsoc.o" -c -
o "nuclei_sdk/SoC/evalsoc/Common/Source/GCC/startup_evalsoc.o" "../
nuclei_sdk/SoC/evalsoc/Common/Source/GCC/startup_evalsoc.S"
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/Drivers/
evalsoc_uart.c
Invoking: GNU RISC-V Cross C Compiler
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/
evalsoc_common.c
Building file: ../nuclei_sdk/SoC/evalsoc/Common/Source/
system_evalsoc.c
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/Drivers/evalsoc_uart.d" -
```

```
MT"nuclei_sdk/SoC/evalsoc/Common/Source/Drivers/evalsoc_uart.o" -c
-o "nuclei_sdk/SoC/evalsoc/Common/Source/Drivers/evalsoc_uart.o"
"../nuclei_sdk/SoC/evalsoc/Common/Source/Drivers/evalsoc_uart.c"
Invoking: GNU RISC-V Cross C Compiler
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/evalsoc_common.d" -
MT"nuclei_sdk/SoC/evalsoc/Common/Source/evalsoc_common.o" -c -o
"nuclei_sdk/SoC/evalsoc/Common/Source/evalsoc_common.o" "../
nuclei_sdk/SoC/evalsoc/Common/Source/evalsoc_common.c"
Invoking: GNU RISC-V Cross C Compiler
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"nuclei_sdk/SoC/evalsoc/Common/Source/system_evalsoc.d" -
MT"nuclei_sdk/SoC/evalsoc/Common/Source/system_evalsoc.o" -c -o
"nuclei_sdk/SoC/evalsoc/Common/Source/system_evalsoc.o" "../
nuclei_sdk/SoC/evalsoc/Common/Source/system_evalsoc.c"
Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/stat.c
Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/write.c


Building file: ../application/main.c
Invoking: GNU RISC-V Cross C Compiler
riscv64-unknown-elf-gcc -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
```

```
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -D__IDE_RV_CORE=n307fd -DBOOT_HARTID=0 -DRUNMODE_IC_EN=0
-DRUNMODE_DC_EN=0 -DRUNMODE_CCM_EN=0 -
DDOWNLOAD_MODE=DOWNLOAD_MODE_ILM -DDOWNLOAD_MODE_STRING=\"ILM\" -
I"C:\NucleiStudio_workspace\test\nuclei_sdk\NMSIS\Core\Include" -
I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Common\Include"
-I"C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Incl
-I"C:\NucleiStudio_workspace\test\application" -std=gnu11 -MMD -MP
-MF"application/main.d" -MT"application/main.o" -c -o "application/
main.o" "../application/main.c"
Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/GCC/
startup_evalsoc.S

Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/times.c

Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/GCC/
intexc_evalsoc_s.S

Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/Drivers/
evalsoc_uart.c

Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/
evalsoc_common.c

Finished building: ../nuclei_sdk/SoC/evalsoc/Common/Source/
system_evalsoc.c

Finished building: ../application/main.c

Building target: test.elf
Invoking: GNU RISC-V Cross C++ Linker
riscv64-unknown-elf-g++ -march=rv32imafdc -mabi=ilp32d -
mtune=nuclei-300-series -mcmodel=medlow -msave-restore -isystem=/
include/newlib-nano -O2 -ffunction-sections -fdata-sections -fno-
common -g -T "C:
\NucleiStudio_workspace\test\nuclei_sdk\SoC\evalsoc\Board\nuclei_fpga_eval\Sour
-nostartfiles -nodefaultlibs -Xlinker --gc-sections -Wl,-
Map,"test.map" -Wl,--check-sections -Wl,--no-warn-rwx-segments -u
_isatty -u _write -u _sbrk -u _read -u _close -u _fstat -u _lseek -
u errno -o "test.elf" ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/chown.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
clock_getres.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
clock_gettime.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/clock_settime.o ./nuclei_sdk/SoC/evalsoc/Common/Source/
```

```
Stubs/newlib/close.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/environ.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/
newlib/errno.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
execve.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
exit.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
fork.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
fstat.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
getpid.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
gettimeofday.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
isatty.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
kill.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
link.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
lseek.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
open.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
read.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
readlink.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
sbrk.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
stat.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
symlink.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
times.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
unlink.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
wait.o ./nuclei_sdk/SoC/evalsoc/Common/Source/Stubs/newlib/
write.o  ./nuclei_sdk/SoC/evalsoc/Common/Source/GCC/
intexc_evalsoc.o ./nuclei_sdk/SoC/evalsoc/Common/Source/GCC/
intexc_evalsoc_s.o ./nuclei_sdk/SoC/evalsoc/Common/Source/GCC/
startup_evalsoc.o  ./nuclei_sdk/SoC/evalsoc/Common/Source/Drivers/
evalsoc_uart.o  ./nuclei_sdk/SoC/evalsoc/Common/Source/
evalsoc_common.o ./nuclei_sdk/SoC/evalsoc/Common/Source/
system_evalsoc.o  ./application/main.o   -lstdc++ -lc_nano -lgcc
Finished building target: test.elf

Invoking: GNU RISC-V Cross Create Flash Image
riscv64-unknown-elf-objcopy -O ihex "test.elf"  "test.hex"
Invoking: GNU RISC-V Cross Create Listing
riscv64-unknown-elf-objdump --source --all-headers --demangle --
line-numbers --wide "test.elf" > "test.lst"
Invoking: GNU RISC-V Cross Print Size
riscv64-unknown-elf-size --format=berkeley "test.elf"
   text    data     bss     dec     hex filename
   8824    1272    4592   14688    3960 test.elf
Finished building: test.siz
Finished building: test.hex


Finished building: test.lst
```

```
17:00:23 Build Finished. 0 errors, 0 warnings. (took 5s.75ms)
```

以下为org.eclipse.cdt.managedbuilder.core.headlessbuild所提供的参数，以供参考。

```
   -data       {/path/to/workspace}
   -remove     {[uri:/]/path/to/project}
   -removeAll  {[uri:/]/path/to/projectTreeURI} Remove all projects
under URI
   -import     {[uri:/]/path/to/project}
   -importAll  {[uri:/]/path/to/projectTreeURI} Import all projects
under URI
   -build      {project_name_reg_ex{/config_reg_ex} | all}
   -cleanBuild {project_name_reg_ex{/config_reg_ex} | all}
   -markerType Marker types to fail build on {all | cdt |
marker_id}
   -no-indexer Disable indexer
   -verbose    Verbose progress monitor updates
   -printErrorMarkers Print all error markers
   -I          {include_path} additional include_path to add to
tools
   -include    {include_file} additional include_file to pass to
tools
   -D          {prepoc_define} addition preprocessor defines to
pass to the tools
   -E          {var=value} replace/add value to environment
variable when running all tools
   -Ea         {var=value} append value to environment variable
when running all tools
   -Ep         {var=value} prepend value to environment variable
when running all tools
   -Er         {var} remove/unset the given environment variable
   -T          {toolid} {optionid=value} replace a tool option
value in each configuration built
   -Ta         {toolid} {optionid=value} append to a tool option
value in each configuration built
   -Tp         {toolid} {optionid=value} prepend to a tool option
value in each configuration built
   -Tr         {toolid} {optionid=value} remove a tool option value
in each configuration built
               Tool option values are parsed as a string, comma
separated list of strings or a boolean based on the options type
```

OpenOCD烧写程序时报错Error:Device ID 8xle2g8a6d is not known as FESPI capable¶

问题说明¶

Nuclei Studio 2023.10版中烧写程序时有报以下错误：

参见这个 https://github.com/riscv-mcu/hbird-sdk/issues/8

```
Info : Using libusb driver
Info : clock speed 1000 kHz
Info : JTAG tap: riscv.cpu tap/device found: 0x1e200a6d (mfg:
0x536 (Nuclei System Technology Co Ltd), part: 0xe200, ver: 0x1)
Info : [riscv.cpu] Found 0 triggers
halted at 0x200000b2 due to debug interrupt
Info : Examined RISCV core; XLEN=32, misa=0x40001105
[riscv.cpu] Target successfully examined.
Info : starting gdb server for riscv.cpu on 3333
Info : Listening on port 3333 for gdb connections
Error: Device ID 0x1e200a6d is not known as FESPI capable
Error: auto_probe failed
```

解决方案¶

因为在openocd 2023.10中，将`flash bank $_FLASHNAME`从`fespi`修改为了`nuspi`，需要工程中的openocd配置文件中的`fespi`修改为了`nuspi`， 以蜂鸟工程为例，将 `hbird_sdk/SoC/hbirdv2/Board/mcu200t/openocd_hbirdv2.cfg`修改为如下配置，工程即可正常使用。

```
adapter_khz     1000

interface ftdi
ftdi_vid_pid 0x0403 0x6010
ftdi_oscan1_mode off

transport select jtag

ftdi_layout_init 0x0008 0x001b
ftdi_layout_signal nSRST -oe 0x0020 -data 0x0020
ftdi_layout_signal TCK -data 0x0001
ftdi_layout_signal TDI -data 0x0002
```

```
ftdi_layout_signal TDO -input 0x0004
ftdi_layout_signal TMS -data 0x0008
ftdi_layout_signal JTAG_SEL -data 0x0100 -oe 0x0100

set _CHIPNAME riscv
jtag newtap $_CHIPNAME cpu -irlen 5

set _TARGETNAME $_CHIPNAME.cpu
target create $_TARGETNAME riscv -chain-position $_TARGETNAME
$_TARGETNAME configure -work-area-phys 0x80000000 -work-area-size
10000 -work-area-backup 1

set _FLASHNAME $_CHIPNAME.flash
flash bank $_FLASHNAME nuspi 0x20000000 0 0 0 $_TARGETNAME
# Set the ILM space also as flash, to make sure it can be add
breakpoint with hardware trigger
#flash bank onboard_ilm nuspi 0x80000000 0 0 0 $_TARGETNAME

# Expose Nuclei self-defined CSRS range
770-800,835-850,1984-2032,2064-2070
# See https://github.com/riscv/riscv-gnu-toolchain/issues/
319#issuecomment-358397306
# Then user can view the csr register value in gdb using: info reg
csr775 for CSR MTVT(0x307)
riscv expose_csrs 770-800,835-850,1984-2032,2064-2070

init
#reset
if {[ info exists pulse_srst]} {
  ftdi_set_signal nSRST 0
  ftdi_set_signal nSRST z
}
halt
# We must turn on this because otherwise the IDE version debug
cannot download the program into flash
flash protect 0 0 last off
```

关于dhrystone在IDE上跑分和NSDK 0.5.0命令行跑分不一致的问题¶

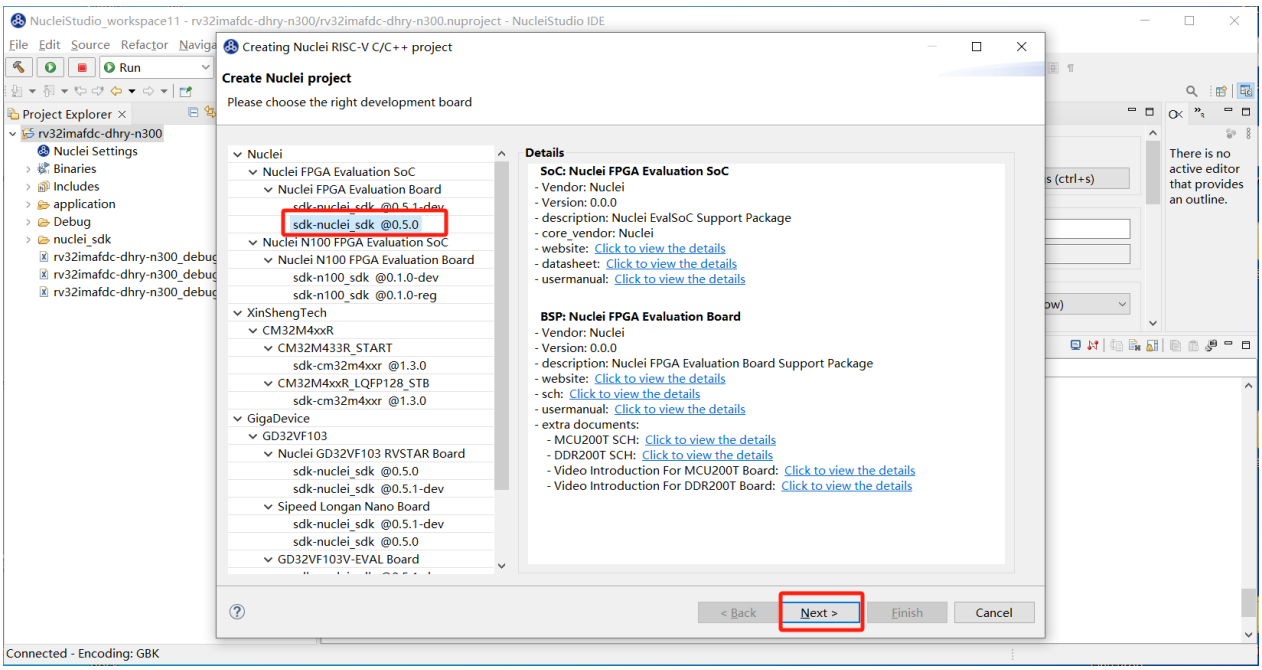
问题说明¶

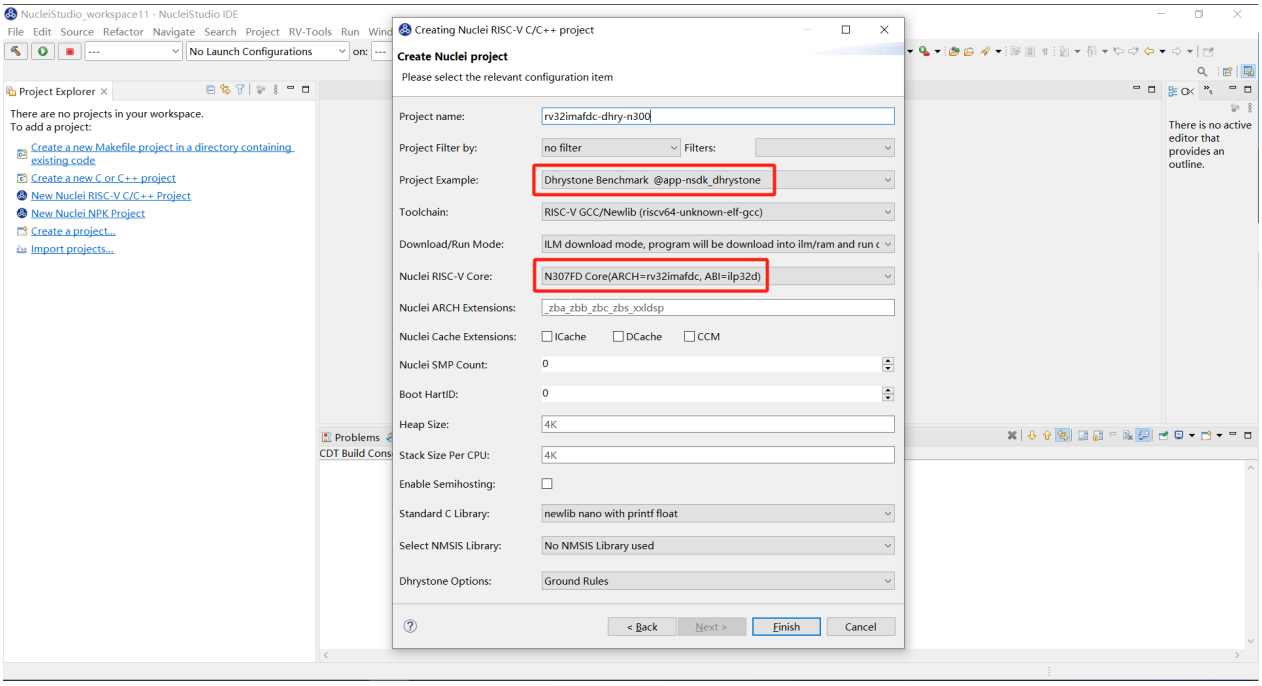
在0.5.0版本的sdk-nuclei_sdk中，为了IDE上使用libncrt库的时候编译有些程序不报错，设置了会默认带上`-msave-restore`。 但在创建dhrystone用例工程时，选择使用newlib库后，该选项会导致跑分降低，不符合CPU的真实跑分。

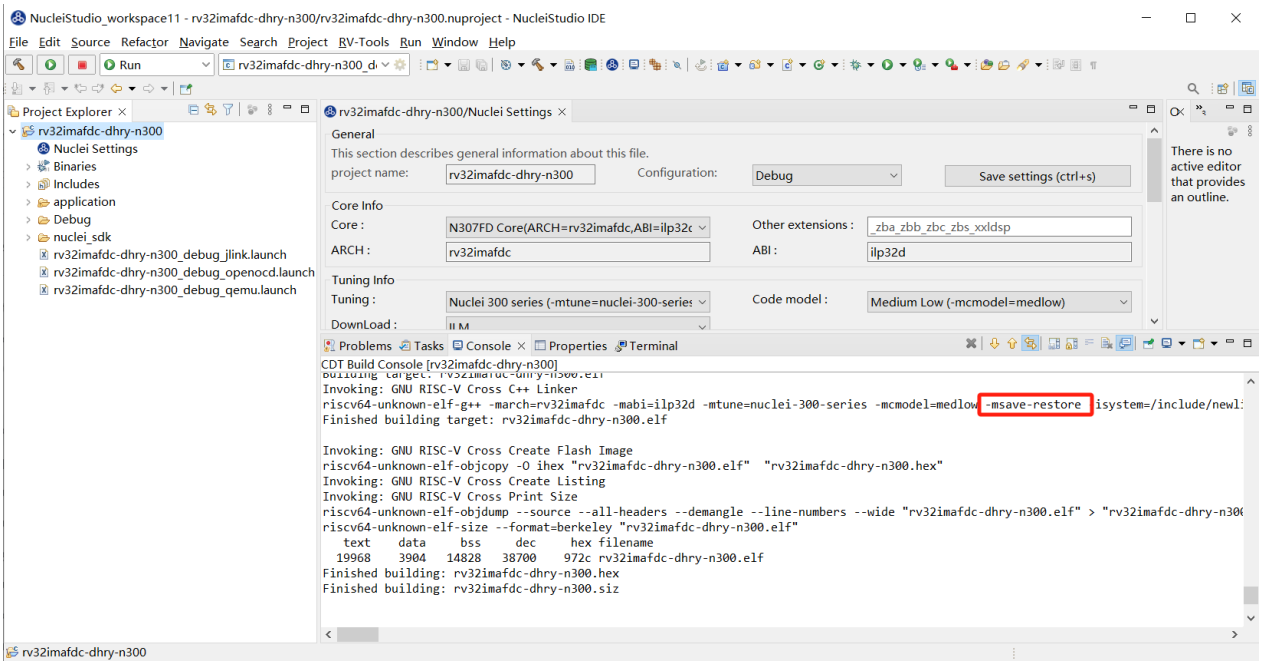
解决方案¶

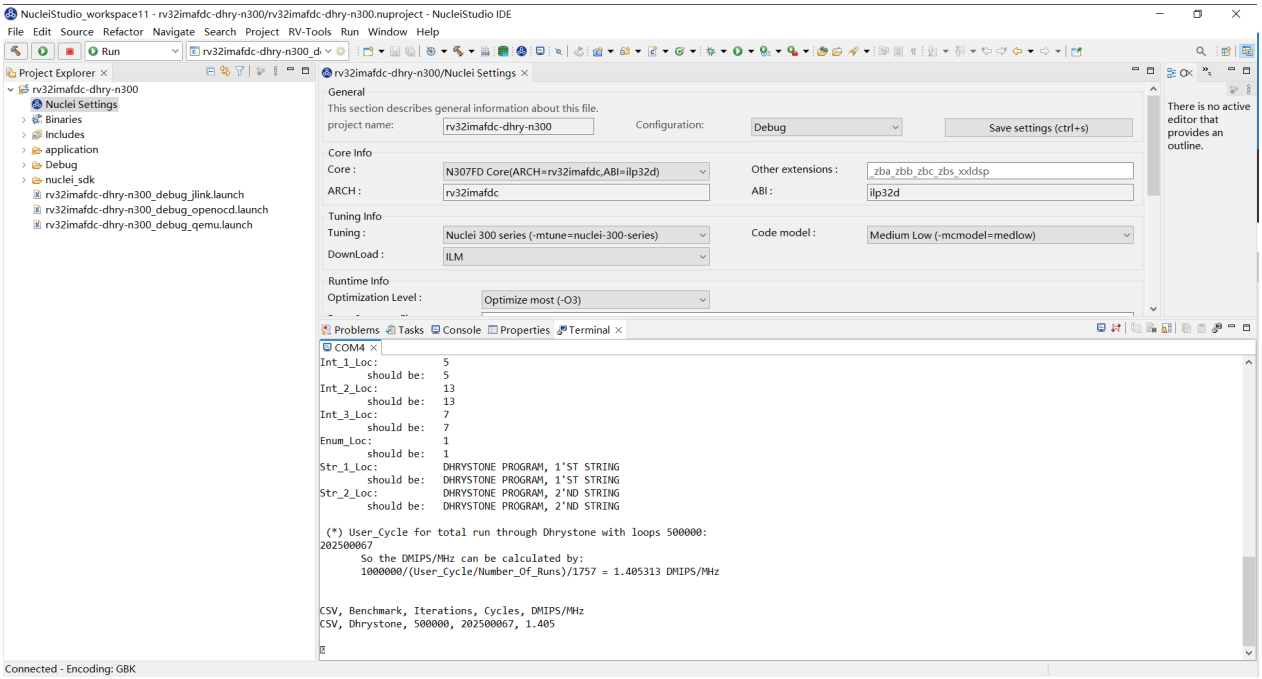
在跑分的时候，需要在对应项目的`Properties -> C/C++ Build -> Settings`中，取消对`Small prologue/epilogue(-msave-restore)`的选中。 具体流程和示例图如下：

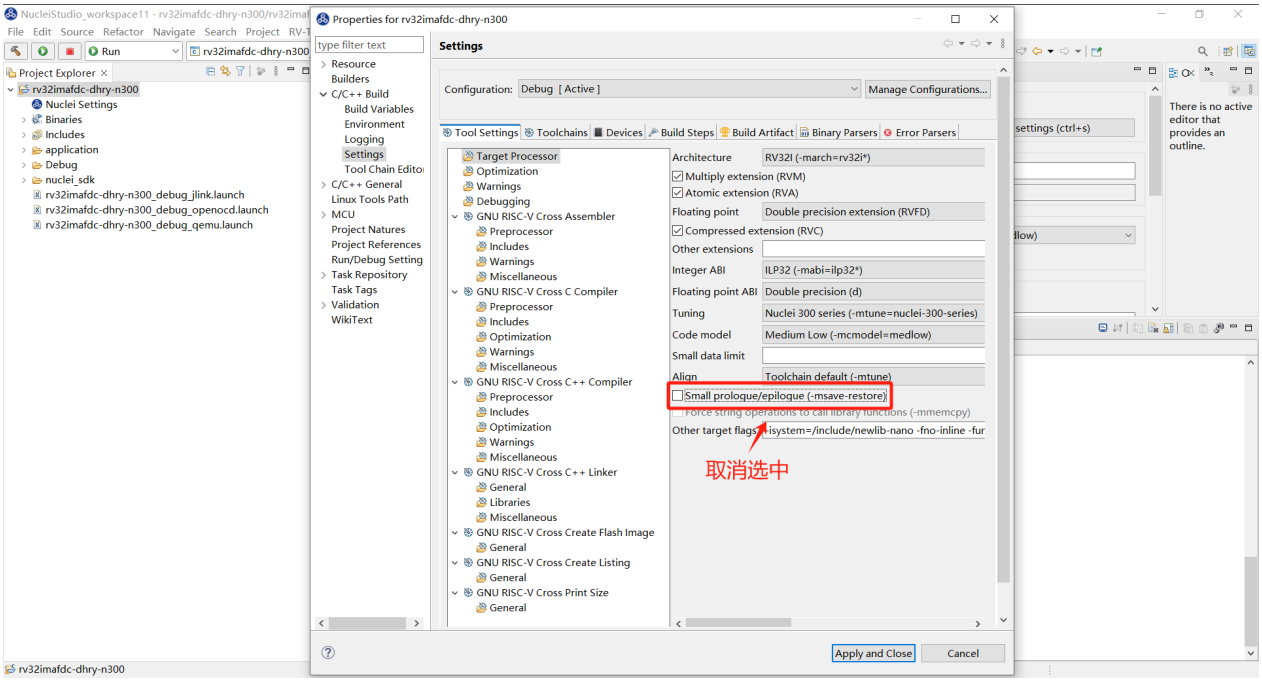
1. 下载sdk-nuclei_sdk 0.5.0 NPK组件包。
2. 新建一个Nuclei RISCV-V C/C++ project。
3. 在新建项目的过程中，选中Dhrystone Benchmark和N307FD Core,其他选项默认设置即可。此时直接编译运行，跑分为1.405。
4. 但实际需要跑分时，要先取消选中`-msave-restore`选项，该跑分结果为1.664。

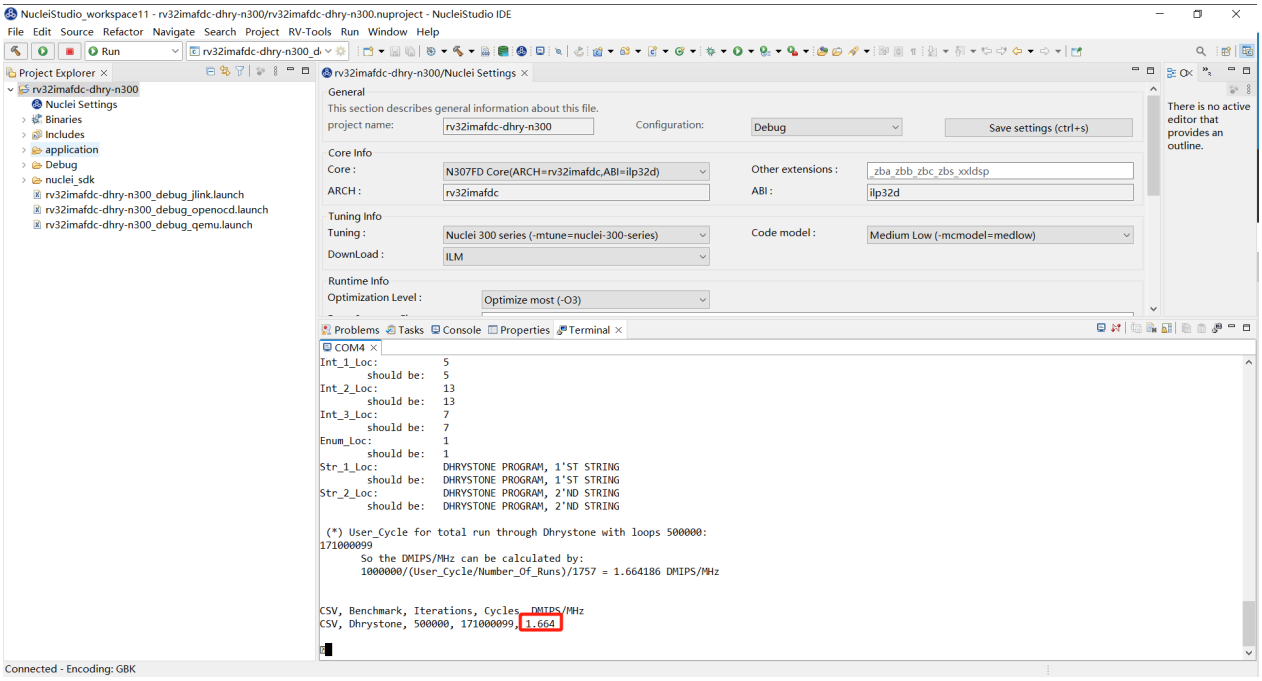










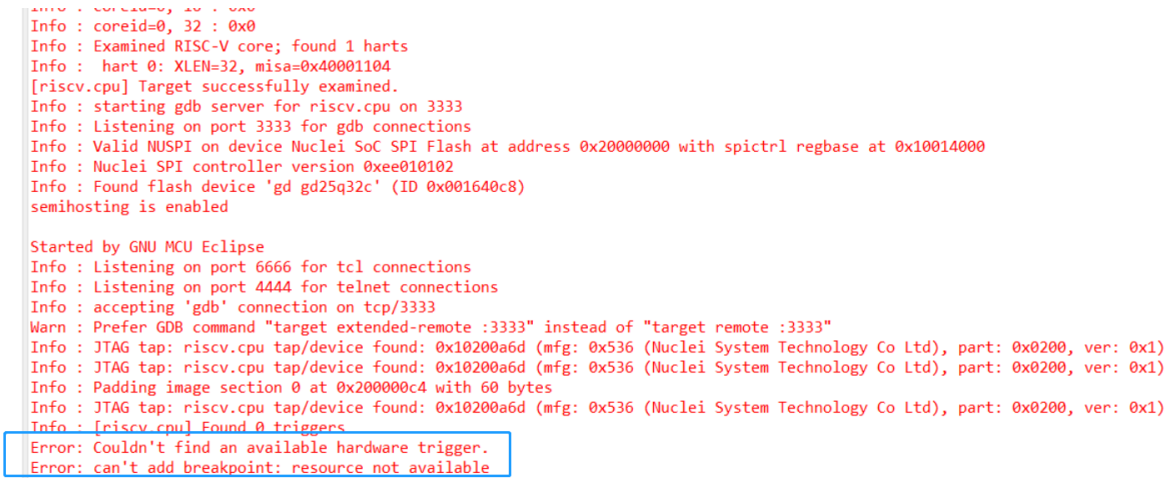


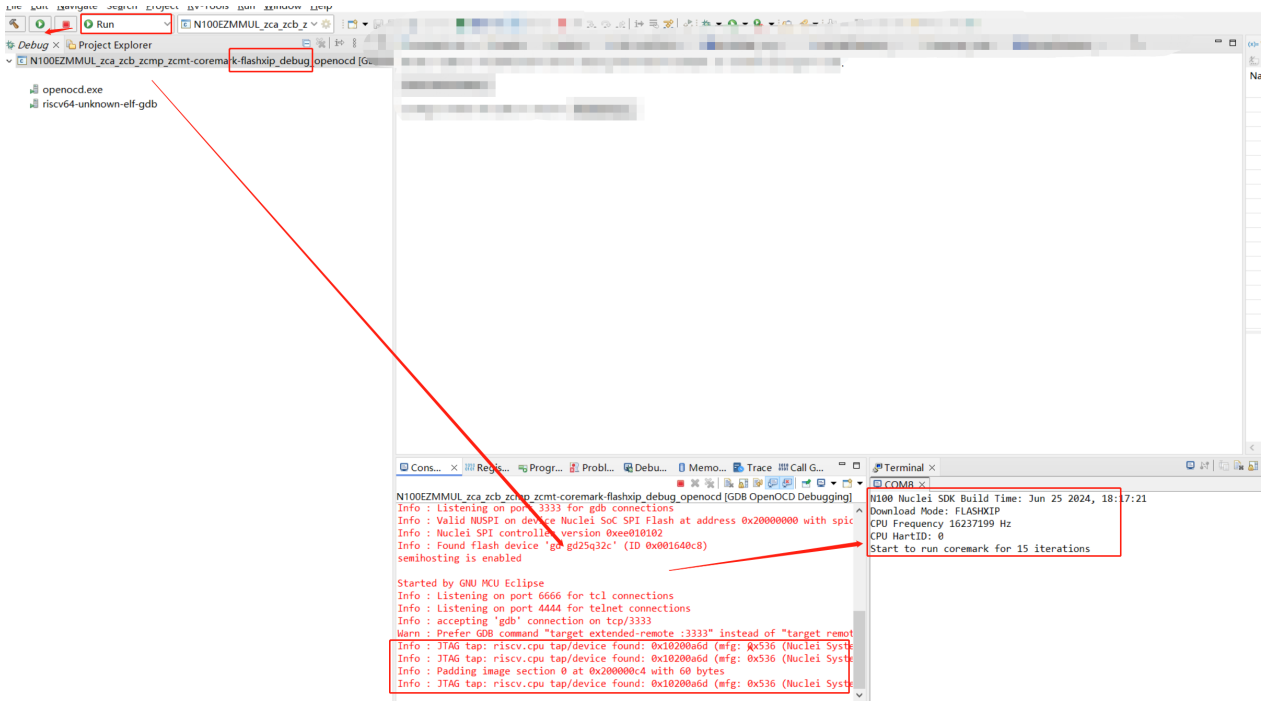
Error: Couldn't find an available hardware trigger / Error: can't add breakpoint: resource not available¶

问题说明¶

在NucleiStudio中使用OpenOCD调试hbird/hbirdv2处理器（不支持硬件断点）或者 Nuclei 100 系列的处理器时，当程序运行在Flash/FlashXip下时，会报Error。

```
Error: Couldn't find an available hardware trigger.
Error: can't add breakpoint: resource not available
```





是因为所运行的CPU不支持硬件断点，导致程序运行在Flash上的时候，IDE调试功能无法正常工作，这个是IDE会需要打一个临时断点的缘故导致的。如果需要下载并运行程序，切换到Run运行模式可以正常运行程序。

解决方案¶

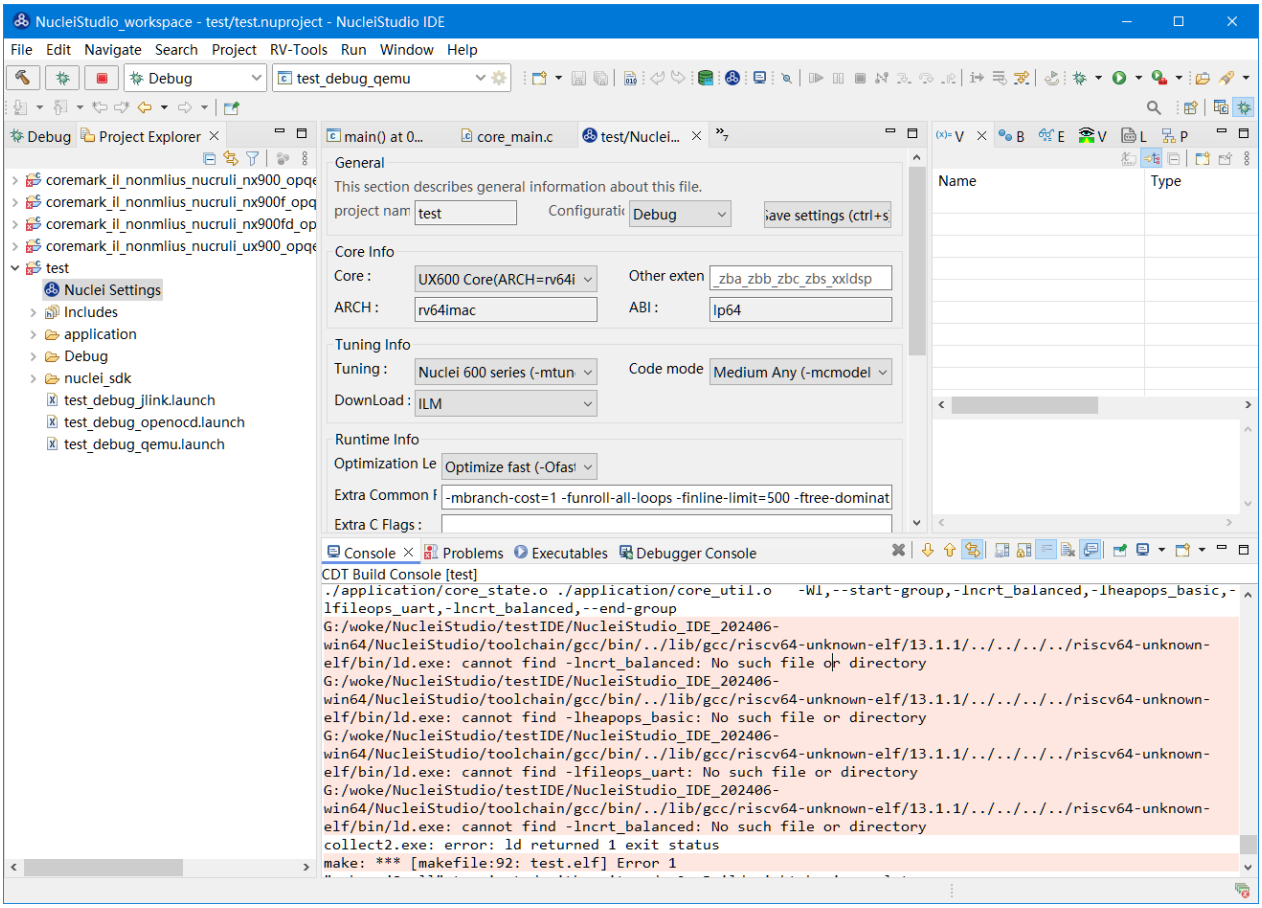
当在调试此类型处理器时，如果需要调试的话，就需要将程序编译运行在RAM上。

cannot find -lncrt_balanced: No such file or directory¶

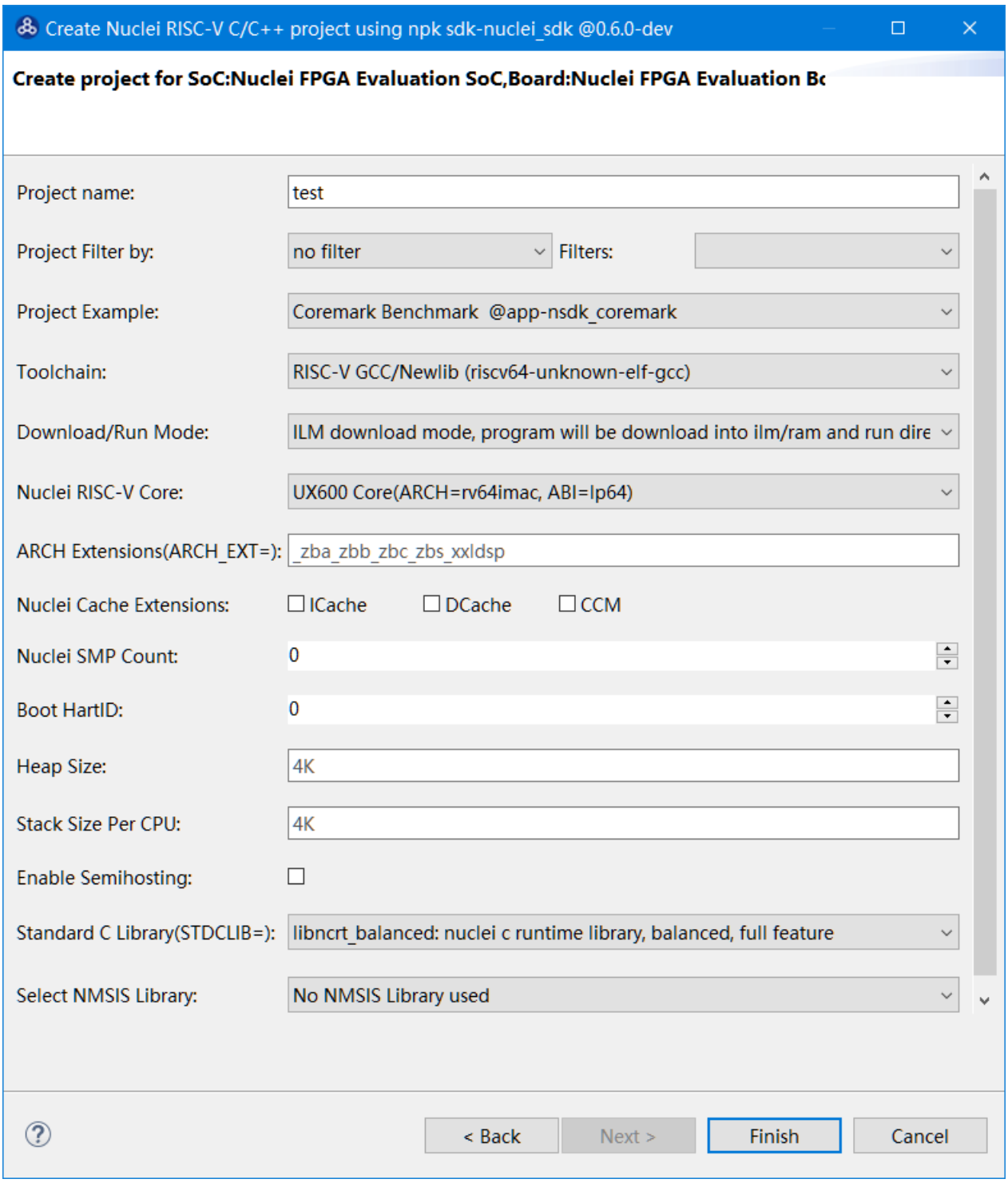
问题说明¶

在NucleiStudio中使用编译工程时有报错信息如下：

```
G:/NucleiStudio/toolchain/gcc/bin/../lib/gcc/riscv64-unknown-elf/
13.1.1/../../../../riscv64-unknown-elf/bin/ld.exe: cannot find -
lncrt_balanced: No such file or directory
G:/NucleiStudio/toolchain/gcc/bin/../lib/gcc/riscv64-unknown-elf/
13.1.1/../../../../riscv64-unknown-elf/bin/ld.exe: cannot find -
lheapops_basic: No such file or directory
G:/NucleiStudio/toolchain/gcc/bin/../lib/gcc/riscv64-unknown-elf/
13.1.1/../../../../riscv64-unknown-elf/bin/ld.exe: cannot find -
lfileops_uart: No such file or directory
G:/NucleiStudio/toolchain/gcc/bin/../lib/gcc/riscv64-unknown-elf/
13.1.1/../../../../riscv64-unknown-elf/bin/ld.exe: cannot find -
lncrt_balanced: No such file or directory
```



是因为在创建工程时，我们创建了一个64位的工程，同时在Standard C Library时，选择了带-lncrt_balanced、-lfileops_uart的扩展，而此类扩展又不支持64位，导致编译不通过。



解决方案¶

`-lncrt_balanced`、`-lfileops_uart`不支持64位处理器，在创建此类处理器工程时，避免使用libncrt库。

UnsatisfiedLinkError of swt-win32-4965r8.dll on Windows 7¶

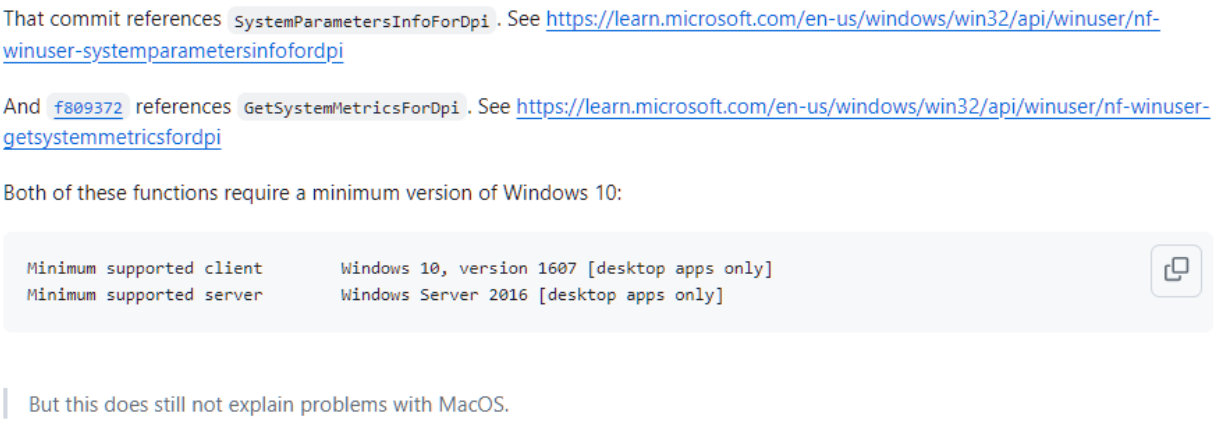
问题说明¶

用户在Windows 7、Windows 8下使用NucleiStudio 2024.06时，发现启动不了，在NucleiStudio\configuration目录的日志中可以看到以下报错内容：

```
!ENTRY org.eclipse.osgi 4 0 2024-07-16 10:41:57.010
!MESSAGE Application error
!STACK 1
java.lang.UnsatisfiedLinkError: Could not load SWT library.
Reasons:
    C:\NucleiStudio\configuration\org.eclipse.osgi\492\0\.cp\swt-
win32-4965r11.dll: 找不到指定的程序。
    no swt-win32 in java.library.path: C:\NucleiStudio;C:
\Windows\Sun\Java\bin;C:\Windows\system32;C:\Windows;C:/
NucleiStudio//plugins/
org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.
0.3.v20240426-1530/jre/bin/server;C:/NucleiStudio//plugins/
org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.
0.3.v20240426-1530/jre/bin;C:\Java\JCDK3.0.4_ClassicEdition\bin;C:
\Java\jdk1.6.0_26\bin;C:\Java\jdk1.6.0_26\lib;C:
\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:
\Windows\System32\WindowsPowerShell\v1.0\;C:\Program
Files\TortoiseSVN\bin;C:\Program Files (x86)\Microsoft SQL
Server\90\Tools\binn\;D:\Python25;C:\NucleiStudio;;.
    no swt in java.library.path: C:\NucleiStudio;C:
\Windows\Sun\Java\bin;C:\Windows\system32;C:\Windows;C:/
NucleiStudio//plugins/
org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.
0.3.v20240426-1530/jre/bin/server;C:/NucleiStudio//plugins/
org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.
0.3.v20240426-1530/jre/bin;C:\Java\JCDK3.0.4_ClassicEdition\bin;C:
\Java\jdk1.6.0_26\bin;C:\Java\jdk1.6.0_26\lib;C:
\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:
\Windows\System32\WindowsPowerShell\v1.0\;C:\Program
Files\TortoiseSVN\bin;C:\Program Files (x86)\Microsoft SQL
Server\90\Tools\binn\;D:\Python25;C:\NucleiStudio;;.
    C:\Users\username\.swt\lib\win32\x86_64\swt-win32-4965r11.dll:
找不到指定的程序。
    Can't load library: C:
```

```
\Users\username\.swt\lib\win32\x86_64\swt-win32.dll
    Can't load library: C:
\Users\username\.swt\lib\win32\x86_64\swt.dll
    C:\Users\username\.swt\lib\win32\x86_64\swt-win32-4965r11.dll:
找不到指定的程序。

    at org.eclipse.swt.internal.Library.loadLibrary(Library.java:
345)
    at org.eclipse.swt.internal.Library.loadLibrary(Library.java:
254)
    at org.eclipse.swt.internal.C.<clinit>(C.java:19)
    at
org.eclipse.swt.internal.win32.STARTUPINFO.<clinit>(STARTUPINFO.java:
42)
    at org.eclipse.swt.widgets.Display.<clinit>(Display.java:149)
    at
org.eclipse.ui.internal.Workbench.createDisplay(Workbench.java:721)
    at org.eclipse.ui.PlatformUI.createDisplay(PlatformUI.java:185)
    at
org.eclipse.ui.internal.ide.application.IDEApplication.createDisplay(IDEApplica
182)
    at
org.eclipse.ui.internal.ide.application.IDEApplication.start(IDEApplication.jav
125)
    at
org.eclipse.equinox.internal.app.EclipseAppHandle.run(EclipseAppHandle.java:
208)
    at
org.eclipse.core.runtime.internal.adaptor.EclipseAppLauncher.runApplication(Ecl
143)
    at
org.eclipse.core.runtime.internal.adaptor.EclipseAppLauncher.start(EclipseAppLa
109)
    at
org.eclipse.core.runtime.adaptor.EclipseStarter.run(EclipseStarter.java:
439)
    at
org.eclipse.core.runtime.adaptor.EclipseStarter.run(EclipseStarter.java:
271)
    at java.base/
jdk.internal.reflect.DirectMethodHandleAccessor.invoke(DirectMethodHandleAccess
103)
    at java.base/java.lang.reflect.Method.invoke(Method.java:580)
    at org.eclipse.equinox.launcher.Main.invokeFramework(Main.java:
668)
    at org.eclipse.equinox.launcher.Main.basicRun(Main.java:605)
    at org.eclipse.equinox.launcher.Main.run(Main.java:1481)
```

是因为在eclipse 2024.06版本中，有使用到一些特性，而该特性对操作系统有要求，可以参考 https://github.com/eclipse-platform/eclipse.platform.swt/issues/1252


That commit references `SystemParametersInfoForDpi`. See https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-systemparametersinfofordpi

And `f809372` references `GetSystemMetricsForDpi`. See https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-getsystemmetricsfordpi

Both of these functions require a minimum version of Windows 10:

```
Minimum supported client       Windows 10, version 1607 [desktop apps only]
Minimum supported server       Windows Server 2016 [desktop apps only]
```

> But this does still not explain problems with MacOS.

并且在eclipse的官方文档中，针对eclipse测试的操作系统中也做了说明，对某些版本的操作系统不再做兼容。可以参考 https://eclipse.dev/eclipse/development/plans/eclipse_project_plan_4_32.xml#target_environments

Operating System	Version	Hardware	JRE	Windowing System
Windows	10 11	x86 64-bit	OpenJDK 17.0.8 (LTS) OpenJDK 21 (LTS) Oracle Java 17.0.8 (LTS) Oracle Java 21 (LTS)	Win32
Red Hat Enterprise Linux	9.0	x86 64-bit aarch64	OpenJDK 17.0.8 (LTS) OpenJDK 21 (LTS) Oracle Java 17.0.8 (LTS) Oracle Java 21 (LTS)	GTK 3
		Power 64-bit LE	OpenJDK 17.0.8 (LTS)	
SUSE Linux Enterprise Server	15 SP4	x86 64-bit	OpenJDK 17.0.8 (LTS) OpenJDK 21 (LTS)	GTK 3
		Power 64-bit LE	OpenJDK 17.0.8 (LTS)	
Ubuntu Long Term Support	22.04	x86 64-bit aarch64	OpenJDK 17.0.8 (LTS) OpenJDK 21 (LTS)	GTK 3
Apple macOS	12 13	x86 64-bit	OpenJDK 17.0.8 (LTS) OpenJDK 21 (LTS) Oracle Java 17.0.8 (LTS) Oracle Java 21 (LTS)	Cocoa
	12 13	M1 (arm64)	OpenJDK 17.0.8 (LTS) OpenJDK 21 (LTS)	

而NucleiStudio 2024.06是基于eclipse 2024.06，所以也会有同类型的问题。

解决方案¶

请在windows 10或以上的版本操作系统上使用 NucleiStudio 2024.06。

如果想在Windows 7、Windows 8等低版本的操作系统上使用NucleiStudio，可以考虑使用NucleiStudio 2024.02及以下版本。

使用 Profiling 功能时可能遇到的一些问题¶

目前使用 Profiling 功能可能遇到一些问题，记录如下:

- 问题1：日志打印中报片上内存不足，没有充足内存来存放 gprof/gcov 数据
- 问题2：采用串口输出的方式收集数据，打印被冲掉，Console 或 Terminal 收集的数据不全，导致数据解析失败，弹出 `No files have been generated` 错误弹框
- 问题3：删掉 `gmon.out` 文件，再次解析时，弹出 `No files have been generated` 错误弹框

问题1：日志打印中报片上内存不足，没有充足内存来存放 gprof/gcov 数据¶

gprof/gcov data 需要存到片上内存上，占用内存的大小与用例规模有关(几十到几百KB不等)，需要确保片上内存足够大。

解决方案¶

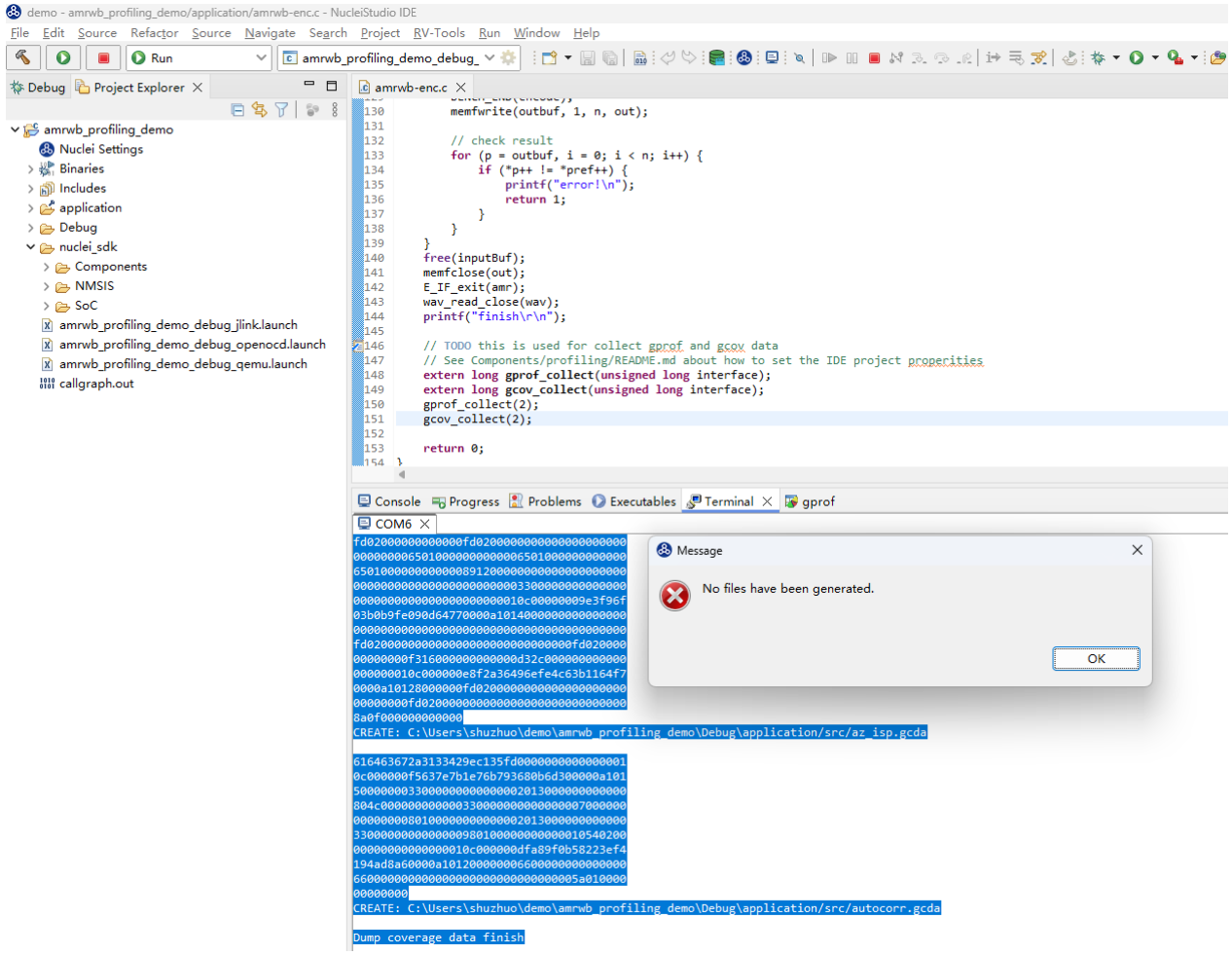
首先需要确认软件配置的内存大小与硬件实际大小相匹配（ilm/sram/flash/ddr/），否则需要适配软件链接脚本内存布局：
比如，如果是 `DOWNLOAD=ilm` 模式下载，可以按硬件的 ilm 与 dlm 大小适配。 对于 `nuclei sdk 0.6.0` 版本，修改的文件为`nuclei-sdk/SoC/evalsoc/Board/nuclei_fpga_eval/Source/GCC/gcc_evalsoc_ilm.ld`

```
INCLUDE evalsoc.memory

MEMORY
{
  ilm (rxa!w) : ORIGIN = ILM_MEMORY_BASE,   LENGTH =
ILM_MEMORY_SIZE
  ram (wxa!r) : ORIGIN = DLM_MEMORY_BASE,   LENGTH =
DLM_MEMORY_SIZE
}
```

如果 DOWNLOAD=ilm 模式内存不足，可以使用内存大一点的下载方式（如 DOWNLOAD=ddr）。

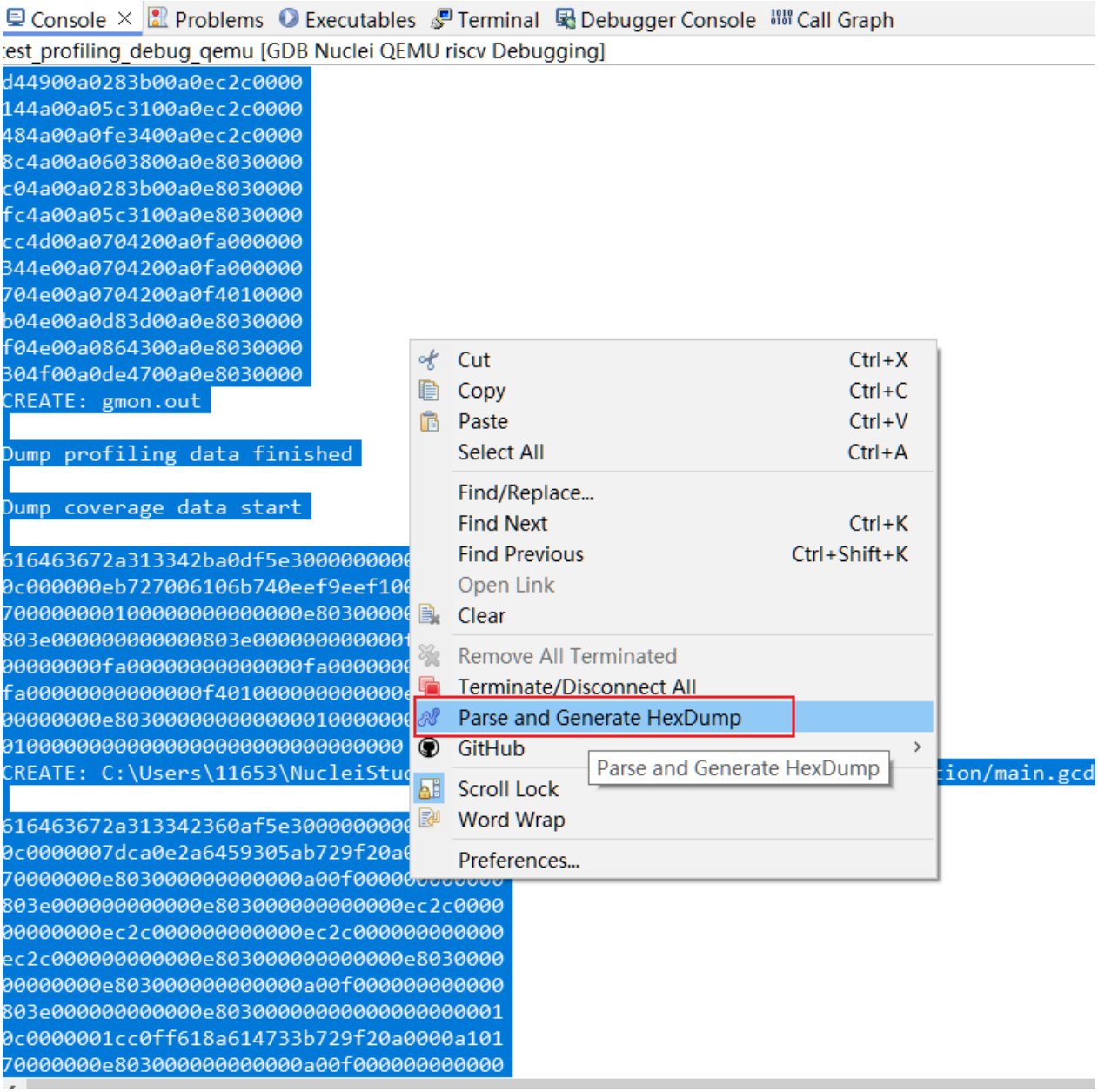
问题2：Console 或 Terminal 收集的数据不全导致数据解析时失败¶

在 NucleiStudio 2024.06 中，当选择使用串口输出的方式使用 Profiling 功能时，可能使用 Parse and Generate Hexdump 解析数据时 弹出 No files have been generated 错误弹框，最后没有生成对应的 gmon.out 文件或者 *.gcno 文件。这可能是因为串口数据被冲掉，导致数据不完整从而解析失败



确认方法：

需确保串口开始时的打印没有被冲掉，参考Nuclei Studio使用Profiling功能进行性能调优举例



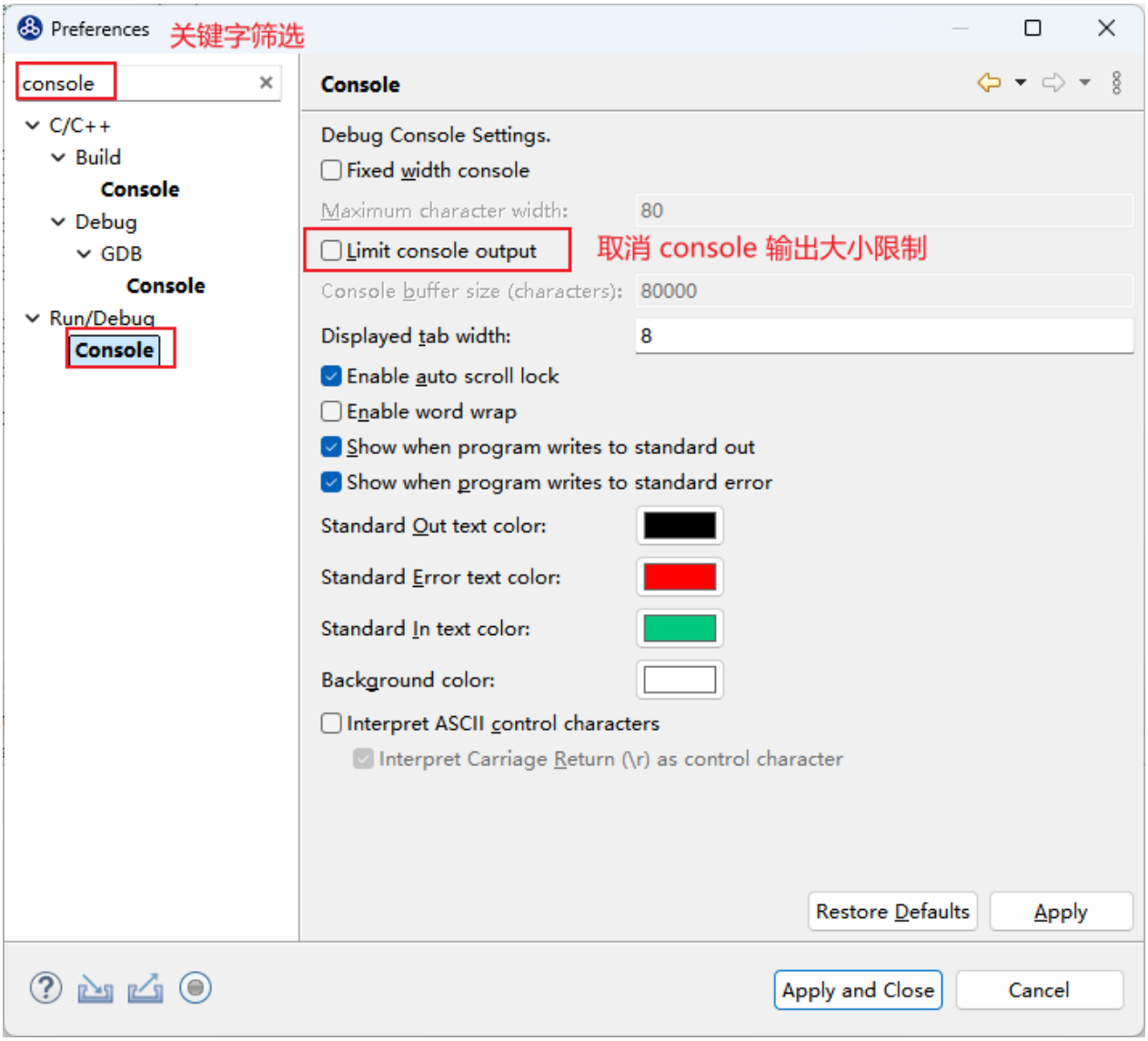
解决方案¶

因为在Console或者Terminal中，对输出的内容条数有限制，当输出的内容长度超过限制时，前面的内容会被冲掉，导致内容不完整，这样会解析失败。

需要调节 Console 或 Terminal 输出大小限制，确保数据没有被冲掉。

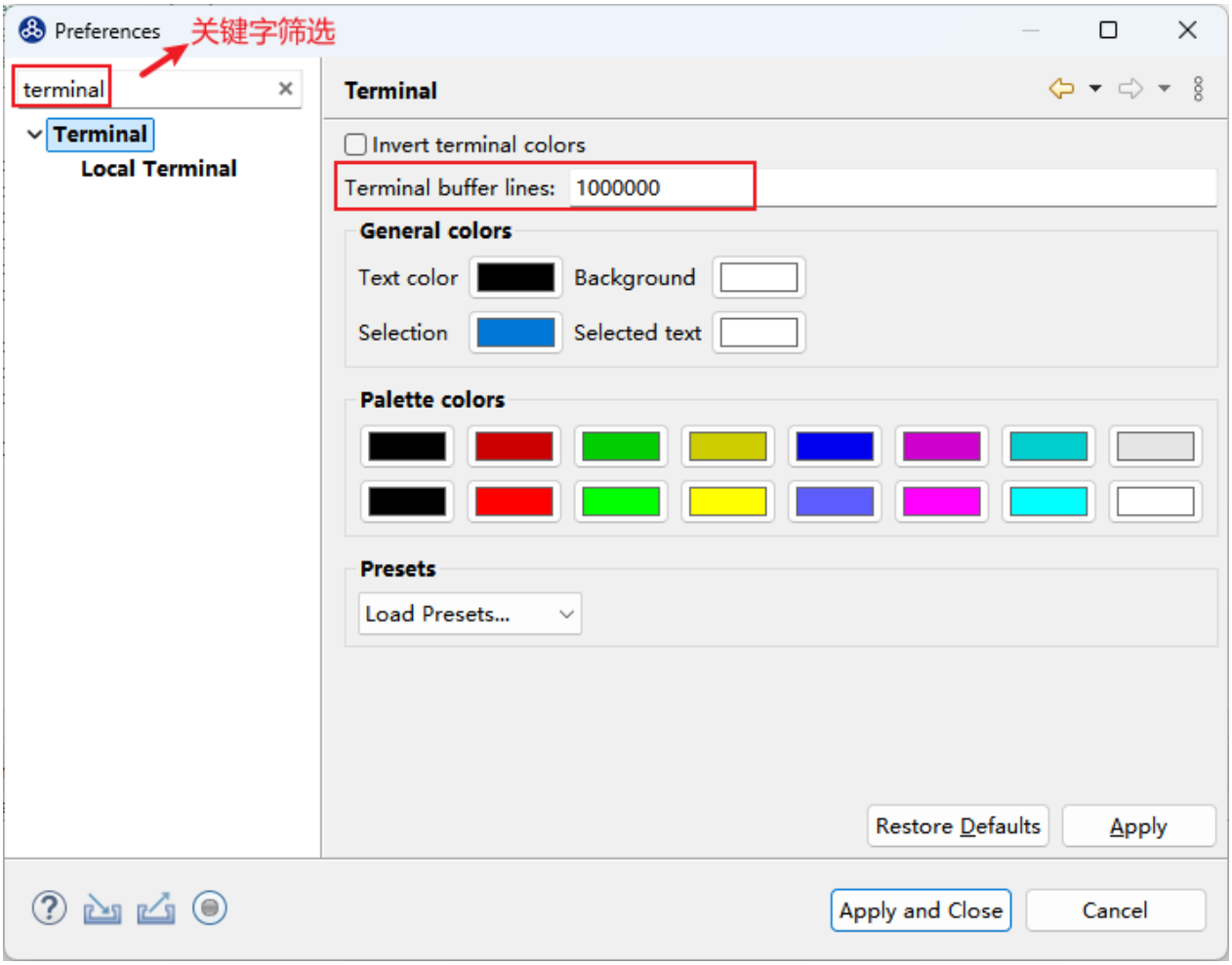
- 建议将Console中输出内容条限修改为不受限制。

`Window->Preference` 进入如下界面：



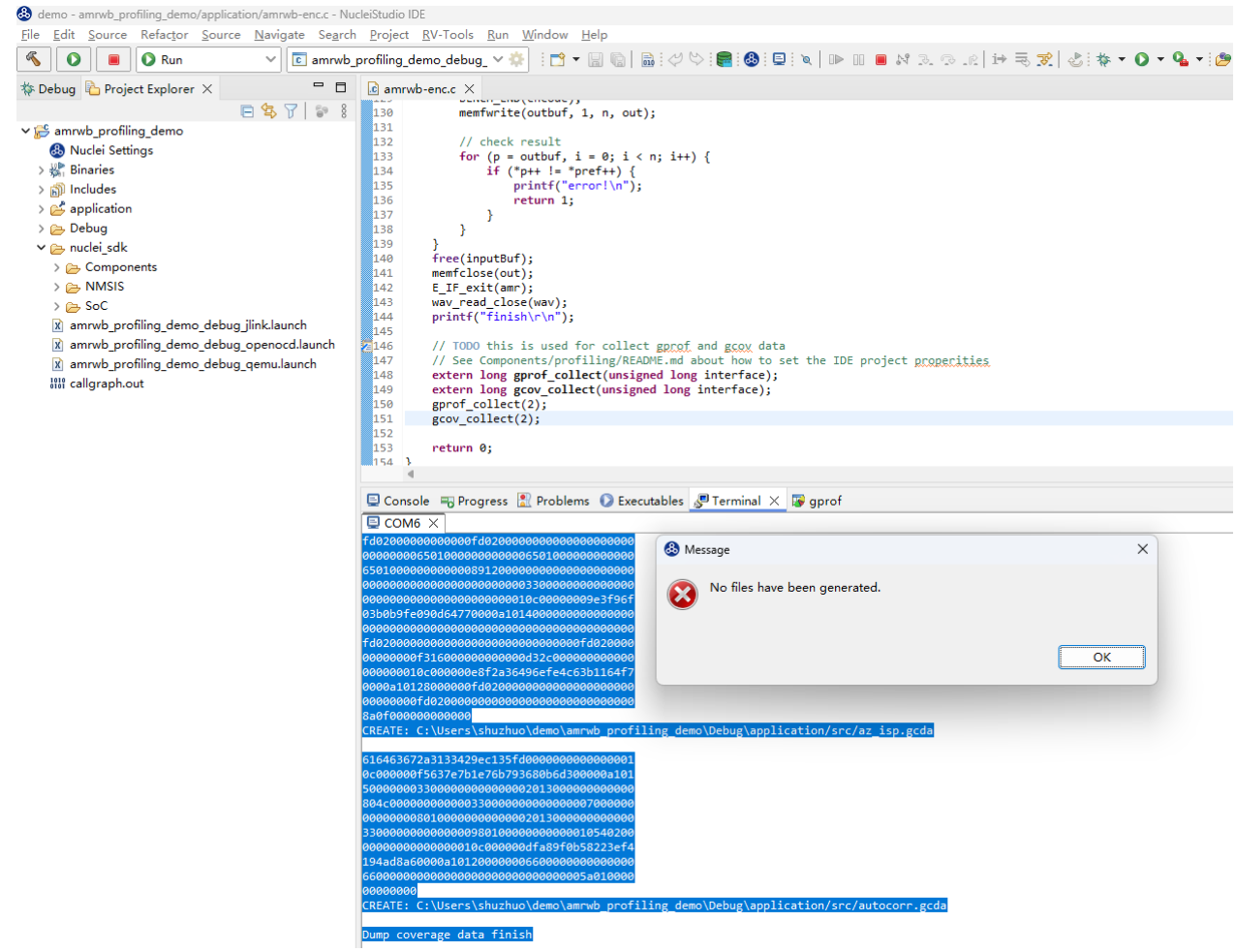
- 建议将Terminal中输出内容条限修改为一个较大的值。

`Window->Preference` 进入如下界面：



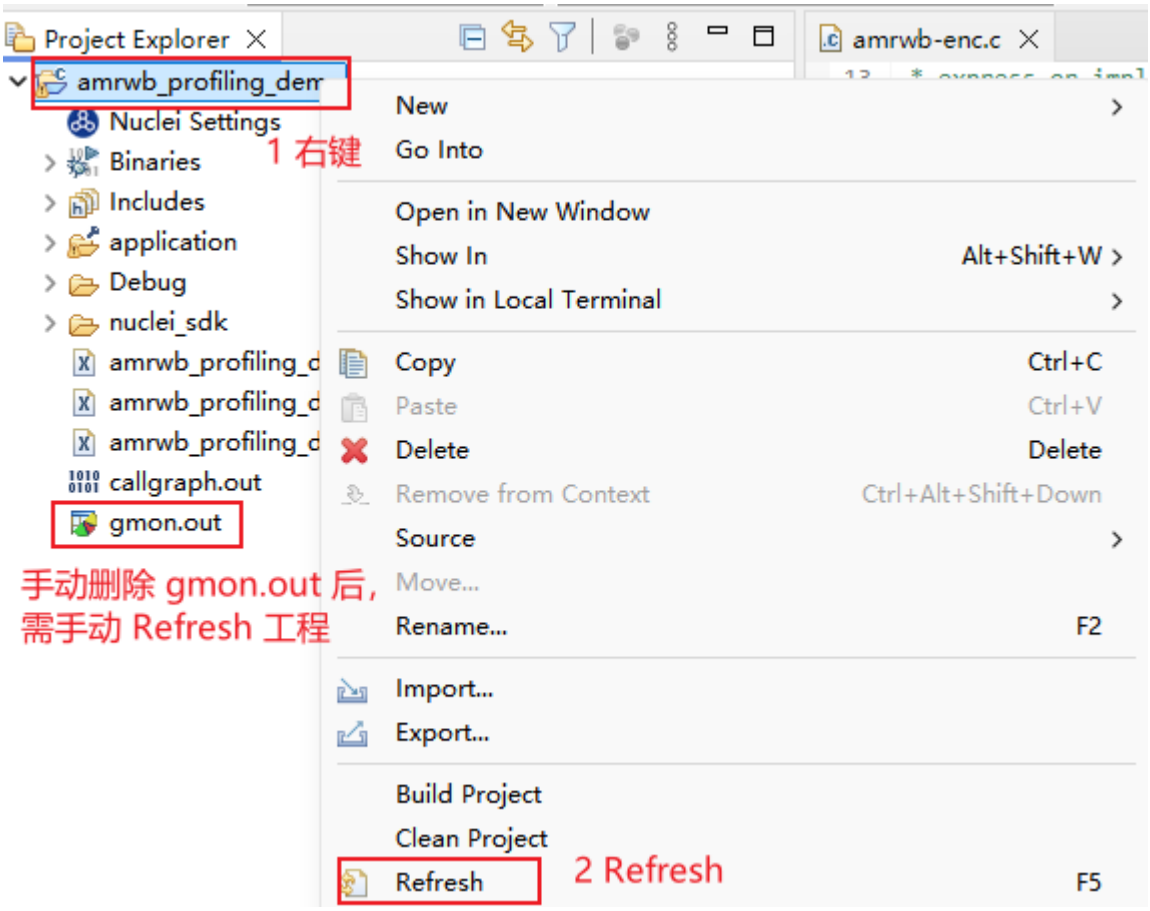
问题3：删掉 gmon.out 文件，再次解析，弹出 No files have been generated 错误弹框¶

手动删掉工程文件夹下的 gmon.out 文件，再次解析时出现 `No files have been generated` 的错误弹框



解决方案¶

手动删掉 gmon.out 文件后，需要手动刷新一下工程。



Nuclei Studio使用Profiling功能进行性能调优举例¶

文档是基于 Nuclei Studio 的 2024.06 Windows 版本实测。

问题说明¶

Nuclei Studio 2024.06 提供 Profiling 功能、Call Graph 功能 以及 Code coverage 功能，方便用户使用。 简单描述如下：

- Profiling 功能：基于 binutils gprof 工具，可用于分析函数调用关系、调用次数、以及运行时间；通过 Profiling 抓取热点函数可以用来分析程序的瓶颈，以便进行性能优化。
- Call Graph 功能：基于 Profiling 功能，将函数调用关系、调用次数、以及运行时间用图展示出来，方便开发人员分析。
- Code coverage 功能：基于 gcc 编译器提供 gcov 工具，可用来查看源码文件的代码覆盖率，帮助开发人员确定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 NucleiStudio_User_Guide.pdf 相关章节对这几个功能已经有较详细的描述，这篇文档以一个例子来展示它们的实际应用。

解决方案¶

1 环境准备¶

所需材料：

- Nuclei Studio：NucleiStudio 2024.06，以 Windows 版本为例
- 用例： 以 AMR-WB-enc 即自适应多速率宽带编码音频算法为例，用户可以移植自己的用例

基于 nuclei-sdk v0.6.0 移植 amrwbenc 裸机用例：

打开 Nuclei Studio 建立 amrwbenc 工程，然后移植 amrwbenc 源码，最终用例可正常运行。用户可以移植自己的用例，不同用例移植的细节各不相同，这一步不是这篇文档的重点，略过。

2 Profiling 功能¶

Nuclei studio 中 Profiling 功能基于 binutils gprof 工具。编译时需带特定的编译选项 -pg 来编译指定源码文件，编译成功后得到 ELF 文件， 然后在实际开发板上运行并收集需要的 gmon.out 文

件，最终在 IDE 上以图形化的方式展示。所以还需要在用例末尾添加 gprof 数据收集代码，有两种方式：

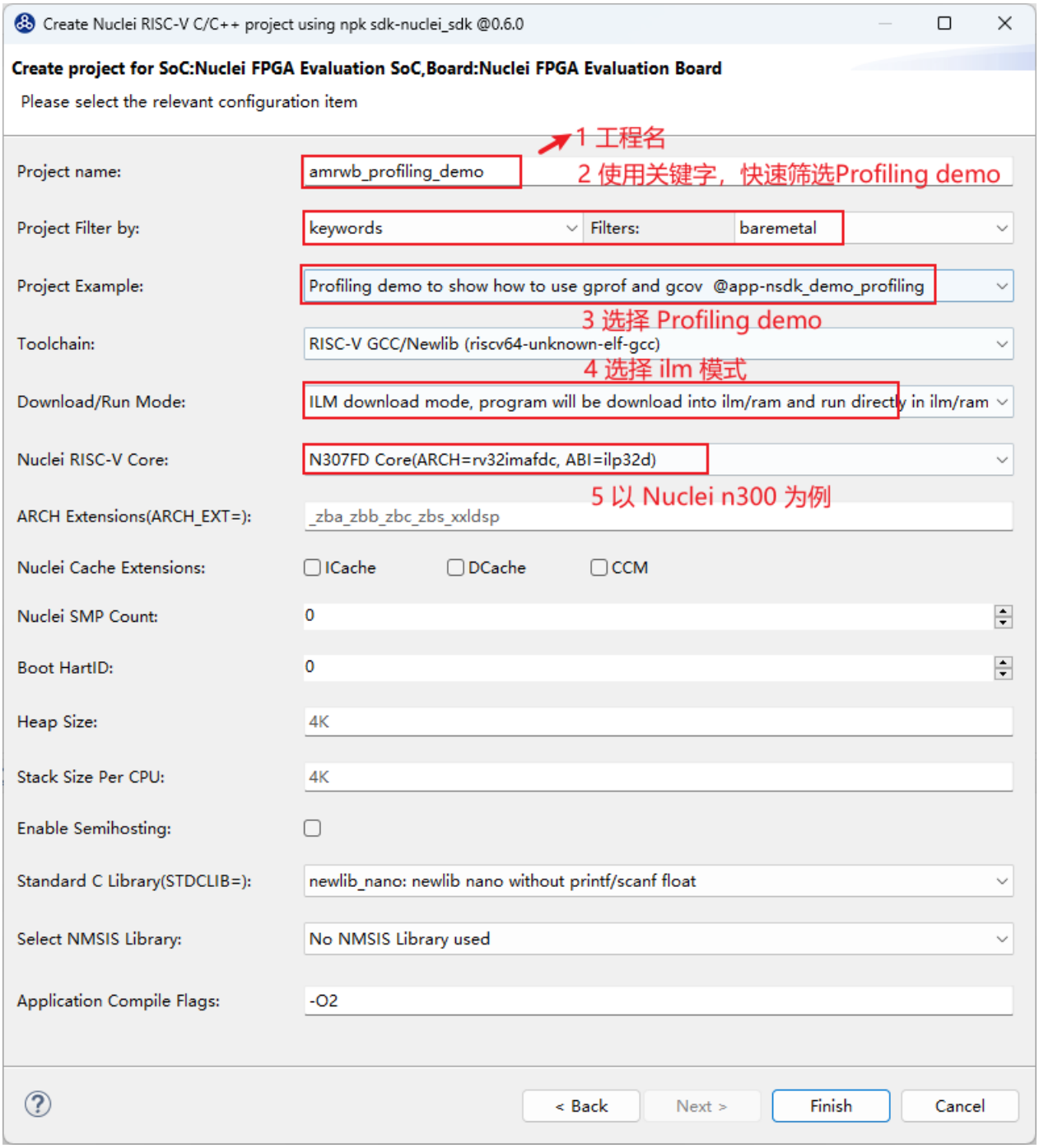
- 方式1：移植 gprof 数据收集代码到自己的工程中，代码可以参考 Profiling README
- 方式2：基于 Nuclei Studio 中的 Profiling demo 进行改造，即用自己的用例替换掉 Profiling demo 工程的的用例部分

下面示例采用后一种方法进行演示：

step1：新建 Profiling demo 工程

`File->New->New Nuclei RISC-V C/C++ Project`，选择 `Nuclei FPGA Evalution Board->sdk-nuclei_sdk @0.6.0`

注意： Nuclei SDK 需选择 0.6.0 及以后版本才支持 Profiling 与 Code coverage 功能

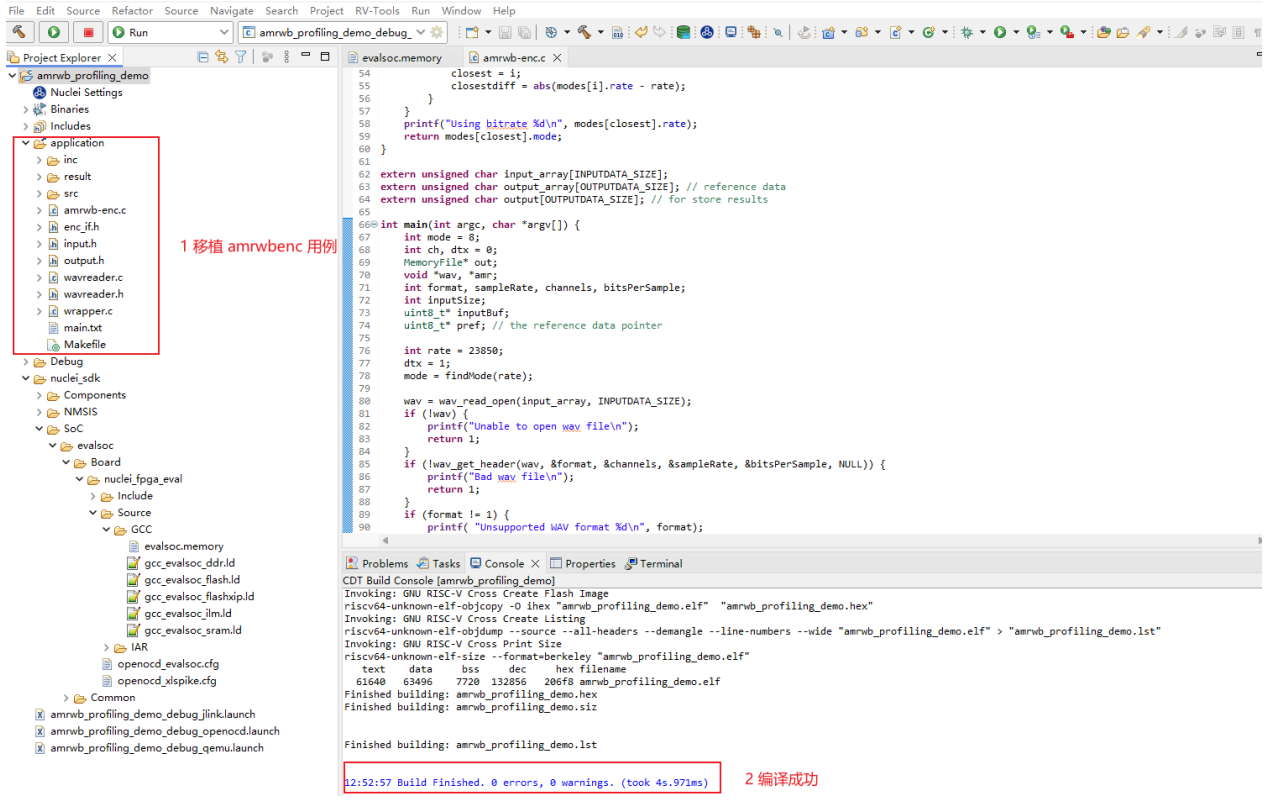


step2：基于 Profiling demo 工程移植 amrwbenc 裸机用例

删掉 Profiling demo 工程中 application 中的原始用例，替换成 amrwbenc 用例，形成如下目录结构，并确保能编译成功。

这里提供本示例使用的工程，有兴趣可以下载使用：
优化前的工程下载链接

下载 zip 包后，可以直接导入到 Nuclei Studio 中运行(导入步骤：`File->Import->Existing Projects into Workspace->Next->Select archive file->`选择zip压缩包`->next`即可)



step3：在用例结尾处添加 grof 数据收集代码，并添加 -pg 编译选项，重新编译代码

在 main 函数的结尾处添加 gprof 数据收集代码：

```
int main(int argc, char *argv[]) {
    /*
     * 代码省略
     */

    /*
     * 在main函数的结尾处添加gprof数据收集代码
     */
    // TODO this is used for collect gprof and gcov data
    // See Components/profiling/README.md about how to set the IDE
project properities
    extern long gprof_collect(unsigned long interface);
    gprof_collect(2);

    return 0;
}
```

收集 gprof data 有三种方式，通过入参不同进行区分：

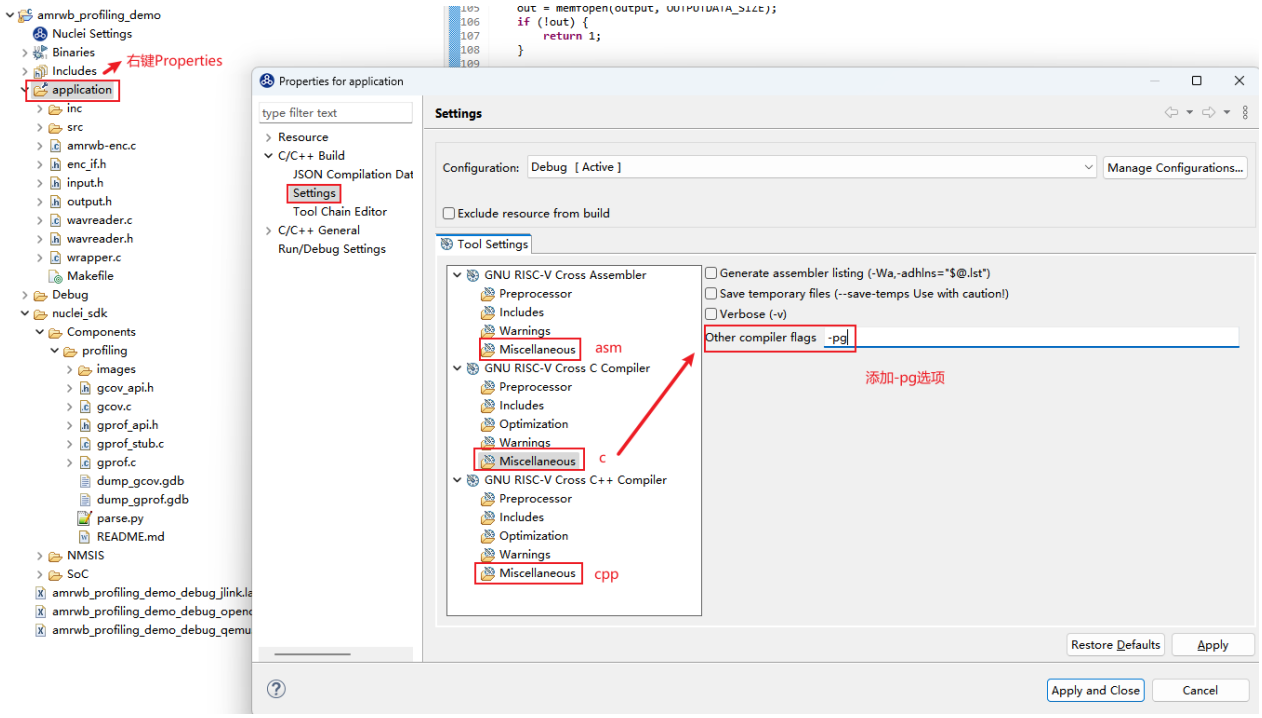
- gprof_collect(0)：在缓冲区中收集 gprof 或 gcov 数据，在调试程序时可以使用 GDB 脚本转储 gcov 或 gprof 二进制文件

- gprof_collect(1)：使用 semihost 直接将 gprof 或 gcov 数据写入文件中
- gprof_collect(2)：直接在 Console 或 Serial Terminal 中打印 gcov 或 gprof 数据，然后可以通过IDE中 `Parse and Generate HexDump` 功能进行解析数据并保存到PC上

详情可参考 Profiling README，这里以将 gprof data 打印到串口（Console 或 Serial Terminal）为例。

对需要进行profiling的代码添加 `-pg` 编译选项，重新编译代码：

注意： 选择 application, 对关键代码添加 `-pg` 编译选项，这个用例只有 C 代码，只对 C 代码添加 `-pg` 编译选项即可



step4：运行程序

有几种方式可以运行程序：

- qemu 模拟器（不需要硬件，简单跑一下流程，测试结果不准确）
- 上板测试 （基于定时器采集数据）
- 基于 xl_cpumodel (Nuclei Near Cycle Model)，参考: 通过Profiling展示Nuclei Model NICE/VNICE指令加速

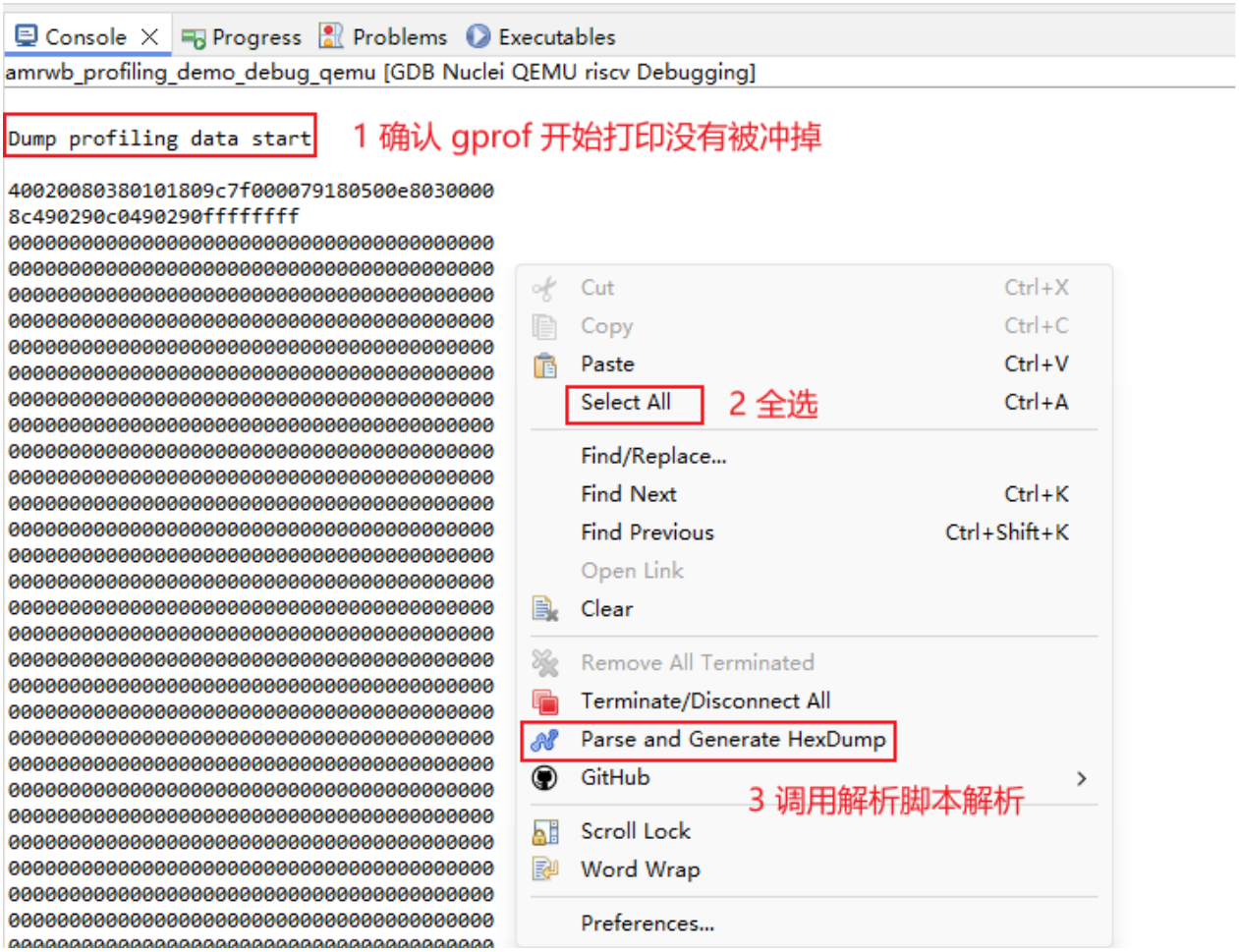
这一篇文章只介绍 qemu 仿真与上板测试两种方式，qemu 收集的数据打印到 Console 口，上板实际运行输出到 Nuclei Studio 的 Serial Terminal 口。

step5：解析 gprof 数据

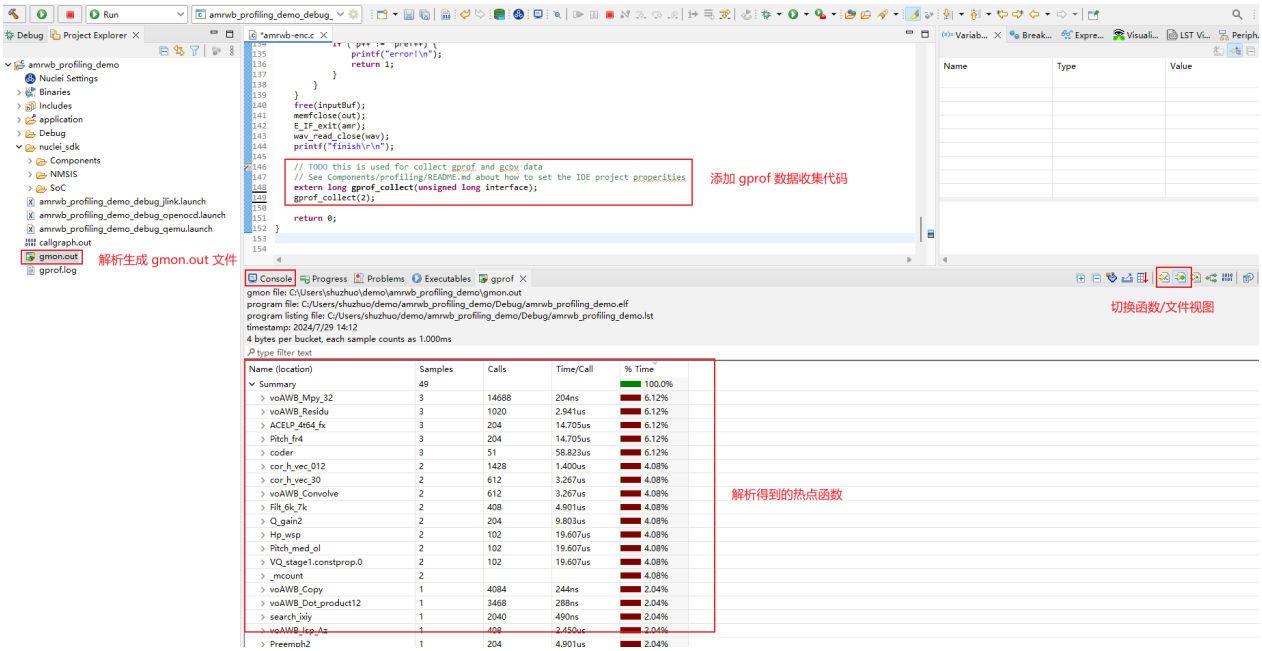
开始解析 gprof 数据。 **注意：** 这一步可能遇到一些问题，解决方法可参考 Profiling与 Code coverage 功能可能遇到的问题

- 在 qemu 上测试, log 打印到 Console 口

注意: qemu 仅用来模拟展示，如果希望得到准确的热点函数，需要上板测试。



解析完成后，会在当前工程目录下生成 gmon.out，双击打开展示：

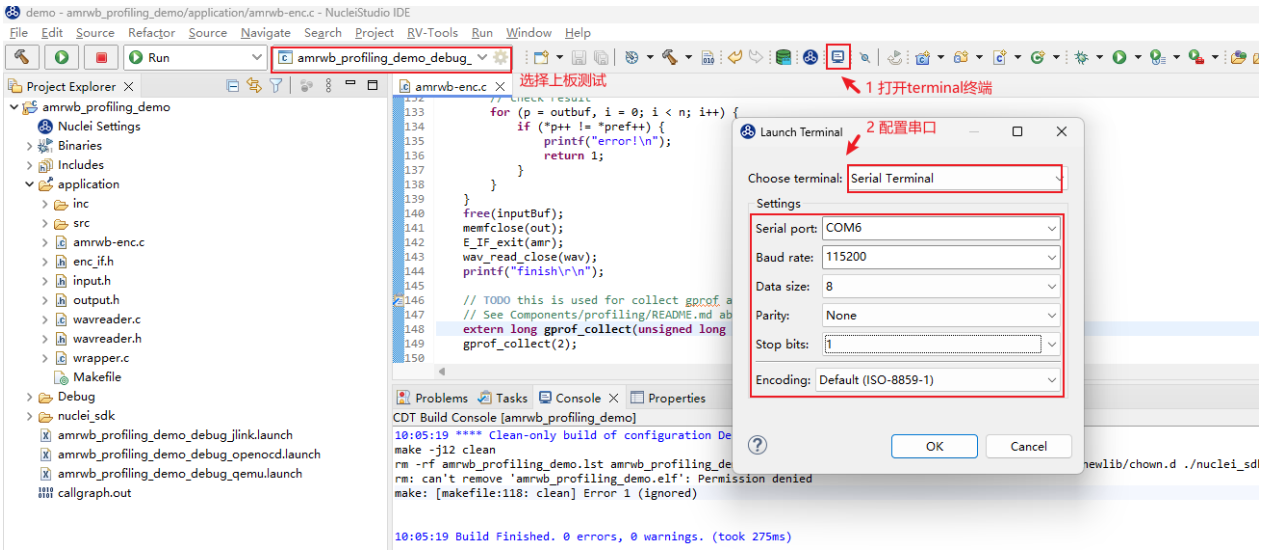


- 上板测试

上板测试的步骤与 qemu 类似，唯一不同的是 gprof 数据输出到 Serial Terminal 上。

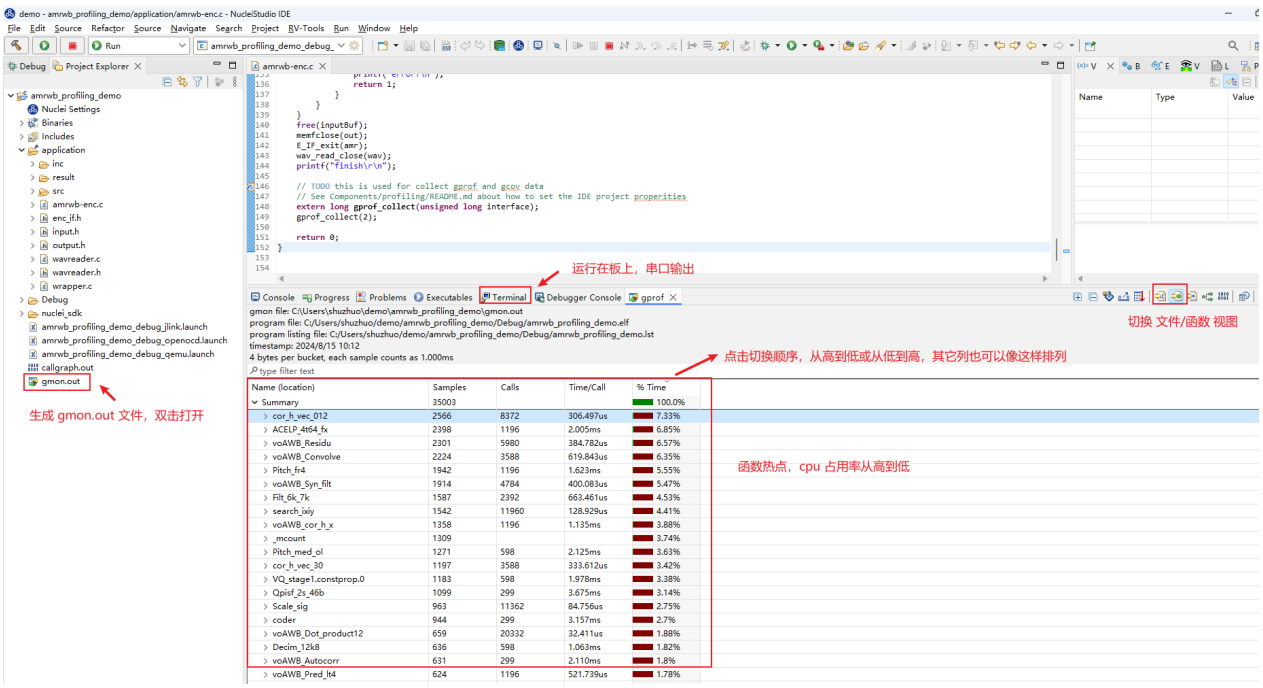
配置 Serial Terminal:

注意:如果串口工具已经打开，确保每次运行 gprof 前，清除掉串口打印（鼠标右键-> Clear Terminal），避免对数据解析产生影响。



同样，全选 log，右键选择`Parse and Generate HexDump` 功能，就会在工程文件夹下生成 gmon.out 文件， 刷新工程后，就可以双击打开这个gmon.out 文件。

如下图是在**板子上实际运行**得到的 gprof 数据：



从而得到 TOP5 热点函数为（实际上板测试）：

```
cor_h_vec_012
ACELP_4t64_fx
voAWB_Residu
voAWB_Convolve
voAWB_Syn_filt
```

获得热点函数后，可以从热点函数入手开始优化，优化 TOP 函数往往可以事半功倍。

step6：优化热点函数

有如下几种方法优化热点函数：

- 调节编译器参数，针对整个工程或单独算子使用 O2/O3/Ofast 等优化等级，开启 `-finline-functions -funroll-all-loops` 等优化选项
- 针对算法进行优化，使用更好的算法实现热点函数
- 使用 RISC-V 扩展指令（RVP/RVV 扩展等）优化

这里以 RVP 扩展为例，按照热点函数从高到低，用 RVP 扩展来优化。需要确定所用硬件支持 RVP 扩展。

举例如下：

TOP1 热点函数为 `cor_h_vec_012`，分析函数，尝试使用 RVP 扩展优化：

如下以 #if defined __riscv_xxldspn3x 隔开的代码表示使用 Nuclei N3 P 扩展指令优化的代码。 其中__RV_DSMALDA 是一条 Nuclei N3 P扩展指令，实现了 一次完成 4 笔 int16 相乘，最后累加，结果存放到 int64 变量中。

这些指令Intrinsic API可参考 Nuclei P 扩展指令Intrinsic API

具体的RVP指令手册，请联系芯来科技获取。

优化后的工程如下，可以与优化之前的工程做对比，只优化了cor_h_vec_012 算子:

优化后的工程下载链接

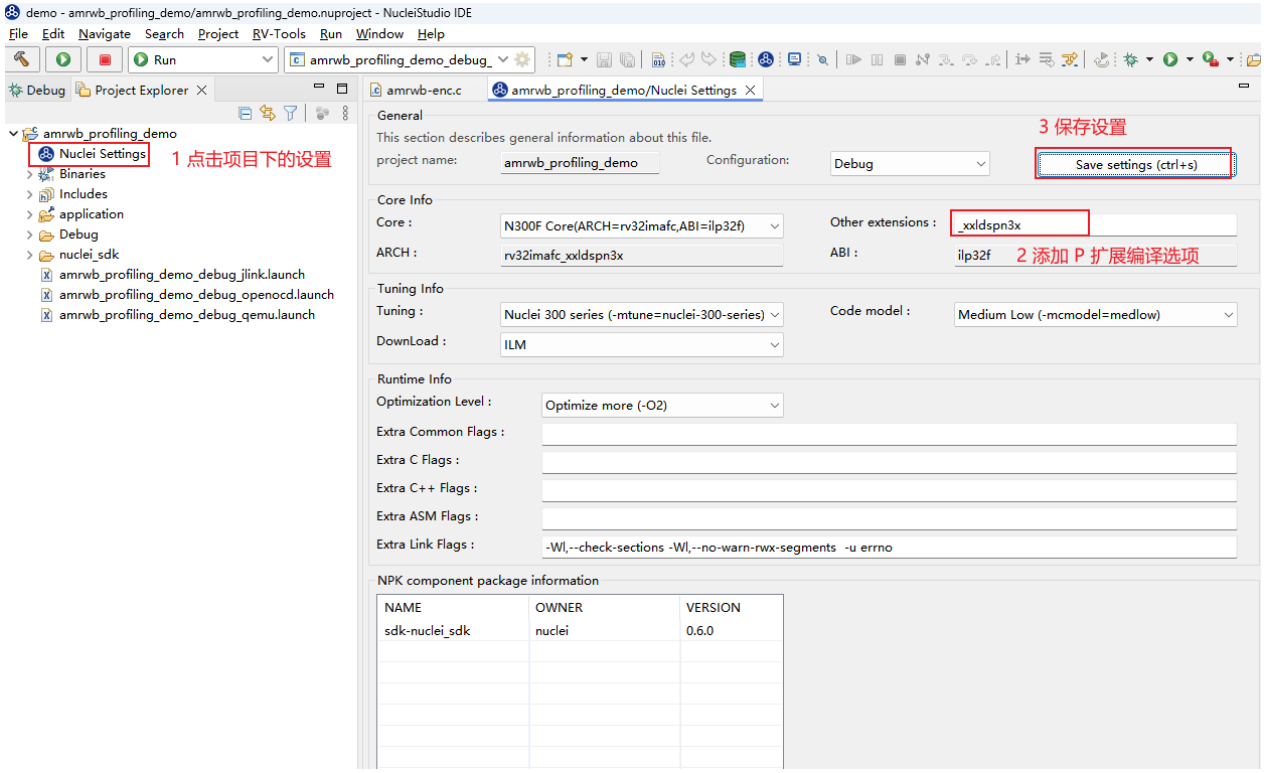
使用 Nuclei N3 P 扩展指令优化的代码片段如下：

```
void cor_h_vec_012(
        Word16
h[],                            /* (i) scaled impulse
response                    */
        Word16 vec[],                           /* (i) scaled vector
(/8) to correlate with h[] */
        Word16 track,                           /* (i) track to
use                             */
        Word16 sign[],                          /* (i) sign
vector                                  */
        Word16 rrixix[]
[NB_POS],               /* (i) correlation of h[x] with h[x]      */
        Word16 cor_1[],                         /* (o) result of
correlation (NB_POS elements) */
        Word16 cor_2[]                          /* (o) result of
correlation (NB_POS elements) */
        )
{
    Word32 i, j, pos, corr;
    Word16 *p0, *p1, *p2,*p3,*cor_x,*cor_y;
    Word32 L_sum1,L_sum2;
    cor_x = cor_1;
    cor_y = cor_2;
    p0 = rrixix[track];
    p3 = rrixix[track+1];
    pos = track;

    for (i = 0; i < NB_POS; i+=2)
    {
        p1 = h;
        p2 = &vec[pos];
#if defined __riscv_xxldspn3x
        Word32 tmp1, tmp2;
```

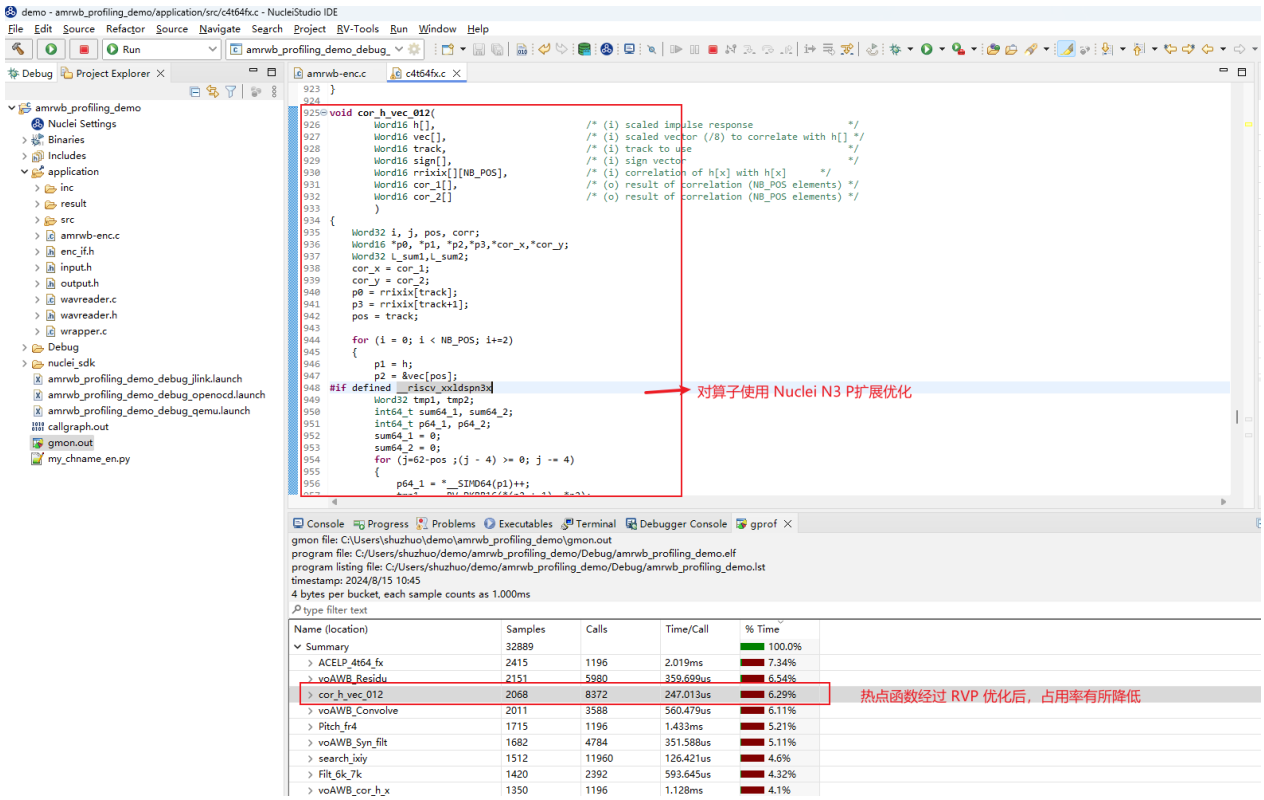
```
        int64_t sum64_1, sum64_2;
        int64_t p64_1, p64_2;
        sum64_1 = 0;
        sum64_2 = 0;
        for (j=62-pos ;(j - 4) >= 0; j -= 4)
        {
            p64_1 = *__SIMD64(p1)++;
            tmp1 = __RV_PKBB16(*(p2 + 1), *p2);
            tmp2 = __RV_PKBB16(*(p2 + 3), *(p2 + 2));
            p64_2 = __RV_DPACK32(tmp2, tmp1);
            sum64_1 = __RV_DSMALDA(sum64_1, p64_1, p64_2);

            tmp1 = __RV_PKBB16(*(p2 + 2), *(p2 + 1));
            tmp2 = __RV_PKBB16(*(p2 + 4), *(p2 + 3));
            p64_2 = __RV_DPACK32(tmp2, tmp1);
            sum64_2 = __RV_DSMALDA(sum64_2, p64_1, p64_2);
            p2 += 4;
        }
        L_sum1 = (Word32)sum64_1;
        L_sum2 = (Word32)sum64_2;
        for ( ;j >= 0; j--)
        {
            L_sum1 += *p1 * *p2++;
            L_sum2 += *p1++ * *p2;
        }
#endif
        L_sum1 += *p1 * *p2;
        L_sum1 = (L_sum1 << 2);
        L_sum2 = (L_sum2 << 2);

        corr = (L_sum1 + 0x8000) >> 16;
        cor_x[i] = vo_mult(corr, sign[pos]) + (*p0++);
        corr = (L_sum2 + 0x8000) >> 16;
        cor_y[i] = vo_mult(corr, sign[pos + 1]) + (*p3++);
        pos += STEP;

        p1 = h;
        p2 = &vec[pos];
#if defined __riscv_xxldspn3x
        sum64_1 = 0;
        sum64_2 = 0;
        for (j=62-pos ;(j - 4) >= 0; j -= 4)
        {
            p64_1 = *__SIMD64(p1)++;
            tmp1 = __RV_PKBB16(*(p2 + 1), *p2);
            tmp2 = __RV_PKBB16(*(p2 + 3), *(p2 + 2));
            p64_2 = __RV_DPACK32(tmp2, tmp1);
```

```
            sum64_1 = __RV_DSMALDA(sum64_1, p64_1, p64_2);

            tmp1 = __RV_PKBB16(*(p2 + 2), *(p2 + 1));
            tmp2 = __RV_PKBB16(*(p2 + 4), *(p2 + 3));
            p64_2 = __RV_DPACK32(tmp2, tmp1);
            sum64_2 = __RV_DSMALDA(sum64_2, p64_1, p64_2);
            p2 += 4;
        }
        L_sum1 = (Word32)sum64_1;
        L_sum2 = (Word32)sum64_2;
        for ( ;j >= 0; j--)
        {
            L_sum1 += *p1 * *p2++;
            L_sum2 += *p1++ * *p2;
        }
#endif
        L_sum1 += *p1 * *p2;
        L_sum1 = (L_sum1 << 2);
        L_sum2 = (L_sum2 << 2);

        corr = (L_sum1 + 0x8000) >> 16;
        cor_x[i+1] = vo_mult(corr, sign[pos]) + (*p0++);
        corr = (L_sum2 + 0x8000) >> 16;
        cor_y[i+1] = vo_mult(corr, sign[pos + 1]) + (*p3++);
        pos += STEP;
    }
    return;
}
```

这个算子进行 P 扩展优化后，**编译时务必带上 dsp 扩展选项进行编译**，如下图所示：



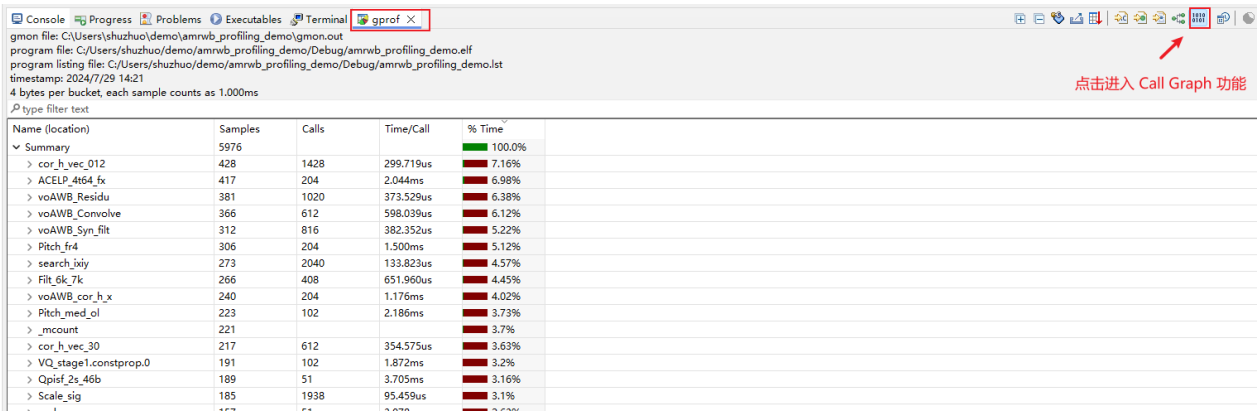
CLean Project 并重新编译，重新跑一次profiling，可以看到优化效果，`cor_h_vec_012` 函数占用率有所下降，函数调用时间也有所减少。



注意： 上述仅提供简单的示例，用户可以依次对热点函数进行分析并优化，运行过程中由于采样等原因， 导致 TOP 函数分布有所波动，这是正常的，最终精确的分析需要统计最终的总 cycle 数，然后计算提升比。

2 Call Graph 功能¶

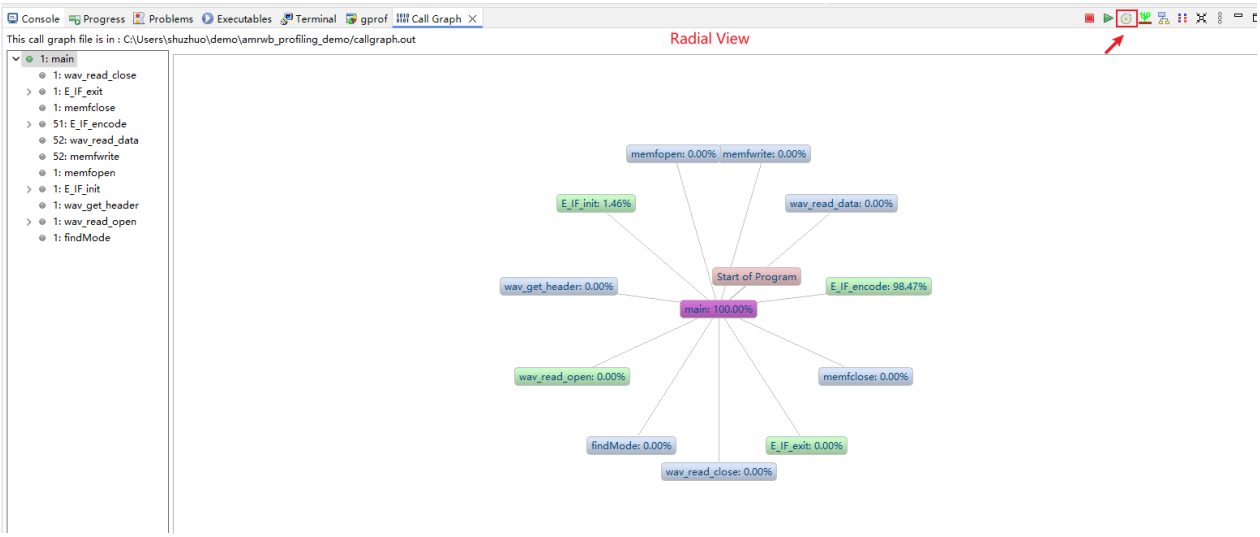
Nuclei Studio 中 Call Graph 主要是通过分析 Profiling 的数据来获取到程序中函数的调用关系。



Call Graph 功能包括如下几种视图：

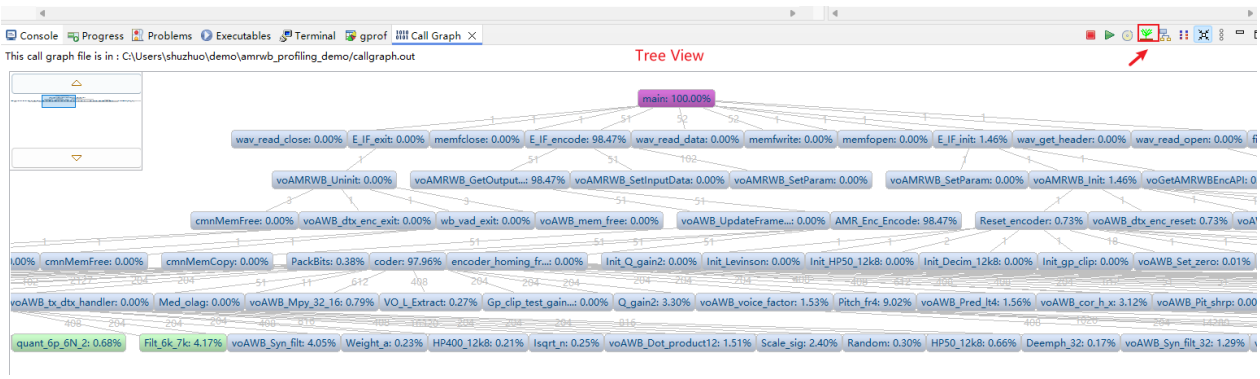
- Radial View

本视图中展示了程序的调用关系。



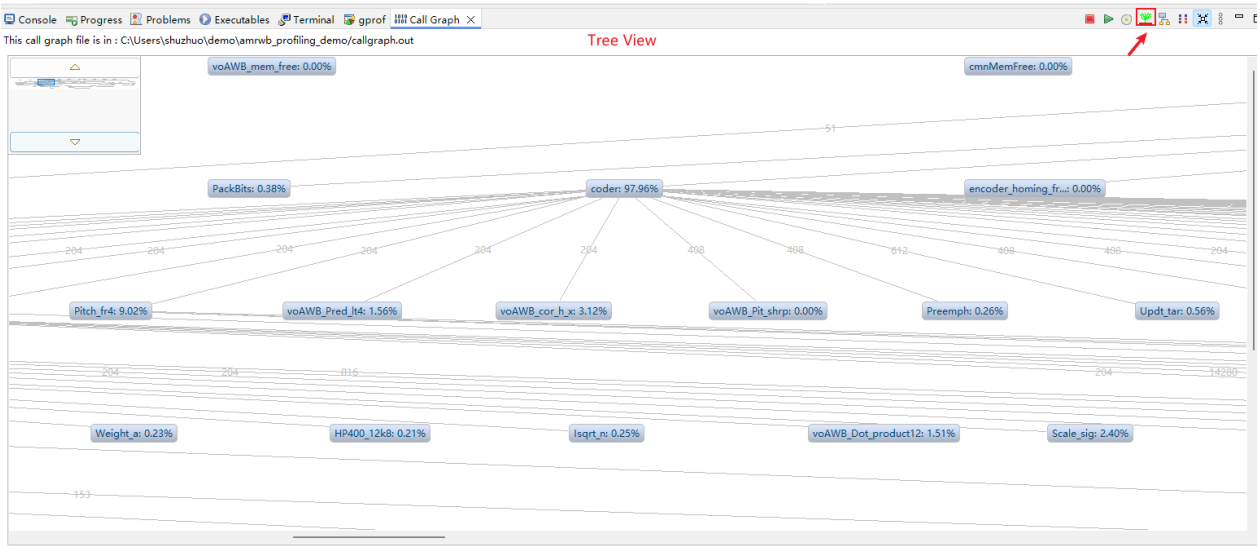
- Tree View

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



- Level View

与 Tree View 有点类似，展示了程序的调用关系以及调用次数。



- Aggregate View

以方图的方式，非常直观的展示了程序的耗时关系。

3 Code coverage 功能¶

Nuclei studio 中 Code coverage 功能基于 gcc 编译器提供的 gcov 工具，编译时需带特定的编译选项 `-coverage` 来编译指定源码文件，编译成功后得到 ELF 文件，然后在实际开发板上运行并收集需要的 coverage 文件(gcda/gcno 文件)，最终在 IDE 上以图形化的方式展示。

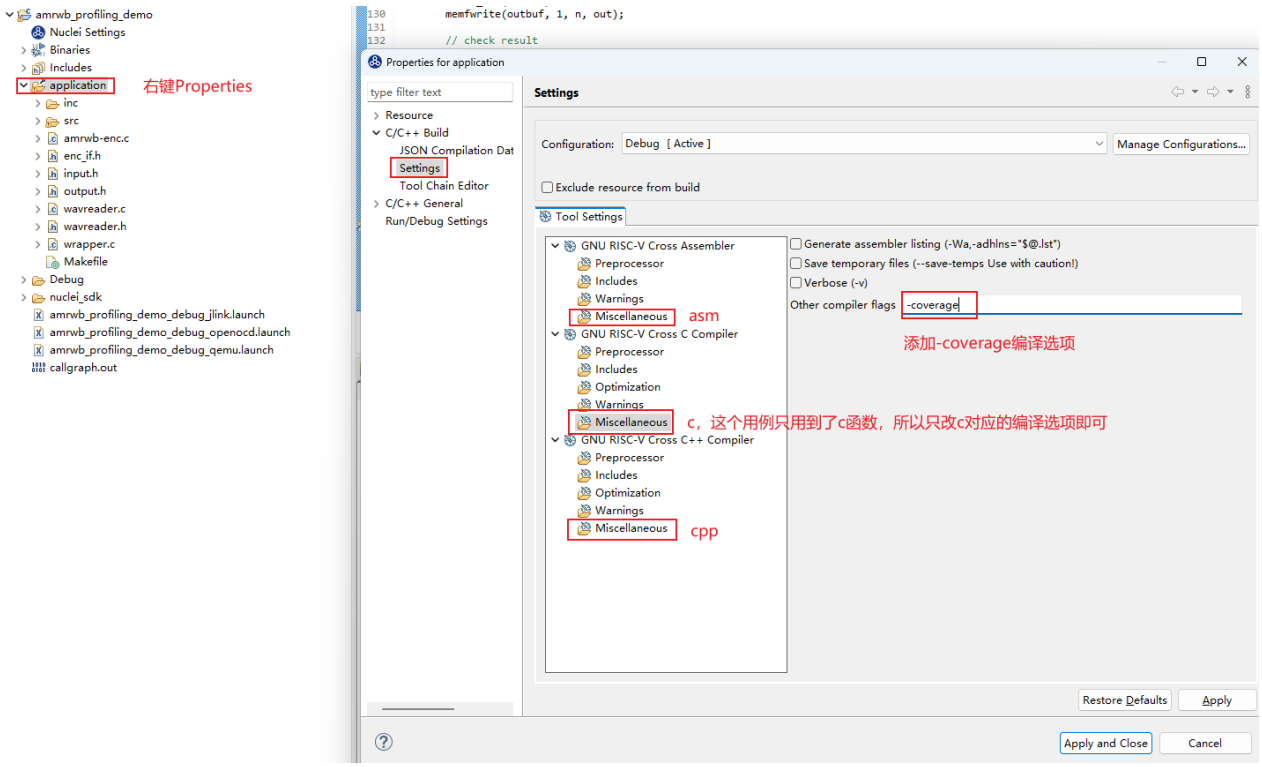
使用方法与 Profiling 功能类似，这里仅对不同的地方进行说明：

step1：新建 Profiling demo 工程
step2：基于 Profiling demo 工程移植 amrwbenc 裸机用例
step3：添加 gcov 数据收集代码，并添加 -coverage 编译选项，重新编译代码

在main函数的结尾处添加gprof数据收集代码：

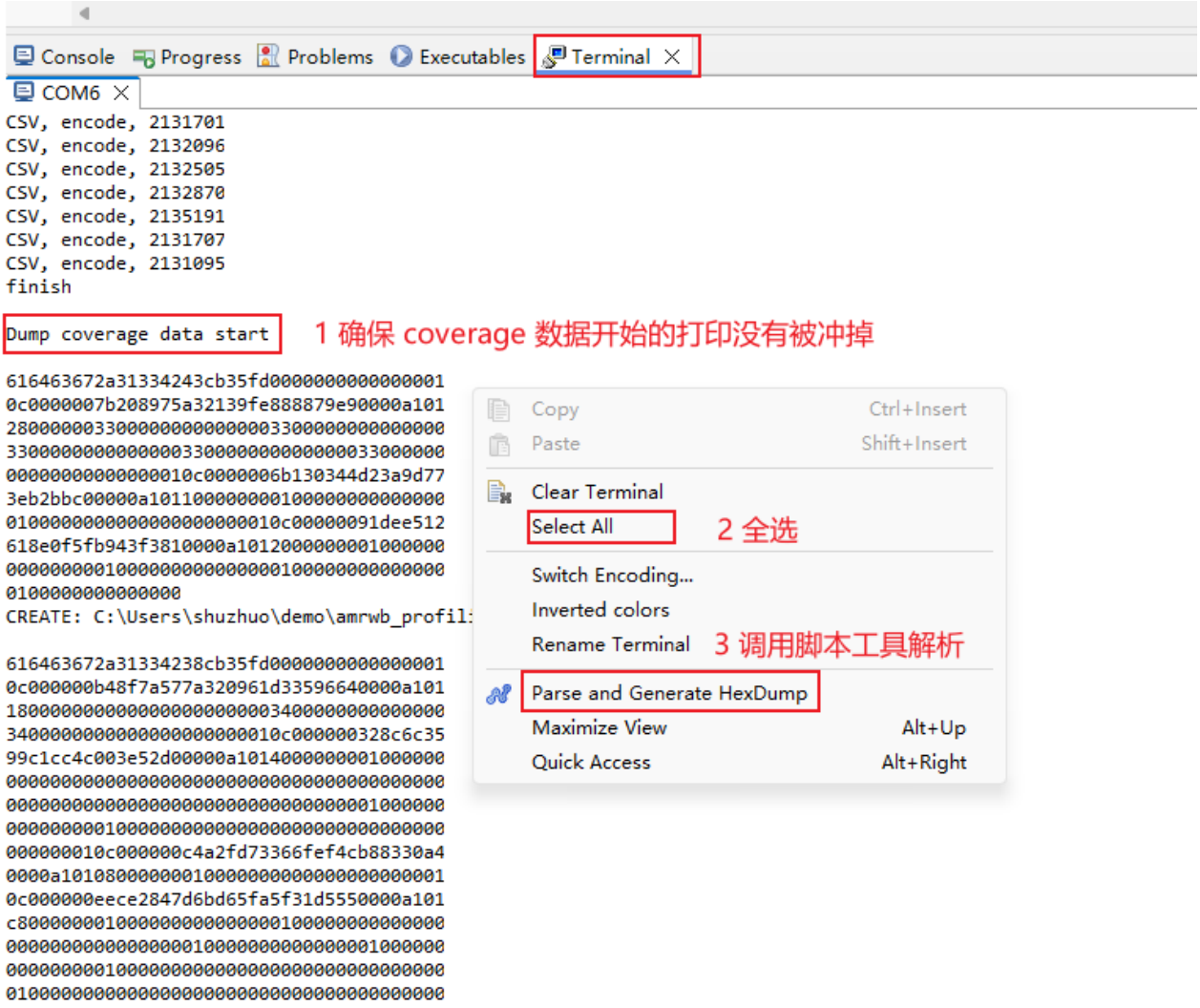
```
int main(int argc, char *argv[]) {
    /*
     * 代码省略
     */

    /*
     * 在main函数的结尾处添加 gcov 数据收集代码
     */
    // TODO this is used for collect gprof and gcov data
    // See Components/profiling/README.md about how to set the IDE
project properities
    extern long gcov_collect(unsigned long interface);
    gcov_collect(2);

    return 0;
}
```

添加`-coverage`编译选项，重新编译代码：

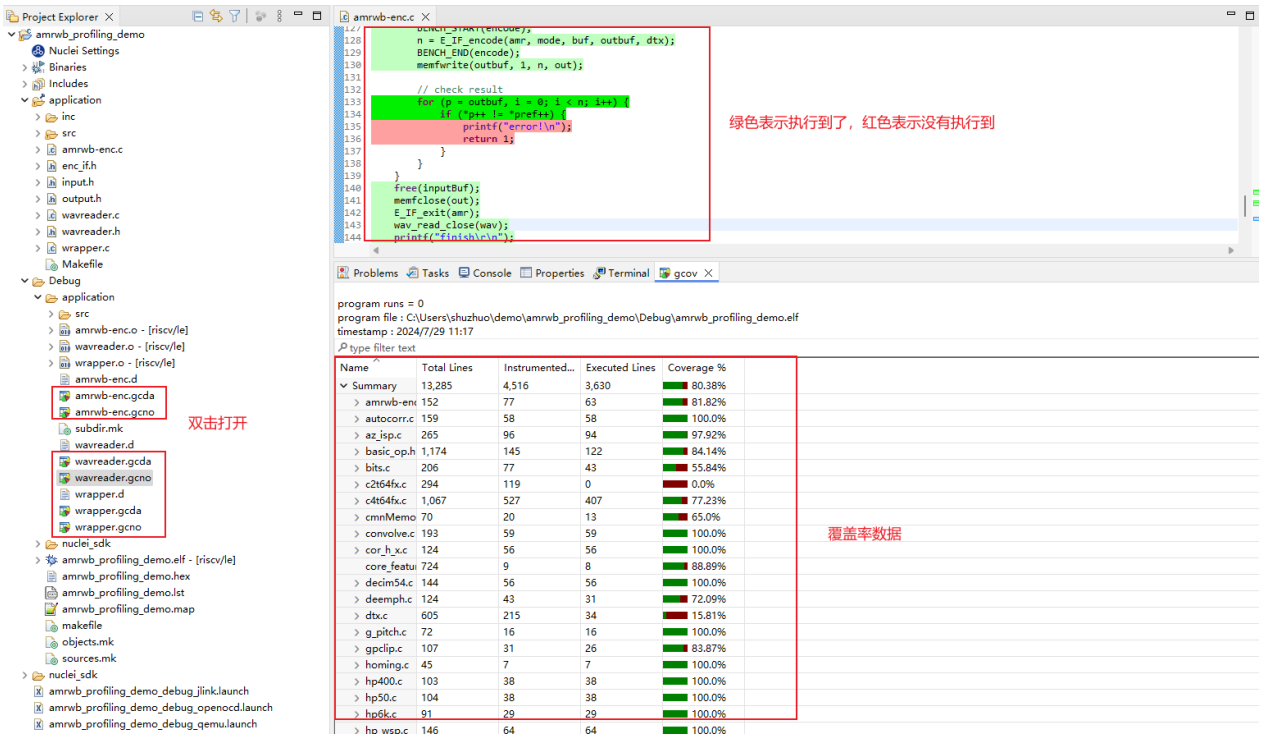


step4：运行程序

可以在qemu中模拟运行，或者上板实际运行都可以（统计覆盖率，不涉及到性能分析，所以使用qemu 或者上板测试都可以）。



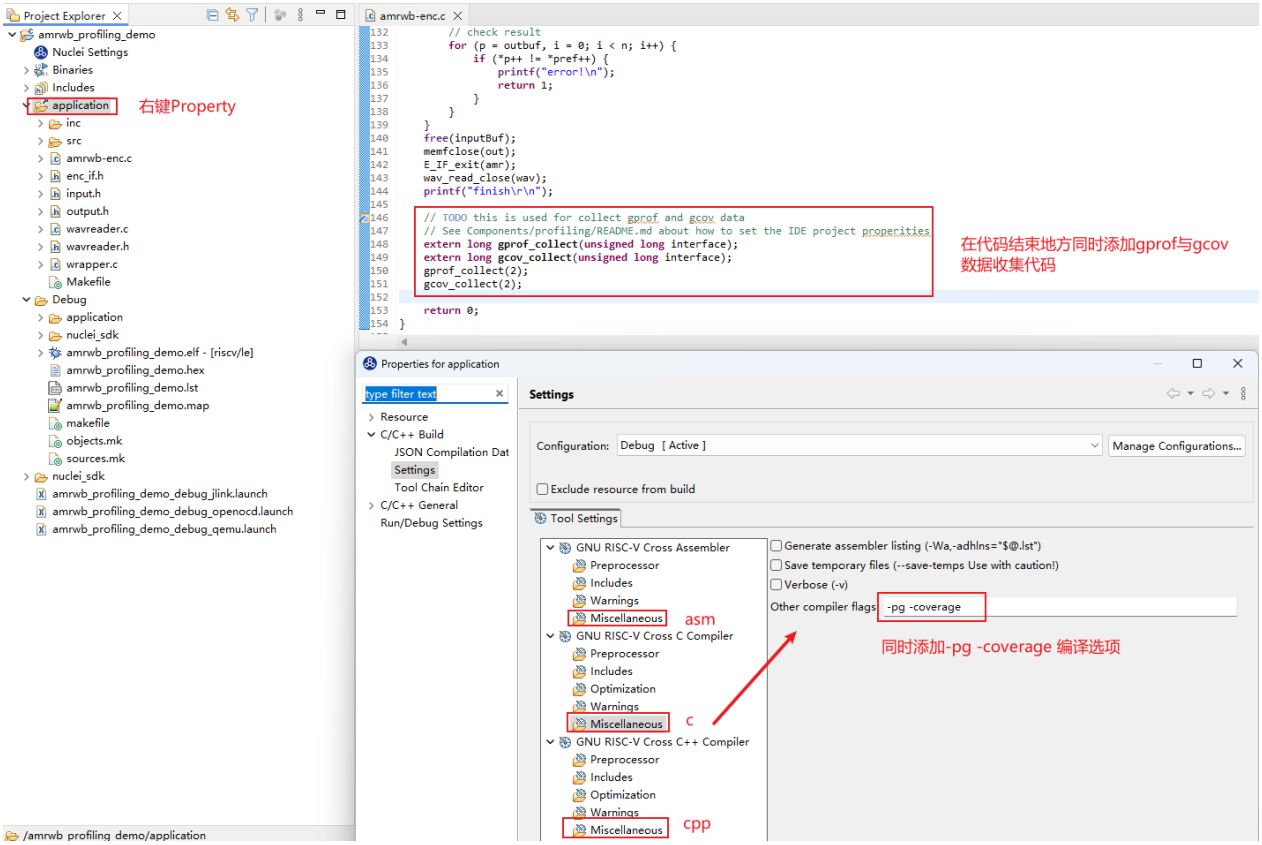
解析之后，在Debug->application文件夹下生成了 gcda 与 gcno 文件，双击打开即可



4 补充¶

1. Profiling 与 Code coverage 功能可以同时打开，只需添加一起收集 Profiling 数据与 Code coverage 数据的代码，并在编译时添加 `-pg -coverage` 编译选项。

```
    // TODO this is used for collect gprof and gcov data
    // See Components/profiling/README.md about how to set the IDE
project properities
    extern long gprof_collect(unsigned long interface);
    extern long gcov_collect(unsigned long interface);
    gprof_collect(2);
    gcov_collect(2);
```



1. 使用Profiling可能遇见的问题：

2. 片上内存不足，打印日志中有错误打印，gprof/gcov data 需要占用一定大小空间

3. Console 或 Terminal 收集的数据不全导致解析数据不正确，需确认数据没有被冲掉，需要调节 Console 或 Terminal 输出大小限制
4. 手动删掉 gmon.out 文件，再次解析，弹出 No files have been generated 错误弹框

上述具体解决方法可参考 Profiling与 Code coverage 功能可能遇到的问题

通过Profiling展示Nuclei Model NICE/VNICE指令加速¶

> 由于 Nuclei Model 仅支持Linux版本，所以此文档的测试都是基于 Nuclei Studio 的 Linux版本 (>= 2024.06) 完成的。

背景描述¶

Nuclei Model Profiling¶

在Nuclei Studio使用Profiling功能进行性能调优举例中已经通过 qemu 以及上板测试两种运行方式展示了 如何在IDE中导入特定程序进行 Profiling，此文档中的一部分将介绍如何针对 Nuclei Model 完成 Profiling。

Nuclei Model Profiling 的优势:

- 无需使用开发板等硬件
- model 中内建了 gprof 功能，无需 Profiling 库和 `gcc -pg` 选项就可以产生 Profiling 文件
- 采取了指令级别的采样，可以进行指令级别的 Profiling 分析

在NucleiStudio_User_Guide.pdf相关章节对 Nuclei Model 如何仿真性能分析配置已经有较详细的描述，此文档以一个例子来展示其实际应用。

NICE/VNICE 自定义指令加速¶

NICE/VNICE使得用户可以结合自己的应用扩展自定义指令，将芯来的标准处理器核扩展成为面向领域专用的处理器，NICE 具体编码规则可以参考 `Nuclei_RISC-V_ISA_Spec.pdf` 中的 `NICE Introduction`。NICE 适用于无需使用 RISCV Vector 的自定义指令，VNICE 适用于需要使用 RISCV Vector 的自定义指令。

demo_nice/demo_vnice介绍了 Nuclei 针对 NICE/VNICE 的 demo 应用 是如何编译运行的，此文档将通过改造一个更为常见的 AES 加解密的例子，重点说明该如何使用 NICE/VNICE 指令替换热点函数以及如何在 model 里实现 NICE/VNICE 指令，然后通过 Nuclei Studio 的 Profiling 功能分析替换前后的程序性能。

解决方案¶

环境准备¶

Nuclei Studio：NucleiStudio 2024.06 Linux

Model Profiling¶

工程创建方式有两种：

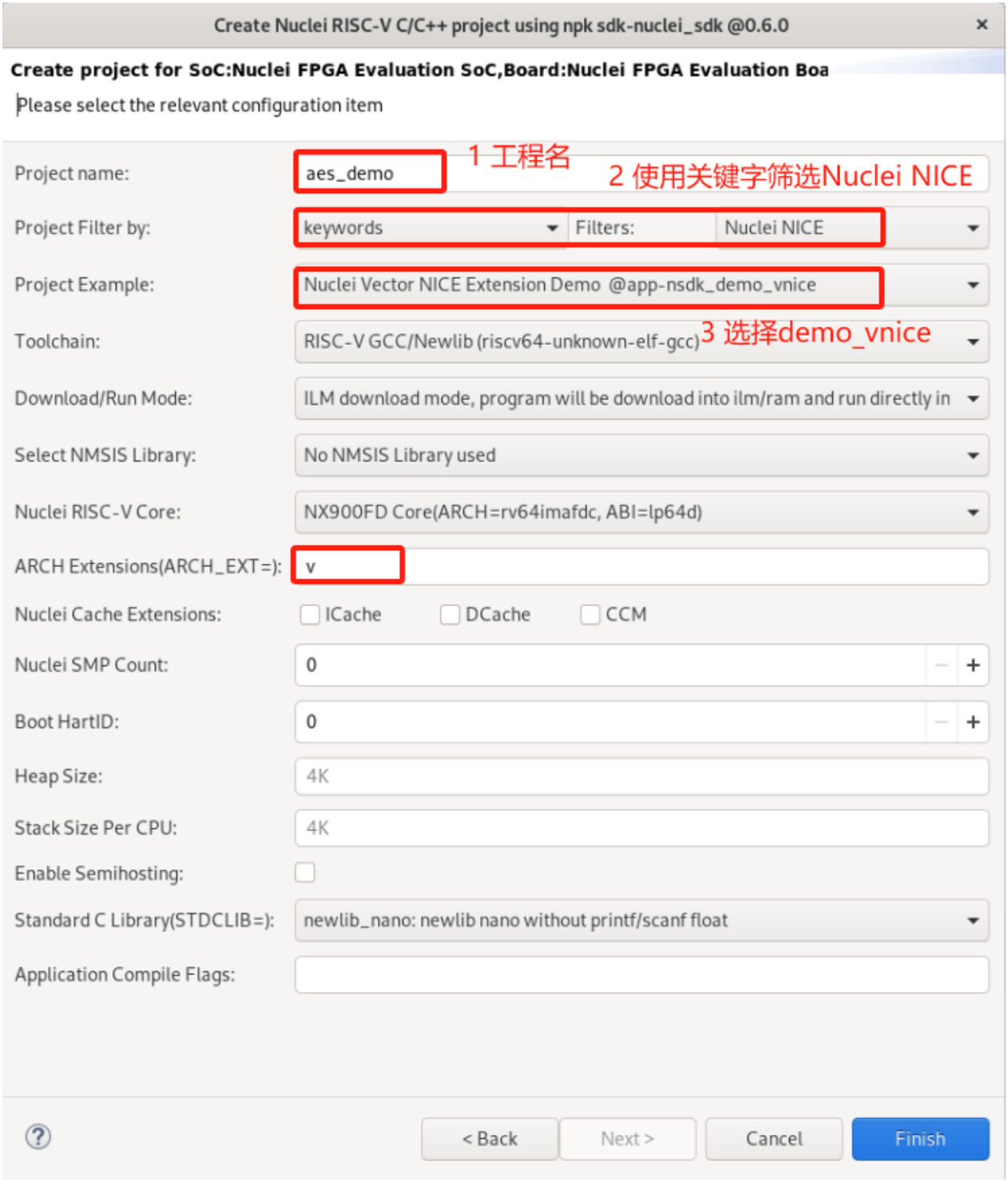
- 方式1：用户可以使用 Nuclei Studio 中的 `demo_nice` 或 `demo_vnice` 模板来移植改造自己的 NICE/VNICE 程序
- 方式2：用户导入自己的工程到 Nuclei Studio 中，然后再添加NICE 内嵌汇编头文件、NICE CSR 使能等代码

此文档将采取前一种方式创建工程，由于此 demo 会用到 VNICE 指令，故创建 `demo_vnice` 工程，然后将 AES 加解密程序移植替换到其中。

step1：新建 demo_vnice 工程¶

File->New->New Nuclei RISC-V C/C++ Project，选择Nuclei FPGA Evalution Board->sdk-nuclei_sdk @0.6.0

注意： Nuclei SDK 需选择 0.6.0 及以后版本

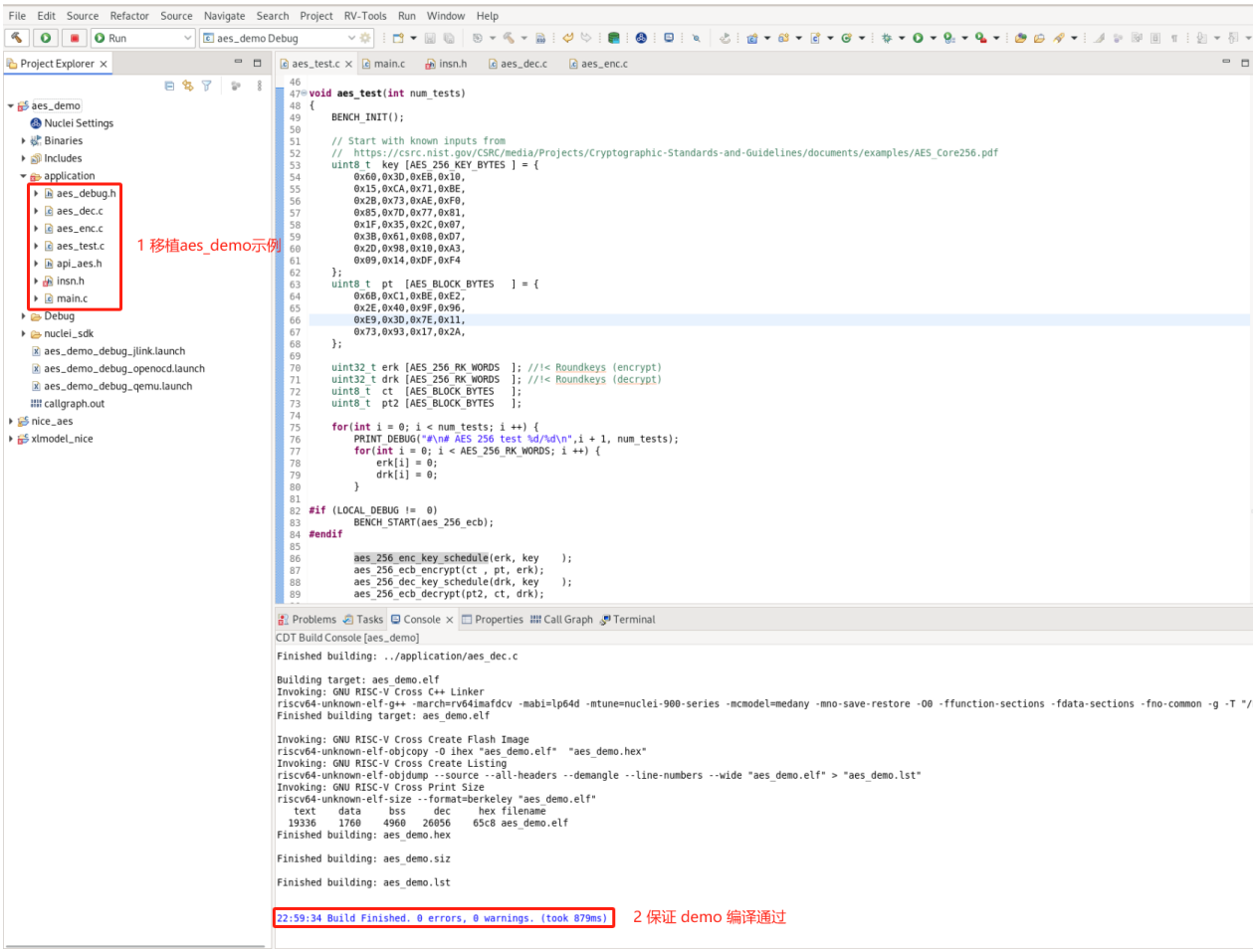


step2：基于 demo_vnice 工程移植 aes_demo 裸机用例¶

移植 aes_demo 时，需要保留 `demo_vnice` 中的 `insn.h` 内嵌汇编头文件框架，方便后续添加自定义的 NICE/VNICE 指令，在 `main.c` 中需要保留 NICE/VNICE 指令执行前的 CSR 使能代码：

```
__RV_CSR_SET(CSR_MSTATUS, MSTATUS_XS);
```

其余 demo_vnice 工程中 application 原始用例可删除，替换成 aes_demo 用例，形成如下目录结构，并确保能够编译通过。



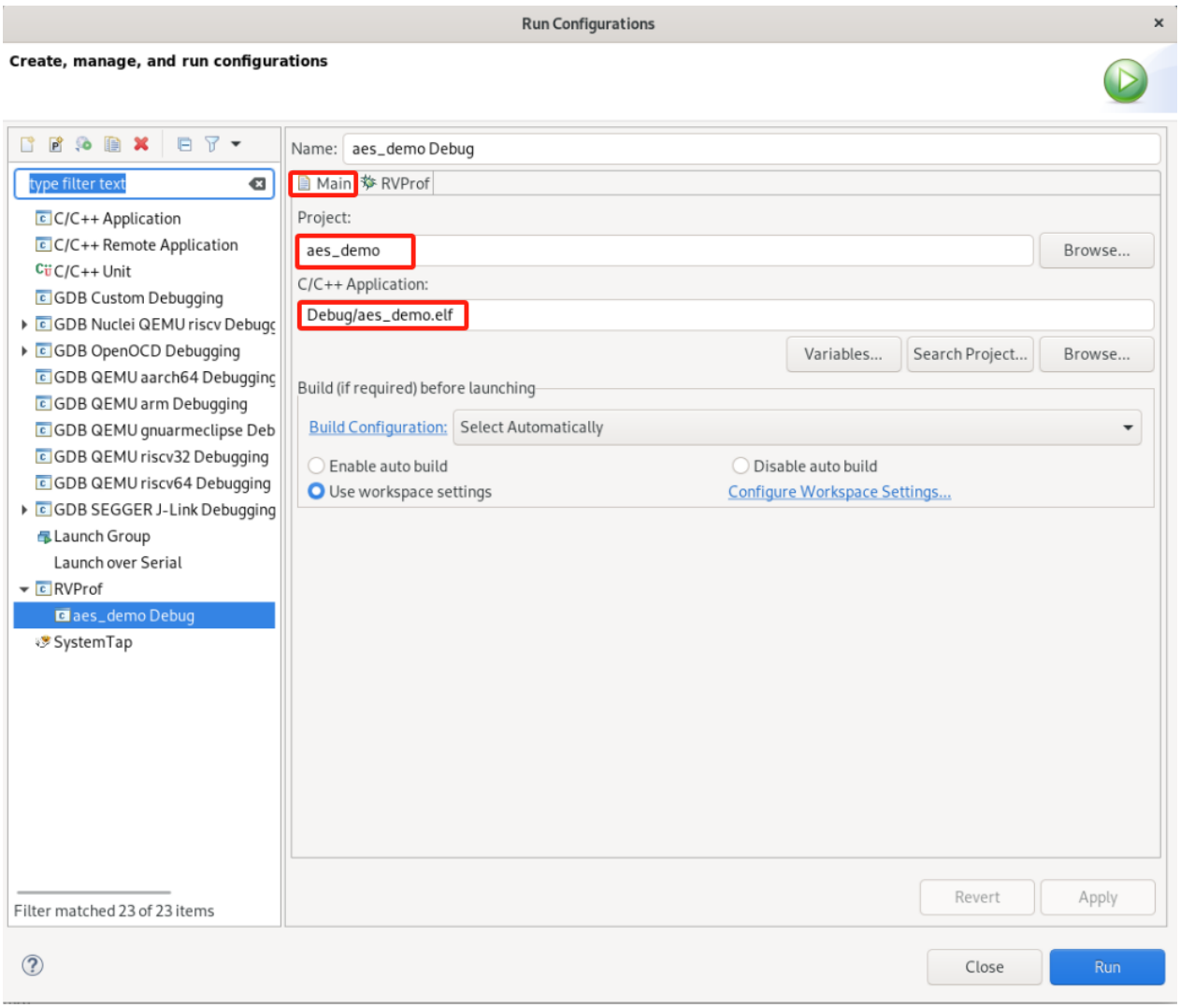
用户可以下载我们移植好的 AES 加解密 demo：优化前AES工程链接下载

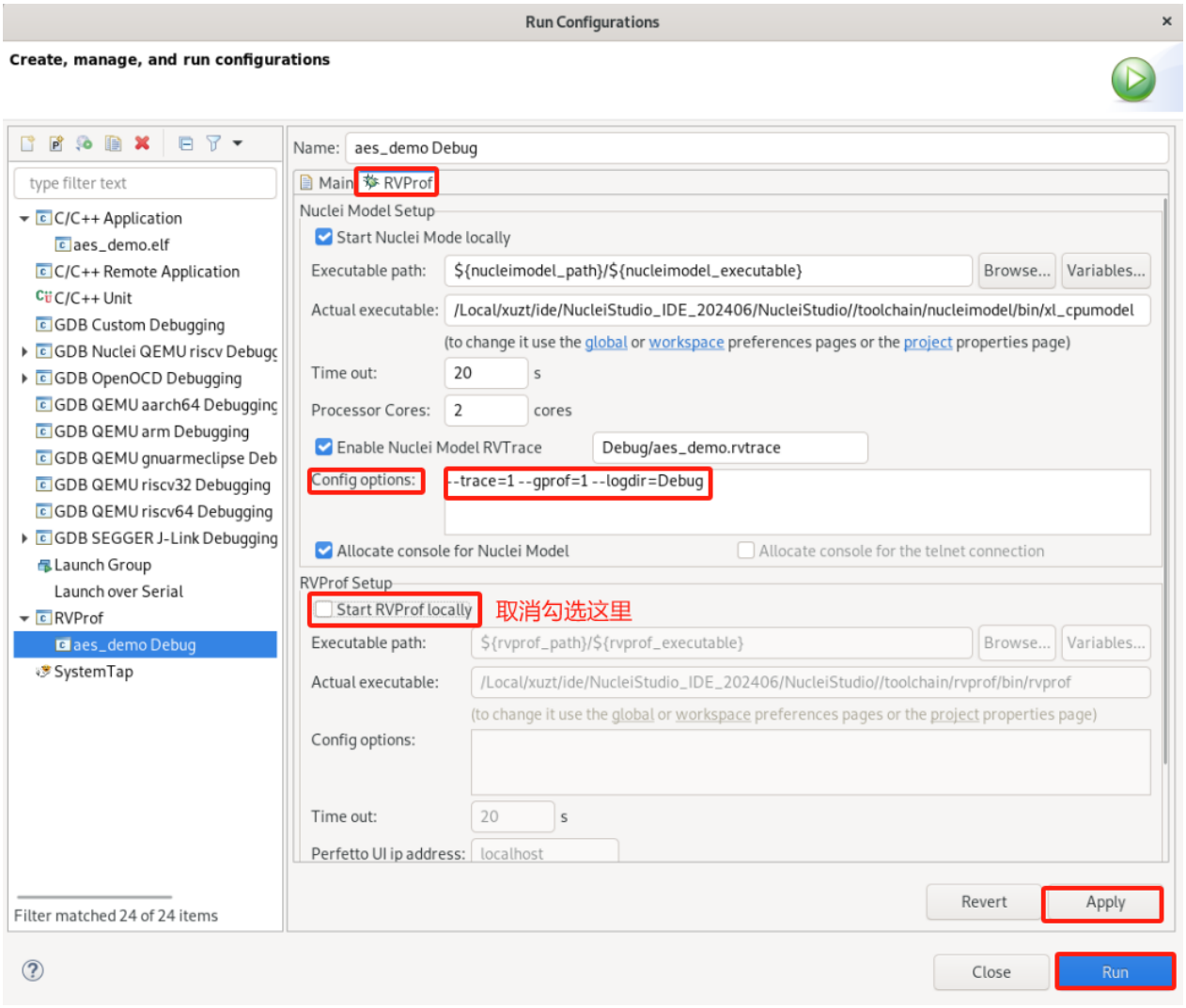
下载 zip 包后，可以直接导入到 Nuclei Studio 中运行(导入步骤：File->Import->Existing Projects into Workspace->Next->Select archive file->选择zip压缩包->next即可)

step3：model 仿真程序¶

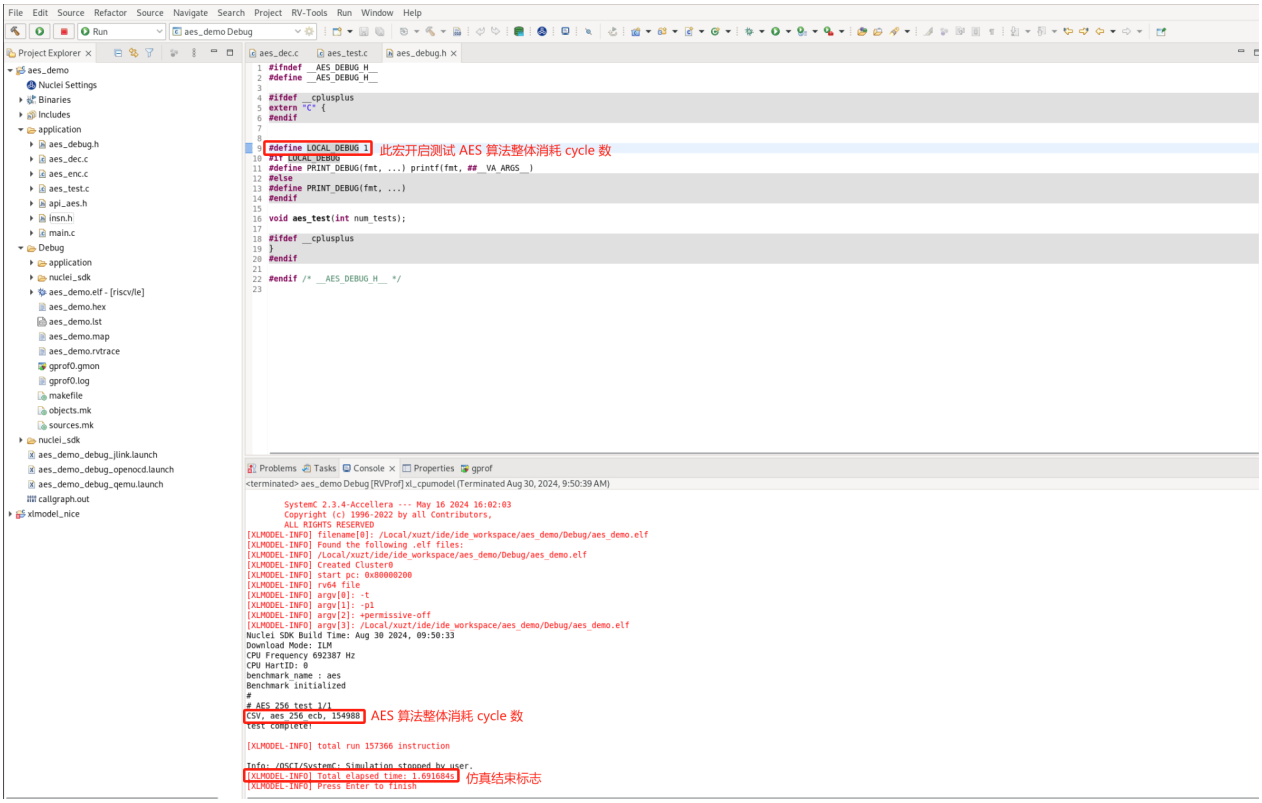
首先将 aes_debug.h 中的 LOCAL_DEBUG 打开，准备测试 AES 算法的整体 cycle 数。

Nuclei Model 仿真程序需要配置 Nuclei Studio 中的 RVProf 运行配置，打开 Nuclei Studio 主菜单栏的 Run 选项的 Run Configurations 后，先在 Main 选项卡中选择编译好的 elf 文件路径，然后在 RVProf 选项卡 的Config options 中完成 model 运行配置 --trace=1 --gprof=1 --logdir=Debug，--trace=1 表示开启 rvtrace，--gprof=1 表示开启 gprof 功能生成 *.gmon 文件，--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，取消勾选 Start RVProf locally，然后点击 Apply 和 Run，model 就开始运行程序了。

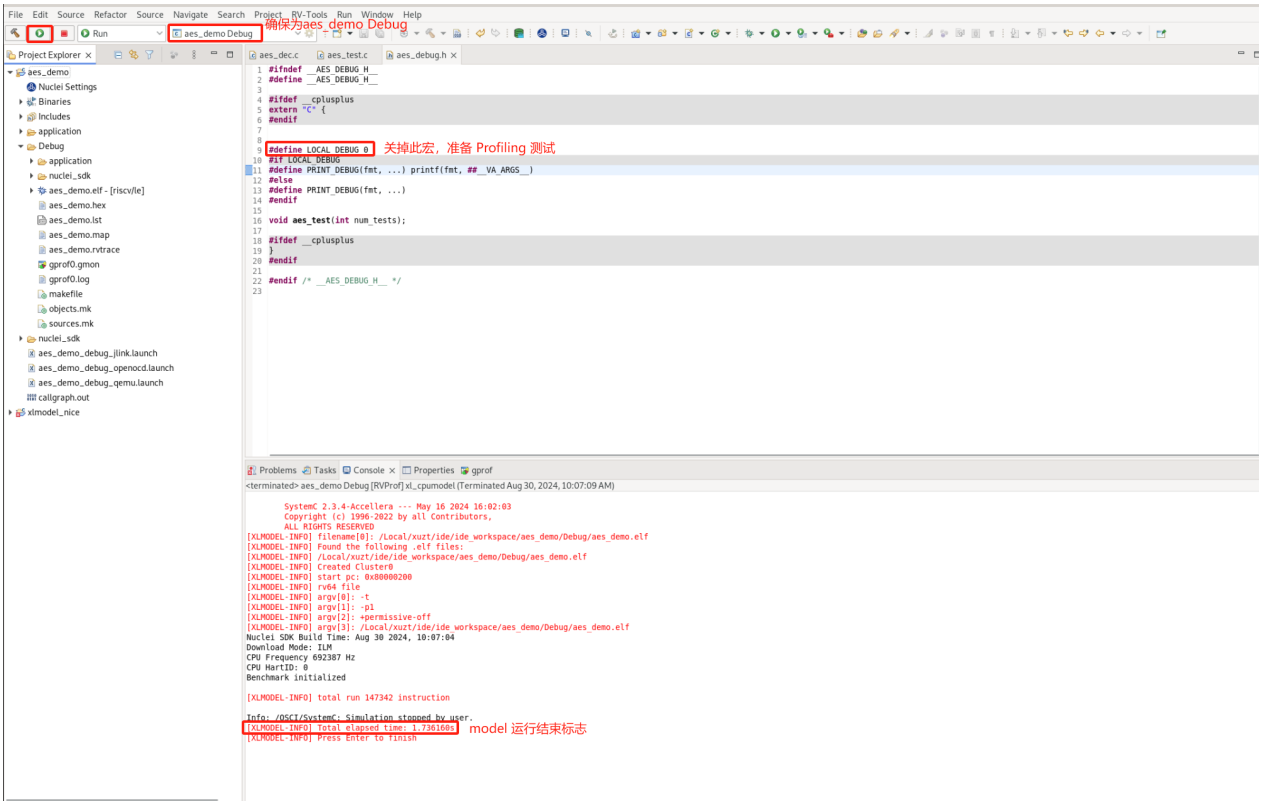




在 Console 中会看到 `Total elapsed time` 说明 model 已经完成仿真了，得到 AES 算法整体消耗 154988 cycle。



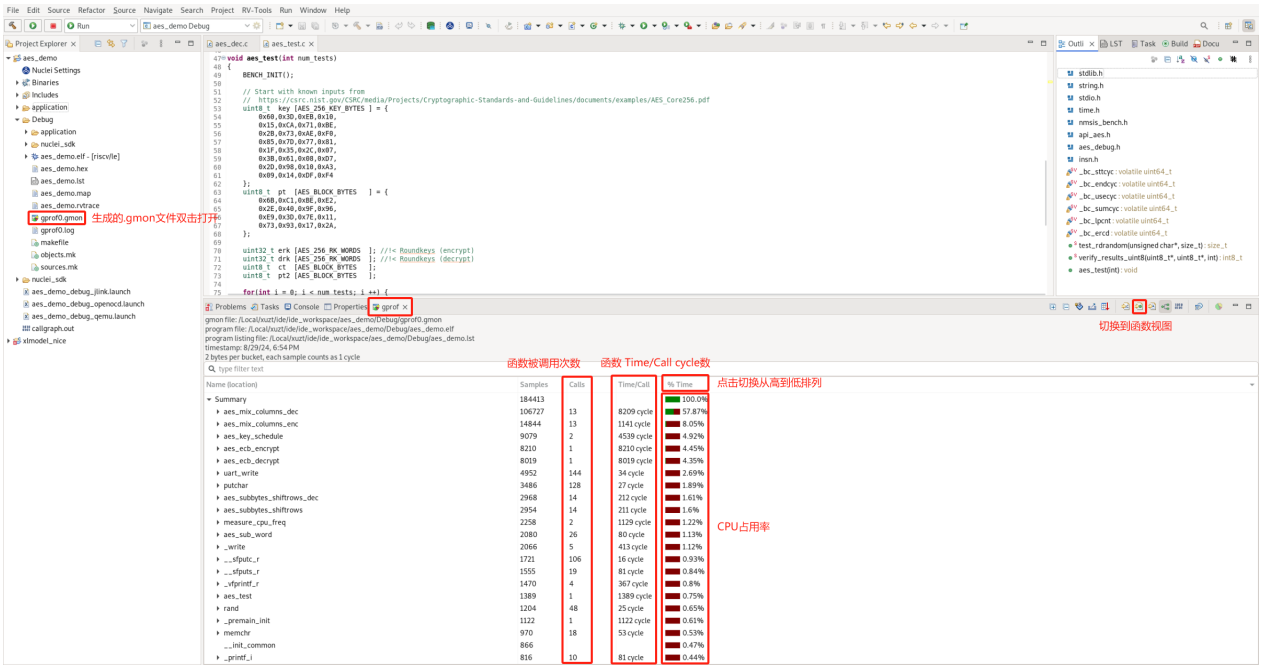
将 aes_debug.h 中的 LOCAL_DEBUG 关掉去掉程序打印，为了准确测试 Profiling 数据，确保 Nuclei Studio 的 launch bar 为 aes_demo Debug, 重新 Run model：



step4：解析 gprof 数据¶

model 仿真程序完成后，双击打开生成的 `gprof*.gmon` 文件，切换到函数视图，点击 `% Time` 从高到低排列函数 CPU 占用率。

注意： `Time/Call` 显示的是每个函数的函数体 text 段的 cycle 数，并不是整个函数的 cycle 数，是不计入其中子函数占用的 cycle 数的。



从而得到 CPU 占用率最高的 TOP5 热点函数为：

```
aes_mix_columns_dec
aes_mix_columns_enc
aes_key_schedule
aes_ecb_decrypt
aes_ecb_encrypt
```

注意： 此时需要备份当前的 `aes_demo` 工程，改名为 `aes_demo_nice` 工程，这样可以在 Nuclei Studio 中同时打开两个工程，方便添加 NICE/VNICE 指令优化后的工程和原 `aes_demo` 工程进行 Profiling 比较。

step5：NICE/VNICE 指令替换¶

用户需要在备份的 `aes_demo_nice` 工程下，研究热点函数算法特点，将其替换为 NICE/VNICE 指令，从而提升整体程序性能。

在包含 AES 加解密的 TOP5 热点函数的 `aes_dec.c` 和 `aes_dec.c` 两个C文件中 `#include "insn.h"` 以便添加 NICE/VNICE 指令替换。

TOP1 热点函数为 aes_mix_columns_dec，实现了 AES 算法解密的逆混合列，输入一个状态矩阵，经过计算后原地址输出一个计算后的状态矩阵，实现了 Load 数据、逆混合运算以及 Store 数据，代码如下：

```
static void aes_mix_columns_dec(
    uint8_t     pt[16]       //!< Current block state
){
    // Col 0
    for(int i = 0; i < 4; i ++) {
        uint8_t b0,b1,b2,b3;
        uint8_t s0,s1,s2,s3;

        s0 = pt[4*i+0];
        s1 = pt[4*i+1];
        s2 = pt[4*i+2];
        s3 = pt[4*i+3];

        b0 = XTE(s0) ^ XTB(s1) ^ XTD(s2) ^ XT9(s3);
        b1 = XT9(s0) ^ XTE(s1) ^ XTB(s2) ^ XTD(s3);
        b2 = XTD(s0) ^ XT9(s1) ^ XTE(s2) ^ XTB(s3);
        b3 = XTB(s0) ^ XTD(s1) ^ XT9(s2) ^ XTE(s3);

        pt[4*i+0] = b0;
        pt[4*i+1] = b1;
        pt[4*i+2] = b2;
        pt[4*i+3] = b3;
    }
}
```

由于输入输出地址一样，可以考虑用一条 NICE 指令替换，指令的 opcode、funct3 和 funct7 都可以在编码位域中自定义，该指令设置 opcode 为 Custom-0，funct3 设置为0，funct7 设置为0x10，寄存器只使用到 rs1 描述入参地址，不需要使用 rd 和 rs2，指令写到 insn.h 中，内嵌汇编如下：

```
__STATIC_FORCEINLINE void custom_aes_mix_columns_dec(uint8_t* addr)
{
    int zero = 0;
    asm volatile(".insn r 0xb, 0, 0x10, x0, %1, x0" : "=r"(zero) :
"r"(addr));
}
```

用户可以在 insn.h 中定义一个 USE_NICE 的宏选择是否使用 NICE ，在 aes_dec.c 改写 aes_mix_columns_dec 如下 ：

```
static void aes_mix_columns_dec(
    uint8_t     pt[16]       //!< Current block state
){

#ifdef USE_NICE
    custom_aes_mix_columns_dec(pt);
#else
    // Col 0
    for(int i = 0; i < 4; i ++) {
        uint8_t b0,b1,b2,b3;
        uint8_t s0,s1,s2,s3;

        s0 = pt[4*i+0];
        s1 = pt[4*i+1];
        s2 = pt[4*i+2];
        s3 = pt[4*i+3];

        b0 = XTE(s0) ^ XTB(s1) ^ XTD(s2) ^ XT9(s3);
        b1 = XT9(s0) ^ XTE(s1) ^ XTB(s2) ^ XTD(s3);
        b2 = XTD(s0) ^ XT9(s1) ^ XTE(s2) ^ XTB(s3);
        b3 = XTB(s0) ^ XTD(s1) ^ XT9(s2) ^ XTE(s3);

        pt[4*i+0] = b0;
        pt[4*i+1] = b1;
        pt[4*i+2] = b2;
        pt[4*i+3] = b3;
    }
#endif
}
```

TOP2 热点函数为 aes_mix_columns_enc，和 TOP1 类似，实现的是 AES 加密的逆混合列，同样也是输入一个状态矩阵，经过计算后原地址输出一个计算后的状态矩阵：

```
static void aes_mix_columns_enc(
    uint8_t     ct [16]       //!< Current block state
){
    for(int i = 0; i < 4; i ++) {
        uint8_t b0,b1,b2,b3;
        uint8_t s0,s1,s2,s3;

        s0 = ct[4*i+0];
        s1 = ct[4*i+1];
        s2 = ct[4*i+2];
        s3 = ct[4*i+3];
```

```
        b0 = XT2(s0) ^ XT3(s1) ^    (s2) ^    (s3);
        b1 =    (s0) ^ XT2(s1) ^ XT3(s2) ^    (s3);
        b2 =    (s0) ^    (s1) ^ XT2(s2) ^ XT3(s3);
        b3 = XT3(s0) ^    (s1) ^    (s2) ^ XT2(s3);

        ct[4*i+0] = b0;
        ct[4*i+1] = b1;
        ct[4*i+2] = b2;
        ct[4*i+3] = b3;
    }
}
```

考虑到指令实现可能无法只用1条指令完成，可使用2条 VNICE 指令替换此算法，第一条 load 16 byte 数据到 Vector 寄存器，第二条再完成计算以及 store。

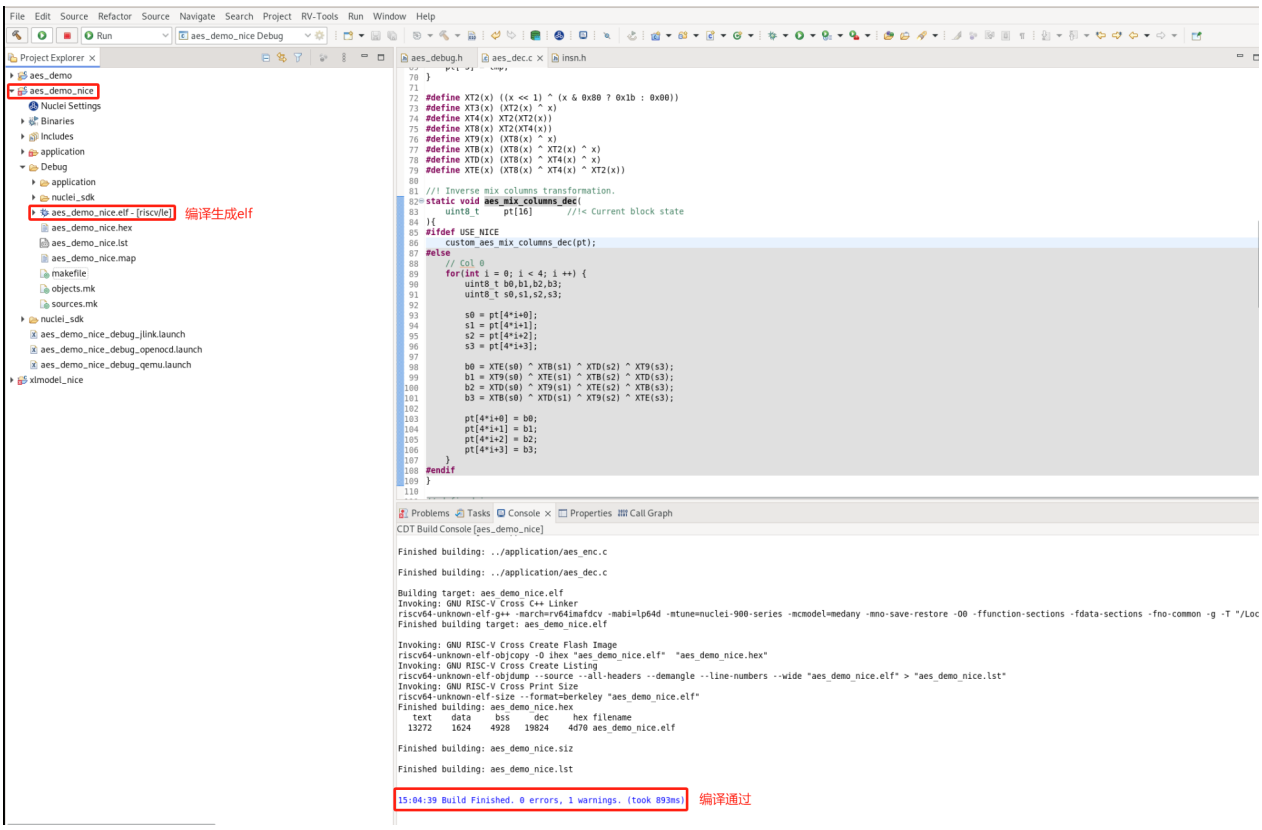
指令的 `opcode`、`funct3` 和 `funct7` 仍然可以在编码位域中自定义，第一条指令使用 `rd` 描述 Vector 寄存器，`rs1` 描述入参地址，第二条指令使用 `rs1` 描述入参地址，`rs1` 描述入参 Vector 寄存器，两条 VNICE 指令的内嵌汇编写到 `insn.h` 中，定义如下：

```
__STATIC_FORCEINLINE vint8m1_t __custom_vnice_load_v_i8m1
(uint8_t* addr)
{
    vint8m1_t rdata ;
    asm volatile(".insn r 0xb,4,0,%0,%1,x0"
            : "=vr"(rdata)
            : "r"(addr)
            );
    return rdata;
}

__STATIC_FORCEINLINE void __custom_vnice_aes_mix_columns_enc_i8m1
(uint8_t *addr, vint8m1_t data)
{
    int zero = 0;
    asm volatile(".insn r 0xb,4,1,x0,%1,%2"
            : "=r"(zero)
            : "r"(addr)
            , "vr"(data)
            );
}
```

用户通过定义 Vector 寄存器以及使用上定义好的 VNICE 指令内嵌汇编改写 `aes_enc.c` 中的 `aes_mix_columns_enc` 如下：

```
static void aes_mix_columns_enc(
    uint8_t     ct [16]       //!< Current block state
){
#ifdef USE_NICE
    uint32_t blkCnt = 16;
    size_t l;
    vint8m1_t vin;
    for (; (l = __riscv_vsetvl_e8m1(blkCnt)) > 0; blkCnt -= l) {
        vin = __custom_vnice_load_v_i8m1(ct);
        __custom_vnice_aes_mix_columns_enc_i8m1(ct, vin);
    }
#else
    for(int i = 0; i < 4; i ++) {
        uint8_t b0,b1,b2,b3;
        uint8_t s0,s1,s2,s3;

        s0 = ct[4*i+0];
        s1 = ct[4*i+1];
        s2 = ct[4*i+2];
        s3 = ct[4*i+3];

        b0 = XT2(s0) ^ XT3(s1) ^    (s2) ^    (s3);
        b1 =    (s0) ^ XT2(s1) ^ XT3(s2) ^    (s3);
        b2 =    (s0) ^    (s1) ^ XT2(s2) ^ XT3(s3);
        b3 = XT3(s0) ^    (s1) ^    (s2) ^ XT2(s3);

        ct[4*i+0] = b0;
        ct[4*i+1] = b1;
        ct[4*i+2] = b2;
        ct[4*i+3] = b3;
    }
#endif
}
```

修改后的程序代码编译通过：(aes_demo_nice 工程)

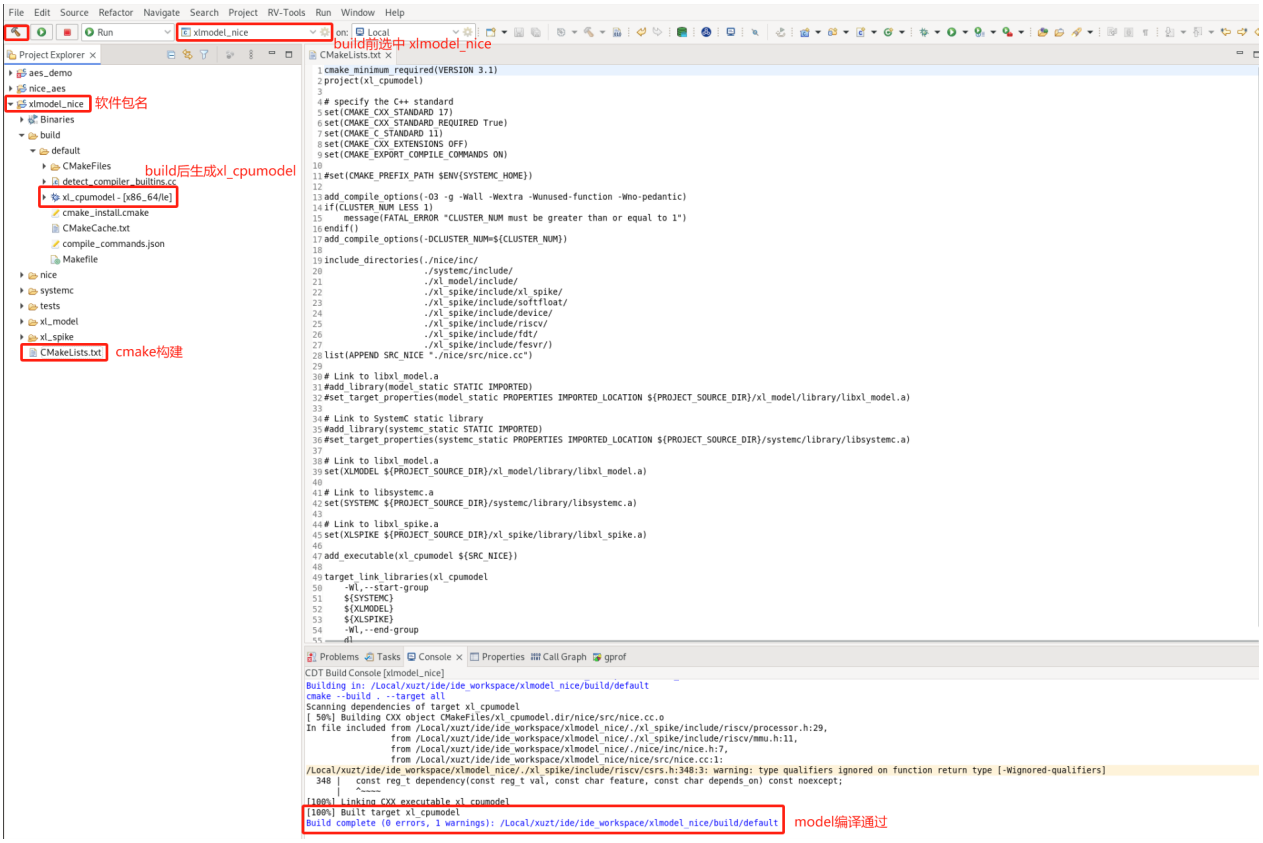


step6：在 Nuclei Model 中实现 NICE/VNICE 指令¶

首先需要下载支持用户配置自定义 NICE/VNICE 指令的原始 Nuclei Model 软件包原始model软件包下载，解压软件包为 `xlmodel_nice`，然后将其导入 Nuclei Studio。

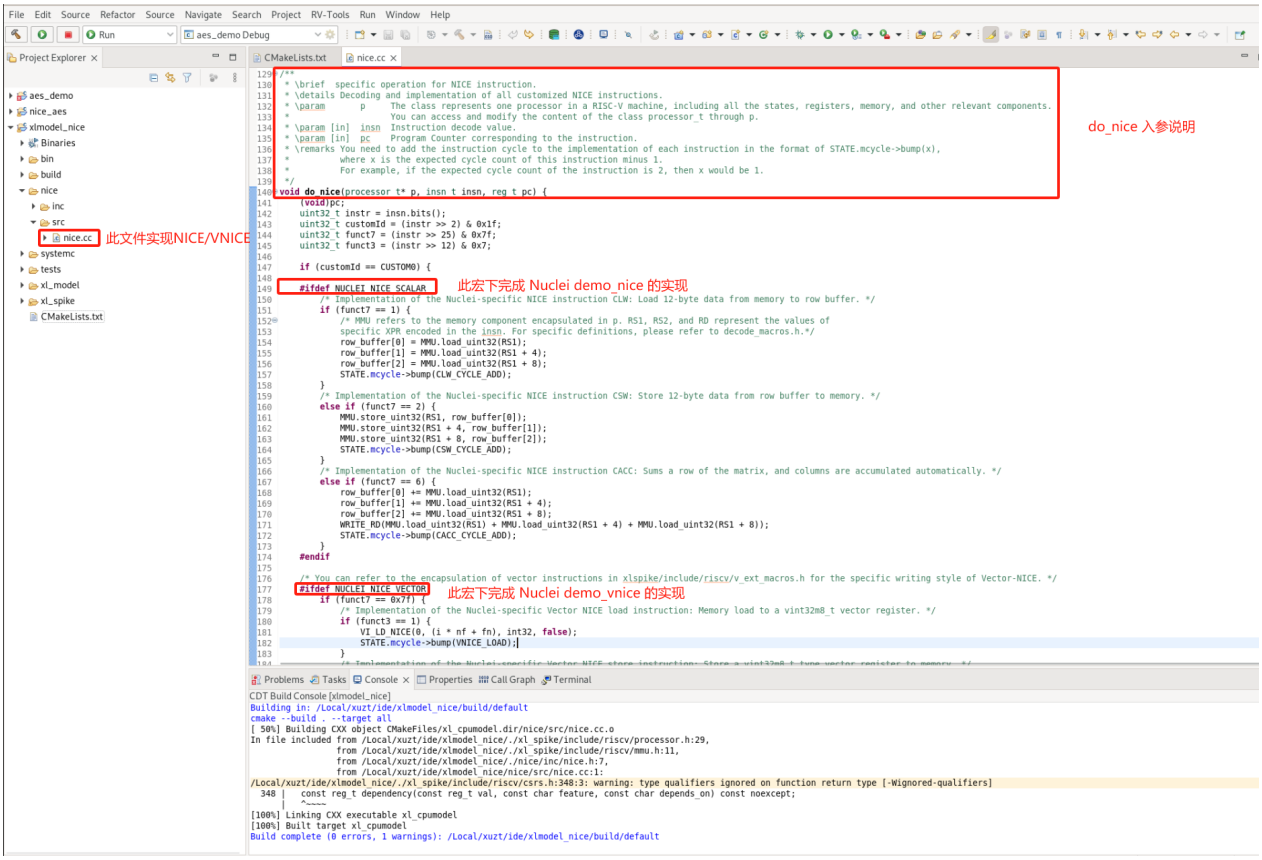
导入步骤：File->Import->Projects from Folder or Archive->Next->Directory->选择`xlmodel_nice`->Finish即可

如何使用 Nuclei Model 以及查看 `xlmodel_nice` 软件包的目录结构可以参考Nuclei Model介绍，`xlmodel_nice` 是由CMake构建的，用户无需修改即可编译，在 编译前选择 Nuclei Studio 的 launch bar 的 `xlmodel_nice`，然后点击编译，确保软件包本身编译通过，编译生成的 elf 文件所在路径为 `build/default/xl_cpumodel`：



打开 `nice.cc` 文件，用户需要用该文件的 `do_nice` 函数实现所有自定义的 NICE/VNICE 指令，当前 `do_nice` 里包含了针对 `demo_nice` 或 `demo_vnice` 的 Nuclei定义的 NICE/VNICE 指令，用户可以参考其中注释完成自己的自定义指令。

注意： 当用户编写自定义 NICE/VNICE 指令时，需要关掉和 Nuclei `demo_nice/demo_vnice` 对应的 `NUCLEI_NICE_SCALAR/NUCLEI_NICE_VECTOR` 宏，以免和用户自定义的指令编码相冲突。

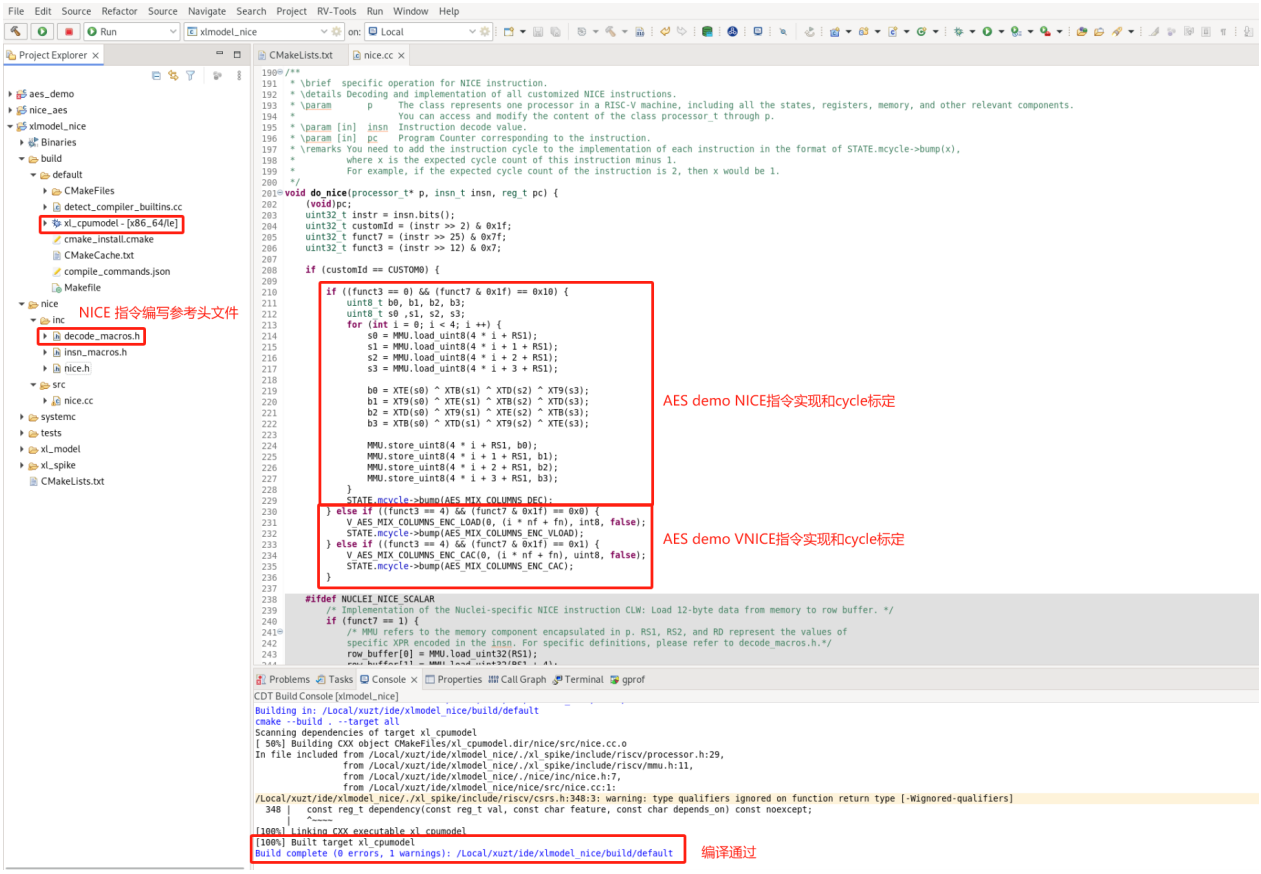


AES demo 中定义的 NICE/VNICE 指令实现如下图，通过指令的 opcode、funct3 和 funct7 编写条件判断语句指定该条指令，然后在其中实现指令行为以及指令 cycle 数添加。

NICE 指令实现中，MMU 宏表示 memory 访问，load memory 使用 MMU.load_uint<n>，store memory 使用 MMU.store_uint<n>，RD、RS1、RS2、RS3 宏表示其对应标量寄存器中的值，FRS1、FRS2、FRS3 宏表示其对应浮点寄存器中的值，这些宏的使用可以参考 nice/inc/decode_macros.h。

VNICE 指令实现中仍然是用 MMU 宏访问 memory，只不过 Vector 寄存器数据会存储在 P.VU.elt 类中，用户可以参考 xlspike/include/riscv/v_ext_macros.h 完成相关代码编写。

在指令实现完后，将自定义指令需要的 cycle 数 n 直接标定：STATE.mcycle->bump(n); 即可，这里根据硬件通过 NICE/VNICE 实现此算法的理论值，标定 custom_aes_mix_columns_dec 为 7 cycle，__custom_vnice_load_v_i8m1 为 1 cycle，__custom_vnice_aes_mix_columns_enc_i8m1 为 2 cycle。



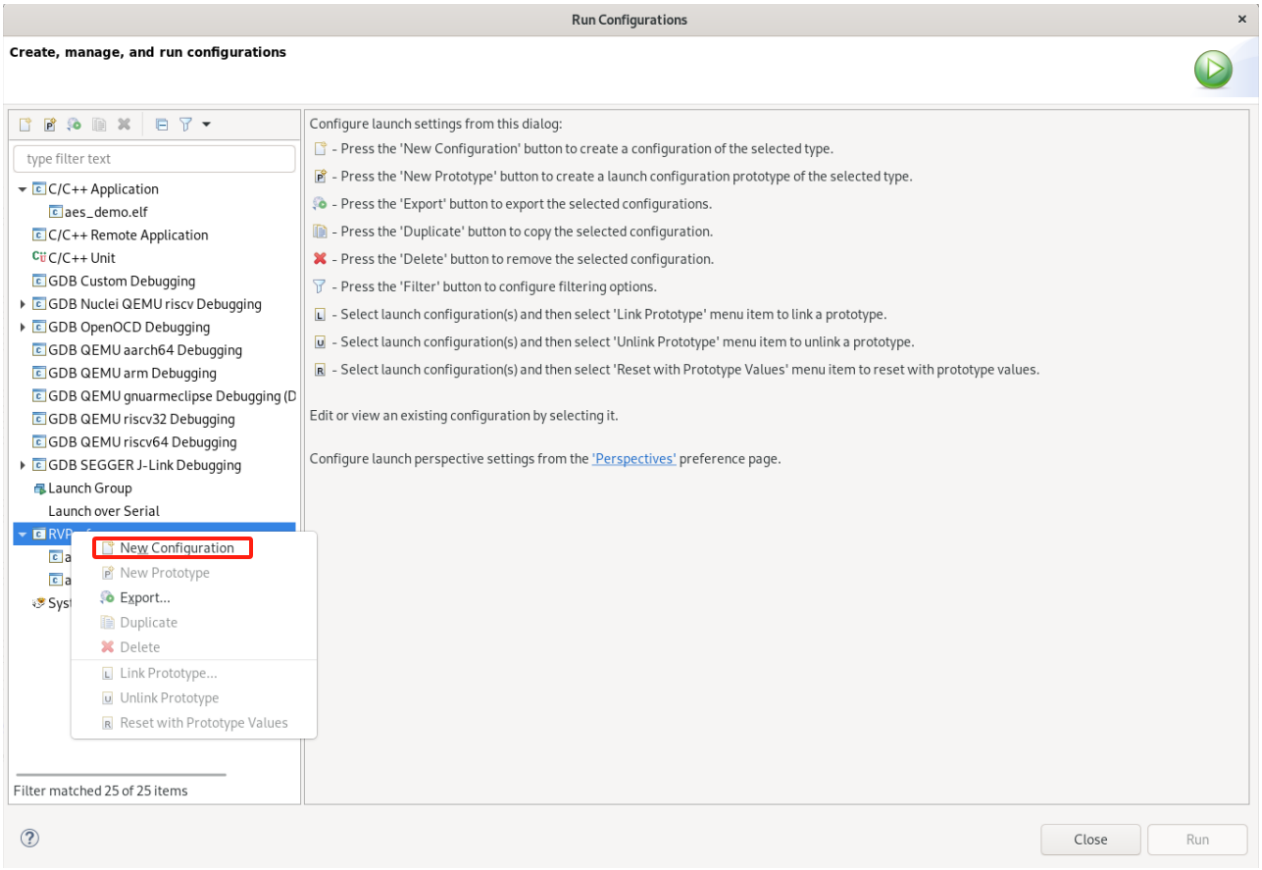
以上介绍了用户如何从原始 Nuclei Model 软件包添加自定义 NICE/VNICE 指令，接下来需要将新编译出的 model 可执行程序导入到 Nuclei Studio 中，为了不和 Nuclei Studio 原始 model 名称混淆，可以将 model 导入到 `NucleiStudio/toolchain/nucleimodel/bin_aes/` 的创建路径下，我们提供了两种 model 可执行程序获取方式：

1. 实现 AES demo NICE/VNICE 指令的 Nuclei model 软件包添加AES NICE指令model软件包，编译后将 `xl_cpumodel` 可执行程序导入上述路径。
2. 编译好的 model 的可执行程序 xl_cpumodel，直接导入上述路径。

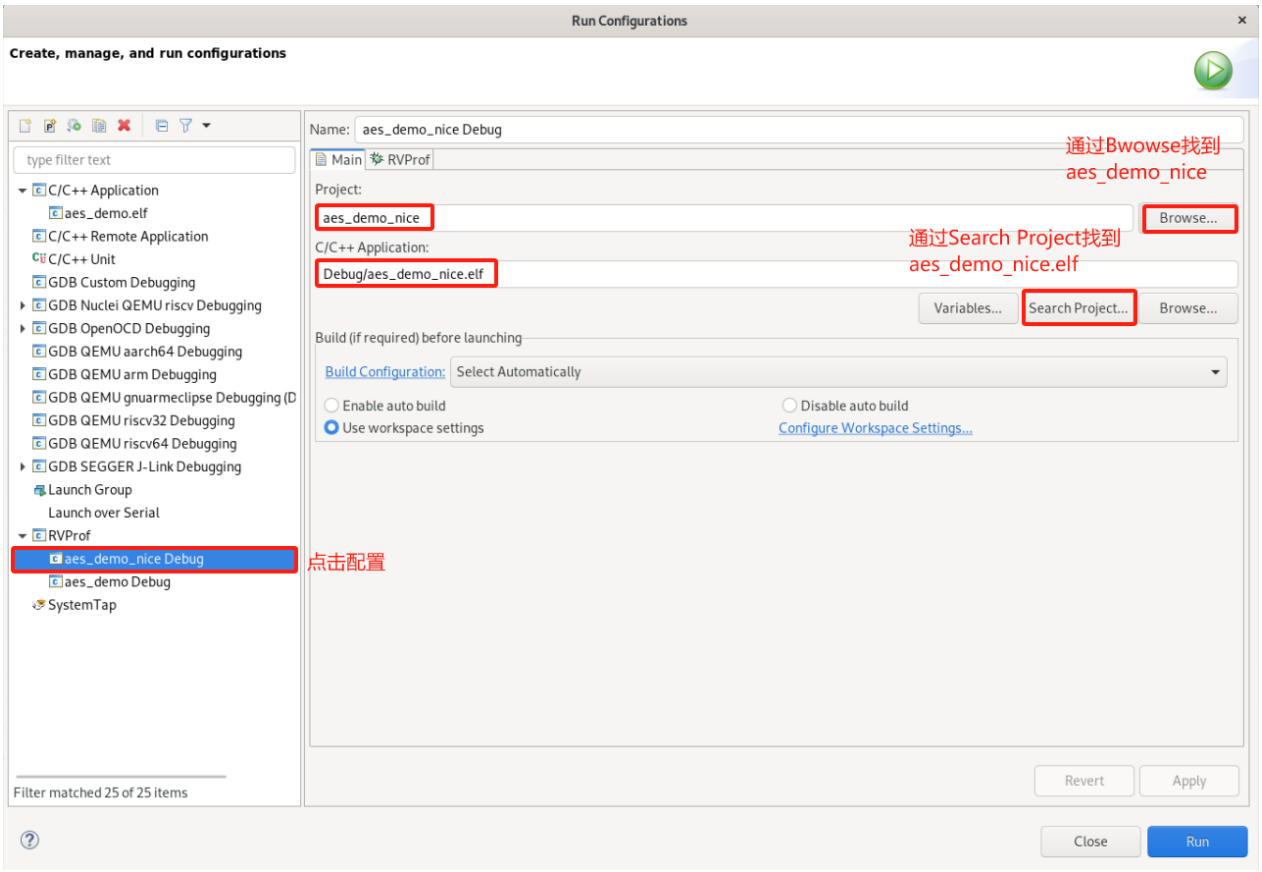
step7：热点函数再分析¶

注意： 请务必完成 step6 中介绍的实现了 NICE/VNICE 指令的 model 导入 Nuclei Studio 中才能用 model Run `aes_demo_nice` 工程。

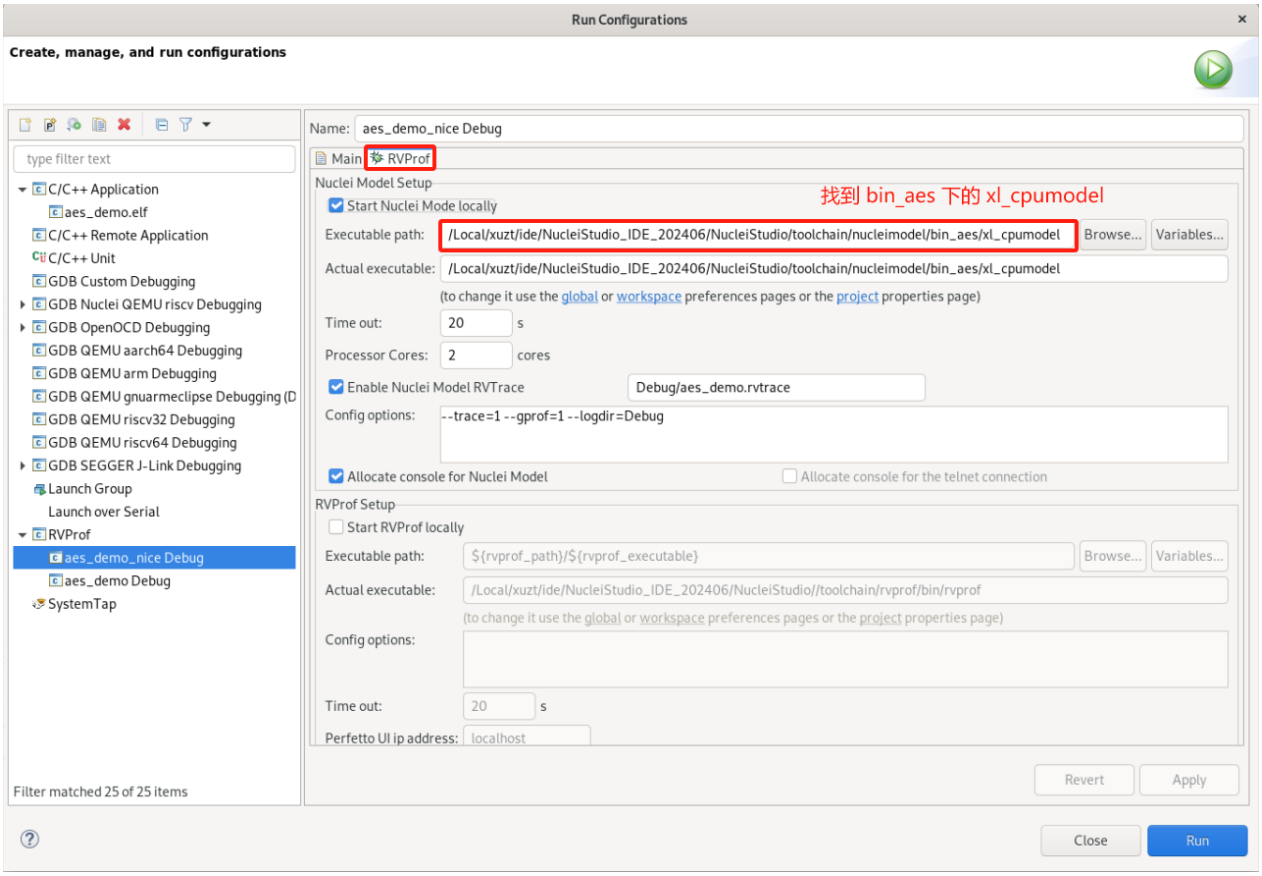
首先打开 Nuclei Studio 主菜单栏的 Run 选项的 `Run Configurations`，model 配置需要重新添加一份 Nuclei Studio 中的 RVProf 运行配置 `aes_demo_nice Debug`：



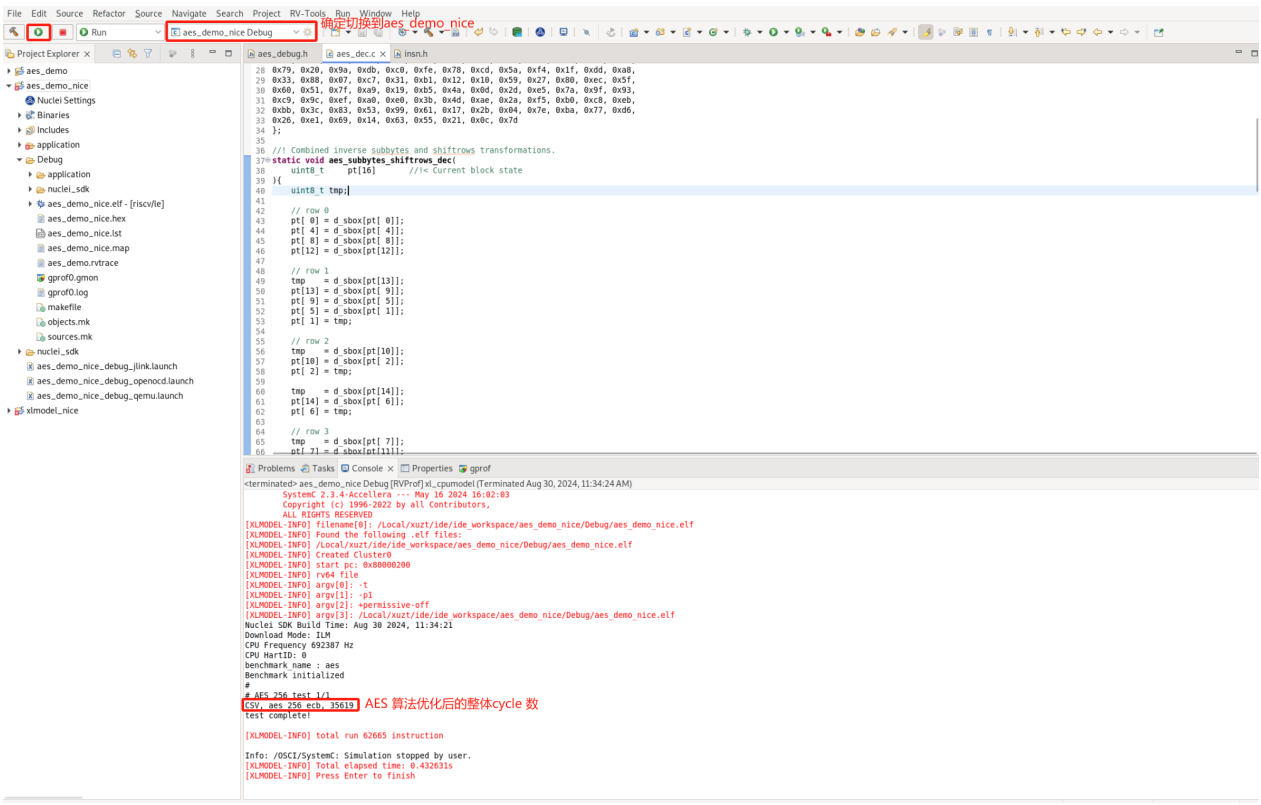
将 `Main` 选项卡的 `Project` 通过 `Browse` 改为 `aes_demo_nice`，`C/C++ Application` 通过 `Search Project` 改为 `aes_demo_nice.elf`:



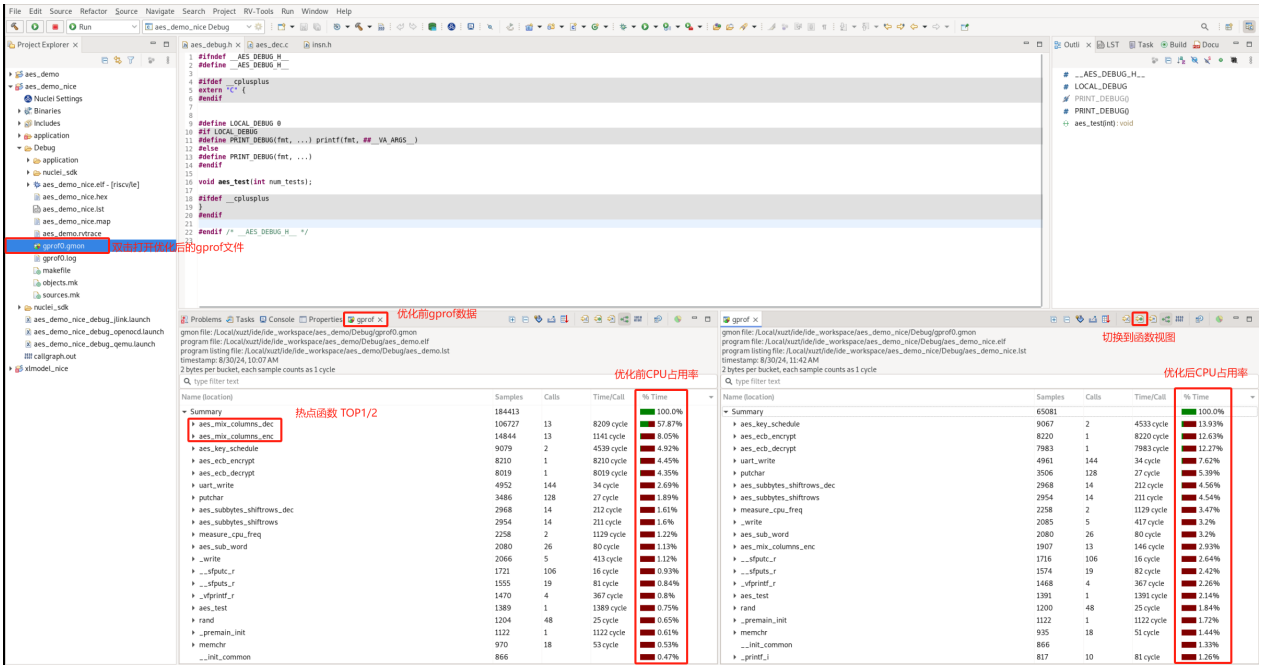
然后将 RVProf 选项卡中的 model 执行路径 Executable path 改为 step6 中新修改 model 的执行路径：.../NucleiStudio/toolchain/nucleimodel/bin_aes/xl_cpumodel:



运行前将 aes_debug.h 中的 LOCAL_DEBUG 打开，测试优化后 AES 算法的整体 cycle 数，选择 Nuclei Studio 的 launch bar 的 aes_demo_nice Debug后 Run model，得到 AES 算法优化后整体消耗 cycle 数从优化前的 154988 降到了 35619 cycle。

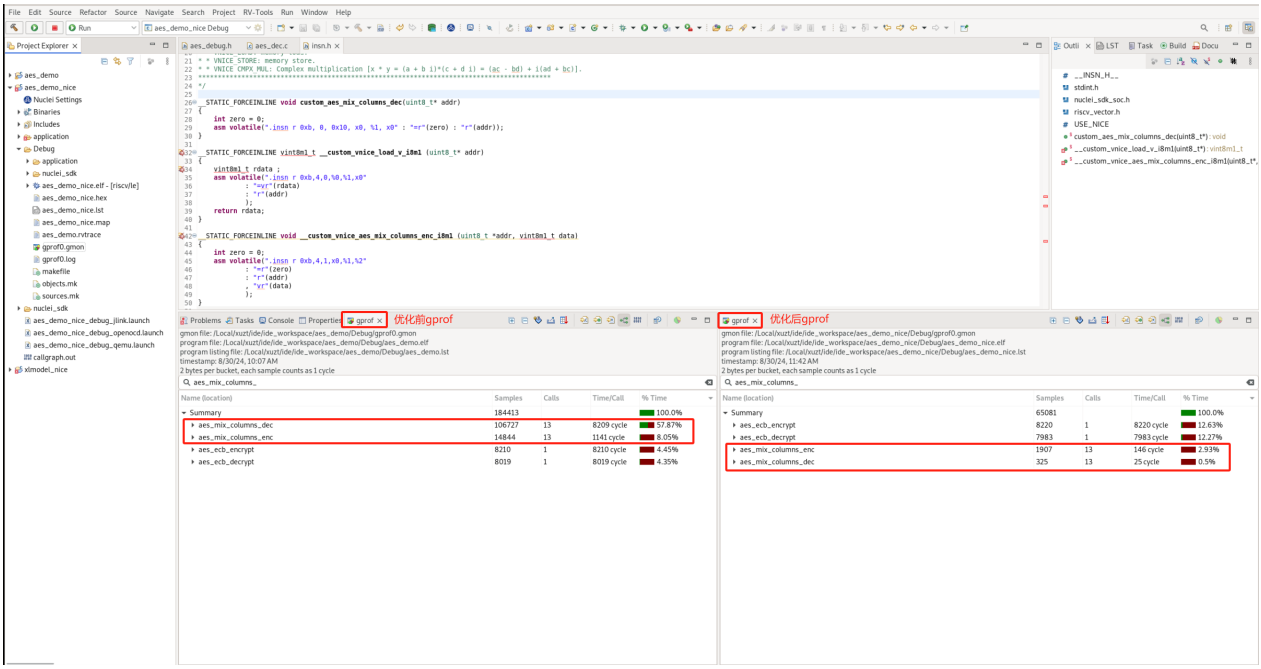


将 aes_debug.h 中的 LOCAL_DEBUG 关掉测试重新 Run model 测试 Profiling 数据，双击 gprof0.gmon 可以看到 CPU 占用率较高的热点函数已经没有 aes_mix_columns_enc 和 aes_mix_columns_dec 了：



搜索 aes_mix_columns_enc 和 aes_mix_columns_dec ，CPU 占用率 aes_mix_columns_enc 从 8.05% 降到了 2.93%，aes_mix_columns_dec 从 57.87% 降到了 0.5%，函数 Time per Call 消耗 cycle 数 aes_mix_columns_enc 从 1141 cycle 降到了 146

cycle，`aes_mix_columns_dec` 从 8209 cycle 降到了 25 cycle，说明了通过 NICE/VNICE 指令替换热点函数可以大幅提高程序算法性能。



数据统计如下：(enc: `aes_mix_columns_enc`, dec: `aes_mix_columns_dec`)

Function	Before Optimization	NICE/VNICE Optimization
CPU Usage % (enc)	8.05	2.93
CPU Usage % (dec)	57.87	0.5
Time per Call Cycles (enc)	1,141	146
Time per Call Cycles (dec)	8,209	25

AES Program Total	Before Optimization	NICE/VNICE Optimization
Cycles	154,988	35,619

AES加解密 NICE/VNICE demo：优化后AES工程链接下载

Nuclei Model结合Nice Wizard快速验证NICE/VNICE指令加速¶

> Nuclei Model 已支持 Windows/Linux 版本，此文档测试都是基于 Nuclei Studio 的 Windows 版本 (>= 2025.02) 完成的。

背景描述¶

xlmodel_nice¶

Nuclei Model 会不断更新提供用户可自定义实现 `NICE/VNICE` 的 `xlmodel_nice` 软件包，用户通过在 `xlmodel_nice/nice/src/nice.cc` 实现指令的具体行为，编译出新的 Nuclei Model 供应用程序配置调用。

Nuclei NICE Wizard¶

Nuclei NICE Wizard 是 Nuclei Studio 上提供的 `NICE/VNICE` 指令生成控件，用户配置好自定义指令后，可以自动生成两个文件：

1. `insn.h`: 指令内嵌汇编头文件，用户需要将此文件的指令内嵌汇编添加到应用程序头文件中
2. `nice.cc`: 指令实现文件，用户需要将此文件的指令 decode 框架添加到 `xlmodel_nice/nice/src/nice.cc` 中

test code¶

在 AI 与深度学习中常见的批量矩阵运算中，存在需要多次处理小矩阵块的场景，此测试将使用标量的多个 4x4 矩阵的乘法和累加操作的算法函数作为 `golden_case`，然后通过配置 NICE Wizard 生成 `NICE/VNICE` 加速指令，分别添加到测试应用程序和 `xlmodel_nice` 软件包工程中重新编译，最后通过运行 Nuclei Model 查看优化后的算法函数的指令数和 cycle 数，以查看 `NICE/VNICE` 加速效果。

解决方案¶

环境准备¶

> Nuclei Studio IDE 集成的 NICE Wizard 相关功能，需要配合 Nuclei CPU Model - NICE Support (xlmodel_nice) 软件包使用。

Nuclei Studio：

- NucleiStudio 202502 Windows
- NucleiStudio 202502 Linux

xlmodel_nice：

- 原始`xlmodel_nice`软件包 Windows
- 原始`xlmodel_nice`软件包 Linux

Nuclei Model运行原始程序¶

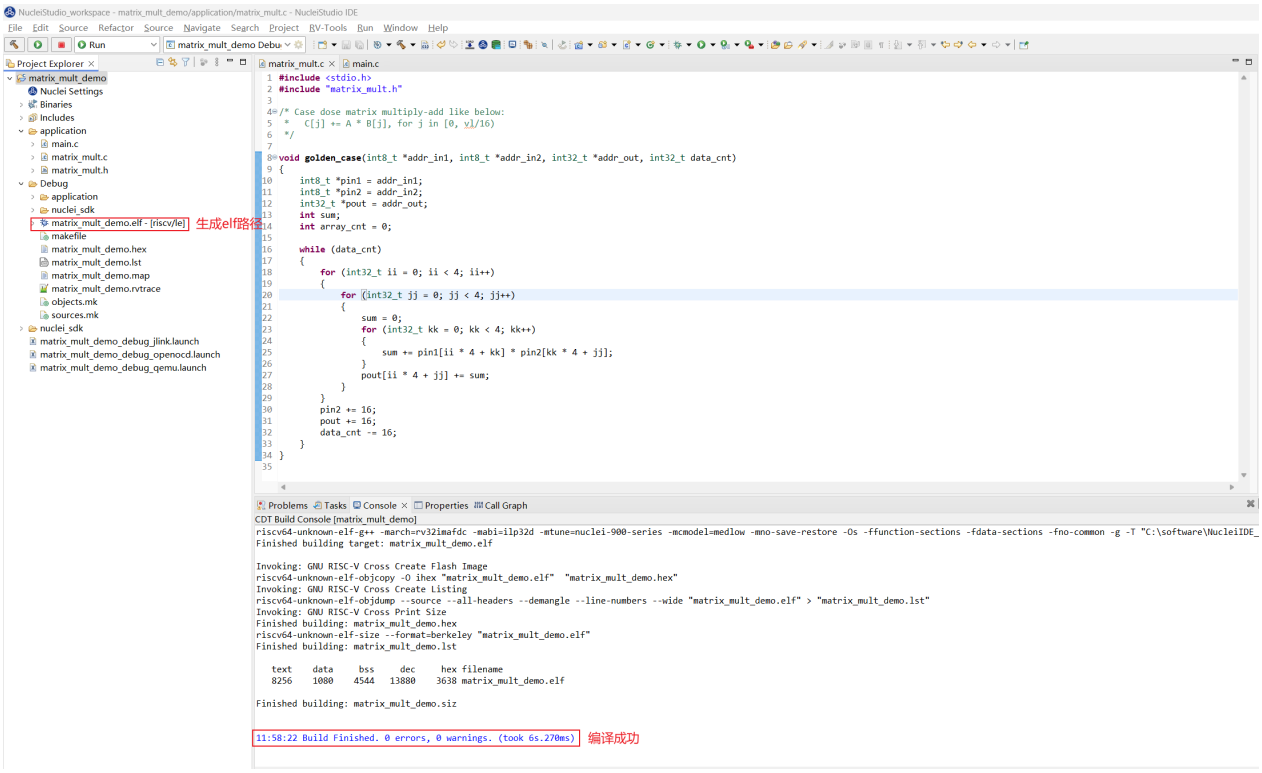
step1：导入 Nuclei SDK 原始工程

优化前的工程下载链接

下载 zip 包后，可以直接导入到 Nuclei Studio 中运行 (导入步骤：`File->Import->Existing Projects into Workspace->Select archive file->`选择`zip`压缩包`->Finish`即可)

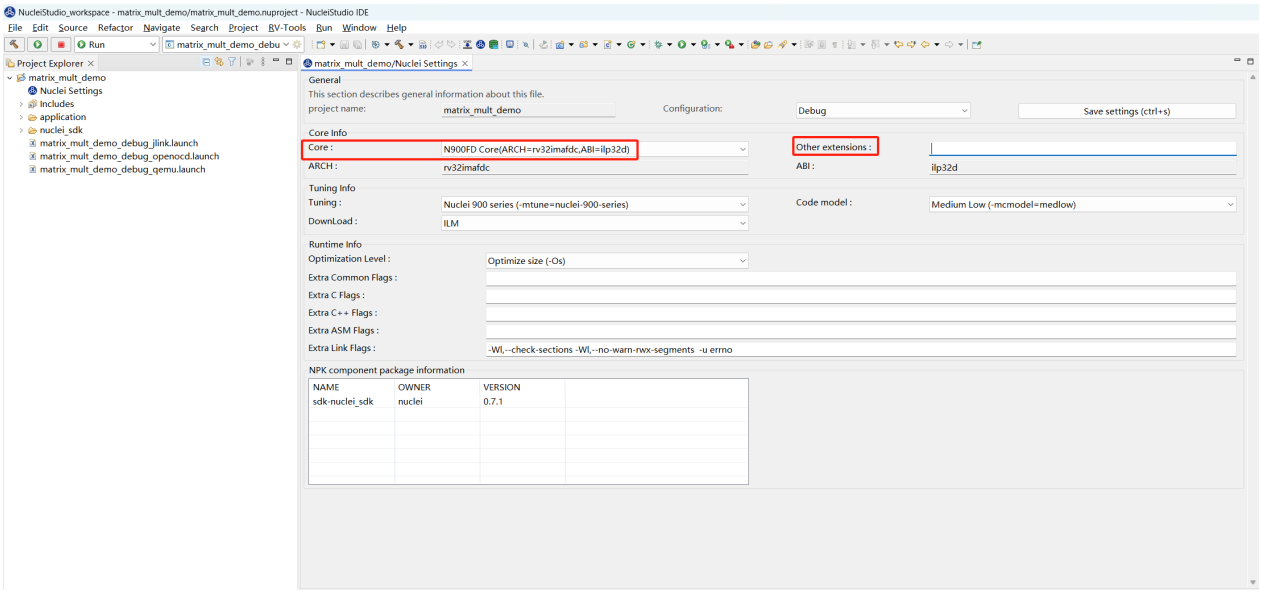
step2：编译 Nuclei SDK 原始工程

编译原始工程，确保编译成功以及在 Debug 下可以找到生成的 elf 文件：



step3：运行 Nuclei SDK 原始工程

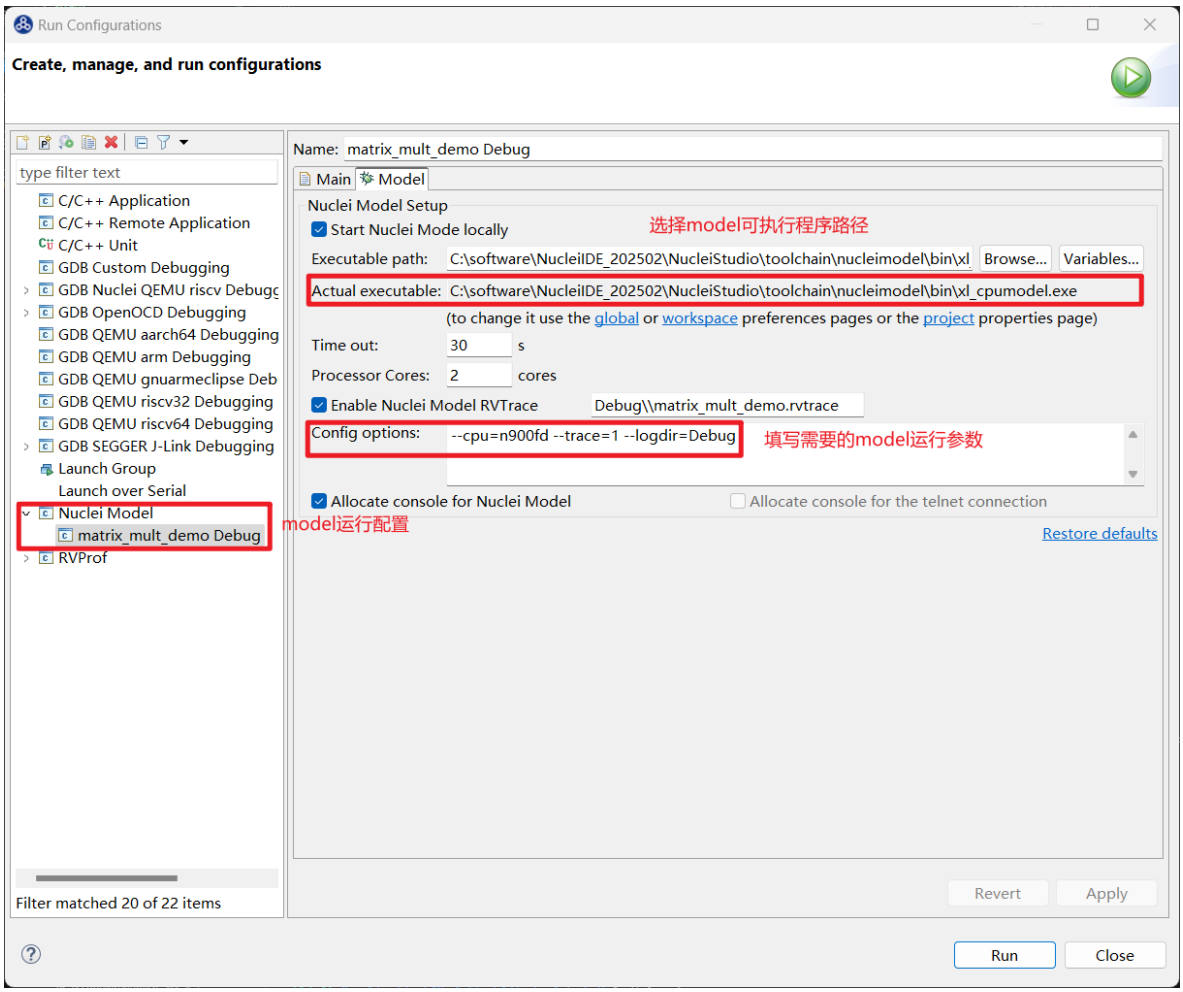
在使用 Nuclei Model 运行程序时，需要先确定工程 Nuclei Settings 中的 Core 配置和 Other extensions 配置，这些配置需要传递给 Model 使用。当前使用的 Core 是 n900fd，Other extensions 未配置。



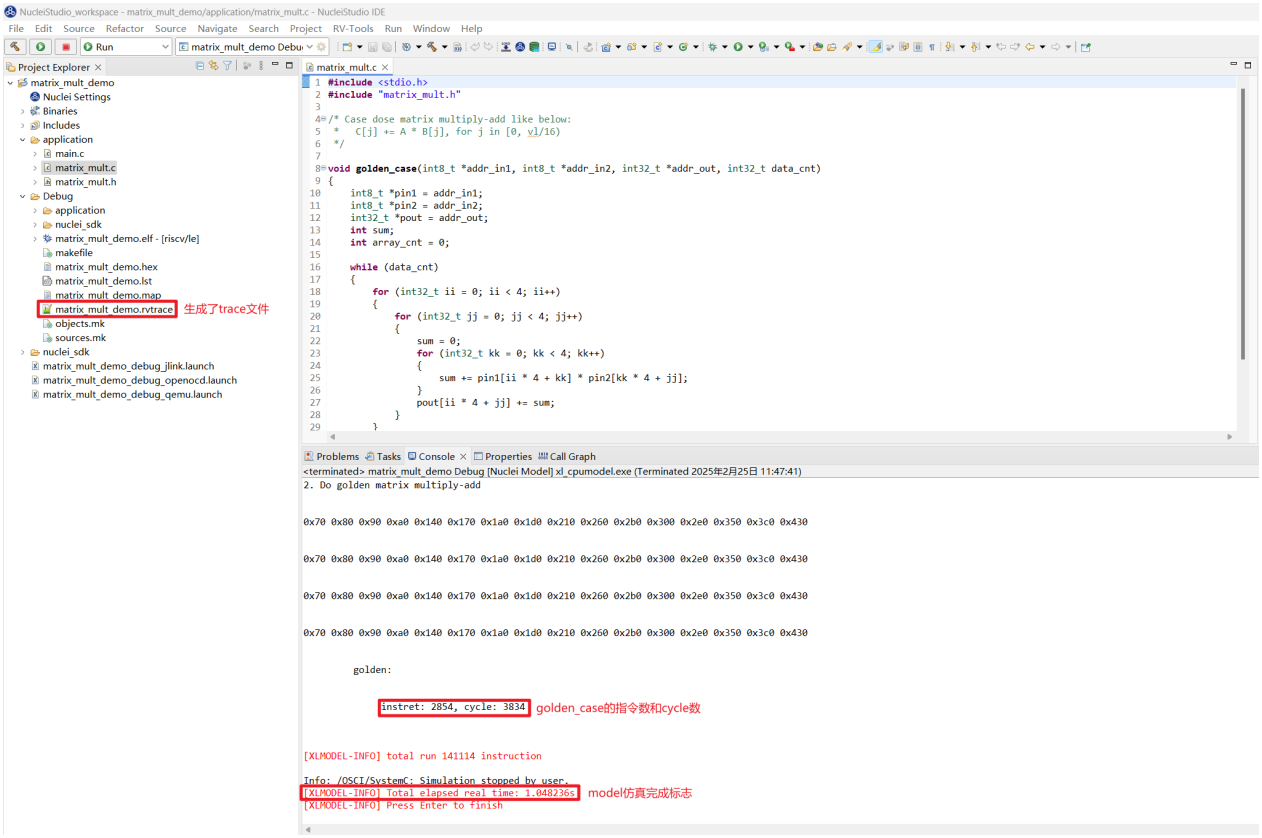
Model 仿真程序需要配置 Nuclei Studio 中的 Nuclei Model 配置项，步骤如下：

1. 打开 Nuclei Studio 主菜单栏的 Run 选项的 Run Configurations
2. 选择 Nuclei Model 配置项，右键选择 New Configuration，会自动生成项目名的 Model 配置页面，launch bar也会同步更新
3. 在右侧 Main 选项卡中点击 Search Project... 选择编译好的 elf 文件
4. 在右侧 Model 选项卡中选择 Browse 找到 Nuclei Model 可执行程序默认路径：NucleiStudio/toolchain/nucleimodel/bin/xl_cpumodel.exe

5. 在右侧 Model 选项卡中的 Config options 中完成 model 运行配置：--cpu=n900fd --trace=1 --logdir=Debug，--cpu 和 --ext 需要保持和 Nuclei Settings 的 Core 和 Other extensions 配置一致，--ext 为空时不传递此参数，--trace=1 表示开启 rvtrace，--logdir=Debug 则表示最终生成的 *.rvtrace 文件存存放路径为当前工程下的 Debug 目录，然后点击 Apply 和 Run，model 就开始运行程序了



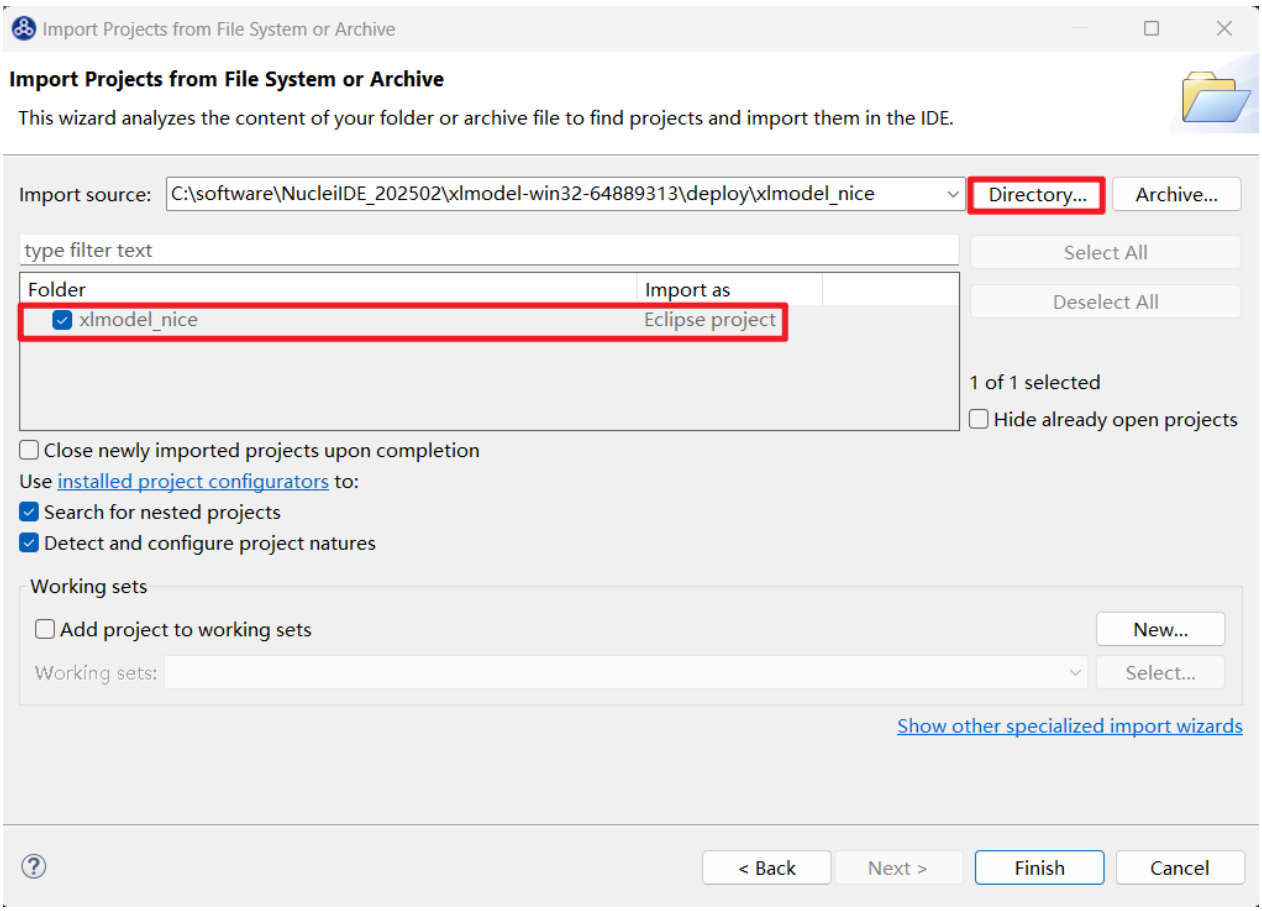
在 Console 中会看到`Total elapsed time`说明 model 已经完成仿真了，程序会提取标量矩阵乘算法函数`golden_case`的执行指令数和 cycle 数如下：



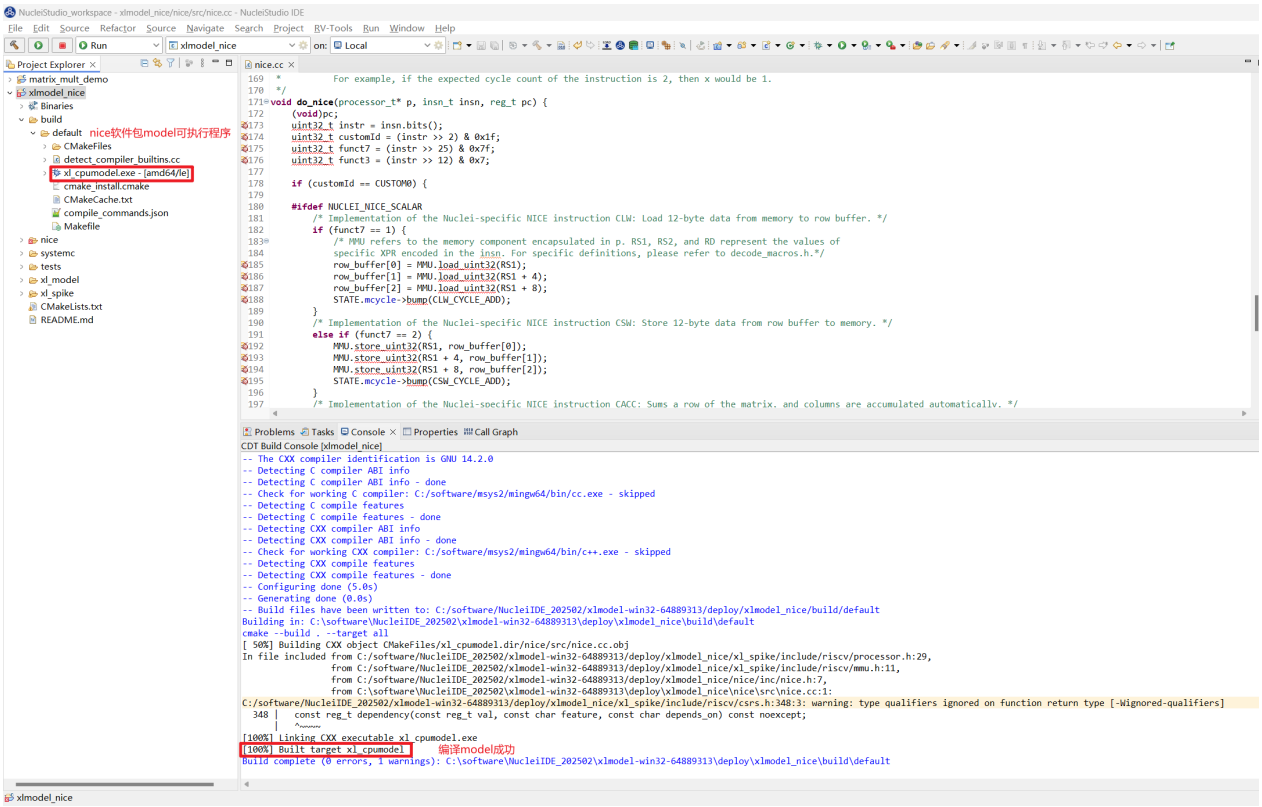
NICE指令替换¶

step1：编译 xlmodel_nice 软件包

下载并解压 `xlmodel_nice` zip 包后，可以直接导入到 Nuclei Studio 中运行 (导入步骤：`File->Import->Projects from Folder or Archive->Next->Directory->`选择`xlmodel_nice`文件夹`->Finish`即可)



在编译 `xlmodel_nice` 前需先配置好 xlmodel 的编译环境 (xlmodel_nice 编译环境配置)，然后编译确保原始软件包可以成功编译生成 model 的可执行程序：



step2：NICE Wizard生成NICE指令替换

应用程序的热点函数可以先用 Nuclei Model Profiling 来定位，具体使用可以参考 通过Profiling展示Nuclei Model NICE/VNICE指令加速，这里不再赘述了。

此用例的热点函数已知是矩阵乘累加，A矩阵某行 * B矩阵某列计算如下：

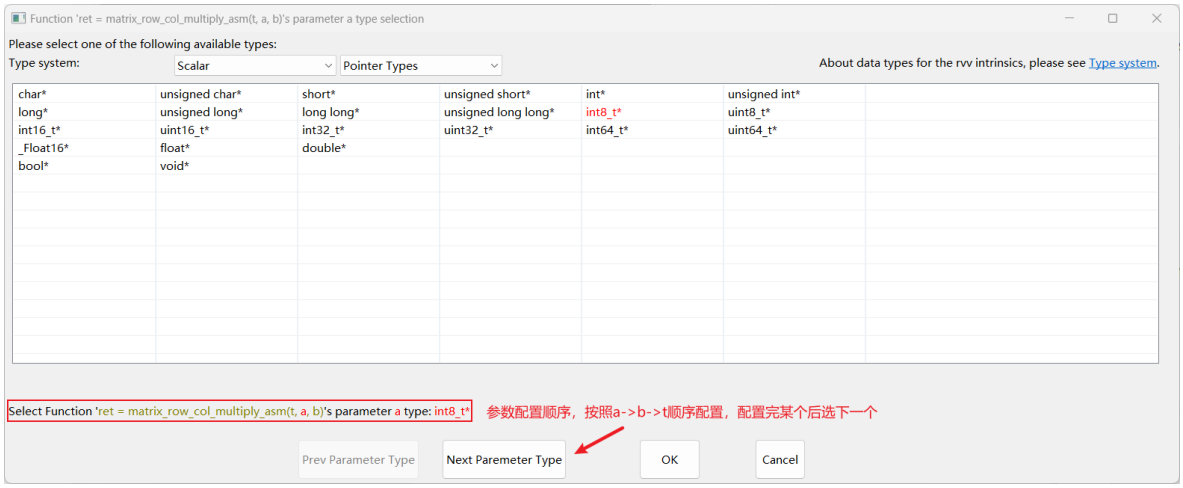
```
for (int32_t kk = 0; kk < 4; kk++)
{
    sum += pin1[ii * 4 + kk] * pin2[kk * 4 + jj];
}
```

此算法完全可以替换成一条 NICE 指令来完成，输入为 sum 值， pin1 地址， pin2 地址，输出为 sum。

接下来用 NICE Wizard 来生成设想的 NICE 指令，用户可以在 Nuclei Studio 的 xlmodel_nice 工程根目录创建一个 aicc.nice 的文件，此文件创建后就会弹出 NICE Wizard 的指令生成窗口，配置生成 NICE 指令步骤如下：

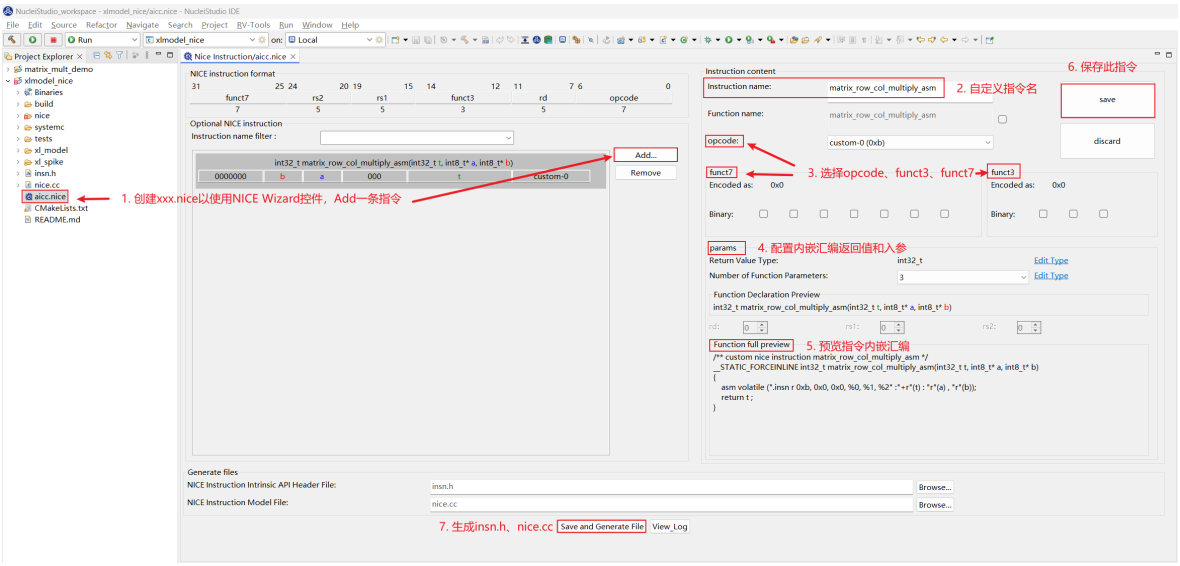
1. 选择 Add 添加一条 NICE 指令，指令格式如左上角 NICE instruction format 所示，首先填写 Instruction name项为 matrix_row_col_multiply_asm 表示矩阵行列乘加操作
2. 依次选择填写 opcode、funct3、funct7
3. params 是指令内嵌汇编的返回值和入参配置，构想的 NICE 指令返回值为 int32_t，入参个数为3个，分别是 int32_t t、int8_t* a、int8_t* b，分别在 params 中设置好

   注意：在 入参的 Edit Type 设置界面中，是按照 a->b->t 的顺序配置的:

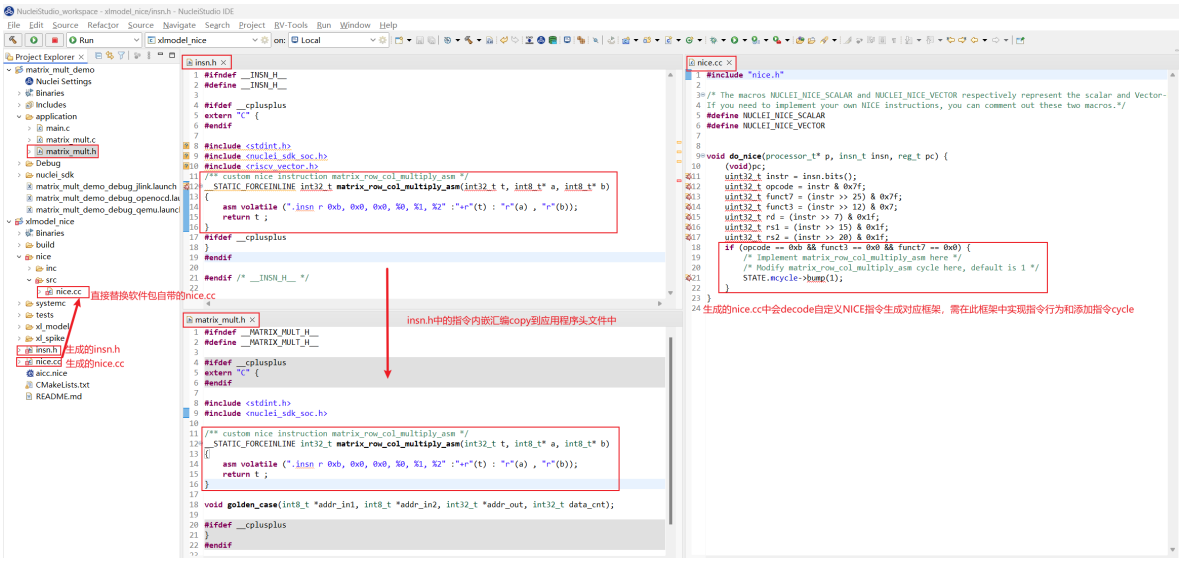


4. 在 Function full preview 中预览指令内嵌汇编格式是否正确，确保没有问题后点击 save，save 完成后可以在左侧指令栏中看到生成好的自定义指令了

5. 点击下方 Save and Generate File，在 aicc.nice 同路径下会生成 insn.h 和 nice.cc



6. 将生成好的 insn.h 中的 NICE 指令内嵌汇编复制到应用程序的头文件中，将生成好的 nice.cc 直接替换 xlmodel_nice/nice/src/nice.cc



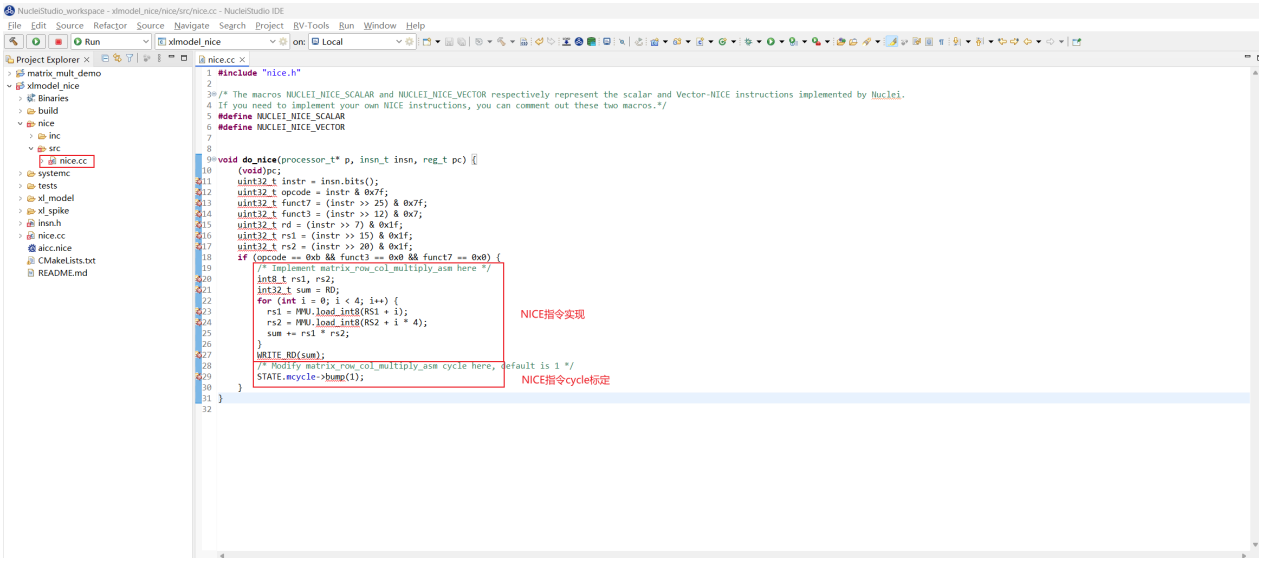
当然也可以将 insn.h 直接生成到应用程序工程路径下引用，这样可以省去每次手动的复制文件内容。

step3：xlmodel_nice实现NICE指令

打开 xlmodel_nice/nice/src/nice.cc 文件，使用 spike 中定义的宏来实现 NICE 指令：MMU 宏表示 memory 访问，load memory 使用 MMU.load_xxx<n>，store memory 使用 MMU.store_xxx<n>，RD、RS1、RS2、RS3 宏表示其对应标量寄存器中的值，写目标寄存器使用 WRITE_RD，这些宏的使用可以参考 nice/inc/decode_macros.h。

在指令实现完后，将自定义指令额外需要的 cycle 数 n 直接标定：`STATE.mcycle->bump(n);` 即可，这里标定此条 `NICE` 指令额外需要 1 cycle，由于指令默认需要 1 cycle，因此此条 `NICE` 指令需要消耗 2 cycle。

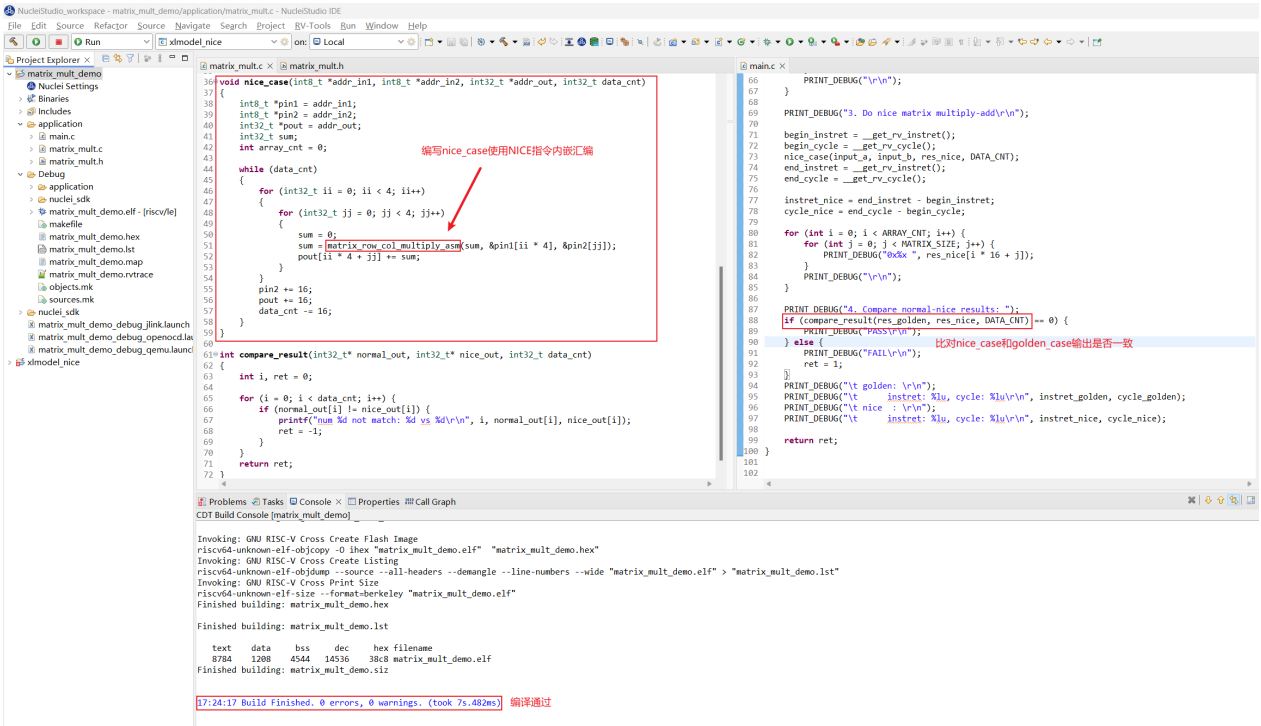
实现的 `NICE` 指令实现和 cycle 标定如下：



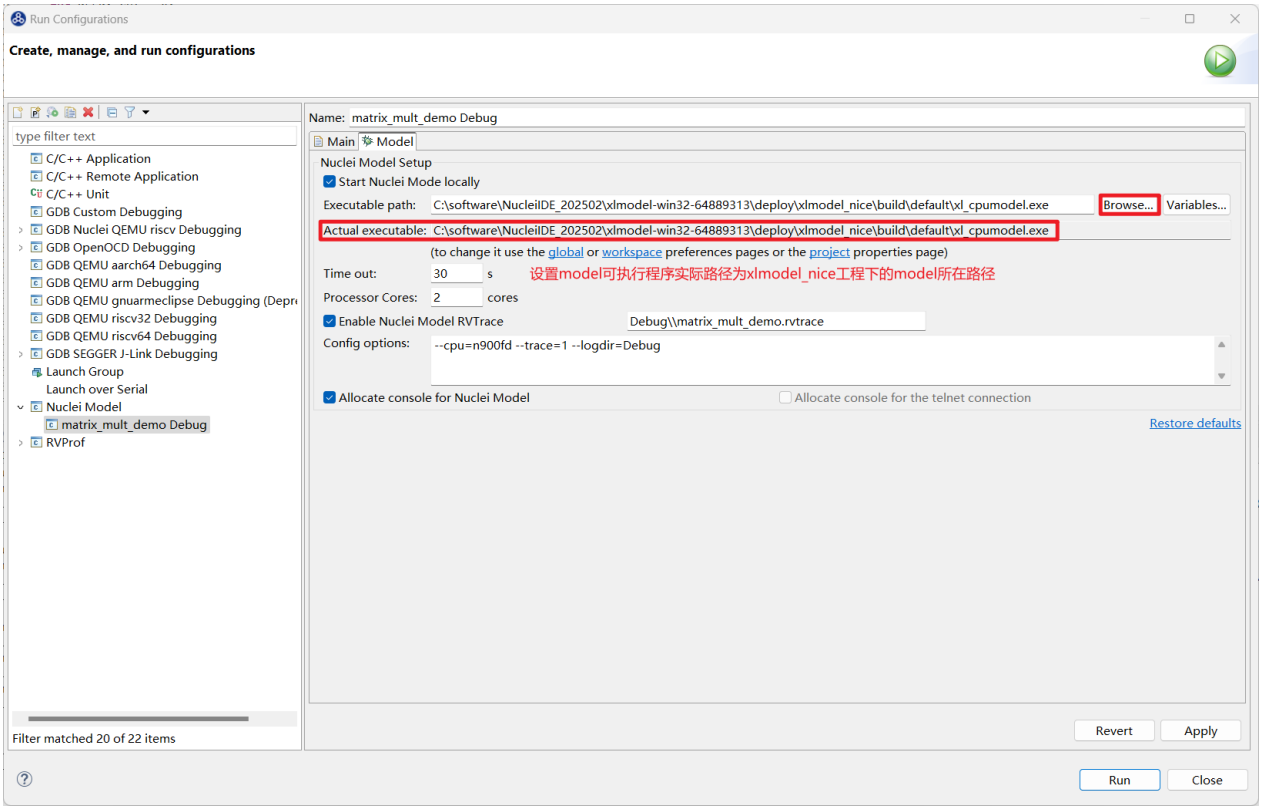
重新编译 `xlmodel_nice` 保证编译通过。

step4：Nuclei Model重新运行程序

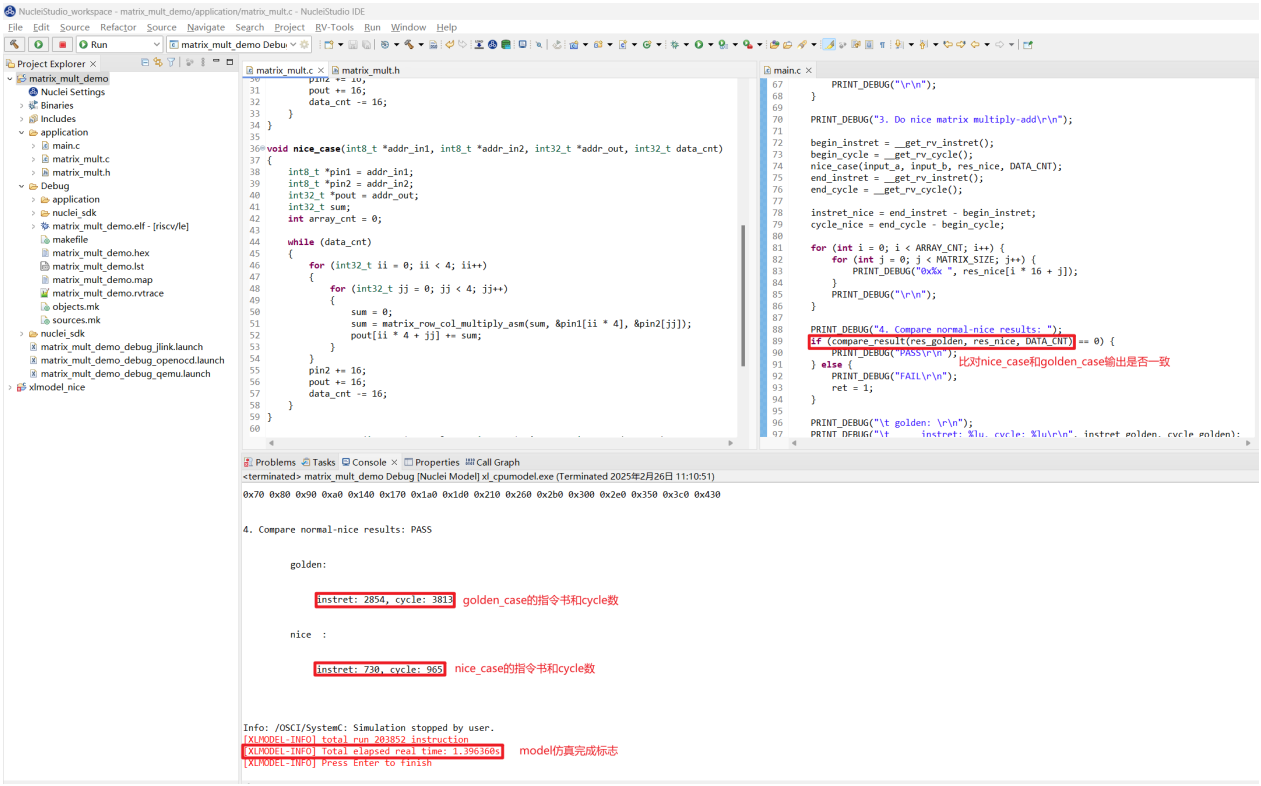
首先需要编写一个带 `NICE` 指令内嵌汇编的算法函数 `nice_case` 方便和 `golden_case` 对比，添加函数输出结果比对，然后重新编译应用程序工程：



因为 model 已经使用 `xlmodel_nice` 重新编译出新的可执行程序了，需要重新配置 Nuclei Studio `Nuclei Model` 配置项中的 model 可执行程序路径为 `xlmodel_nice/build/default/xl_cpumodel.exe`，其余配置不变：



`Apply` 后重新 `Run` 应用程序，可以发现 `nice_case` 和 `golden_case` 输出结果一致，`nice_case` 的指令数和 cycle 数均大幅下降了，构想的 `NICE` 指令实现正确，并优化了原标量算法。



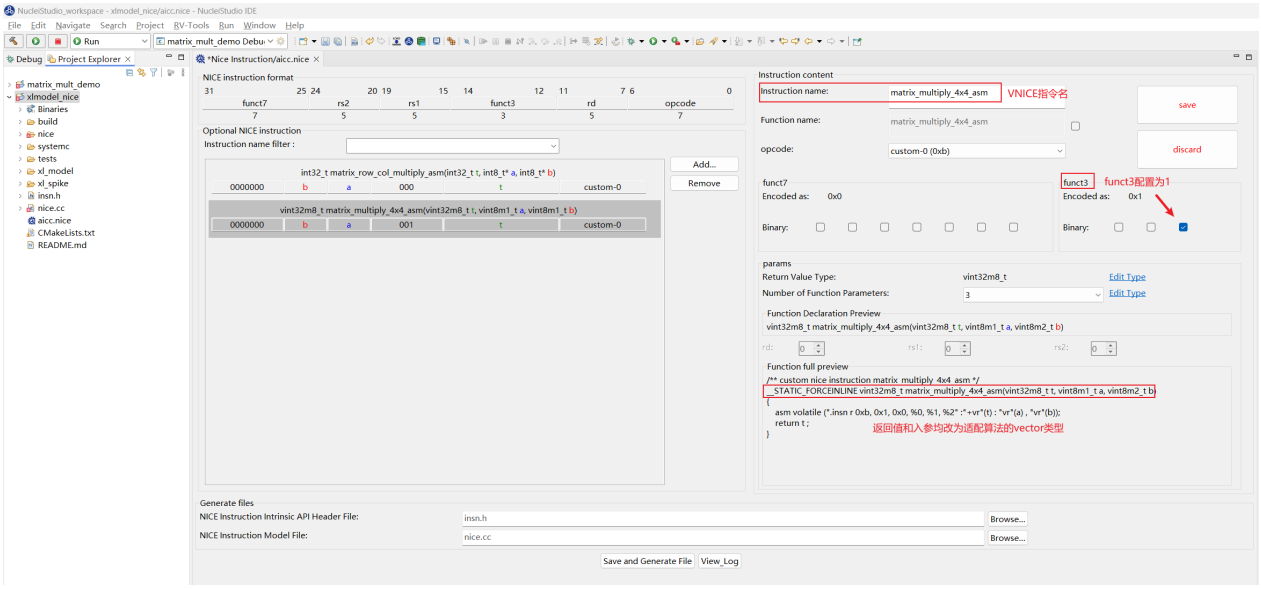
VNICE指令替换¶

step1：NICE Wizard生成VNICE指令替换

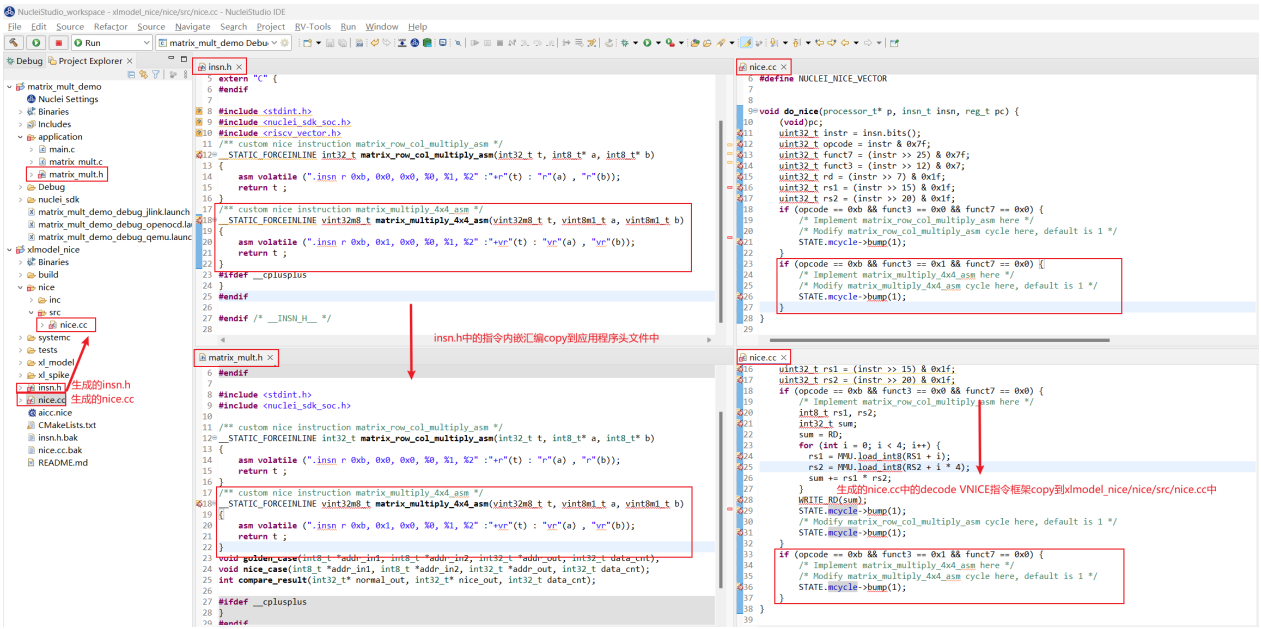
当使用 NICE 指令运算时，每次仅得到的是输出矩阵的一个元素，效率还不够高，如果一次指令操作可以并行处理多个矩阵元素，效率应将进一步提高，很自然会想到使用 Vector 指令来多并行度处理矩阵数据。

构想将完整的 4 * 4 矩阵乘加运算浓缩为一条 Vector 指令，可以使用一条 VNICE 指令来实现此行为，入参为 3 个 4 * 4 的输入矩阵，返回值为 4 * 4 的输出矩阵。

双击 aicc.nice 再次使用 NICE Wizard 配置构想指令，生成指令的步骤和以上生成 NICE 指令相似，不同之处为配置 Instruction name 项为 matrix_multiply_4x4_asm 表示完成的是 4*4 的两矩阵的乘法，配置 funct3 为 1 避免与上条 NICE 指令编码相同，为了匹配和 golden_case 标量对应的 vector 数据类型的输入输出，设置返回值为 vin32m8_t，入参个数为 3，分别是 vin32m8_t、vint8m1_t、vint8m2_t，点击 save 后的配置界面如下：



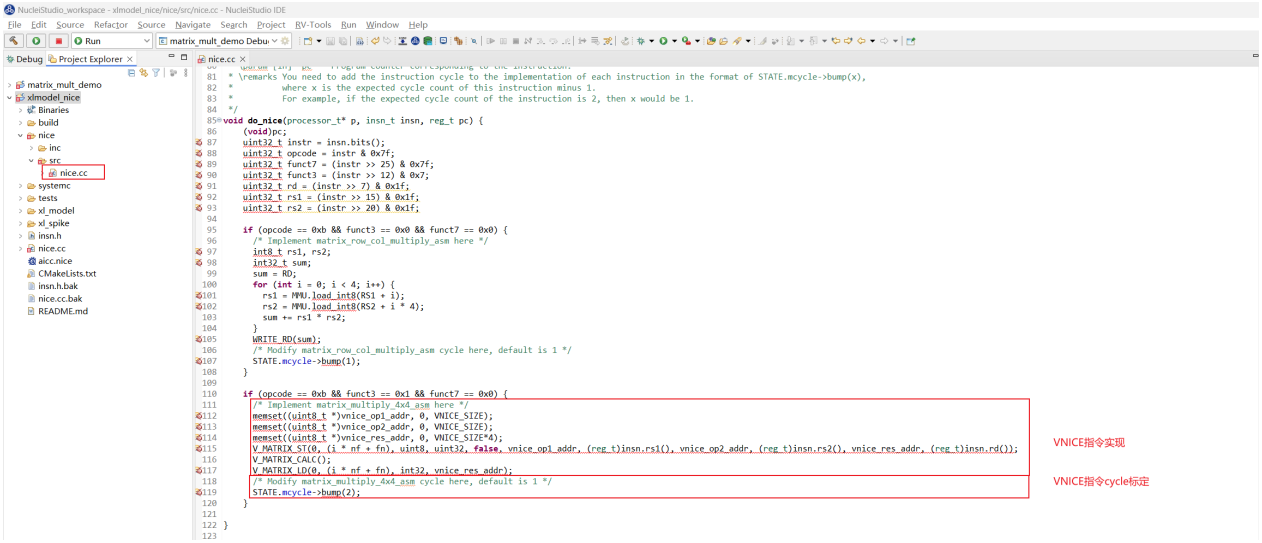
点击下方 Save and Generate File，覆盖之前生成的 insn.h 和 nice.cc，此时在同路径下还会出现 insn.h.bak 和 nice.cc.bak, 这两个文件是上一次保存的 insn.h 和 nice.cc 备份文件不会被用到，再次将生成好的 insn.h 中的 NICE 指令内嵌汇编复制到应用程序的头文件中，将生成好的 nice.cc 中的新指令 decode 框架复制到 xlmodel_nice/nice/src/nice.cc：



step2：xlmodel_nice实现VNICE指令

在 xlmodel_nice/nice/src/nice.cc 中实现 VNICE 指令，V_MATRIX_ST 实现将指令输入的 vector 寄存器 store 到自定义 buffer 中，V_MATRIX_LD 实现将指令输出的结果 load 到 RD 寄存器，V_MATRIX_CALC 实现两矩阵乘加运算，VNICE 指令实现可以参考 spike 中的 vector 指令实现：xlmodel_nice/xl_spike/include/riscv/v_ext_macros.h。

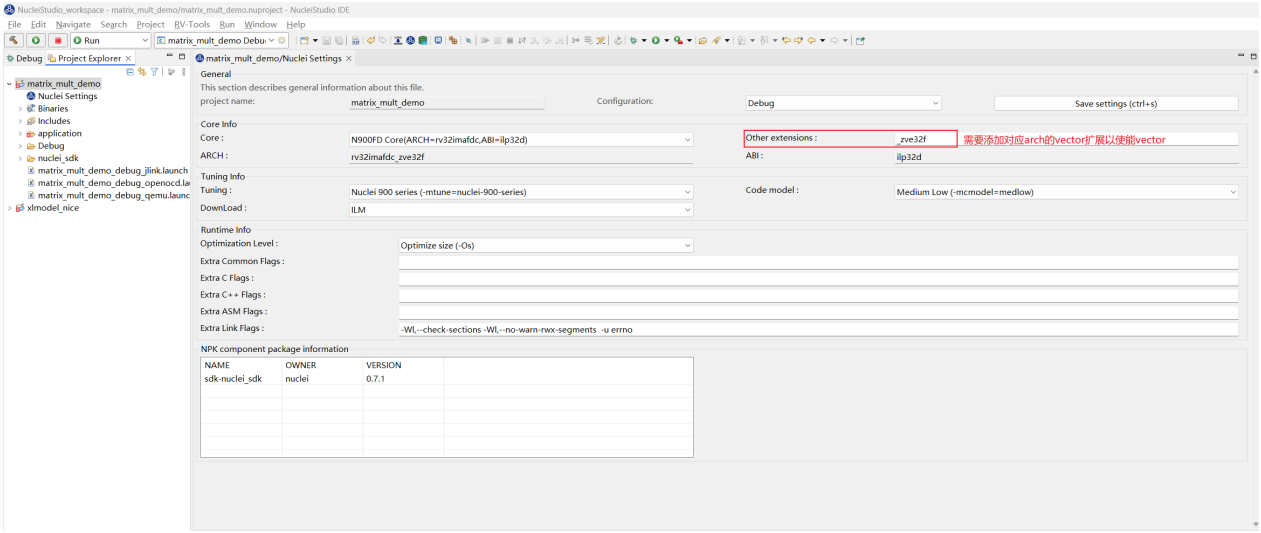
标定此条 VNICE 指令需要 2 cycle，即实际消耗 3 cycle，实现的 VNICE 指令实现和 cycle 标定如下：



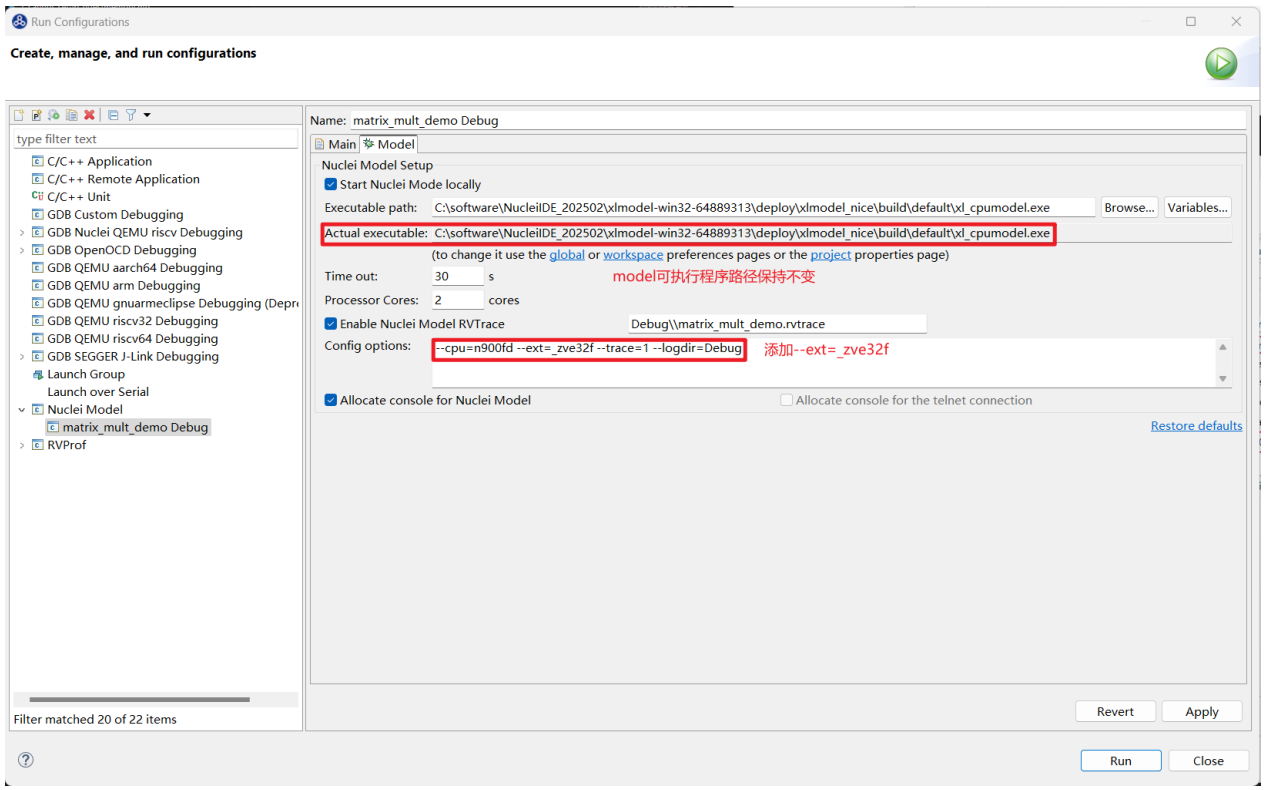
再重新编译 xlmodel_nice 保证编译通过。

step3：Nuclei Model重新运行程序

因为 VNICE 指令的输入输出均为 vector 寄存器，需要配置应用程序的 Nuclei Settings，使能对应 ARCH 的 vector 扩展，这里针对 rv32imafdc 添加 _zve32f 扩展：

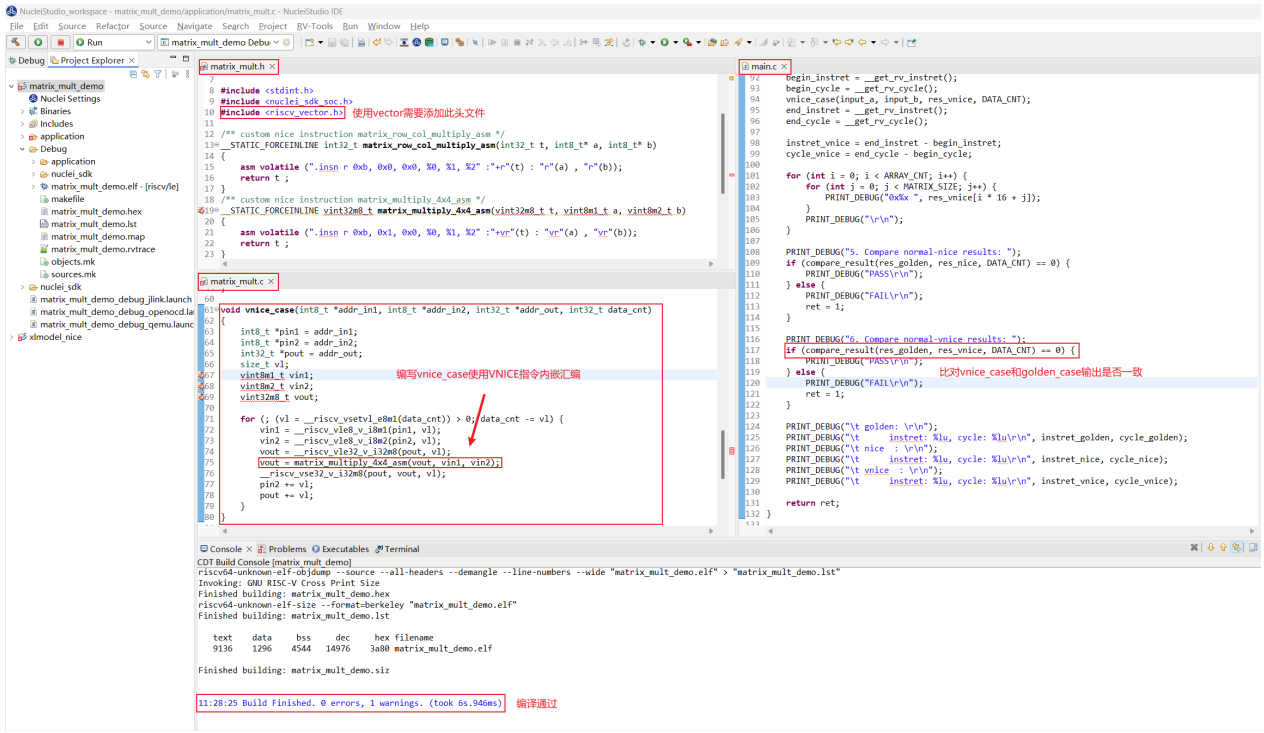


对应的 Nuclei Model 配置项也需要添加 --ext=_zve32f使能 model 的 vector 功能，然后 Apply：

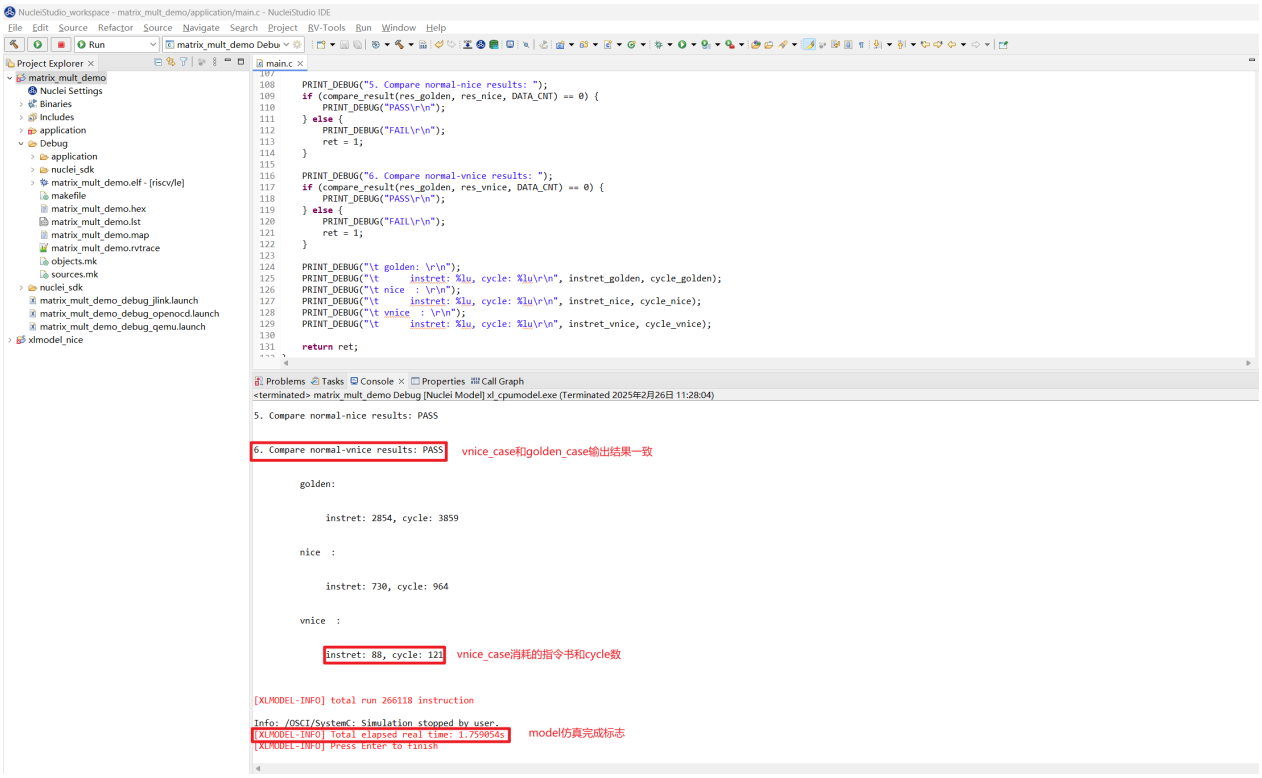


需要编写一个带 `VNICE` 指令内嵌汇编的算法函数 `vnice_case`，`VNICE` 内嵌汇编需要的输入输出需要写相应的 vector intrinsic API 来构造，然后添加和`golden_case` 的结果比对，重新编译应用程序工程。

注意： 在应用程序头文件中需要添加 `#include <riscv_vector.h>` 以使能 vector intrinsic API



重新 Run 应用程序, 可以发现 vnice_case 和 golden_case 输出结果一致，其指令数和 cycle 数相对 nice_case 进一步大幅下降了，构想的 VNICE 指令实现正确，并利用了 vector 的高并行度加速了矩阵乘加算法。



总结¶

下表是实现了 NICE/VNICE 指令优化算法后的 instret/cycle 数据统计，相较于 golden_case, nice_case 优化后的性能提高了约 4 倍，vnice_case 优化后的性能提高了超过 30 倍。

instret/cycle	golden_case	nice_case	vnice_case	golden / nice	golden / vnice	nice / vnice
instret	2854	730	88	3.91	32.43	8.30
cycle	3859	964	121	4.00	31.89	7.97

用户通过研究现有算法的优化策略，就可以将构想快速通过 NICE Wizard 生成相关 NICE/VNICE 指令，再通过 Nuclei Studio 导入 xlmodel_nice 软件包实现指令，编写应用程序指令优化 case，就可以很快的利用 Nuclei Model 验证算法优化效果，整个测试过程只需使用 Nuclei Studio 就可以完成。

优化后的工程下载链接

优化后的xlmodel_nice软件包

Flash Programming使用案例¶

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

解决方案¶

环境准备¶

Nuclei Studio：

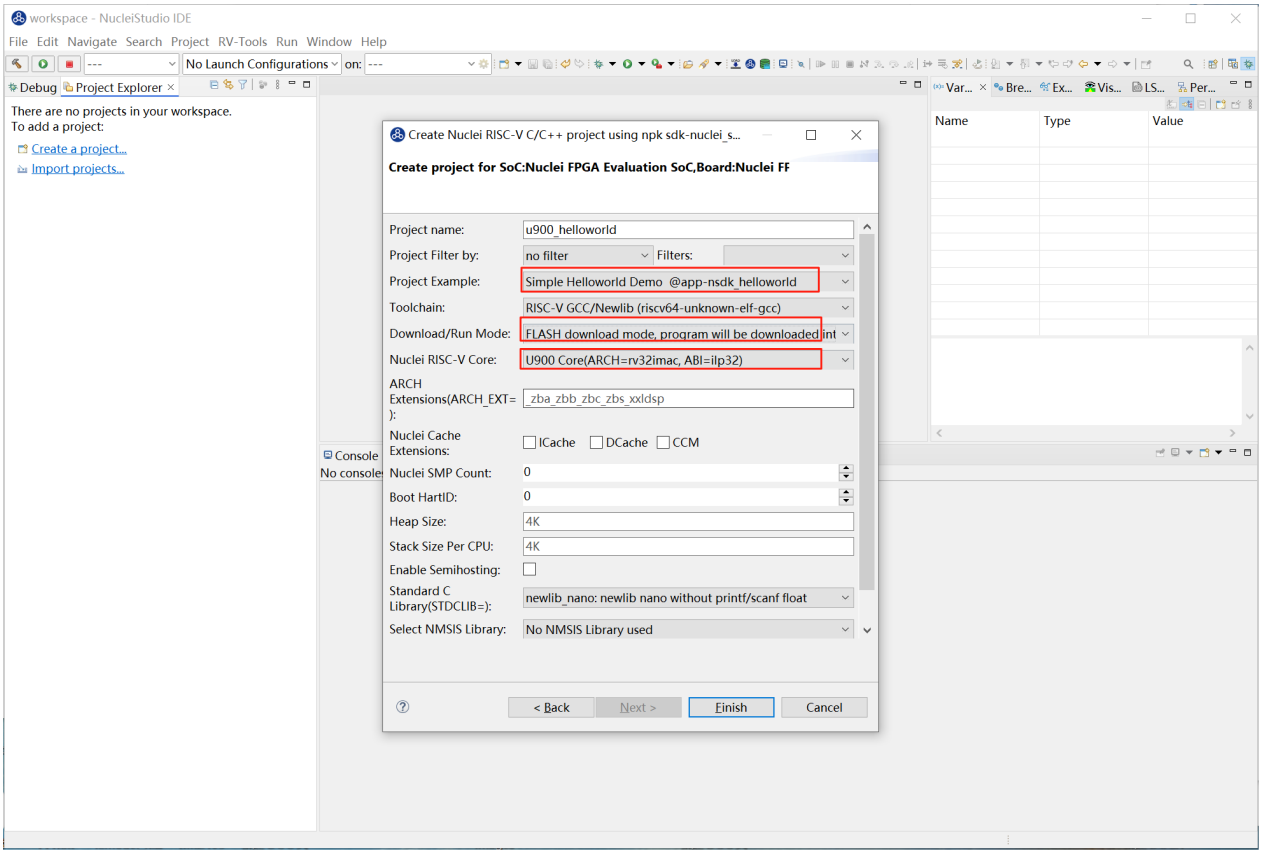
要求版本 >= 202412，下面提供202502版本。

- NucleiStudio 202502 Windows
- NucleiStudio 202502 Linux

Flash Programming 使用演示¶

step1：创建项目，烧写bit

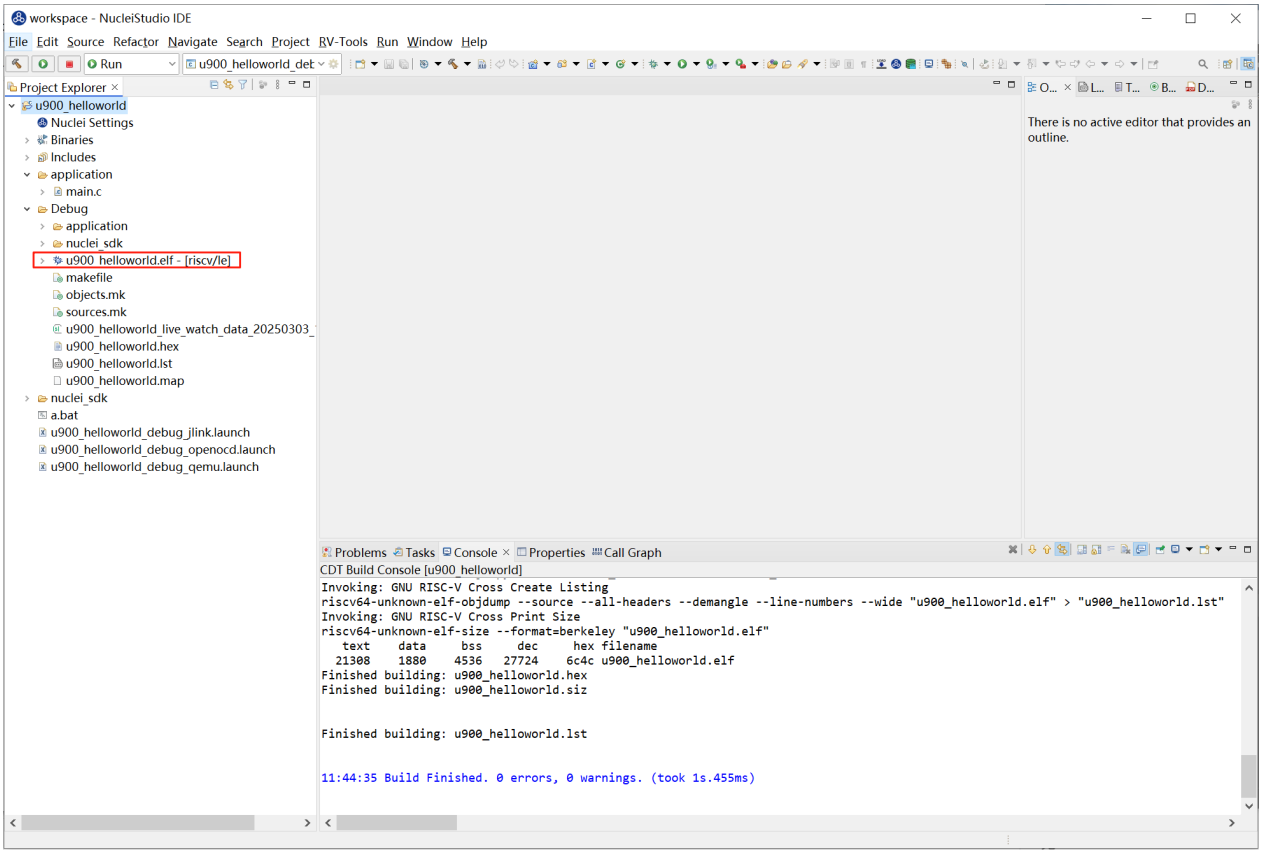
使用0.7.1版本的sdk-nuclei_sdk创建一个u900的helloworld项目，依次选择Simple Helloworld Demo,FLASH下载模式和U900 Core，点击Finsh。



开发板烧写对应的bit即可，这里我们使用trace-u900_best_config_ku060_16M_e85631d489_e82e2771f_202409232110_v3.12.0.bit

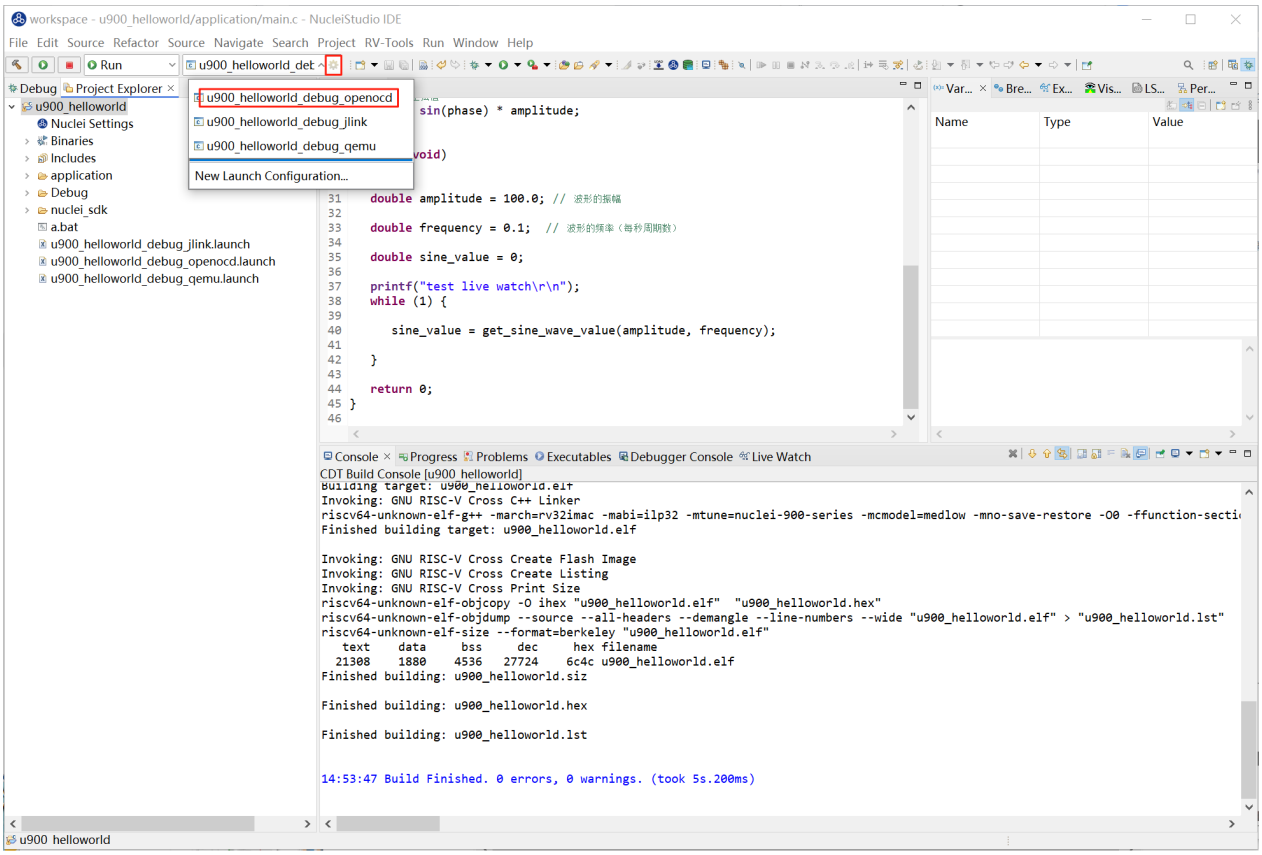
step2：配置编译 Nuclei SDK 原始工程

编译原始工程，确保编译成功以及在 Debug 下可以找到生成的 elf 文件：



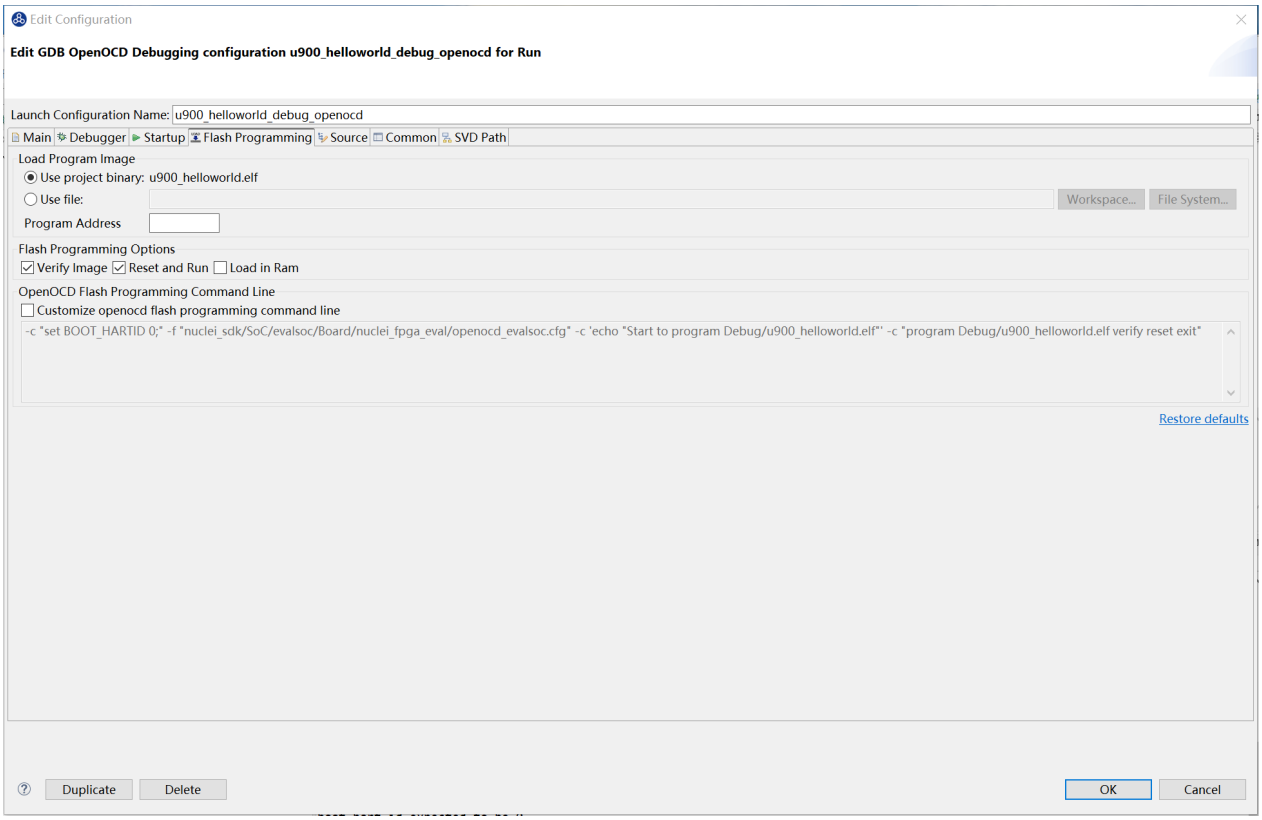
step3：配置Flash Programming选项卡

在Launch Configuration 选中对应调试选项(openocd)，点击edit打开配置页面。



选择 Flash Programming 选项卡，进入配置页面。

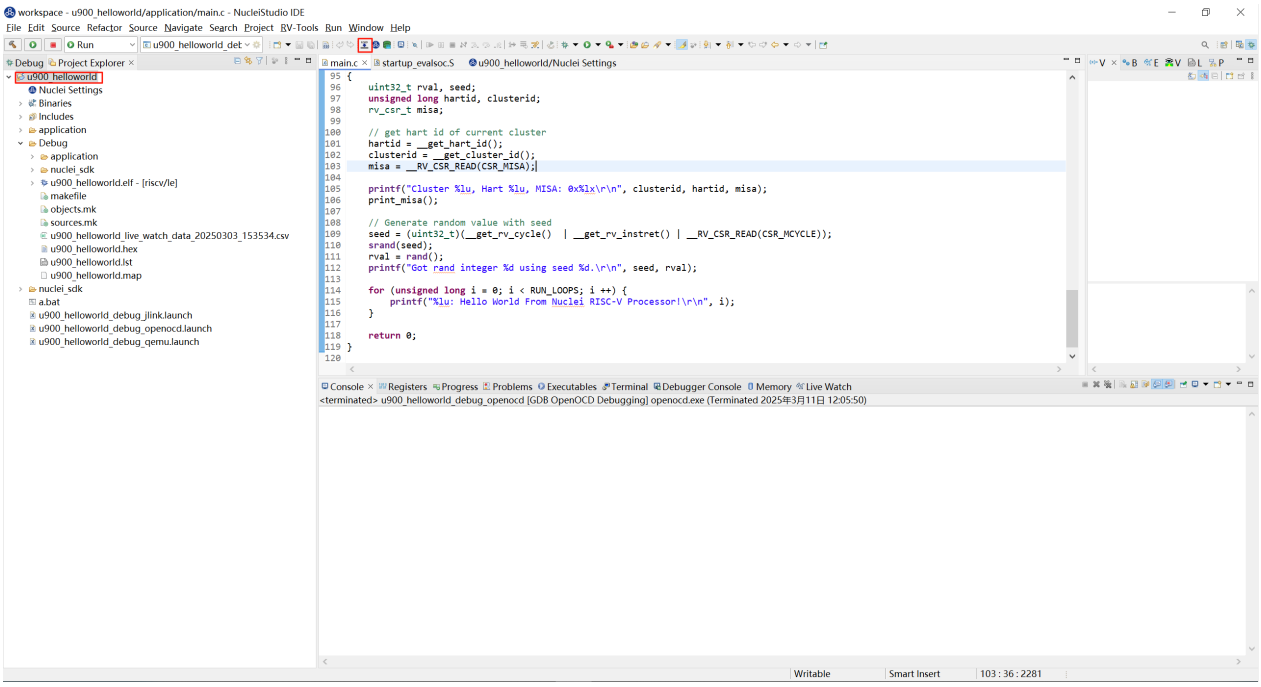
由于是Flash下载模式，这里默认选择的verify image和reset and run即可。



具体配置项内容可参考Nuclei Development Tool Guide

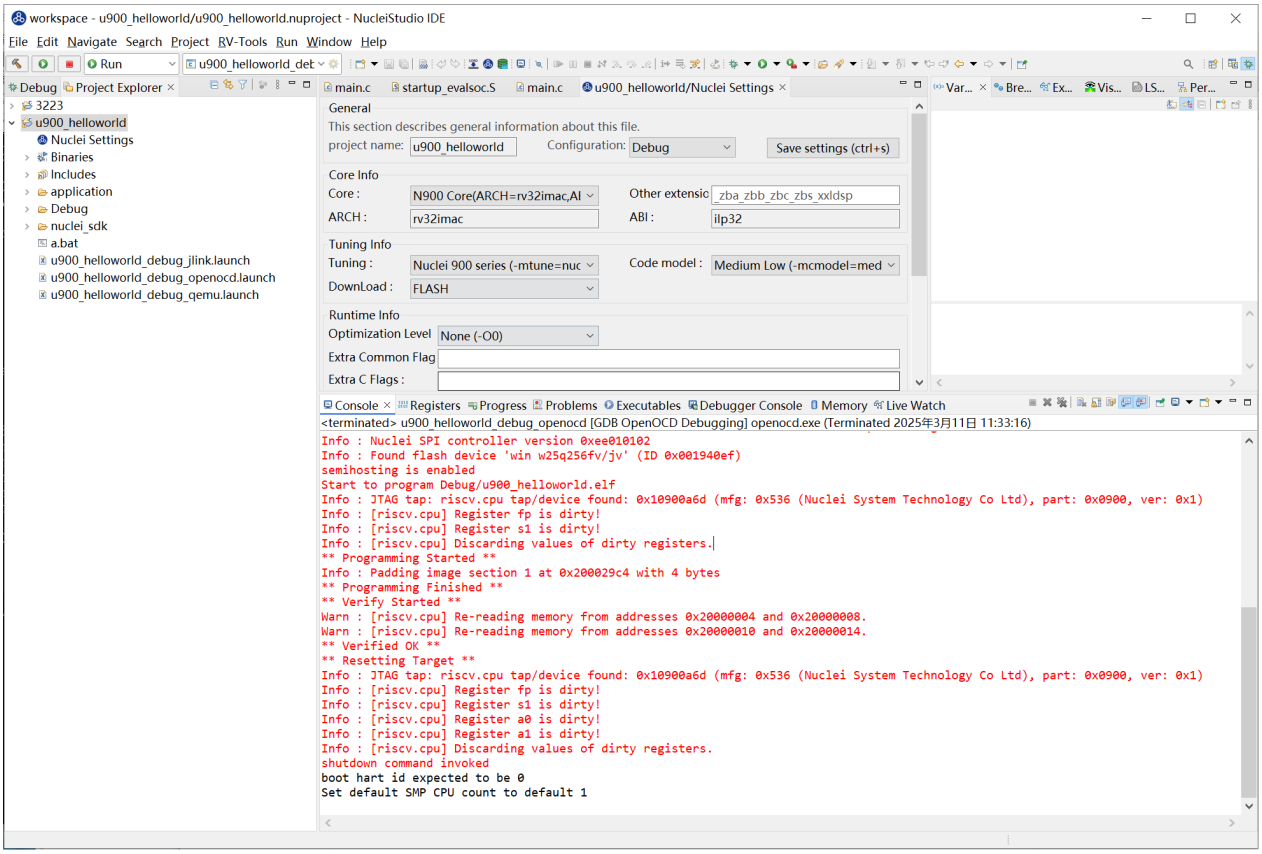
step4：下载

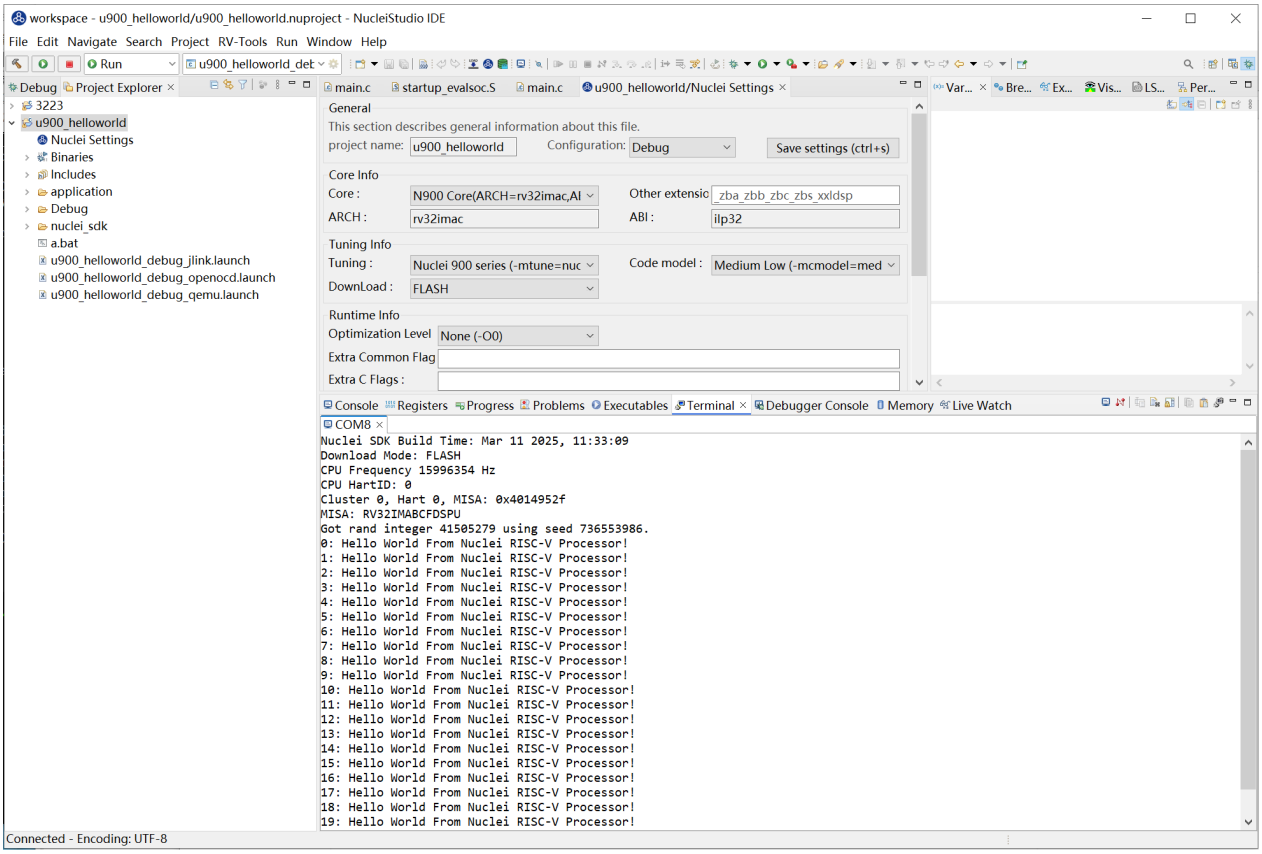
选中项目，点击Flash Programming，下载二进制文件到硬件开发板。



下载成功后，用户可以在 Console 中看到下载结果，确认二进制文件已成功烧录到硬件中。

```
** Programming Started **
Info : Padding image section 1 at 0x200029c4 with 4 bytes
** Programming Finished **
** Verify Started **
Warn : [riscv.cpu] Re-reading memory from addresses 0x20000004 and
0x20000008.
Warn : [riscv.cpu] Re-reading memory from addresses 0x20000010 and
0x20000014.
** Verified OK **
** Resetting Target **
Info : JTAG tap: riscv.cpu tap/device found: 0x10900a6d (mfg:
0x536 (Nuclei System Technology Co Ltd), part: 0x0900, ver: 0x1)
Info : [riscv.cpu] Register fp is dirty!
Info : [riscv.cpu] Register s1 is dirty!
Info : [riscv.cpu] Register a0 is dirty!
Info : [riscv.cpu] Register a1 is dirty!
Info : [riscv.cpu] Discarding values of dirty registers.
shutdown command invoked
```



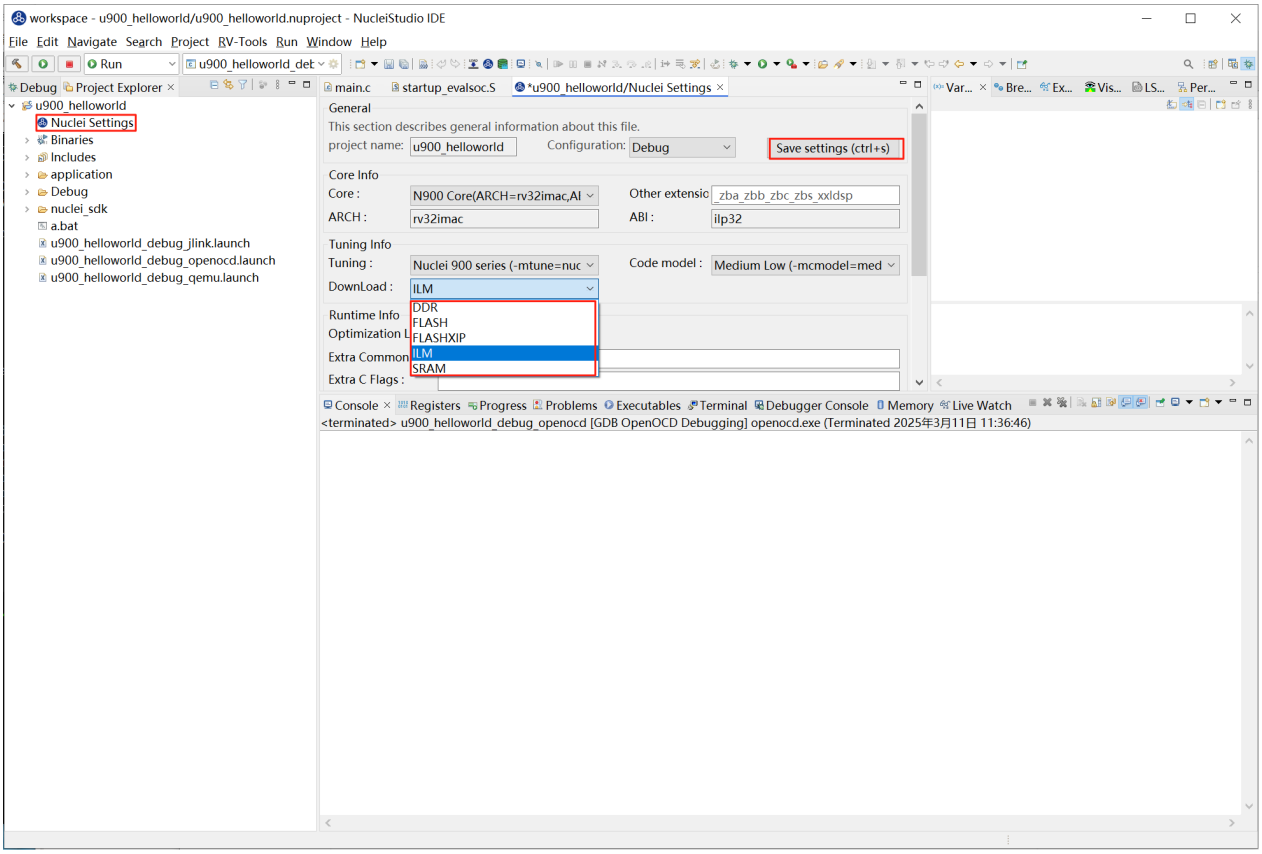


step5：下载到内存的区别

Nuclei Studio有DDR、FLASH、FLASHXIP、ILM、SRAM多种下载模式。

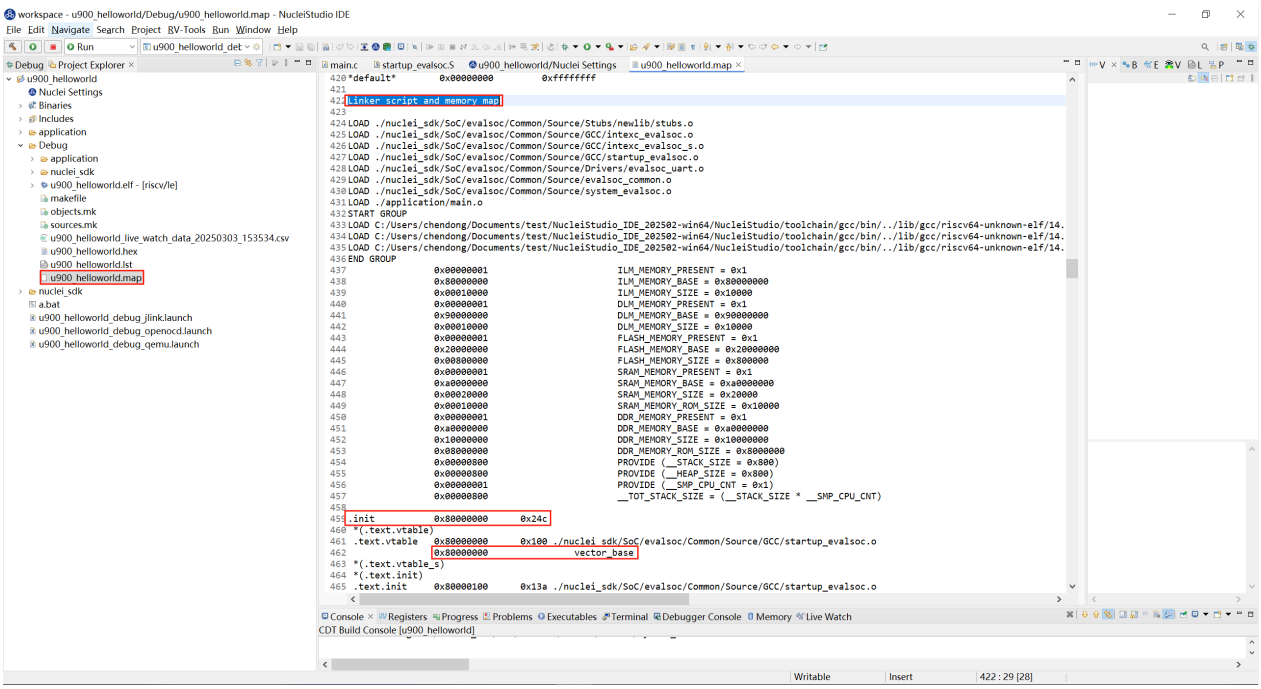
FLASH、FLASHXIP模式按上面的步骤使用即可，而DDR、ILM、SRAM是下载到内存中的与Flash有所区别，下面以ILM为例。

点击Nulcei Settings打开页面，在Download中选择ILM并保存。



重新编译项目，clean project -> build project

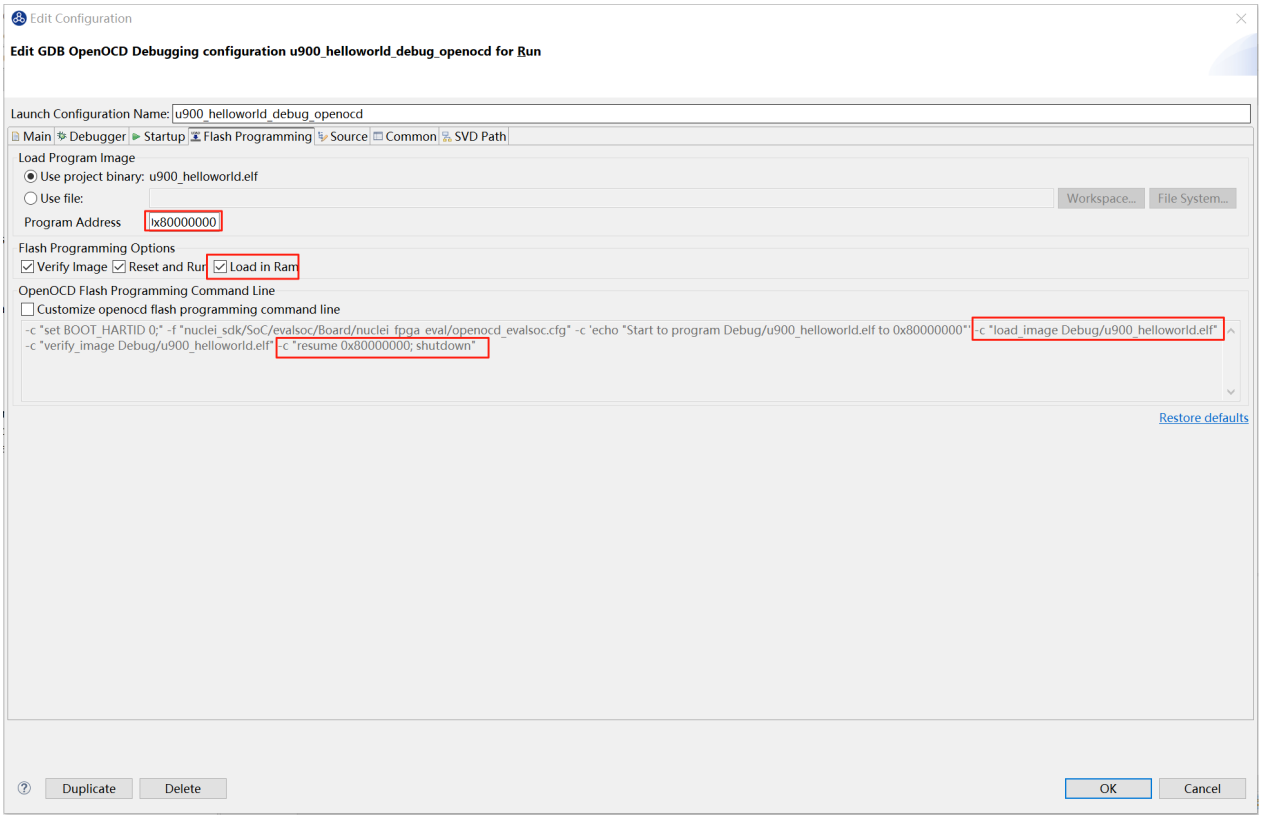
然后打开对应的.map文件，这里是u900_helloworld.map，在里面找到起始加载地址，如下图的0x80000000



打开Flash Programming选项卡，因为是下载到内存，这里要勾选Load in Ram,此时下面的command line会增加load_image命令，

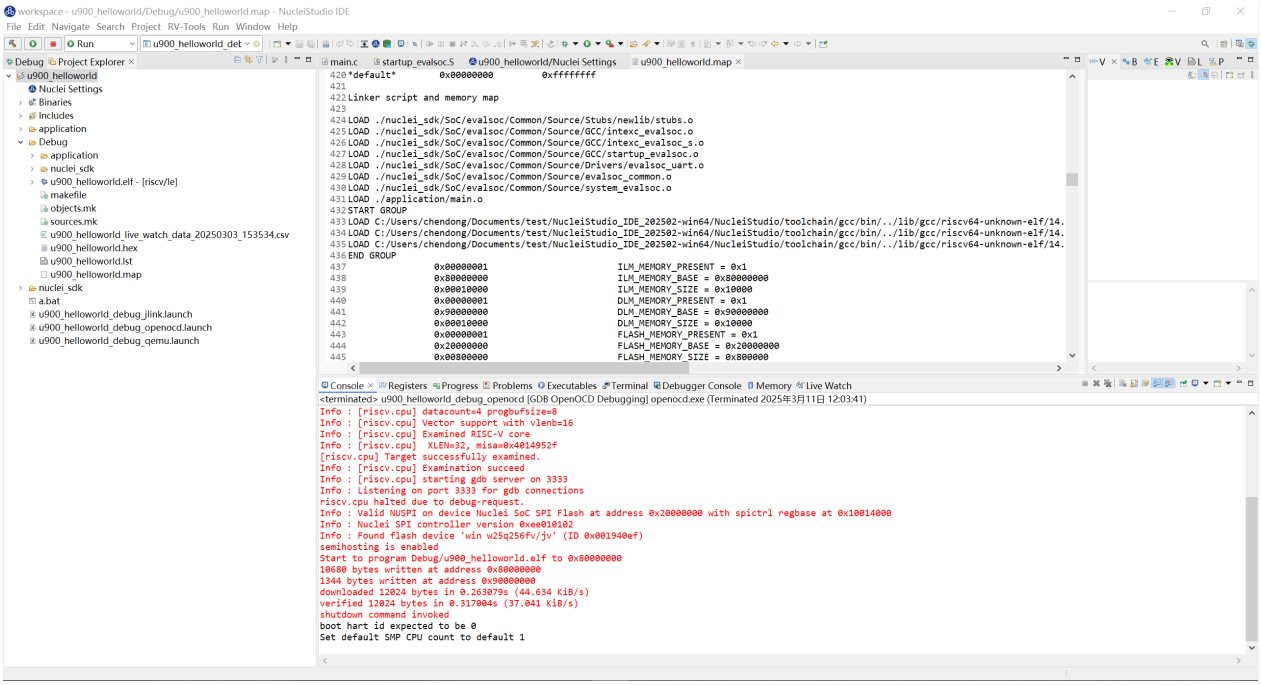
再在Program Address中填入上面获取到的地址0x80000000,command line会带上 resume 0x80000000参数。

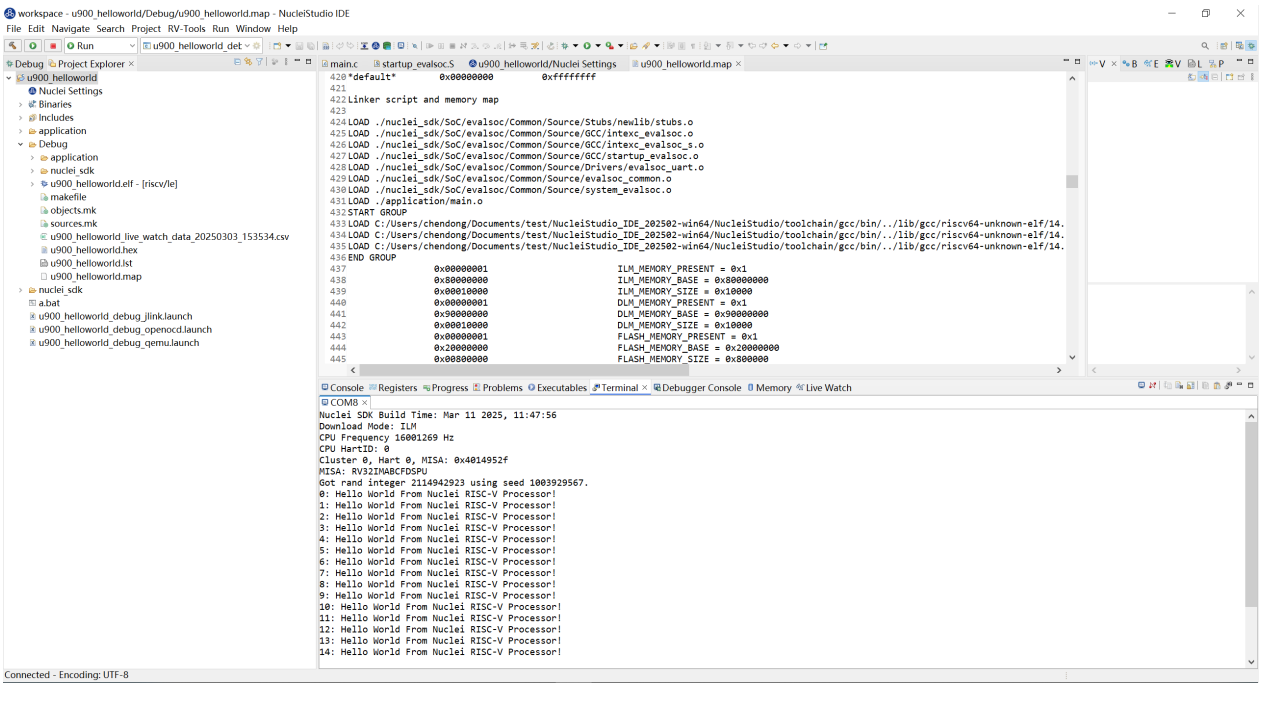
点击OK。



选中项目，点击Flash Programming下载。结果如下。

```
Info : Valid NUSPI on device Nuclei SoC SPI Flash at address
0x20000000 with spictrl regbase at 0x10014000
Info : Nuclei SPI controller version 0xee010102
Info : Found flash device 'win w25q256fv/jv' (ID 0x001940ef)
semihosting is enabled
Start to program Debug/u900_helloworld.elf to 0x80000000
10680 bytes written at address 0x80000000
1344 bytes written at address 0x90000000
downloaded 12024 bytes in 0.263079s (44.634 KiB/s)
verified 12024 bytes in 0.317004s (37.041 KiB/s)
shutdown command invoked
```





step6：可能出现的问题

1. Error: checksum mismatch , attempting binary compare

出现这个错误是因为flash下载和ram下载搞错了，需要在nuclei settings里面进行修改Download模式。

总结¶

Flash Programming 功能为用户提供了一种快速、便捷的方式将编译好的二进制文件下载到硬件开发板中。通过简单的配置，用户可以轻松适配不同的硬件环境，并确保二进制文件的正确烧录。

Live Watch 功能的使用¶

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

背景描述¶

Live Watch 功能依赖 Nuclei OpenOCD >= 2025.02 版本，并且仅支持 Nuclei CPU 配置了 RISC-V SBA 功能。通过 Live Watch，开发者可以在调试过程中实时监控变量的变化，帮助快速定位问题并优化代码性能。

解决方案¶

环境准备¶

Nuclei Studio：

- NucleiStudio 202502 Windows
- NucleiStudio 202502 Linux

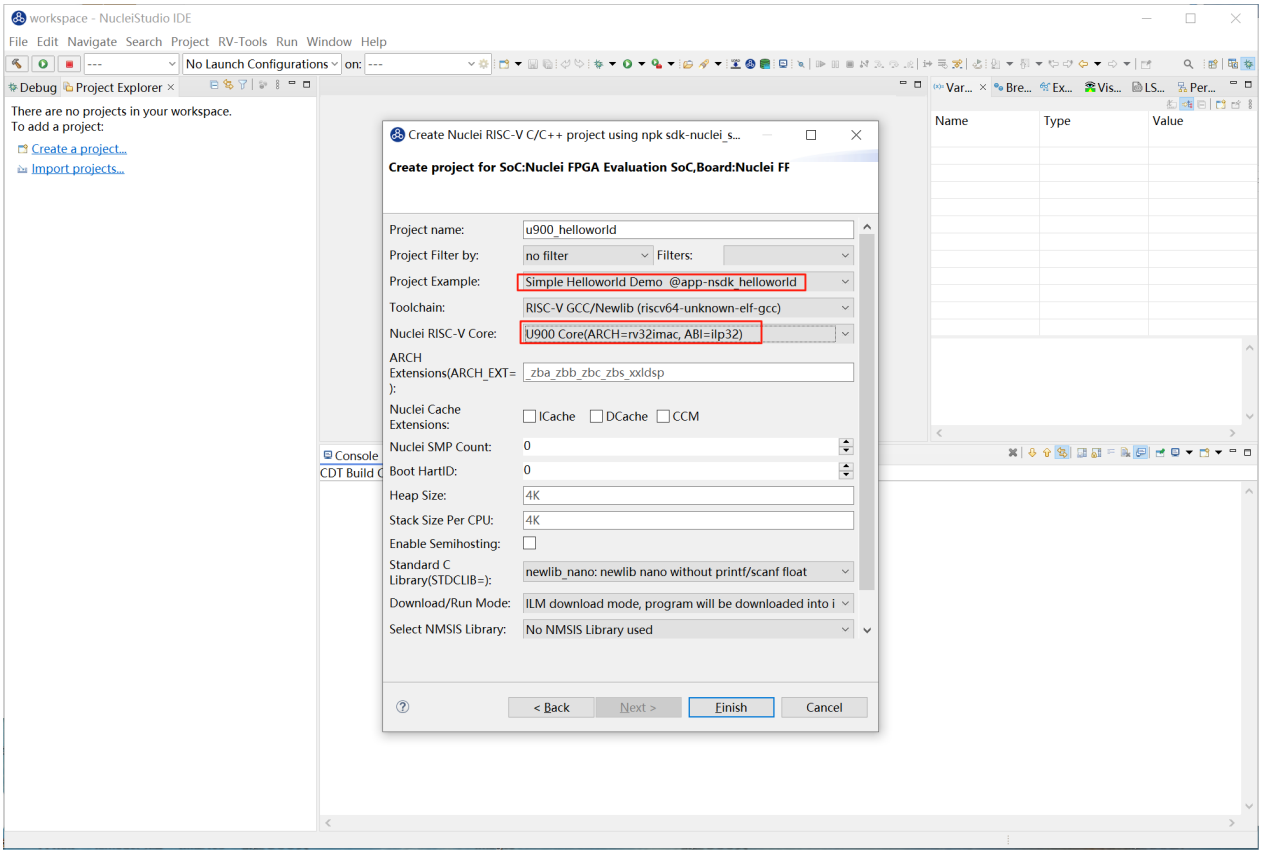
Nuclei OpenOCD：

- 确保安装的 OpenOCD 版本 >= 2025.02，并且支持 RISC-V SBA 功能。

Live Watch 使用演示¶

step1：创建项目，烧写bit

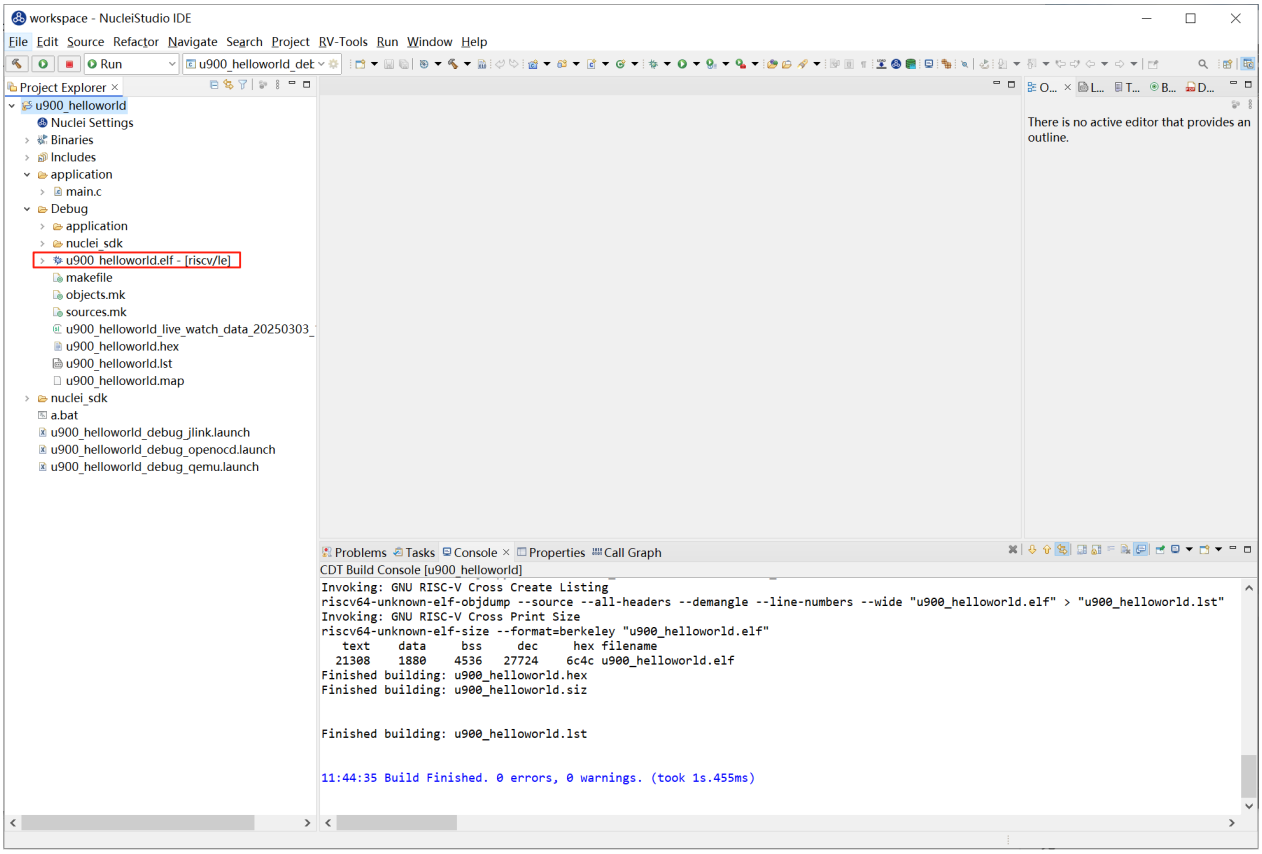
使用0.7.1版本的sdk-nuclei_sdk创建一个u900的helloworld项目，依次选择Simple Helloworld Demo和U900 Core，点击Finsh。



开发板烧写对应的bit即可，这里我们使用trace-u900_best_config_ku060_16M_e85631d489_e82e2771f_202409232110_v3.12.0.bit

step2：编译 Nuclei SDK 原始工程

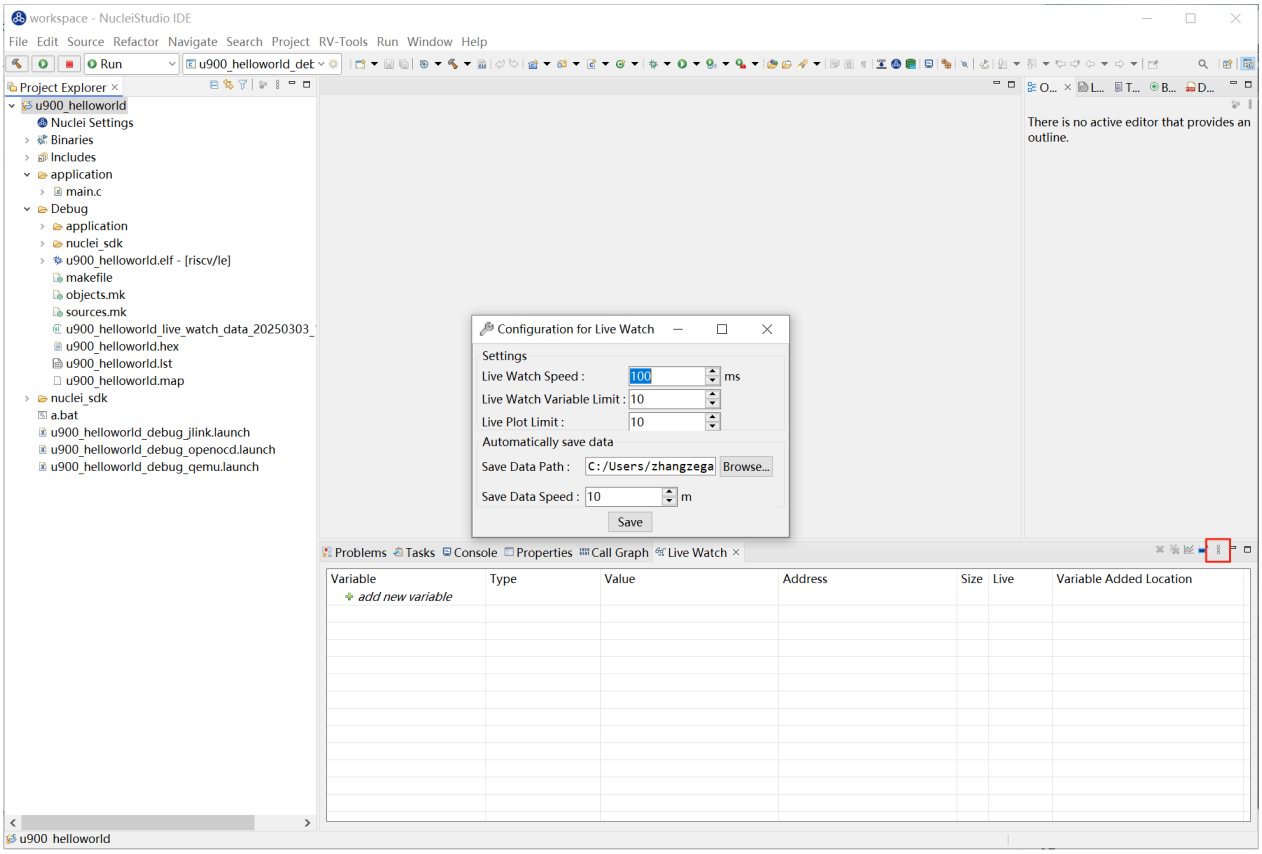
编译原始工程，确保编译成功以及在 Debug 下可以找到生成的 elf 文件：



step3：打开 Live Watch 视图

打开 Live Watch 视图，找到 Live Watch Settings 并根据需要设置相关参数，这里我们直接使用默认值。

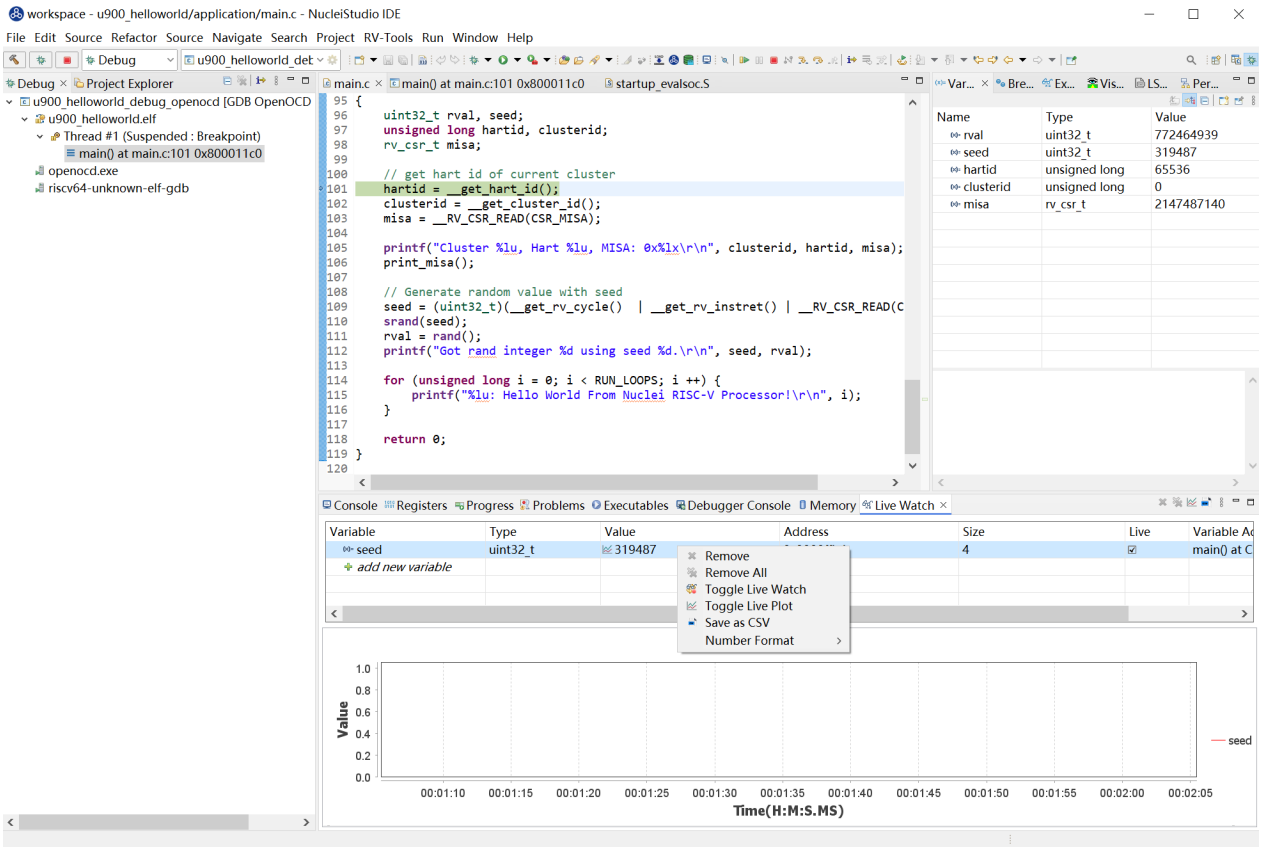
如需配置可参考下图或Nuclei Development Tool Guide



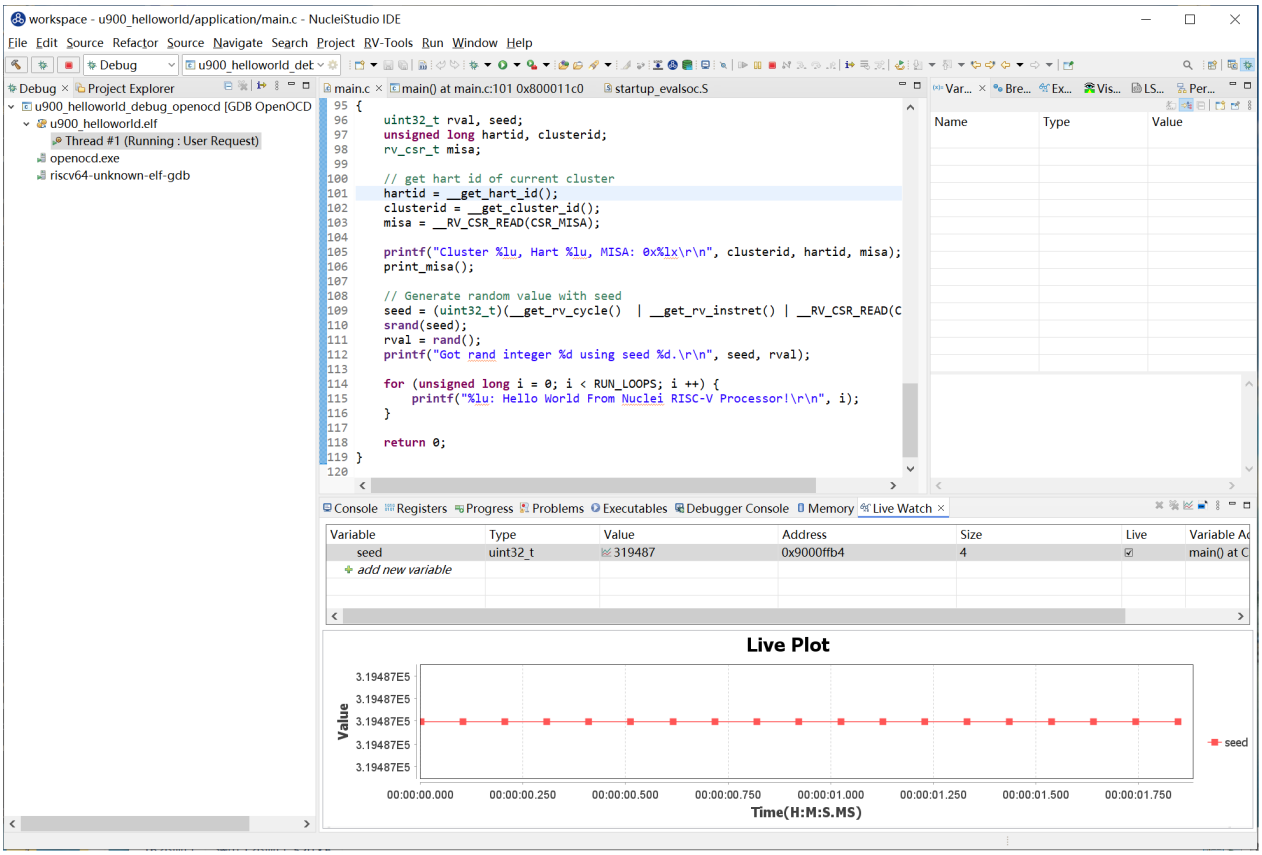
step4：运行Nuclei SDK原始工程

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

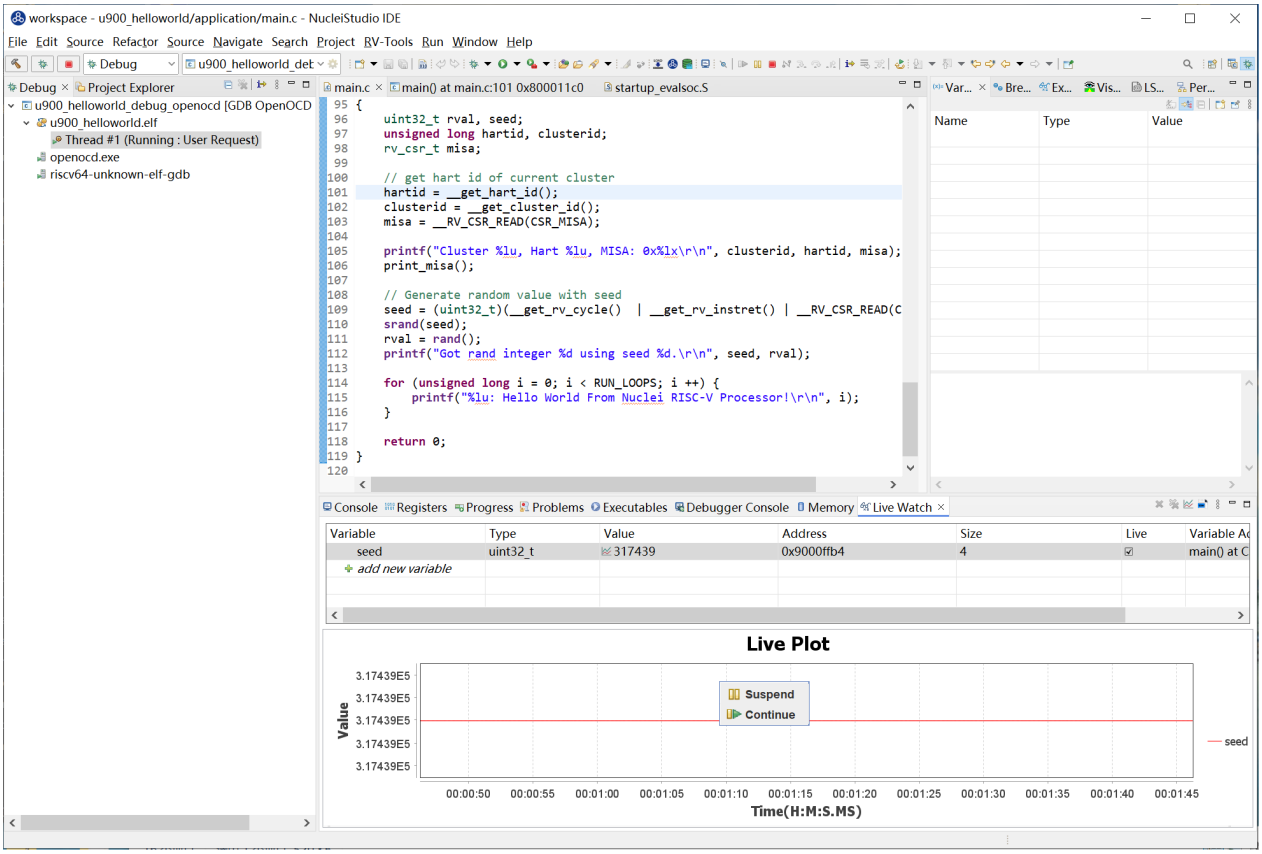
这里想要通过Live　Plot查看变量的变化曲线，选中该条记录，并点击鼠标右键，在弹出的菜单中选中 `Toggle Live Plot` ,Live Plot工具就会弹出。



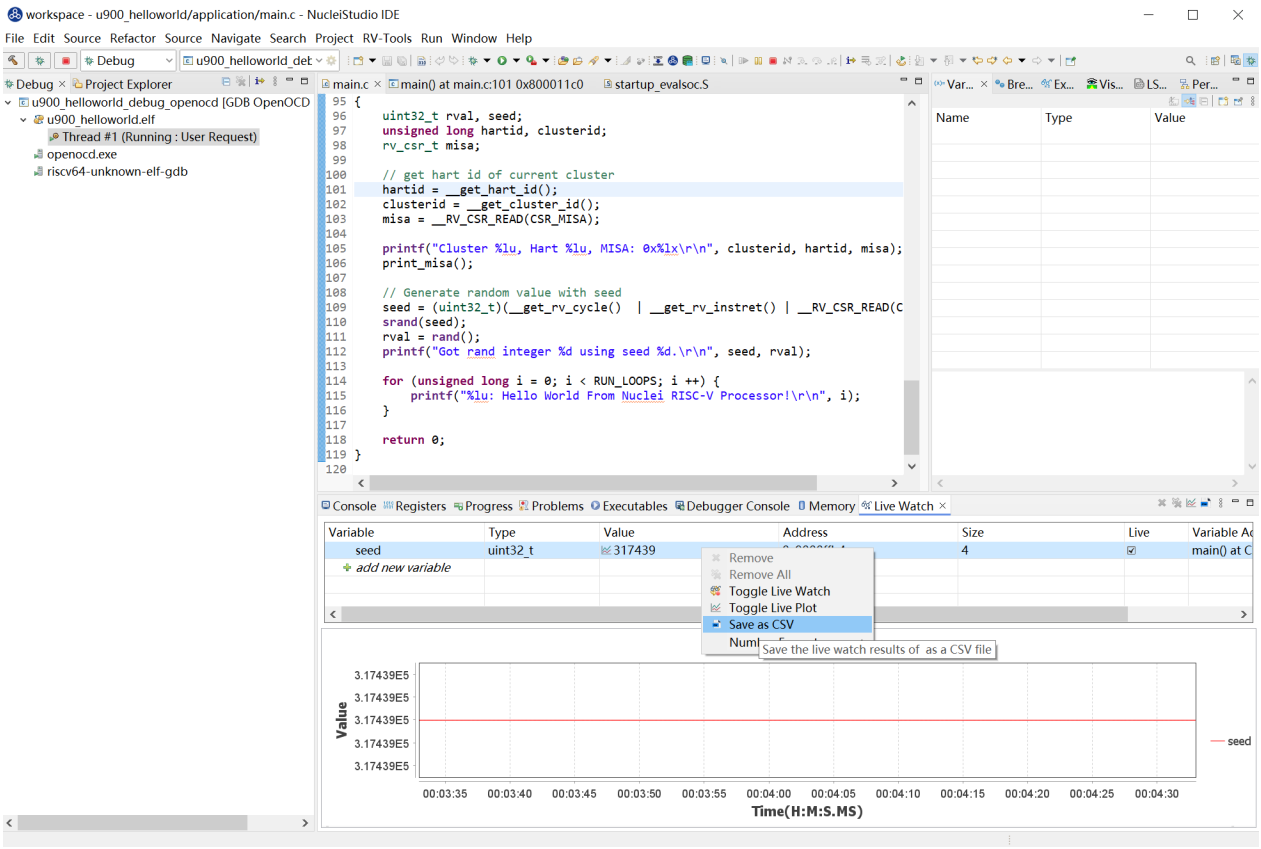
让工程全速运行时，可以看到变量的值，以设定的Live Watch Speed变化，Live Plot绘制的曲线图如下。



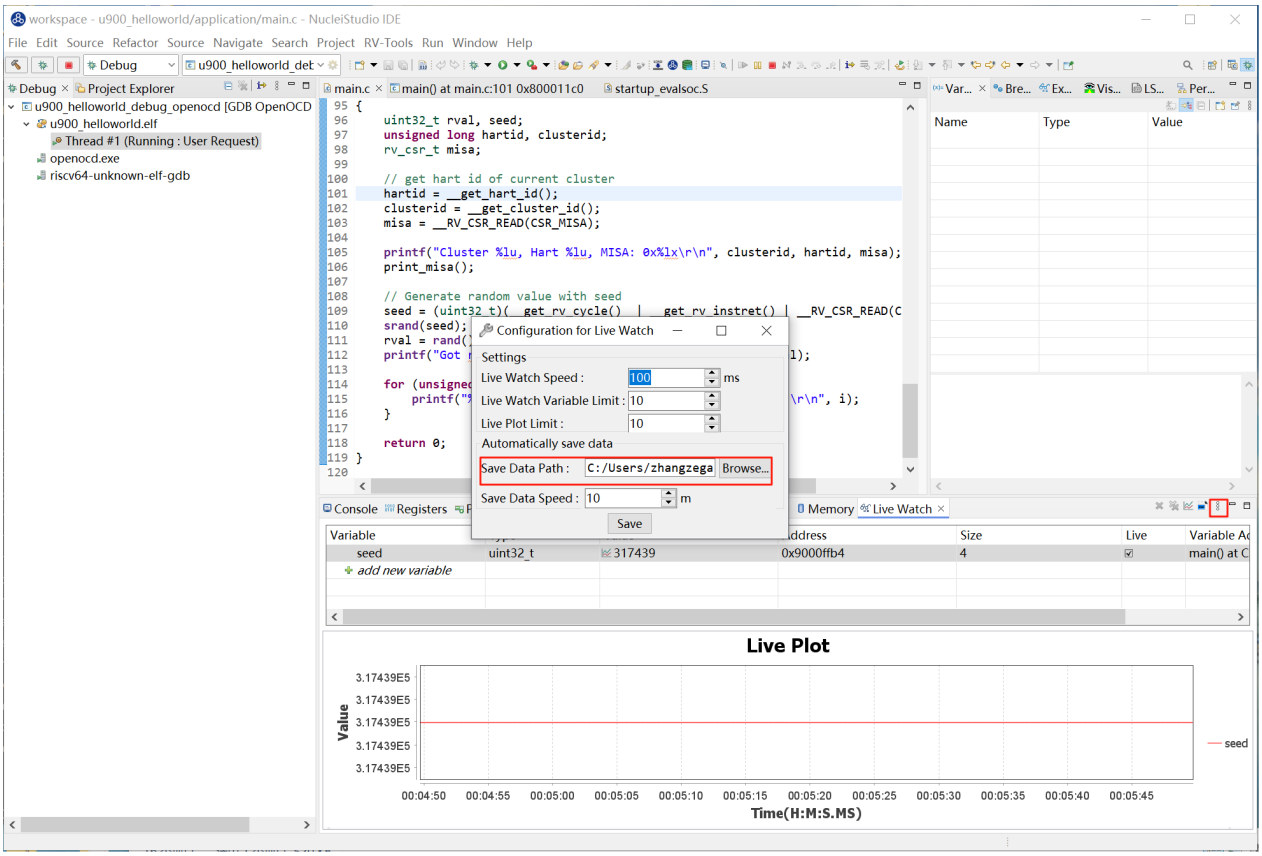
当随着时间数据节点越来越多时，届时会隐藏数据节点。用户可以在Live Plot右键点击Suspend暂停，然后通过滚动鼠标放大曲线，放大到一定倍数会显示节点，鼠标移至节点可查看数据详情；点击 Continue Live Plot则继续绘制曲线。

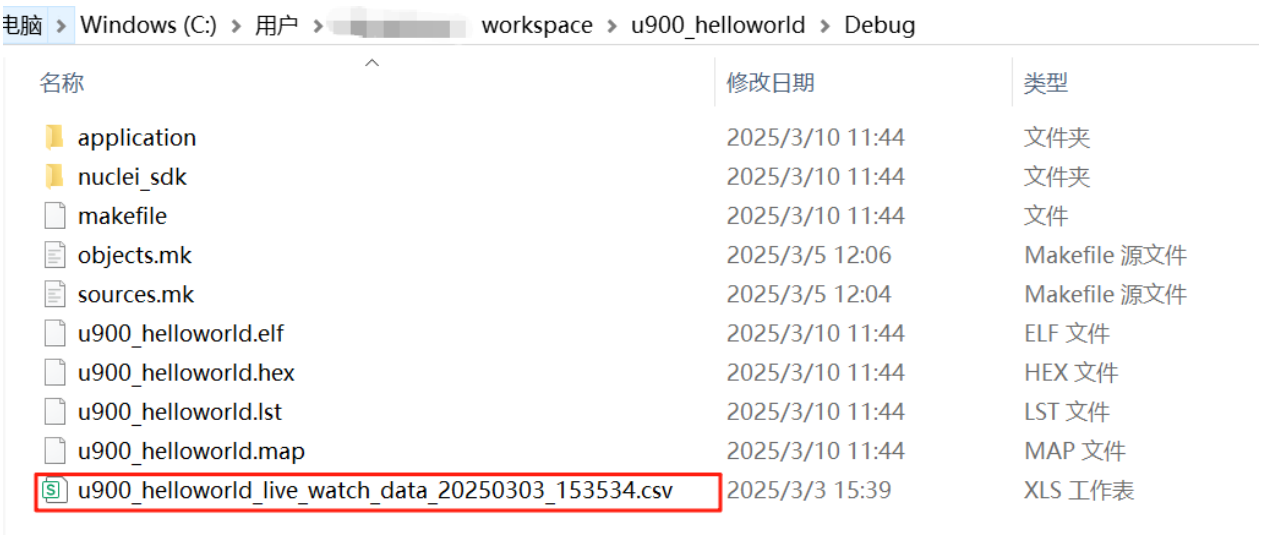


选中seed行,点击鼠标右键，将该变量的结果存为CSV格式文件，用来查阅和使用。



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





总结¶

Live Watch 功能为开发者提供了一个强大的实时监控工具，极大地提升了调试效率和代码优化的能力。通过合理使用 Live Watch，开发者可以更轻松地应对复杂的调试任务，提升开发效率。

在llvm中新增自定义汇编指令教程¶

以下皆以32位指令为例说明

自定义扩展名的识别¶

以下以xnice扩展为例

文件:llvm/lib/Target/RISCV/RISCVFeatures.td

添加内容：

```
def FeatureVendorXnice
    : RISCVExtension<"xnice", 1, 0,
                     "'Xnice' (Xnice extension)">;
def HasVendorXnice
    : Predicate<"Subtarget->hasVendorXnice()">,
    AssemblerPredicate<(all_of FeatureVendorXnice),
                       "'Xnice' (Xnice extension)">;
```

注意：RISCVExtension处第一个xnice为实际llvm编译器可识别的扩展名，后面的1， 0为该扩展的版本号，第二个Xnice只是用于扩展功能描述

自定义汇编指令识别¶

以下以新增一条标准R类型nice指令为例

1、添加对应解码器

文件:llvm/lib/Target/RISCV/Disassembler/RISCVDisassembler.cpp

函数:DecodeStatus RISCVDisassembler::getInstruction32()

内容：

```
TRY_TO_DECODE_FEATURE(RISCV::FeatureVendorXnice,
DecoderTableXnice32,
                      "Xnice extension");
```

2、创建编解码文件

在llvm/lib/Target/RISCV/下创建一个RISCVInstrInfoXnice.td的编解码文件，并在llvm/lib/Target/RISCV/RISCVInstrInfo.td中将该文件include进来

```
include "RISCVInstrInfoXnice.td"
```

3、指令编码

假设该nice指令汇编格式为nice rd, rs1, rs2, 并且使用的是RISC-V预留的custom3区域的编码空间，则编码步骤如下：

- 新建一个XniceInstr的类用于说明XNICE扩展的所有指令的统一格式，由于是R类型指令，所以可以直接从llvm中预先写好的RVInstR类继承而来，否则需要继承其他相匹配的类或者继承基类重新写一个指令类出来， llvm中所有指令类的声明位于llvm/lib/Target/RISCV/RISCVInstrFormats.td
- 通过Predicates限定nice指令所在的扩展（[HasVendorXnice]）以及使用的解码器（DecoderNamespace = "Xnice"）
- 通过def新增一个指令说明，只需要通过XniceInstr以及填充该类在声明时缺少的参数即可完成一条指令的编码，例如自定义的R类型指令只需要再次声明func7，func3以及汇编指令名

完整示例如下：

```
let hasSideEffects = 0, mayLoad = 0, mayStore = 0 in {
 class XniceInstr<bits<7> funct7, bits<3> funct3, string opcodestr>
     : RVInstR<funct7, funct3, OPC_CUSTOM_3, (outs GPR:$rd),
               (ins GPR:$rs1, GPR:$rs2), opcodestr, "$rd, $rs1,
$rs2">;
 }

 let Predicates = [HasVendorXnice], DecoderNamespace = "Xnice" in {
   def NICE  : XniceInstr<0b1111010, 0b001, "nice">;
 }
```

OPC_CUSTOM_3是llvm/lib/Target/RISCV/RISCVInstrFormats.td中已经预留的宏，如果使用的其他编码空间，则可以直接查找更改

对于def后大写的NICE一般用于intrinsic或者自定向量化等调用，只做汇编时可以只给汇编指令名的大写格式用于区分

另外，以上示例中没有限制指令在RV32/64下的使用场景，因此RV32/64下都可识别，如果需要限定只在RV32下使用，则需要额外在Predicates中指定扩展时同时进行限定，例如[HasVendorXxlczbitop, IsRV32]

使用说明¶

使用时与GCC一样，只需要将`xnice`通过`-march`选项传递给llvm编译器即可，例如`-march=rv32imafdc_xnice`

参考链接¶

llvm table gen语法手册 https://llvm.org/docs/TableGen/ProgRef.html

PLCT关于在LLVM中添加RISC-V的自定义指令的示例 https://www.bilibili.com/video/BV1JR4y1J7he

nuclei自定义vpu指令的扩展识别及汇编实现 https://github.com/riscv-mcu/llvm-project/commit/f5d025b9800f3cd662e93c11eb7c7b0f65ca4472

如何使用芯来提供的DebugMap寄存器分析错误现场¶

首先需要确定硬件支持DebugMap功能¶

core的顶层有一个信号叫做dm_map_enable，这个信号接1表示使能DebugMap功能

> 详情参见 *Nuclei_CPU_Debug_Function_Specification.pdf* 文档的 *Debug Control Interface* 部分内容

什么是DebugMap寄存器¶

DebugMap功能就是当Core被hang的时候可以通过OpenOCD查看core内部的状态，将若干内部状态映射到DM寄存器中，目前只实现了下面三个状态的映射：

- 00: Commit PC(i0 for dual issue)
- 16: ICache miss address(ICache is supported)
- 32: DCache address waiting for retire(DCache is supported)

> 详情参见 *Nuclei_CPU_Debug_Function_Specification.pdf* 文档的 *CFR0 (Custom Feature Register0)* 部分内容

OpenOCD里DebugMap的输出信息¶

在使用OpenOCD连接FPGA/芯片时，经常会看到类似下面这样的输出信息：

```
Info : coreid=0, nuclei debug map reg 00: 0xa00003ac, 16:
0xa0003240, 32: 0x10003014
```

- coreid 表示当前输出的是哪个core的debug-map信息

可能出现的错误现场¶

错误现场一：

```
Info : Using libusb driver
Info : clock speed 1000 kHz
Info : JTAG tap: riscv.cpu tap/device found: 0x10900a6d (mfg:
```

```
0x536 (Nuclei System Technology Co Ltd), part: 0x0900, ver: 0x1)
Info : [riscv.cpu] datacount=4 progbufsize=8
Info : coreid=0, nuclei debug map reg 00: 0xa0000496, 16:
0xa0003140, 32: 0x10002ff8
Error: [riscv.cpu] Unable to halt hart 0. dmcontrol=0x00000001,
dmstatus=0x00400ca2
Error: [riscv.cpu] Fatal: Hart 0 failed to halt during examine()
Warn : target riscv.cpu examination failed
Info : [riscv.cpu] datacount=4 progbufsize=8
Error: Hart 0 doesn't exist.
Error: Fatal: Failed to read s0 from hart 0.
Info : [riscv.cpu] datacount=4 progbufsize=8
Error: Hart 0 doesn't exist.
Error: Fatal: Failed to read s0 from hart 0.
Info : starting gdb server for riscv.cpu on 22800
Info : Listening on port 22800 for gdb connections
Error: Target not examined yet
```

错误现场二：

```
Info : libusb_open() failed with LIBUSB_ERROR_NOT_FOUND
Info : no device found, trying D2xx driver
Info : D2xx device count: 2
Info : Connecting to "(null)" using D2xx mode...
Info : clock speed 1000 kHz
Info : JTAG tap: riscv0.cpu tap/device found: 0x10300a6d (mfg:
0x536 (Nuclei System Technology Co Ltd), part: 0x0300, ver: 0x1)
Info : [riscv0.cpu] datacount=4 progbufsize=2
Info : coreid=0, nuclei debug map reg 00: 0xa0000496, 16:
0xa0003140, 32: 0x10002ff8
Info : Examined RISC-V core; found 1 harts
Info :  hart 0: XLEN=32, misa=0x40001127
[riscv0.cpu] Target successfully examined.
Info : starting gdb server for riscv0.cpu on 3333
Info : Listening on port 3333 for gdb connections
Started by GNU MCU Eclipse
Info : Listening on port 6666 for tcl connections
Info : Listening on port 4444 for telnet connections
Info : accepting 'gdb' connection on tcp/3333
Warn : Prefer GDB command "target extended-remote :3333" instead
of "target remote :3333"
Error: Timed out after 2s waiting for busy to go low
(abstractcs=0x2001004). Increase the timeout with riscv
set_command_timeout_sec.
```

```
Error: Abstract command ended in error 'busy'
(abstractcs=0x2001104)
```

如何正确利用DebugMap分析错误现场¶

- 在出现Core被hang的现象之后，需要在不断电、不复位的情况下再次使用OpenOCD连接FPGA/芯片，此时OpenOCD输出的DebugMap才可被用于分析错误现场
- “00”：当前Commit的PC——用来指示最近正在Commit的PC，通过此信息可以大概推测CPU跑到了什么PC位置
- “16”：配置了ICache的话，记录ICache最近发出去的地址（暂时没有记录ILM的地址），理论上ICache有2个Oustanding，记录的是那个最先发出去还没有返回Response的地址
- “32”：配置了DCache 的话，记录DCache最近发出去的地址（DLM、Mem也可以被记录，暂时没有记录PPI/FIO发出去的地址），理论上DCache有很多个Oustanding，记录的是那个最先发出去还没有返回Response的地址

通过OpenOCD读取其他DebugMap寄存器¶

OpenOCD里有一组 *nuclei expose_cpu_core nuclei examine_cpu_core* 命令，可以使用这两个命令读取其他DebugMap寄存器， 详细参见 https://doc.nucleisys.com/nuclei_tools/openocd/intro.html#debug-map-feature

OpenOCD里的命令实现及使用方法 source code

- 注意 *nuclei expose_cpu_core* 命令需要在init命令之前使用
- *nuclei examine_cpu_core* 在init命令之后使用，也可以在gdb/telent连接上后使用，注意gdb给openocd发送命令需要使用monitor关键词 *monitor nuclei examine_cpu_core*

在binutils中新增自定义汇编指令教程¶

以下皆以32位指令为例说明

自定义扩展名的识别¶

以下以xnice扩展为例

文件:bfd\elfxx-riscv.c

riscv_supported_vendor_x_ext[] 函数:

```
static struct riscv_supported_ext riscv_supported_vendor_x_ext[] ={
  {"xnice",   ISA_SPEC_CLASS_DRAFT, 1, 0, 0},
}
```

Tips:该函数负责添加扩展名称和版本号，其中前面两位1,0为该扩展版本号

riscv_multi_subset_supports 函数:

```
/* Each instuction is belonged to an instruction class
INSN_CLASS_*.
   Call riscv_subset_supports to make sure if the instuction is
valid.  */

bool
riscv_multi_subset_supports (riscv_parse_subset_t *rps,
                 enum riscv_insn_class insn_class)
{
  switch (insn_class)
    {
      case INSN_CLASS_XNICE:
        return riscv_subset_supports (rps, "xnice");
    }
}
```

Tips: switch里面是要添加的内容，添加了xnice扩展的指令所对应的INSN_CLASS_XNICE与xnice扩展之间的联系

riscv_implicit_subsets[] 函数:(可选)

```
/* Please added in order since this table is only run once time.
*/
static struct riscv_implicit_subset riscv_implicit_subsets[] ={
  {"xnice", "+zve32x", check_implicit_always},
}
```

Tips:该函数控制自定义的xnice扩展是否依赖其他扩展，如果不依赖，则不需要添加。假设依赖zve32x扩展，则需要在该函数内按上面形式添加依赖关系，若依赖多个扩展，则在zve32x扩展后面继续添加

文件 include\opcode\riscv.h

```
enum riscv_insn_class
{
  INSN_CLASS_XNICE,
}
```

Tips:该文件主要负责在riscv_insn_class枚举类中，对INSN_CLASS_XNICE进行声明

自定义汇编指令识别¶

以下以新增一条标准R类型nice指令为例

1、添加指令编码

假设该nice指令汇编格式为nice rd, rs1, rs2, 并且使用的是RISC-V预留的custom3区域的编码空间，其编码为：

Inst. format	Func7	rs2	rs1	Func3	rd	opcode
xnice rd,rs1,rs2	1011101	rs2	rs1	001	rd	1111011

- 生成编译器所需的opcode宏(推荐使用riscv-opcodes https://github.com/riscv/riscv-opcodes/tree/master 仓库)

```
git clone https://github.com/riscv/riscv-opcodes.git

cd riscv-opcodes/extensions/unratified/

vim rv_xnice //在该文件夹下创建xnice扩展指令文件(文件名规则是rv_name)，并根据指令模板添加一条指令

nice rd rs1 rs2 31..25=0x5D 14..12=1 6..2=0x1E 1..0=3 //此为需要添加
```

```
的指令

cd ../../

make EXTENSIONS='unratified/rv_xnice'
```

上述步骤后得到了opcode宏，在 riscv-opcodes/encoding.out.h 文件中, 如下所示:

```
#define MATCH_NICE 0xba00107b
#define MASK_NICE 0xfe00707f
```

注意：也可以根据编码手动生成宏，规则为：MATCH_NICE 的编码是未定义位置全为0，其余位置不变。MASK_NICE 的编码是未定义位置全为0，其余位置全为1.

- 在 include\opcode\riscv-opc.h 文件中，添加上述生成的宏

2、添加扩展与扩展指令编码之间的联系

文件：opcodes\riscv-opc.c

riscv_opcodes[]函数:

```
const struct riscv_opcode riscv_opcodes[] =
{
/* name, xlen, isa, operands, match, mask, match_func, pinfo.  */
{"xnice",  0, INSN_CLASS_XNICE,  "d,s,t", MATCH_XNICE, MASK_XNICE,
match_opcode, 0 },
}
```

Tips: 第一个0代表该指令对xlen没有要求。d,s,t 分别代表rd,rs1,rs2, 其中对应的映射关系可在 gas/config/tc-riscv.c 文件 validate_riscv_insn函数中查找

使用说明¶

使用时需要将xnice通过-march选项传递给编译器，例如-march=rv32imafdc_xnice

参考链接:¶

修改binutils在RISC-V上添加汇编指令:
https://blog.cyyself.name/add-compile-instr-for-riscv/

nuclei自定义vpu指令的扩展识别及汇编实现:
https://github.com/riscv-mcu/riscv-binutils-gdb/commit/c8806f4bd8c1a1673ec61ad3badfc3d490fa52f7

OpenOCD 中 Nuclei 交叉触发功能使用指南¶

功能概述¶

为满足 AMP 多核调试中同步暂停(halt)与恢复(resume)的需求，Nuclei RISC-V CPU实现了 cross-trigger 功能，OpenOCD 已集成以下两种同步控制功能：

1. 同步暂停组 (halt_group) - 组内任一核暂停时，其他成员自动同步暂停
2. 同步恢复组 (resume_group) - 组内任一核恢复运行时，其他成员自动同步恢复

基本命令格式：

```
# add target to halt_group
nuclei cti halt_group on $_TARGETNAME0 $_TARGETNAME1

# remove target from halt_group
nuclei cti halt_group off $_TARGETNAME0 $_TARGETNAME1

# add target to resume_group
nuclei cti resume_group on $_TARGETNAME0 $_TARGETNAME1

# remove target from resume_group
nuclei cti resume_group off $_TARGETNAME0 $_TARGETNAME1
```

配置文件示例¶

1. 同步暂停组配置¶

```
adapter_khz     1000

interface ftdi
ftdi_vid_pid 0x0403 0x6010
ftdi_oscan1_mode off

transport select jtag

ftdi_layout_init 0x0008 0x001b
ftdi_layout_signal nSRST -oe 0x0020 -data 0x0020
ftdi_layout_signal TCK -data 0x0001
ftdi_layout_signal TDI -data 0x0002
```

```
ftdi_layout_signal TDO -input 0x0004
ftdi_layout_signal TMS -data 0x0008
ftdi_layout_signal JTAG_SEL -data 0x0100 -oe 0x0100

set _CHIPNAME0 riscv0
jtag newtap $_CHIPNAME0 cpu -irlen 5 -expected-id 0x10900a6d
set _TARGETNAME0 $_CHIPNAME0.cpu
target create $_TARGETNAME0 riscv -chain-position $_TARGETNAME0 -
coreid 0

set _CHIPNAME1 riscv1
jtag newtap $_CHIPNAME1 cpu -irlen 5 -expected-id 0x10900a6d
set _TARGETNAME1 $_CHIPNAME1.cpu
target create $_TARGETNAME1 riscv -chain-position $_TARGETNAME1 -
coreid 0

init
#reset

if {[ info exists pulse_srst]} {
  ftdi_set_signal nSRST 0
  ftdi_set_signal nSRST z
}

# 添加目标到暂停组
nuclei cti halt_group on $_TARGETNAME0 $_TARGETNAME1

foreach t [target names] {
  targets $t
  halt
}
```

2. 同步恢复组配置¶

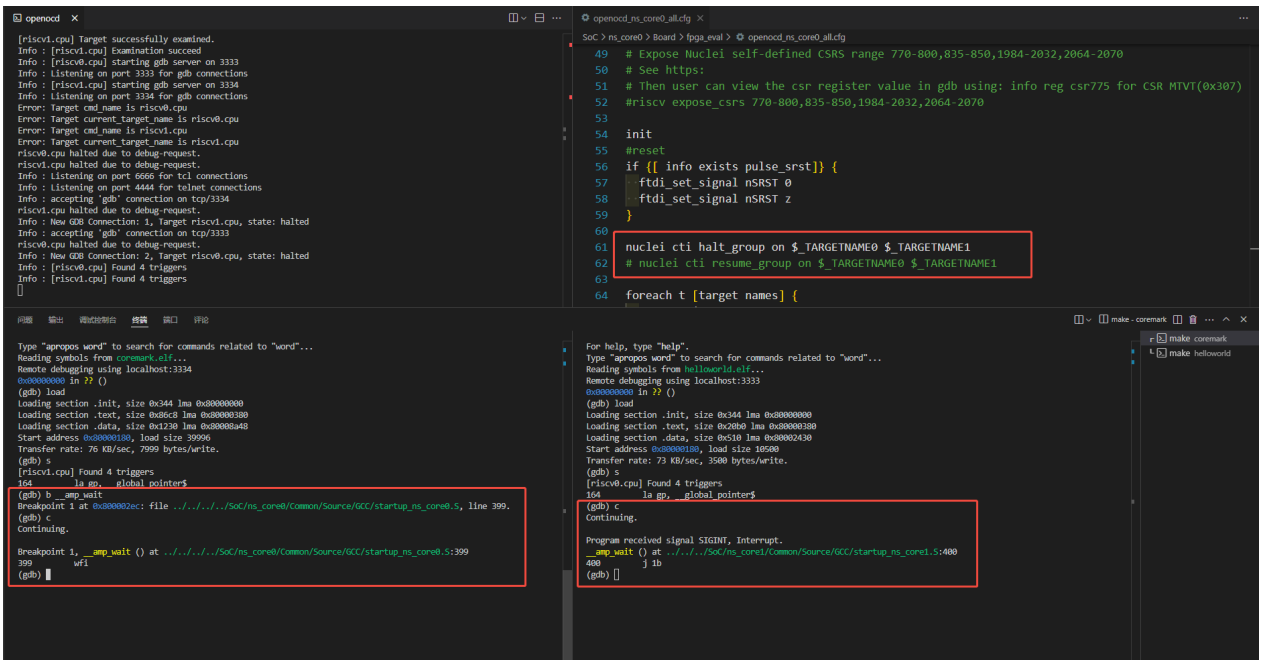
```
adapter_khz     1000

interface ftdi
ftdi_vid_pid 0x0403 0x6010
ftdi_oscan1_mode off

transport select jtag

ftdi_layout_init 0x0008 0x001b
ftdi_layout_signal nSRST -oe 0x0020 -data 0x0020
ftdi_layout_signal TCK -data 0x0001
```

```
ftdi_layout_signal TDI -data 0x0002
ftdi_layout_signal TDO -input 0x0004
ftdi_layout_signal TMS -data 0x0008
ftdi_layout_signal JTAG_SEL -data 0x0100 -oe 0x0100

set _CHIPNAME0 riscv0
jtag newtap $_CHIPNAME0 cpu -irlen 5 -expected-id 0x10900a6d
set _TARGETNAME0 $_CHIPNAME0.cpu
target create $_TARGETNAME0 riscv -chain-position $_TARGETNAME0 -
coreid 0

set _CHIPNAME1 riscv1
jtag newtap $_CHIPNAME1 cpu -irlen 5 -expected-id 0x10900a6d
set _TARGETNAME1 $_CHIPNAME1.cpu
target create $_TARGETNAME1 riscv -chain-position $_TARGETNAME1 -
coreid 0

init
#reset

if {[ info exists pulse_srst]} {
  ftdi_set_signal nSRST 0
  ftdi_set_signal nSRST z
}

# add target to resume_group
nuclei cti resume_group on $_TARGETNAME0 $_TARGETNAME1

foreach t [target names] {
  targets $t
  halt
}
```

命令行验证步骤¶

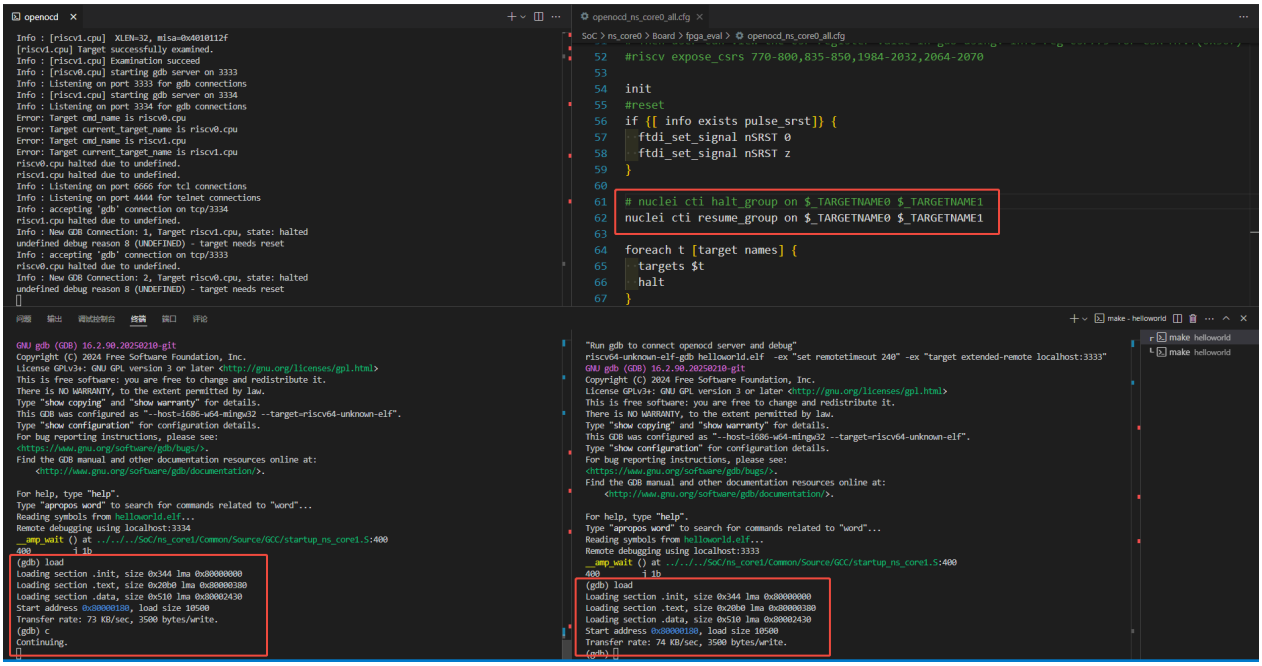
1. 同步暂停组验证¶

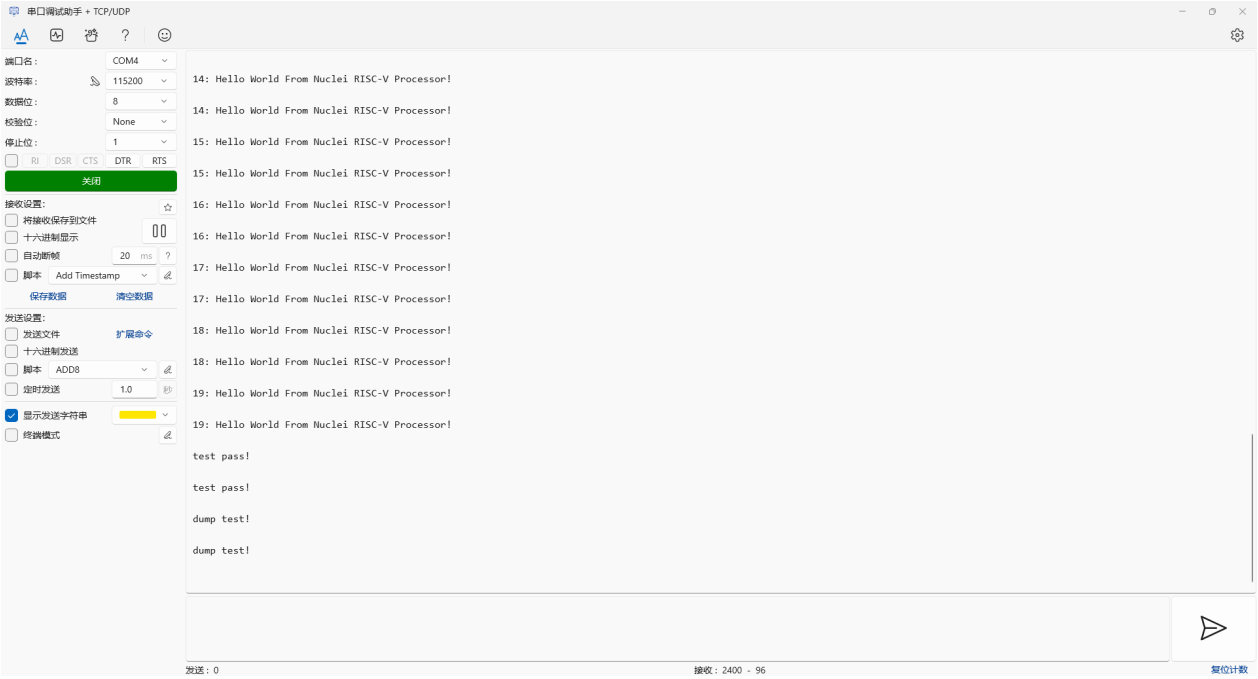
1. 配置文件中已添加目标到 `halt_group`
2. 为两个核心分别加载不同固件
3. 仅在 core0 的 `__amp_wait()` 函数设置断点
4. 执行流程：先恢复 core1，再恢复 core0
5. 验证结果：当 core0 触发断点暂停时，core1 同步暂停



2. 同步恢复组验证¶

1. 配置文件中已添加目标到 `resume_group`
2. 为两个核心加载相同 helloworld 固件
3. 仅向 core0 发送继续运行命令：
4. 验证结果：串口输出显示两个核心同时运行

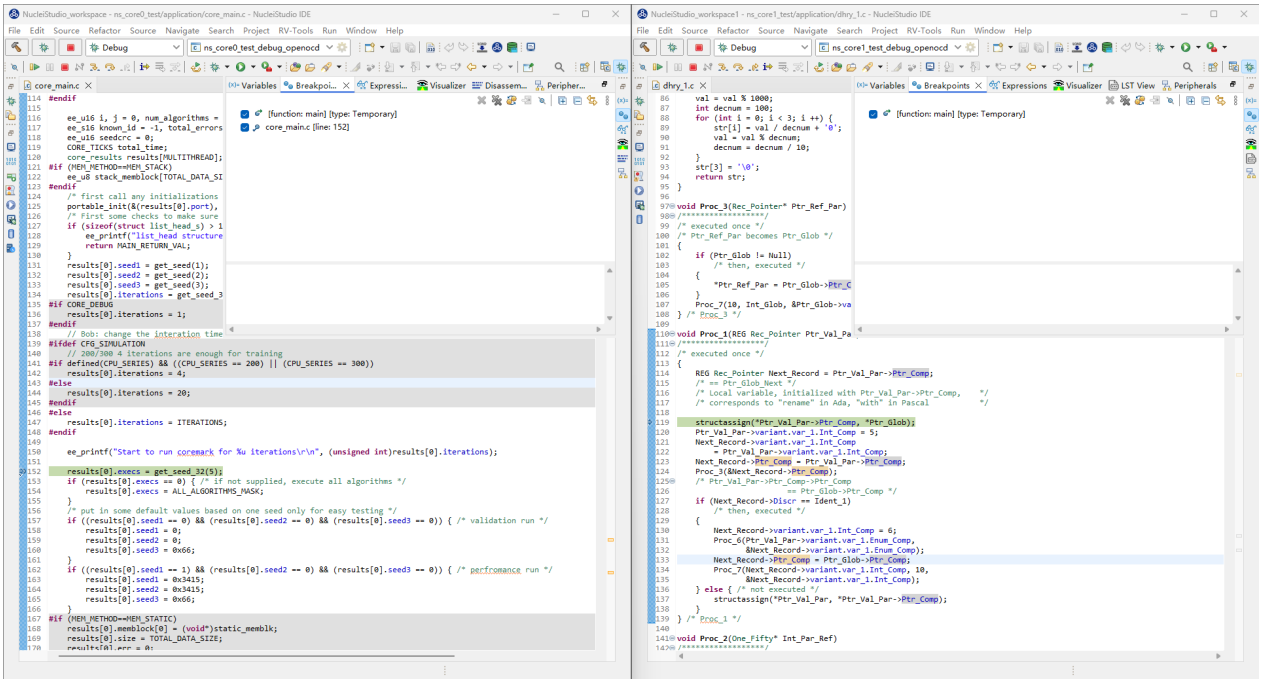




IDE 验证步骤¶

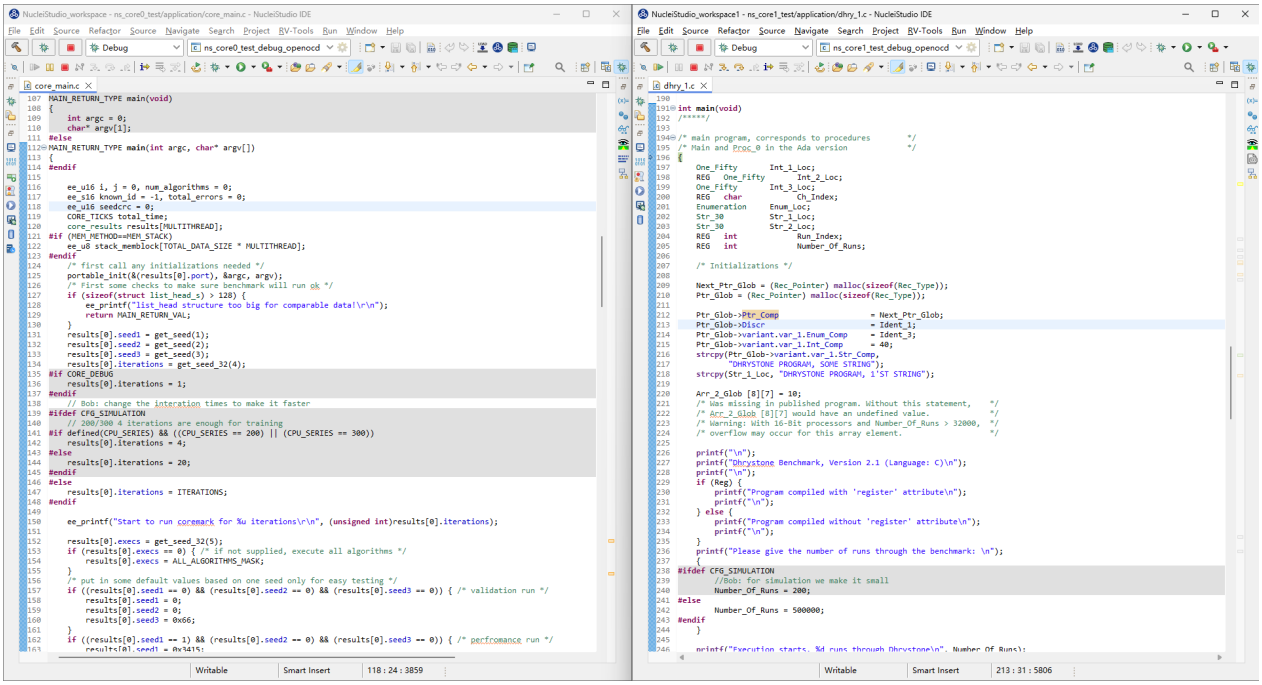
1. 同步暂停组验证¶

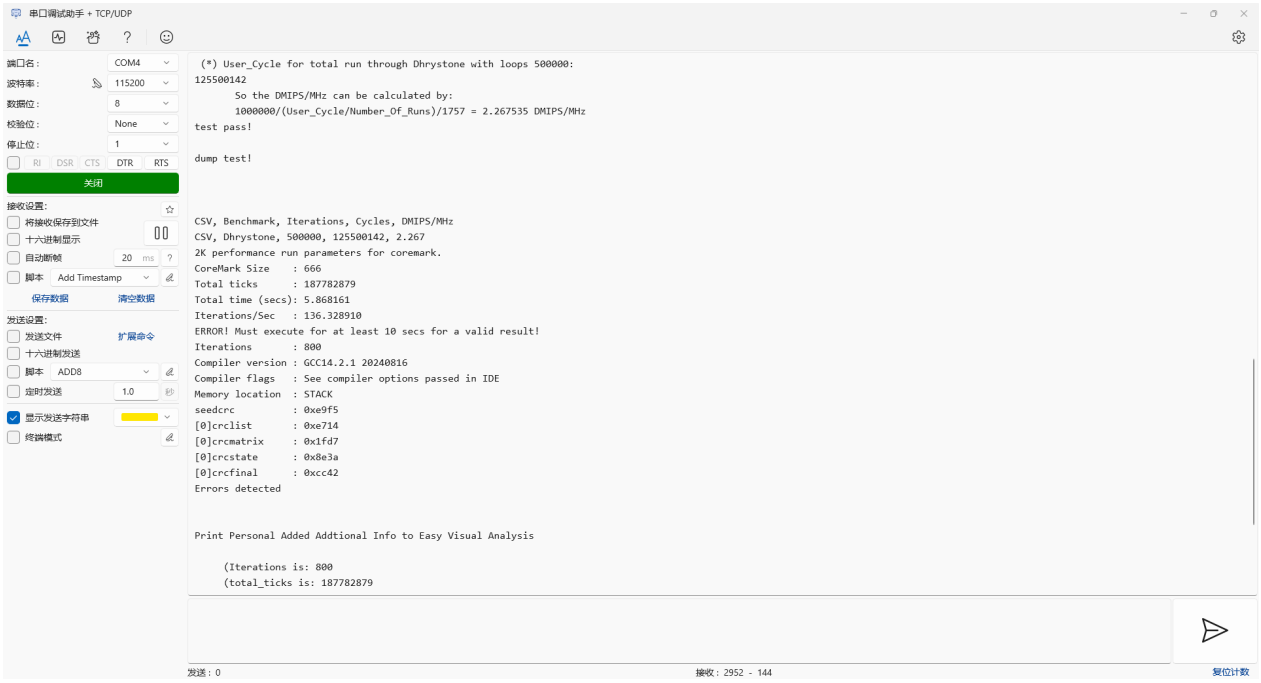
1. 配置文件中已配置 `halt_group`
2. 为两个核心加载不同固件
3. 在 core0 的 `core_main.c` 第 152 行设置断点
4. 操作顺序：
5. 先启动 core1 运行
6. 再启动 core0 运行
7. 验证结果：core0 触发断点时，core1 同步暂停



2. 同步恢复组验证¶

1. 配置文件中已配置 `resume_group`
2. 为两个核心加载不同固件
3. 仅启动 core0 运行
4. 验证结果：串口输出显示两个核心同时运行





OpenOCD对FreeRTOS的调试支持使用指南¶

通过更新您的Nuclei Studio IDE到202502版本和下载0.7.1的sdk-nuclei_sdk,并配合一些下面的修改，就可以使用OpenOCD对FreeRTOS进行调试。

环境准备¶

Nuclei Studio：

- NucleiStudio 202502 Windows
- NucleiStudio 202502 Linux

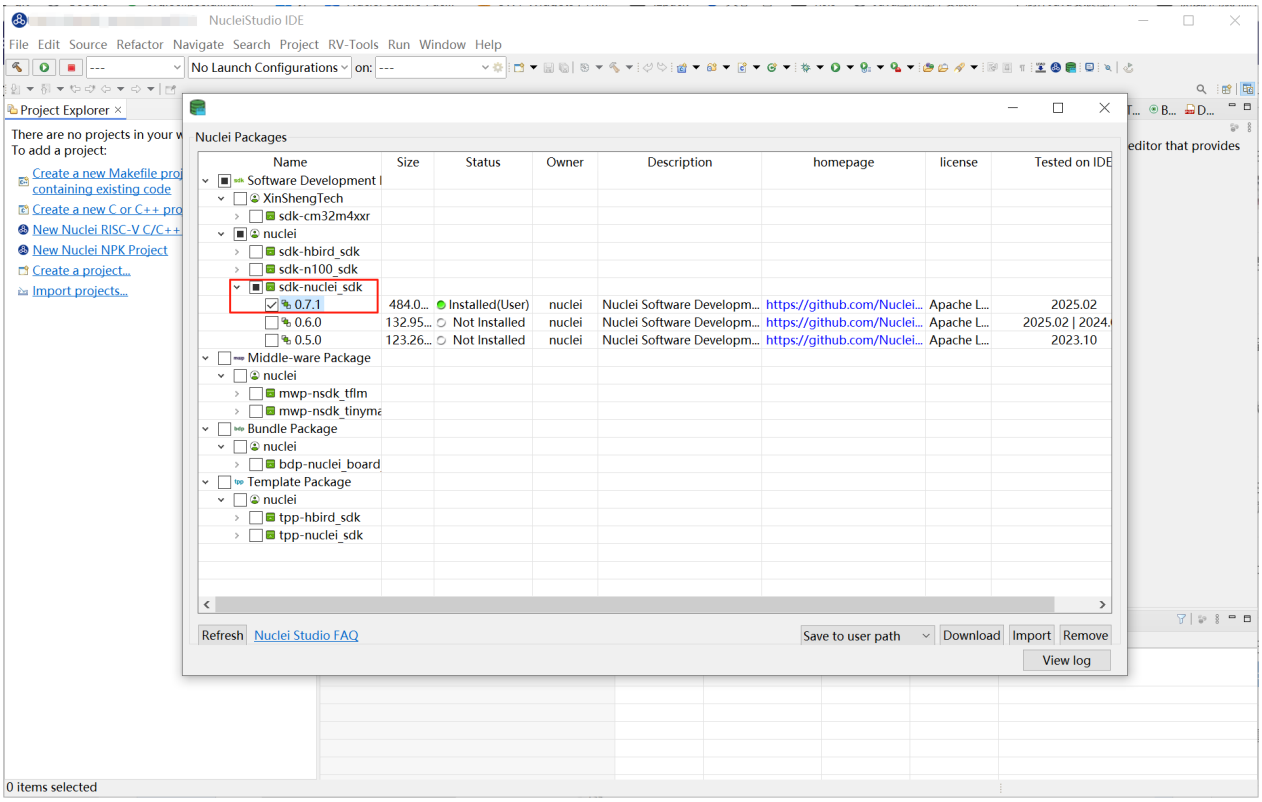
Nuclei OpenOCD：

- 使用NucleiStudio 202502自带的的OpenOCD即可。

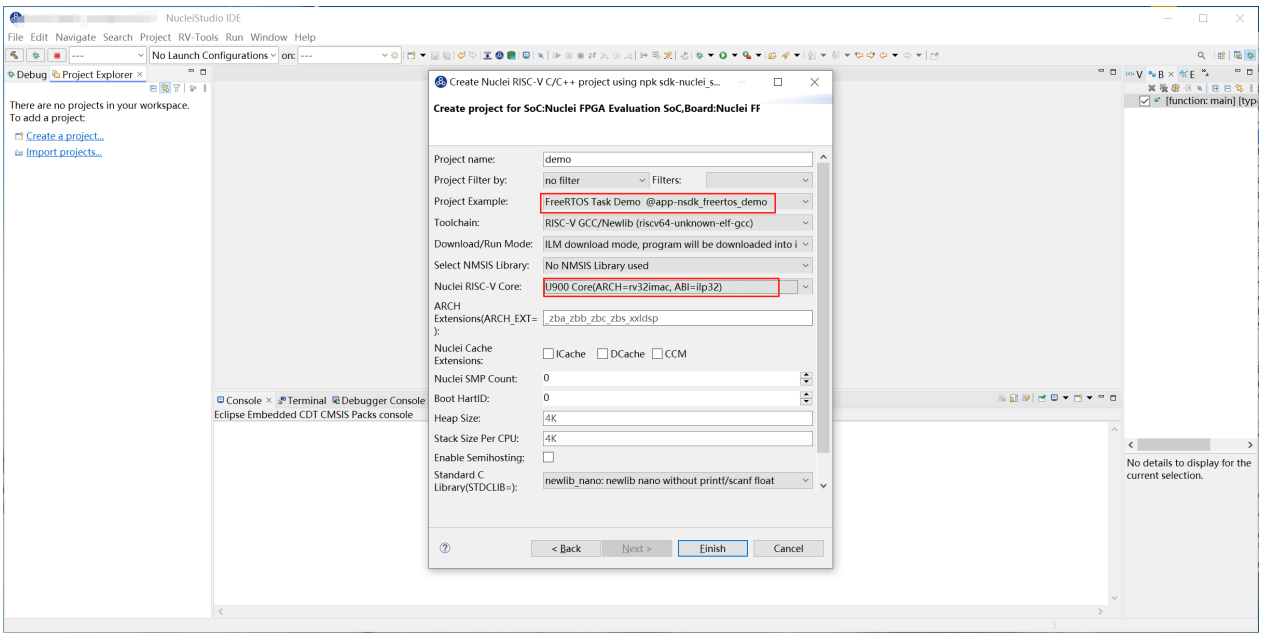
使用步骤¶

step1：创建原始工程

在NucleiStudio IDE下载好0.7.1版本的sdk-nuclei_sdk。



创建一个900的项目，如下图。

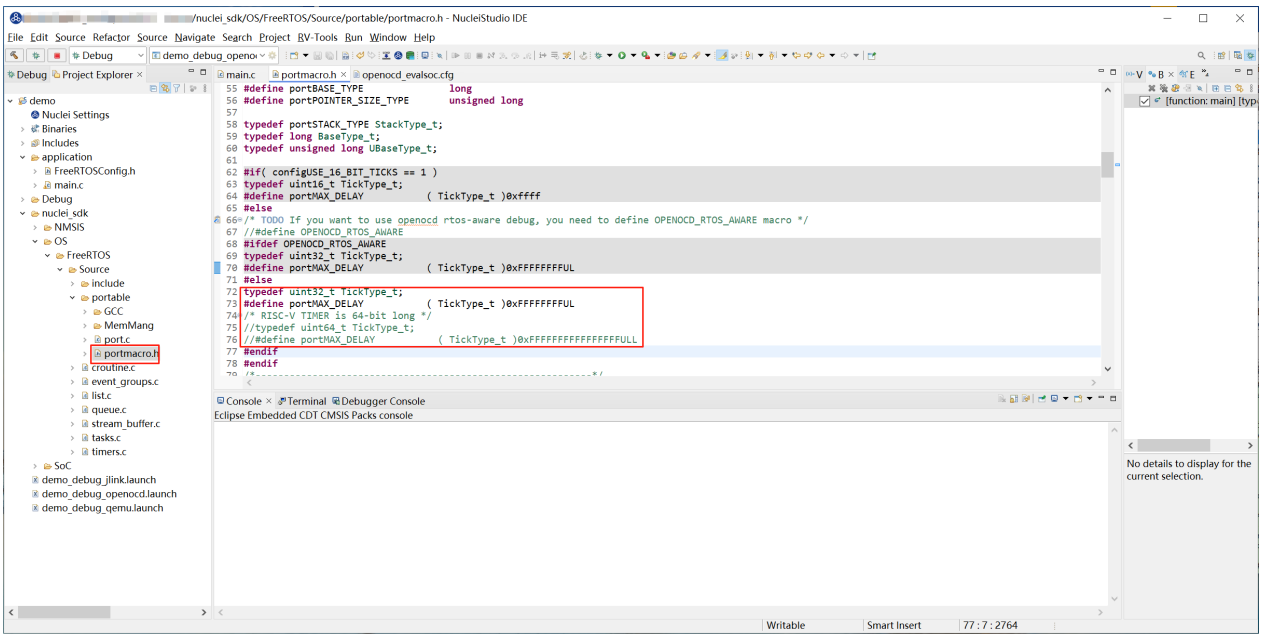


开发板烧写对应的bit即可，这里我们使用
u900_best_config_ku060_50M_c1dd7f44af_915aefa97_202504141013_v4.1.0.bit

step2：修改portmacro.h内容

项目创建好，找到nuclei_sdk\OS\FreeRTOS\Source\portable\portmacro.h, 修改该文件内容如下

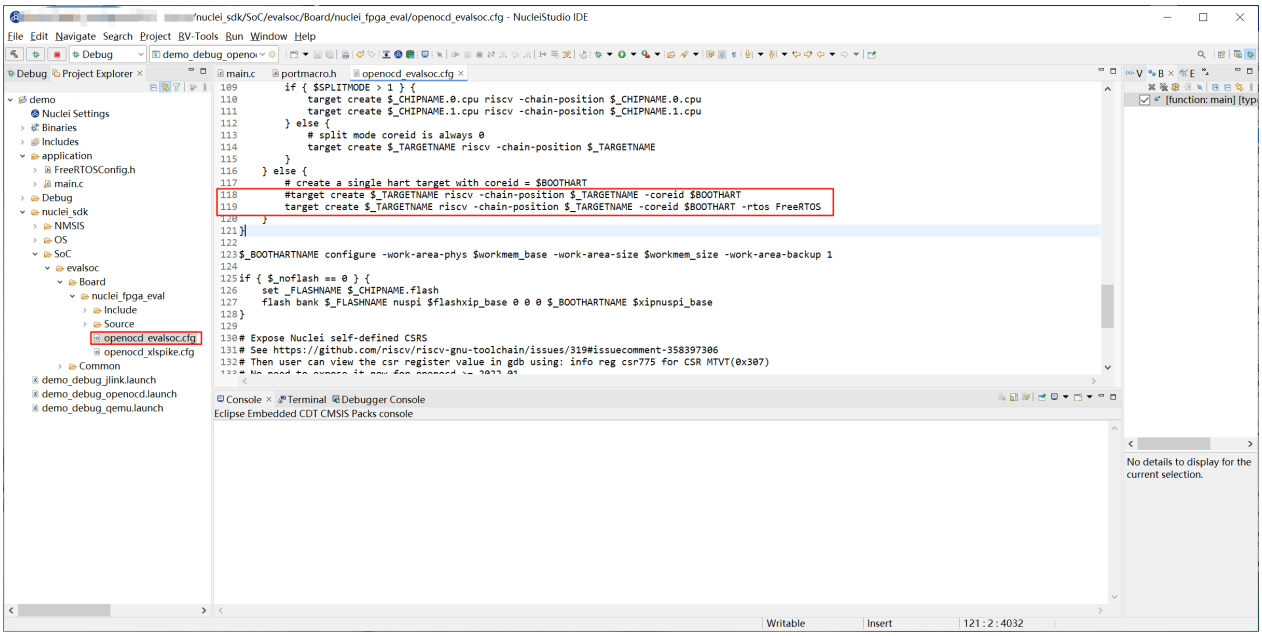
```
typedef uint32_t TickType_t;
#define portMAX_DELAY              ( TickType_t )0xFFFFFFFFUL
/* RISC-V TIMER is 64-bit long */
//typedef uint64_t TickType_t;
//#define portMAX_DELAY
( TickType_t )0xFFFFFFFFFFFFFFFFULL
```



step3：修改openocd_evalsoc.cfg内容

找到nuclei_sdk/SoC/evalsoc/Board/nuclei_fpga_eval/openocd_evalsoc.cfg,修改第118行内容

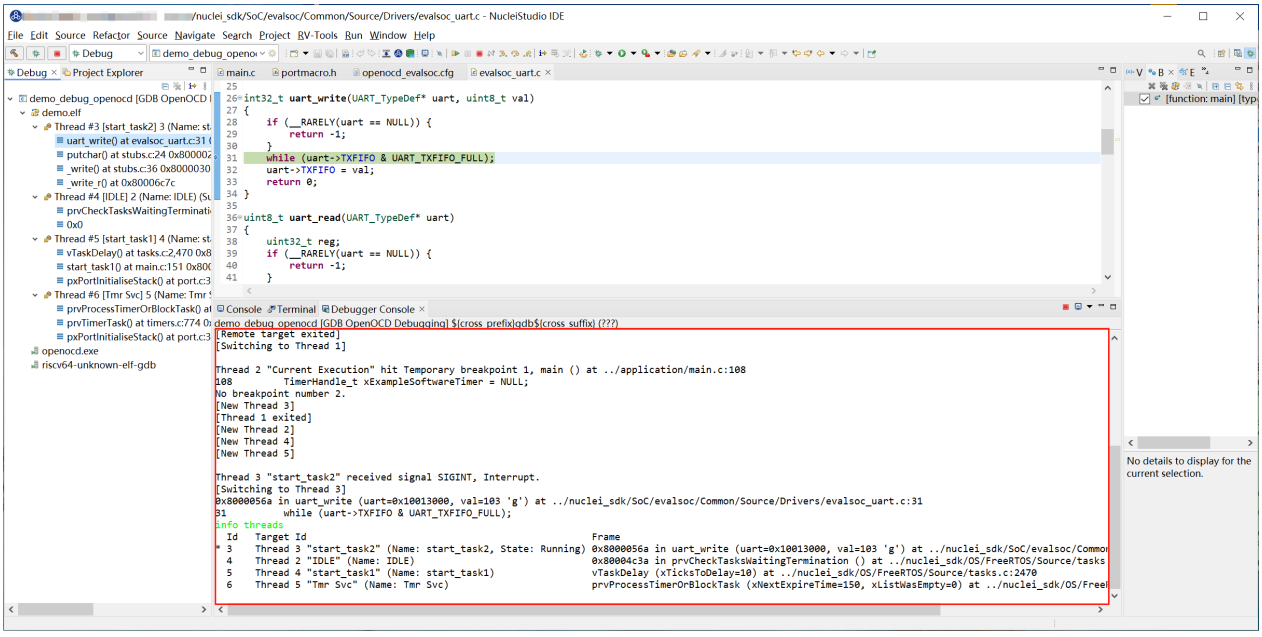
```
target create $_TARGETNAME riscv -chain-position $_TARGETNAME -
coreid $BOOTHART -rtos FreeRTOS
```



step4：openocd调试工程

Debug运行程序，打开Debugger Console视图。

在Debugger Console视图下输入info threads，回车。



使用说明¶

目前支持FreeRTOS,不支持Zephyr、ThreadX、UCOSII。

如何同时使用多个蜂鸟调试器进行调试¶

问题说明¶

芯来科技的蜂鸟调试器采用FTDI-FT2232H作为USB接口转换芯片。 在同时连接多个蜂鸟调试器的情况下，如何区分不同的调试器？如何配置OpenOCD识别指定的蜂鸟调试器？

解决方案¶

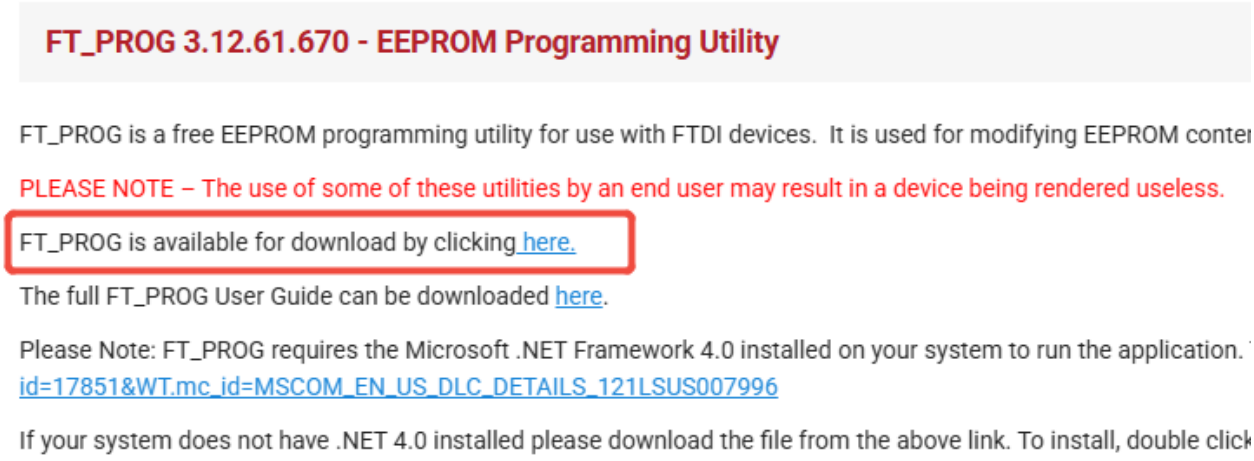
FT2322H提供了一个可配置的串号（Serial Number），可用于区分不同的调试器。

下载FT_PROG¶

FT_PROG是一个用于烧写FT2322H片内的EEPROM的工具。可用于查看和修改FT2322H的串号。

FT_PROG下载地址：https://ftdichip.com/utilities/

从这个页面中可以找到下载链接，如下图所示：

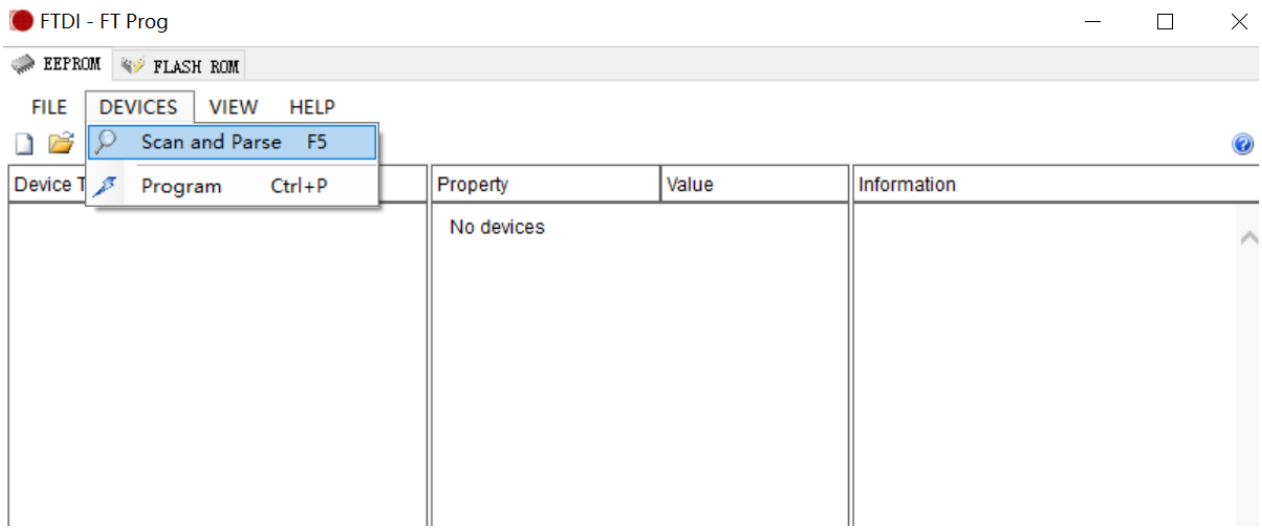


下载并安装后，会在桌面生成FT_Prog工具的图标。

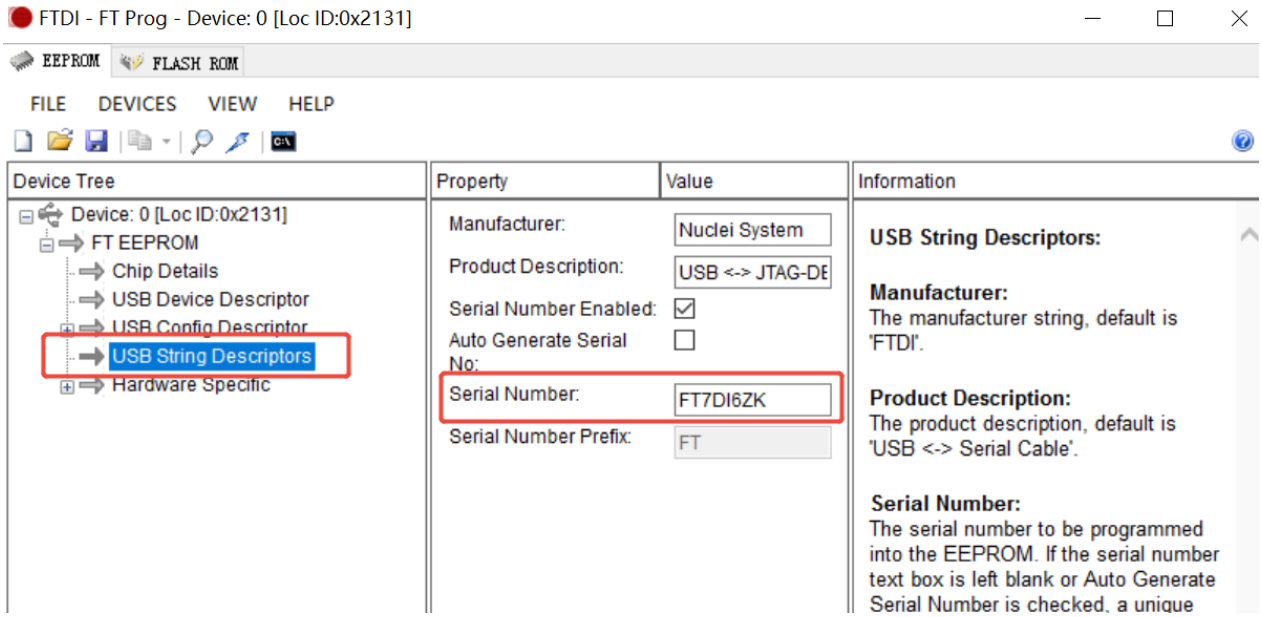
查看串号¶

使用FT_PROG工具，可以查看FT2322H的串号。

1. **连接蜂鸟调试器** 建议在无法区分多个蜂鸟调试器的情况下，先只连接一个蜂鸟调试器。
2. **打开FT_PROG工具** 点击FT_PROG图标打开工具。
3. **扫描设备** 点击菜单栏DEVICES中的Scan and Parse，扫描已连接的蜂鸟调试器。



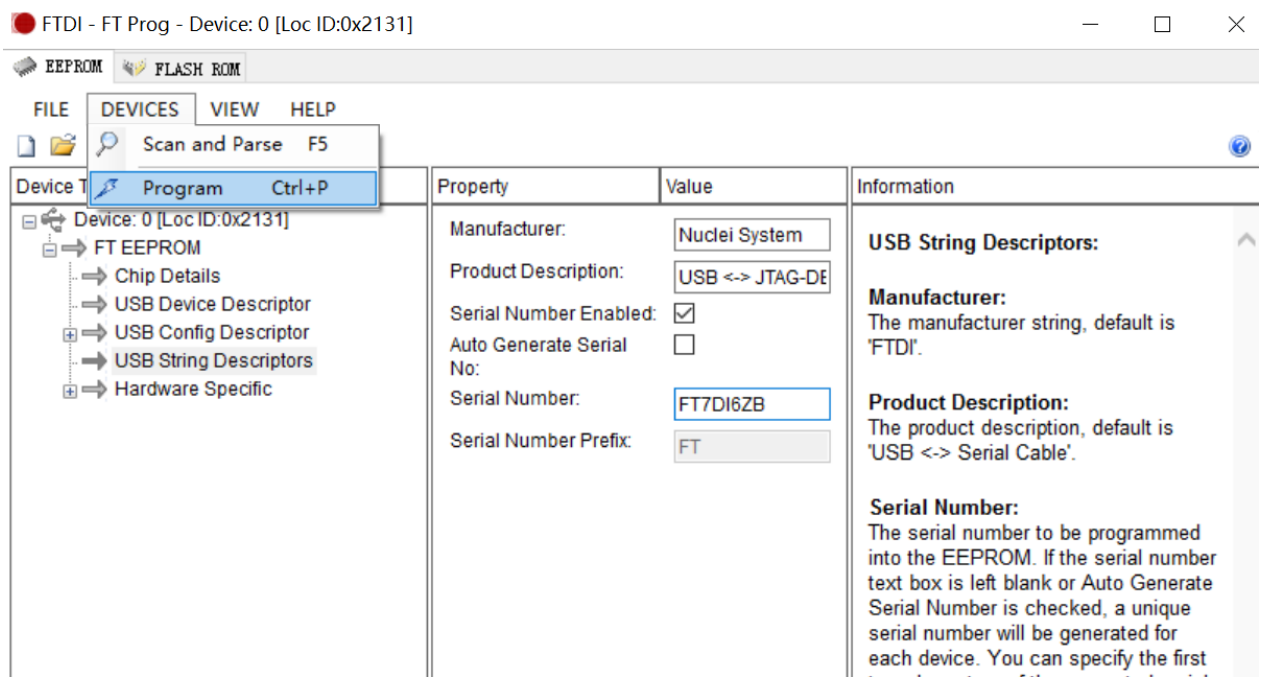
1. **查看串号** 通过`USB String Descriptors`中的`Serial Number`可以查看蜂鸟调试器的串号。



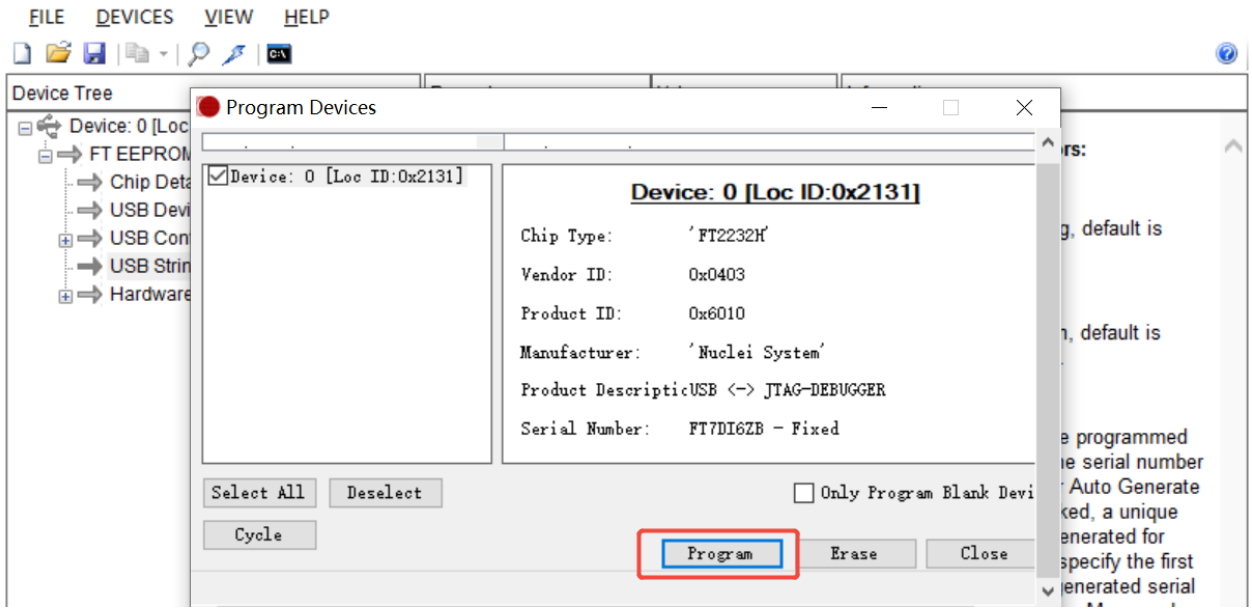
修改串号¶

在查看串号的页面可以修改蜂鸟调试器的串号。

比如下图中，我将原来的串号`FT7DI6ZK`改成了`FT7DI6ZB`



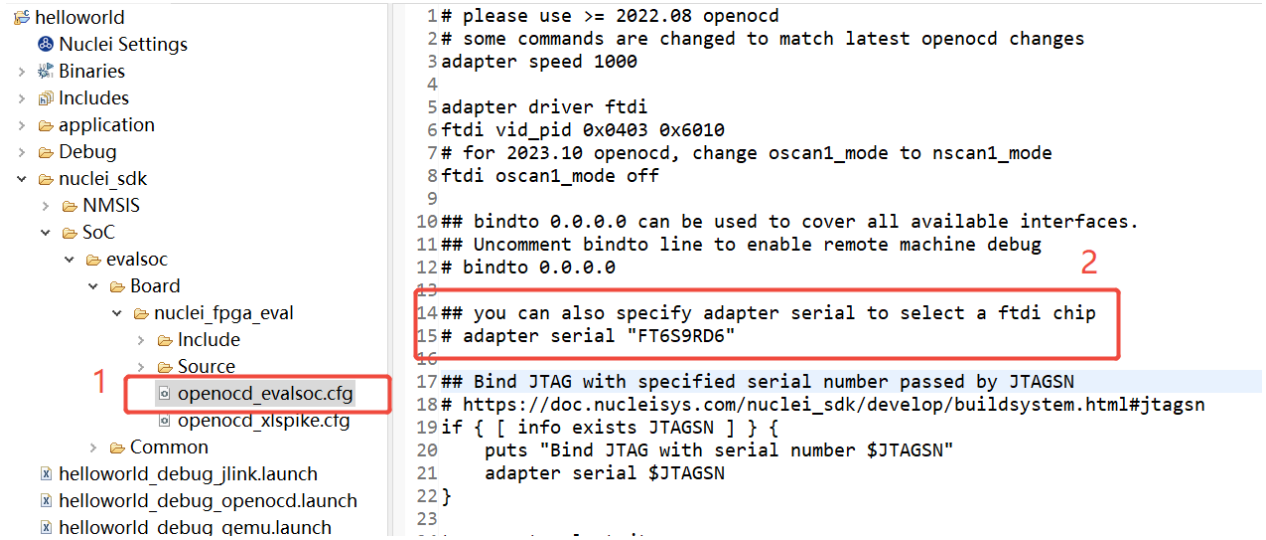
再通过菜单栏DEVICES中的Program选项，可以将修改后的串号写入到FT2322H的EEPROM中。



注意：多个蜂鸟调试器需要分别设置不同的串号来进行区分。

更新OpenOCD配置¶

在使用Nuclei FPGA Evaluation Board时，打开Nuclei Studio中的工程OpenOCD配置文件，可以看到如下内容：



Linux¶

修改openocd_evalsoc.cfg文件，即根据图中红框中的说明进行修改：

```
# 注意要去掉adapter serial前面的注释符号 #
adapter serial "<Serial Number>"
```

其中的<Serial Number>需要替换成实际的串号。

修改后的工程即可使用指定串号的蜂鸟调试器进行调试。

Windows¶

注意：在Windows系统下，需要在实际的串号后加上A才是有效的设置。

例如实际的串号是FT7DI6ZB，那么OpenOCD的配置文件需要添加如下设置：

```
adapter serial "FT7DI6ZBA"
```

参考资料¶

- Nuclei Studio FAQs —— How to select correct FTDI debugger?
- FTDI Utilities
- User Guide for FTDI FT_PROG Utility

Nuclei SDK基于evalsoc快速适配customsoc¶

方案说明¶

Nuclei Eval SoC(简称evalsoc)是芯来科技提供的一款用于评估芯来CPU的SoC，具有On-Chip SRAMs，UART, SPI等；

Nuclei SDK和Nuclei N100 SDK提供基于evalsoc的软件开发平台。客户通过evalsoc评估完芯来CPU后，希望在对应的SDK中快速适配为自己的SoC(本文称为customsoc)。

- Nuclei SDK 主要支持Nuclei 200/300/600/900/1000 series RISC-V CPU, 用于基于这些系列CPU的EvalSoC快速软件评估和开发
- Nuclei N100 SDK 主要支持Nuclei 100 series RISC-V CPU, 用于基于这些系列CPU的EvalSoC快速软件评估和开发

解决方案¶

根据需要移植适配的CPU系列，拉取最新的对应的SDK仓库或者直接使用cpu交付包中的SDK。

环境准备¶

适配修改¶

> 如果通过nuclei_gen工具生成了配套的文件，则直接替换同名文件即可，这样比较简单不出错；如果手动修改，则注意下文提到的文件和修改点

先不要改任何目录名，文件名，按步骤修改如下文件。

1 修改cpu特性描述宏文件¶

SoC/evalsoc/Common/Include/cpufeature.h 文件定义了customsoc支持的特性、参数相关的`#define`宏。CPU交付包中的nuclei_gen工具会自动生成该文件，直接替换即可。

2 修改cpu特性isa配置¶

SoC/evalsoc/cpufeature.mk 文件定义了customsoc的CORE(是否支持单/双精度浮点)ARCH_EXT(是否支持b和v扩展等)。CPU交付包中的nuclei_gen工具会自动生成该文件，直接替换即可。

3 修改链接地址的memory map¶

SoC/evalsoc/Board/nuclei_fpga_eval/Source/GCC/evalsoc.memory 描述了ILM/DLM/FLASH/SRAM/DDR 的BASE address和SIZE以及代码段的大小。CPU交付包中的nuclei_gen工具会自动生成该文件，直接替换即可。

4 修改openocd配置文件¶

openocd会通过jtag与cpu建立gdb server port，供gdb debug和load使用

SoC/evalsoc/Board/nuclei_fpga_eval/openocd_evalsoc.cfg 是openocd的配置描述文件。CPU交付包中的nuclei_gen工具会自动生成该文件，直接替换即可。关键参数如下：

```
# TODO: variables should be replaced by nuclei_gen
set workmem_base    0x80000000
set workmem_size    0x10000
set flashxip_base   0x20000000
set xipnuspi_base   0x10014000
```

5 修改Systimer频率¶

SoC/evalsoc/Common/Include/evalsoc.h 中修改SOC_TIMER_FREQ为customsoc的Systimer的真实频率（需咨询你们SoC硬件设计人员）

```
// 单位是hz 比如32768hz，这里填32768
#define SOC_TIMER_FREQ              customsoc_systimer_freq
```

6 修改CPU主频¶

SoC/evalsoc/Common/Source/system_evalsoc.c 中，SystemCoreClock = get_cpu_freq()自动计算cpu主频(依赖Systimer)，可以直接修改为customsoc的主频

```
// 单位是hz 比如50Mhz，这里填50000000
SystemCoreClock = customsoc_cpu_freq;
```

7 修改串口驱动¶

evalsoc的UART IP是评估版本

evalsoc_uart.c和evalsoc_uart.h里面的uart_xxx API名称不要修改，因为SoC/evalsoc/Common/Source/Stubs下的一些桩函数使用了uart的API

串口驱动位于 SoC/evalsoc/Common/Source/Drivers/evalsoc_uart.c ，SoC/evalsoc/Common/Include/evalsoc_uart.h，如果使用其它串口IP，根据实际的串口寄存器定义适配。

8 修改串口波特率¶

SoC/evalsoc/Common/Source/system_evalsoc.c: `uart_init(SOC_DEBUG_UART, 115200);` 一般波特率为115200

9 修改_premain_init¶

一些在main函数之前执行的初始化可以放在这个函数

如果有 IOMUX 和 PLL 等其他相关的配置，可以在 SoC/evalsoc/Common/Source/system_evalsoc.c: `_premain_init` 函数里面实现；如果没有，可以跳过

10 删除Nuclei内部使用的代码¶

SoC/evalsoc/Common/Source/system_evalsoc.c: `SIMULATION_EXIT` 宏定义是用于Nuclei内部仿真标记，可以定义为空

```
#define SIMULATION_EXIT(ret)    {}
```

11 检查外设地址¶

建议CPU配置时不要修改，保持与evalsoc一致

串口使用的`SOC_DEBUG_UART`定义为`UART0`

- 外设的Base address由`EVALSOC_PERIPS_BASE`决定，`EVALSOC_PERIPS_BASE`在SoC/evalsoc/Common/Include/cpufeature.h(由nuclei_gen工具生成，拷贝覆盖即可)中定义，一般无需再修改
- 外设的offset address在 SoC/evalsoc/Common/Include/evalsoc.h 中定义，搜索 `Peripheral memory map`, 一般无需修改

```
#define UART0_BASE              (EVALSOC_PERIPH_BASE +
0x13000)          /*!< (UART0) Base Address */
#define QSPI0_BASE              (EVALSOC_PERIPH_BASE +
0x14000)          /*!< (QSPI0) Base Address */
#define UART0                   ((UART_TypeDef *) UART0_BASE)
```

测试运行¶

如果以上修改完毕，就可以测试SoC能否正常工作了

> 这里因为是在evalsoc的基础上改的，还没有修改相关地方的名称为customsoc
>
> 所以仍然SOC=evalsoc BOARD=nuclei_fpga_eval

```
# Test helloworld application
## cd to helloworld application directory
cd application/baremetal/helloworld
## clean and build helloworld application for ncstar_eval board
make SOC=evalsoc BOARD=nuclei_fpga_eval clean all
## connect your board to PC and install jtag driver, open UART
terminal
## set baudrate to 115200bps and then upload the built application
## to the fpga board using openocd, and you can check the
## run messsage in UART terminal
make SOC=evalsoc BOARD=nuclei_fpga_eval upload
```

如果可以正常运行打印Hello World From Nuclei RISC-V Processor，那基本没有问题了。如果还需要运行更多case，请参考如下应用示例文档确认是否运行成功。

- Nuclei SDK: https://doc.nucleisys.com/nuclei_sdk/design/app.html
- Nuclei N100 SDK: https://doc.nucleisys.com/nuclei_n100_sdk/design/app.html

调整名称¶

> 重命名的地方有点多，这里就不列举了，最终保证编译通过就可以。

测试通过后，就可以把涉及evalsoc的文件名和目录名修改为customsoc，以及eval/EVAL开头的宏名/文件名替换成custom

```
# 修改完后，再次测试运行
make SOC=customsoc BOARD=nuclei_fpga_custom upload
```

至此，SDK就去掉了eval的logo,成为SDK for custom了。

精简代码¶

因为Nuclei SDK/N100 SDK支持Nuclei多款CPU系列的评估和内部测试，需要考虑非常多的场景，因此存在一些冗余代码，建议在阅读SDK文档并且熟悉代码框架后，再进行精简删除。

IAR工程¶

- IAR的工程有专门的链接脚本，位于`SoC/evalsoc/Board/nuclei_fpga_eval/Source/IAR/*.icf` IAR的链接脚本当前没有通过nuclei_gen工具生成，所以需要手动检查调整`ROM_region32/ILM_region32/RAM_region32`的base address和size, 这里的from就是代表base address，size 表示该region的大小

```
define region ROM_region32 = mem:[from 0x20000000 size 0x800000];
define region ILM_region32 = mem:[from 0x80000000 size 0x10000];
define region RAM_region32 = mem:[from 0x90000000 size 0x10000];
```

- IAR的工程位于`ideprojects/iar`，也是`prebuilt for evalsoc`，在未调整名称之前是可以直接运行的 如果经过了调整名称，路径和文件名都变化了，也需要重新新建工程，建议文本打开ewp文件，搜索“eval”关键词替换

```
diff --git a/ideprojects/iar/baremetal/coremark.ewp b/ideprojects/
iar/baremetal/coremark.ewp
index 3eed66a8..17443eae 100644
--- a/ideprojects/iar/baremetal/coremark.ewp
+++ b/ideprojects/iar/baremetal/coremark.ewp
@@ -434,8 +434,8 @@
                <option>
                    <name>CCIncludePath2</name>
                    <state>$PROJ_DIR$\..\..\..\NMSIS\Core\Include</
state>
-                    <state>$PROJ_DIR$\..\..\..
\SoC\evalsoc\Board\nuclei_fpga_eval\Include</state>
-                    <state>$PROJ_DIR$\..\..\..
\SoC\evalsoc\Common\Include</state>
+                    <state>$PROJ_DIR$\..\..\..
\SoC\customsoc\Board\nuclei_fpga_custom\Include</state>
+                    <state>$PROJ_DIR$\..\..\..
\SoC\customsoc\Common\Include</state>
                    <state>$PROJ_DIR$\..\..\..
\application\baremetal\benchmark\coremark</state>
                </option>
```

IDE工程支持¶

如果希望Nuclei Studio IDE能支持custom soc，需要修改以下文件中涉及eval的名字，npk.yml的语法格式见 2.4. Nuclei Studio NPK 介绍

```
evalsoc/Common/npk.yml
evalsoc/Board/nuclei_fpga_eval/npk.yml
```

参考资料¶

- Nuclei 200/300/600/900/1000 Eval SoC
- Port your SoC into Nuclei SDK
- Nuclei 100 Eval SoC
- Port your SoC into Nuclei N100 SDK

在链接脚本中使用OVERLAY命令¶

问题说明¶

CPU Core内的SRAM具有速度快，容量小，面积大的特点。在嵌入式系统中，这部分Core内的RAM很可能无法放下所有的函数。 为了解决这个问题，有一种方案是在链接脚本中使用OVERLAY命令。
GNU `ld` 提供的OVERLAY命令，可以在同一块内存区域上“叠放”多个段（Section）。若干个段可以共享运行时的 VMA（Virtual Memory Address），只是在运行时需要手动管理这些Overlay的段的加载和卸载。

那么如何在Nuclei Studio IDE中使用OVERLAY命令呢？本文将提供一个示例程序演示如何使用OVERLAY命令。

解决方案¶

示例程序¶

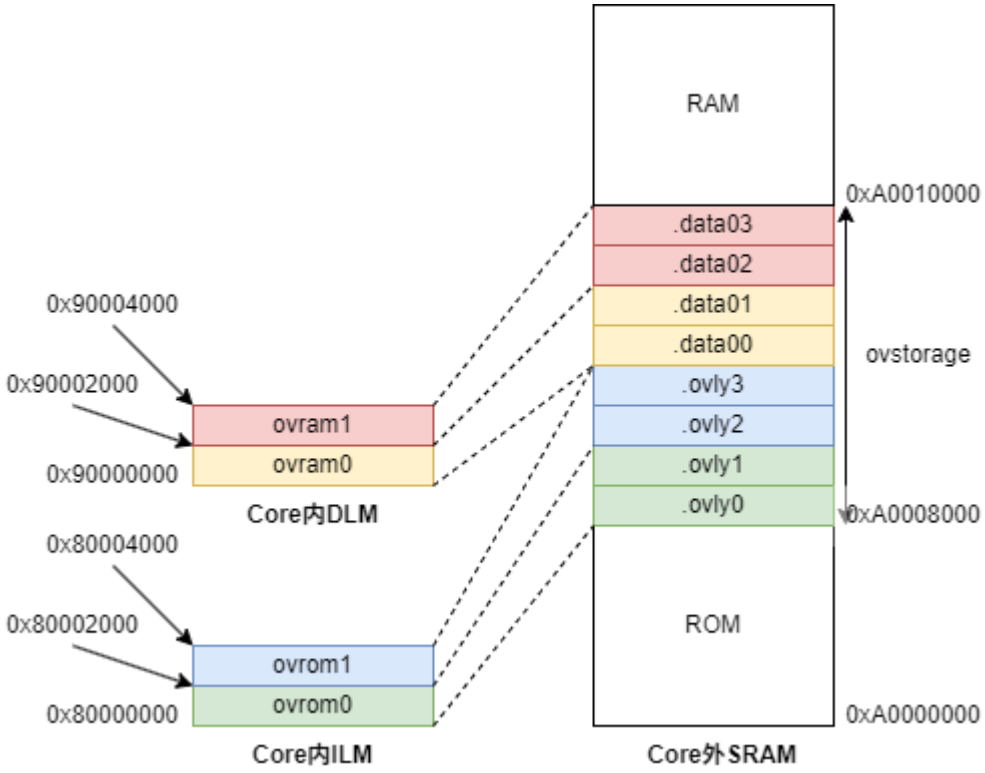
demo_overlay是基于 Nuclei Studio IDE 2025.02 创建的示例工程，支持Linux和Windows两种平台，演示了如何在链接脚本中使用OVERLAY命令。

示例工程中的代码与Overlay Sample Program 中提供的代码基本一致，主要区别是在main函数中增加了测试结果的打印，另外根据evalsoc的地址映射关系 对链接脚本做了修改。

```
├── bar.c
├── baz.c
├── foo.c
├── grbx.c
├── overlays.c
├── ovlymgr.c
└── ovlymgr.h
```

原始代码可以从bminor/binutils-gdb/gdb/testsuite/gdb.base获取。

Overlay布局¶



在我们的evalsoc上，如果程序无法完全放在Core内的ILM/DLM上，就可以采取上图中的方法 将部分Section以Overlay的形式动态加载到ILM/DLM中运行。
`.ovlyx`和`.data0x` 共8个Section的LMA（Load Memory Address）都位于Core外的SRAM上，但它们的VMA都在Core内，并且有部分重叠。

编写链接脚本¶

在MEMORY命令中先划分出需要用到的Memory区域，比如这里4个Core内的区域`ovrom0`, `ovrom1`, `ovram0`, `ovram1`和Core外的`ovstorage`区域。这些Memory区域的大小都可以根据实际的代码和数据的大小进行调整。

```
MEMORY
{
  rom (rxa!w) : ORIGIN =
SRAM_MEMORY_BASE,                          LENGTH =
SRAM_MEMORY_ROM_SIZE
  ovstorage (rwa) : ORIGIN = SRAM_MEMORY_BASE +
SRAM_MEMORY_ROM_SIZE, LENGTH = SRAM_OVLY_STORAGE_SIZE
  ram (wxa!r) : ORIGIN = SRAM_MEMORY_BASE + SRAM_MEMORY_ROM_SIZE +
SRAM_OVLY_STORAGE_SIZE,    LENGTH = SRAM_MEMORY_SIZE -
SRAM_MEMORY_ROM_SIZE - SRAM_OVLY_STORAGE_SIZE
  ovrom0 (rwx) : ORIGIN = ILM_MEMORY_BASE,   LENGTH =
ILM_OVLY_SIZE0
```

```
  ovrom1 (rwx) : ORIGIN = ILM_MEMORY_BASE + ILM_OVLY_SIZE0,
LENGTH = ILM_OVLY_SIZE1
  ovram0 (rwx) : ORIGIN = DLM_MEMORY_BASE,   LENGTH =
DLM_OVLY_SIZE0
  ovram1 (rwx) : ORIGIN = DLM_MEMORY_BASE + DLM_OVLY_SIZE0,
LENGTH = DLM_OVLY_SIZE1
}
```

OVERLAY需要放在SECTION命令中实现。例如下方的代码就是将.ovly0和.ovly1两个段放到ovrom0所 表示的同一个VMA地址区间中，同时它们的LMA则是连续地位于ovstorage所表示的地址区间中。

```
OVERLAY :
{
  .ovly0 { *foo.o(.text .text.*) }
  .ovly1 { *bar.o(.text .text.*) }
} >ovrom0 AT>ovstorage
```

参考Automatic Overlay Debugging， 链接脚本中的以下代码则是将Overlay的段的VMA, LMA和数量以_ovly_table和_novlys变量的形式供C代码访问； 在 ovlymgr.c中进一步实现动态加载和卸载Section的功能。

```
/* _ovly_table used for gdb debug overlay sections */
_ovly_table = .;
  _ovly0_entry = .;
LONG(ABSOLUTE(ADDR(.ovly0)));
LONG(SIZEOF(.ovly0));
LONG(LOADADDR(.ovly0));
LONG(0);
...
_novlys = .;
LONG((_novlys - _ovly_table) / 16);
```

测试结果¶

观察编译后生成的map文件，可以看到相应的代码段和数据段都按照预期的VMA和LMA进行了分配。
例如.ovly0和.ovly1具有相同的VMA 0x80000000，同时它们的LMA分别为0xa0008000和0xa0008028。

```
.ovly0          0x80000000       0x28 load address 0xa0008000
 *foo.o(.text .text.*)
```

```
 .text.foo      0x80000000       0x28 ./application/foo.o
                0x80000000                foo
                [!provide]                      PROVIDE
(__load_start_ovly0 = LOADADDR (.ovly0))
                [!provide]                      PROVIDE
(__load_stop_ovly0 = (LOADADDR (.ovly0) + SIZEOF (.ovly0)))

.ovly1          0x80000000       0x28 load address 0xa0008028
 *bar.o(.text .text.*)
 .text.bar      0x80000000       0x28 ./application/bar.o
                0x80000000                bar
                [!provide]                      PROVIDE
(__load_start_ovly1 = LOADADDR (.ovly1))
                [!provide]                      PROVIDE
(__load_stop_ovly1 = (LOADADDR (.ovly1) + SIZEOF (.ovly1)))
```

main函数中通过OverlayLoad切换调用不同的函数，并在最后将每个函数的返回值累加，校验累加后的结果。

```
/* load .text and .data for `foo` */
OverlayLoad (0);
OverlayLoad (4);
a = foo (1);
/* load .text and .data for `bar` */
OverlayLoad (1);
OverlayLoad (5);
b = bar (1);
/* load .text and .data for `baz` */
OverlayLoad (2);
OverlayLoad (6);
c = baz (1);
/* load .text and .data for `grbx` */
OverlayLoad (3);
OverlayLoad (7);
d = grbx (1);

e = a + b + c + d;
if (e != ('f' + 'o' +'o'
  + 'b' + 'a' + 'r'
  + 'b' + 'a' + 'z'
  + 'g' + 'r' + 'b' + 'x')) {
  printf ("Overlay Test FAIL\r\n");
} else {
  printf ("Overlay Test PASS\r\n");
}
```

通过QEMU仿真或者利用FPGA开发板进行测试，可以看到如下的结果。其中`Overlay Test PASS`表明结果符合预期。

```
Nuclei SDK Build Time: Sep 25 2025, 17:02:19
Download Mode: SRAM
CPU Frequency 16003235 Hz
CPU HartID: 0
Overlay Test PASS
```

注意事项¶

1. 数据段Overlay在要替换Section时，需要保存数据，也就是需要“卸载”的操作，将数据保存到外部SRAM中；而 代码段是只读的，所以不需要Unload。
2. 在示例工程中，ILM/DLM都不会经过Cache，所以不需要考虑Cache一致性的问题。但如果Overlay的Section所在 的VMA是Cacheable的区域，则一般都需要考虑Cache一致性的问题，除非ICache和DCache之间有硬件支持的Snoop。 更多细节可以参考示例工程中`ovlymgr.c`的实现。

参考资料¶

- How Overlays Work
- Overlay Commands
- Automatic Overlay Debugging
- Overlay Sample Program