GigaDevice Semiconductor Inc.

GD32 RISC-V MCU Eclipse development tutorial for Windows

Application Note
AN067

Table of Contents

List of Figures

List of Tables

1. Introduction

This guide introduces how to use Eclipse to develop GD32 MCU. Applicable to GD32 RISC-V series MCU.

2. Development environment

- Development boards: GD32 MCU boards
- Hardware Debugger: J-Link V10 or GD-Link
- Operating system: WIN7/WIN10 64-bit OS
- IDE: eclipse-embedcpp-2021-03-R-win32-x86_64
- Cross toolchains: xpack-riscv-none-embed-gcc-10.1.0-1.1-win32-x64
- Build Tools: gnu-mcu-eclipse-windows-build-tools-2.12-20190422-1053-win64
- GDB server: OpenOCD / J-Link GDB Server CL V7.11a

3. Project development

3.1. New project

Open Eclipse, LAUCH eclipse-workspace. Under "File->New" option, uesr can choose to create a new C/C++ Project and select C Managed Build option.

Figure 3-1. New RISC-V C project

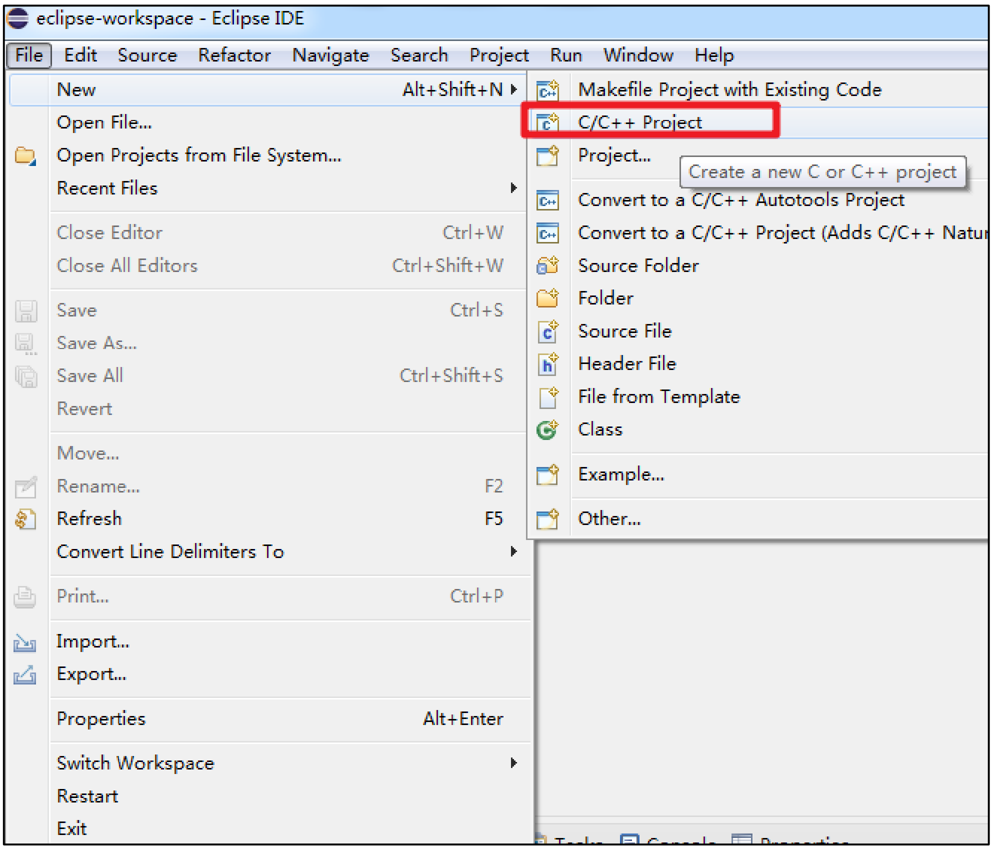
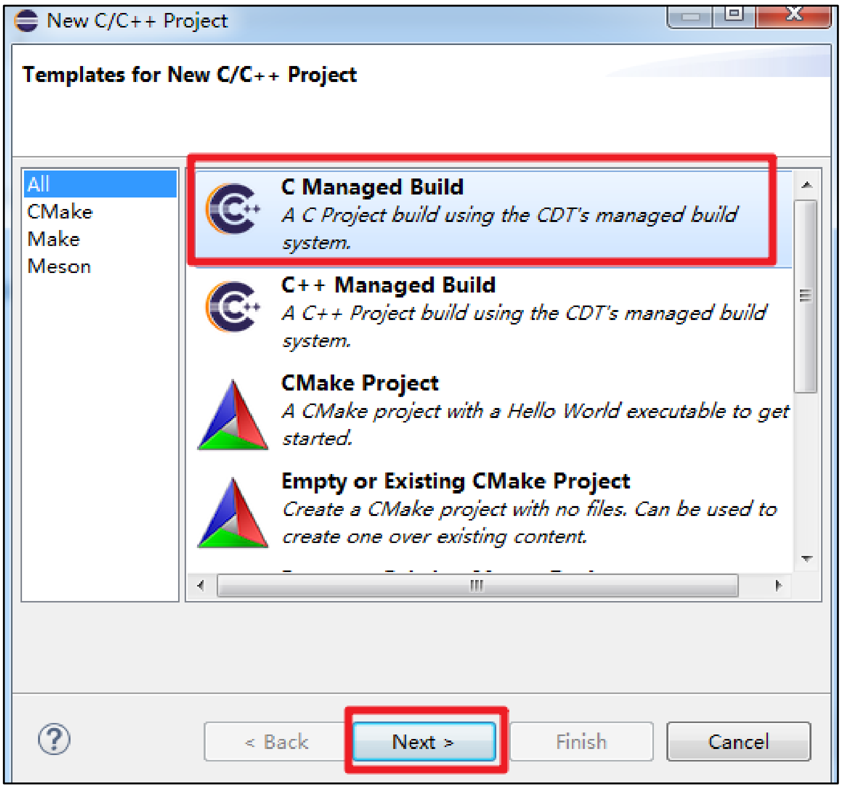


Figure 3-2. Select C Managed Build



New C/C++ Project

Templates for New C/C++ Project

- All
- CMake
- Make
- Meson

C Managed Build
A C Project build using the CDT's managed build system.

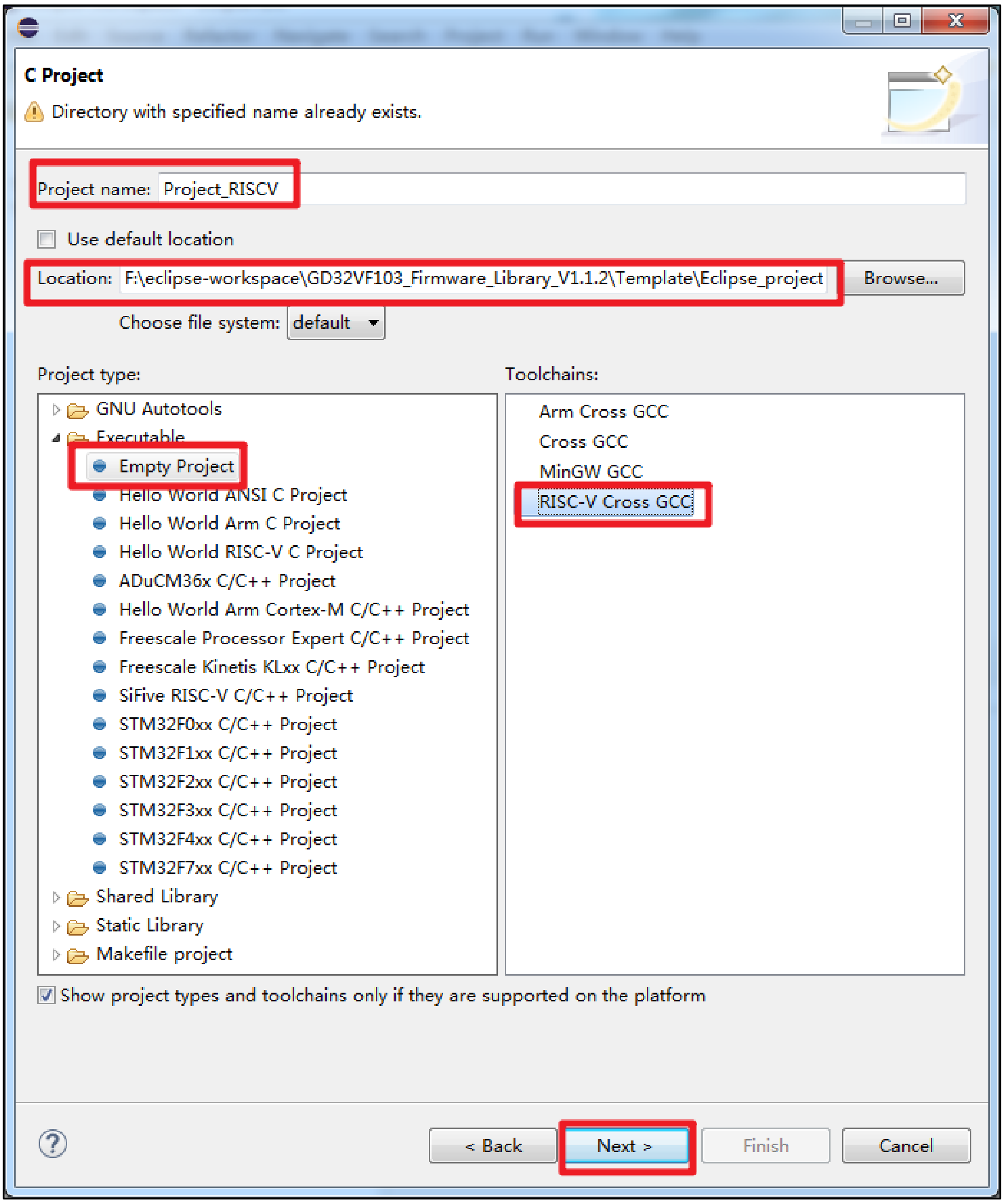
C++ Managed Build
A C++ Project build using the CDT's managed build system.

CMake Project
A CMake project with a Hello World executable to get started.

Empty or Existing CMake Project
Create a CMake project with no files. Can be used to create one over existing content.

Enter the "Project name" and configure the project type. For convenience, it is recommended to put the project in the FW directory. The compilation chain is selected as RISC-V Cross GCC.

Figure 3-3. Create new RISC-V project name and select project storage path



C Project

Directory with specified name already exists.

Project name: Project_RISCV

□ Use default location

Location: F:\eclipse-workspace\GD32VF103_Firmware_Library_V1.1.2\Template\Eclipse_project

Choose file system: default

Project type:

- GNU Autotools
- Executable
  - Empty Project
  - Hello World ANSI C Project
  - Hello World Arm C Project
  - Hello World RISC-V C Project
  - ADuCM36x C/C++ Project
  - Hello World Arm Cortex-M C/C++ Project
  - Freescale Processor Expert C/C++ Project
  - Freescale Kinetis KLxx C/C++ Project
  - SiFive RISC-V C/C++ Project
  - STM32F0xx C/C++ Project
  - STM32F1xx C/C++ Project
  - STM32F2xx C/C++ Project
  - STM32F3xx C/C++ Project
  - STM32F4xx C/C++ Project
  - STM32F7xx C/C++ Project
- Shared Library
- Static Library
- Makefile project

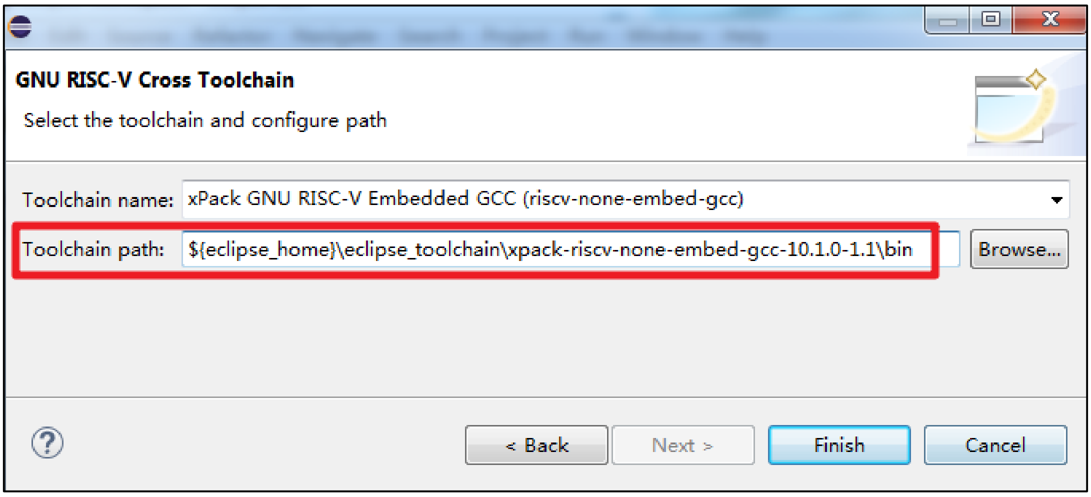
Toolchains:

- Arm Cross GCC
- Cross GCC
- MinGW GCC
- RISC-V Cross GCC

☑ Show project types and toolchains only if they are supported on the platform

If the Eclipse IDE has set the RISC-V Toolchains Path correctly, the path will be automatically selected here. If the Eclipse IDE has not set the RISC-V Toolchains Path, user can also select the absolute path to the RISC-V Toolchains here.

Figure 3-4. Select RISC-V cross toolchain path



GNU RISC-V Cross Toolchain

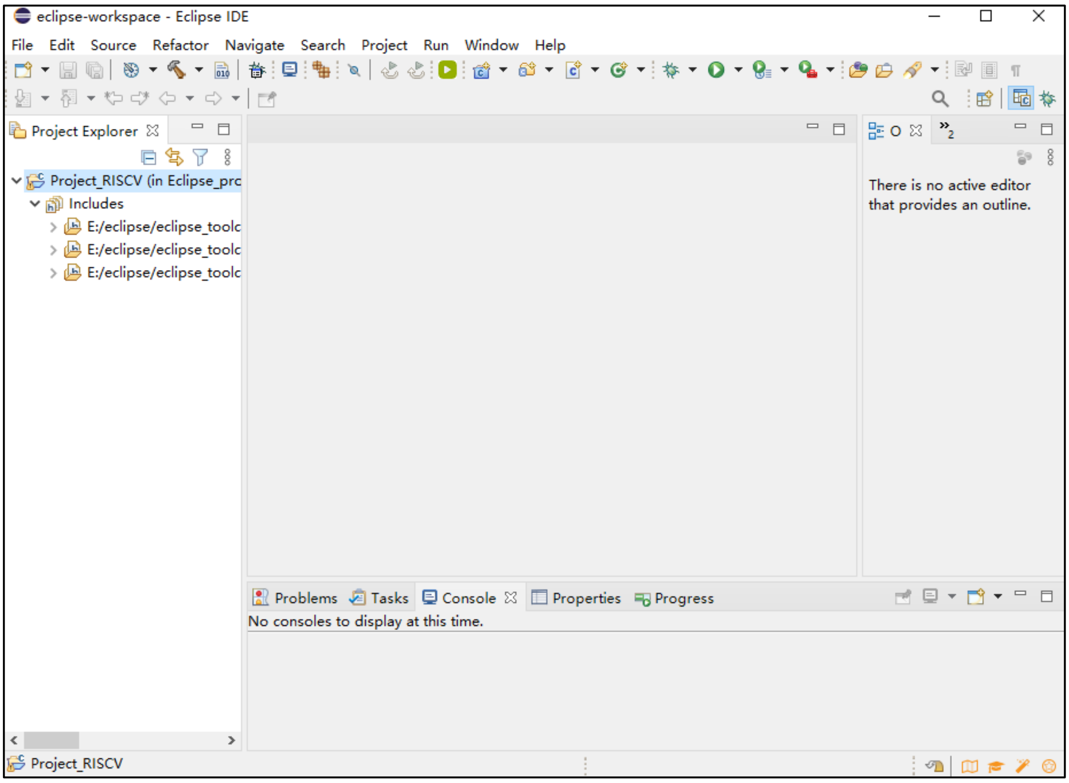
Select the toolchain and configure path

Toolchain name: xPack GNU RISC-V Embedded GCC (riscv-none-embed-gcc)

Toolchain path: ${eclipse_home}\eclipse_toolchain\xpack-riscv-none-embed-gcc-10.1.0-1.1\bin

Click "Finish" until the display interface is shown in *Figure 3-5. Project perspective*. At this point, the establishment of the Project is completed.

Figure 3-5. Project perspective



eclipse-workspace - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Project_RISCV (in Eclipse_prc
Includes
E:/eclipse/eclipse_toolc
E:/eclipse/eclipse_toolc
E:/eclipse/eclipse_toolc
Problems Tasks Console Properties Progress
Project_RISCV

There is no active editor that provides an outline.

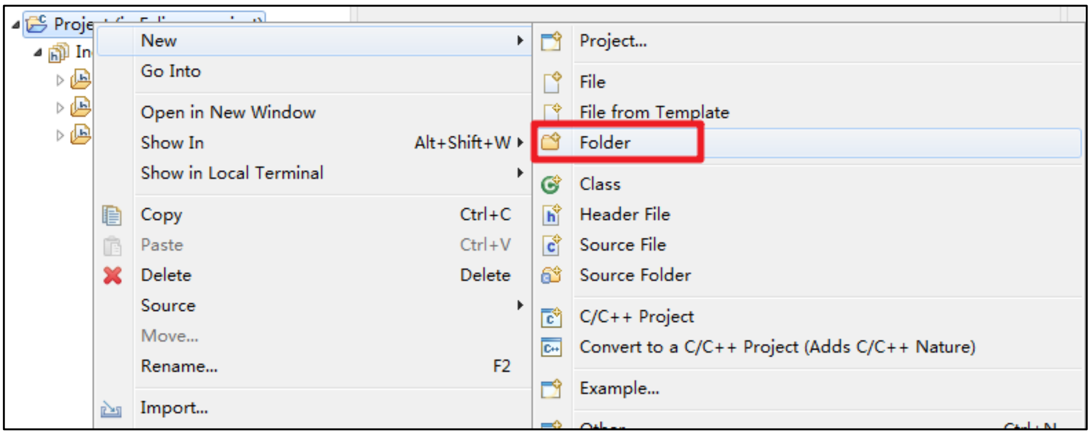
No consoles to display at this time.

3.2. New project folder and add files

3.2.1. Ceate folders and add files manually

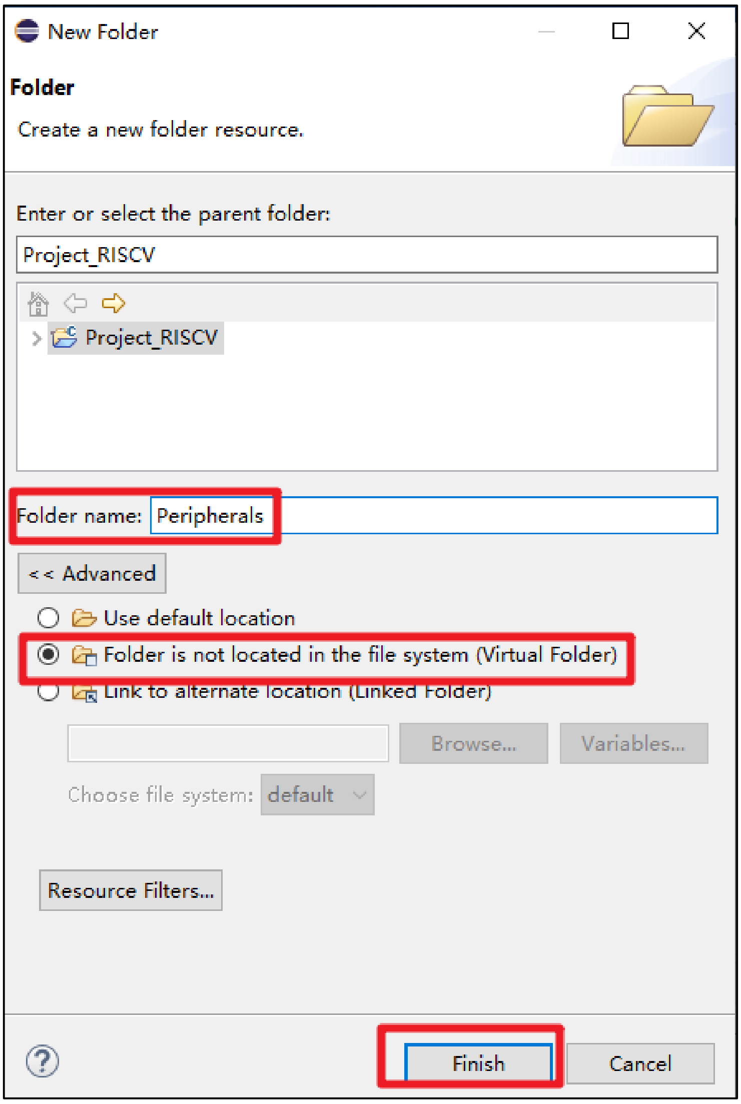
Right-click the project name and select “new->Folder”.

Figure 3-6. New project folder



Create a virtual folder “Peripherals”.

Figure 3-7. New virtual sub-folder



New Folder

Folder

Create a new folder resource.

Enter or select the parent folder:

Project_RISCV

Project_RISCV

Folder name: Peripherals

<< Advanced

○ Use default location

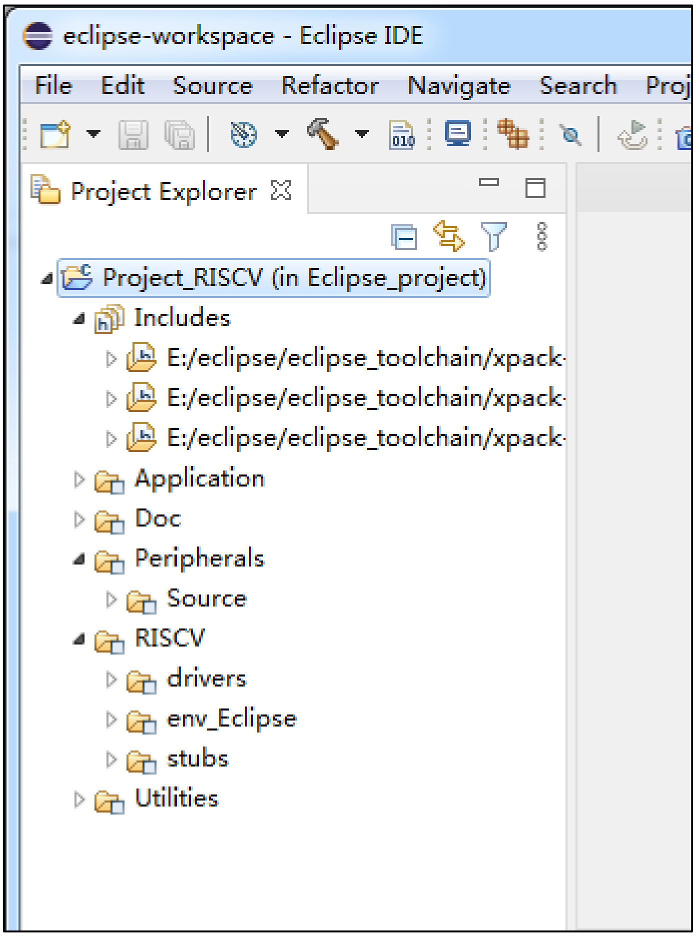
● Folder is not located in the file system (Virtual Folder)

○ Link to alternate location (Linked Folder)

Choose file system: default

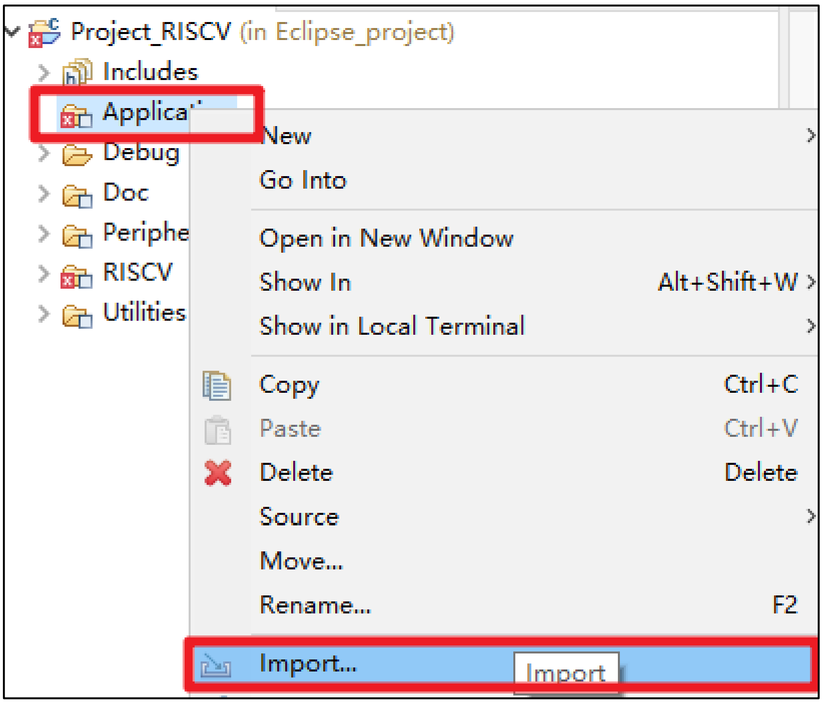
Create the Application, Doc and Utilities folders in the same way. And create a Source sub-folder in the Peripherals folder. Create drivers, env_Eclipse and stubs sub-folders inside the RISCV folder.

Figure 3-8. RISC-V project view


eclipse-workspace - Eclipse IDE
File Edit Source Refactor Navigate Search Proj
Project Explorer
Project_RISCV (in Eclipse_project)
Includes
E:/eclipse/eclipse_toolchain/xpack-
E:/eclipse/eclipse_toolchain/xpack-
E:/eclipse/eclipse_toolchain/xpack-
Application
Doc
Peripherals
Source
RISCV
drivers
env_Eclipse
stubs
Utilities

Right-click “Application” and select the “Import” option to import the file.

Figure 3-9. Add files



Import select “File System”. Select the path of the file to be imported, and tick the file to be imported.

Figure 3-10. Select files to be imported

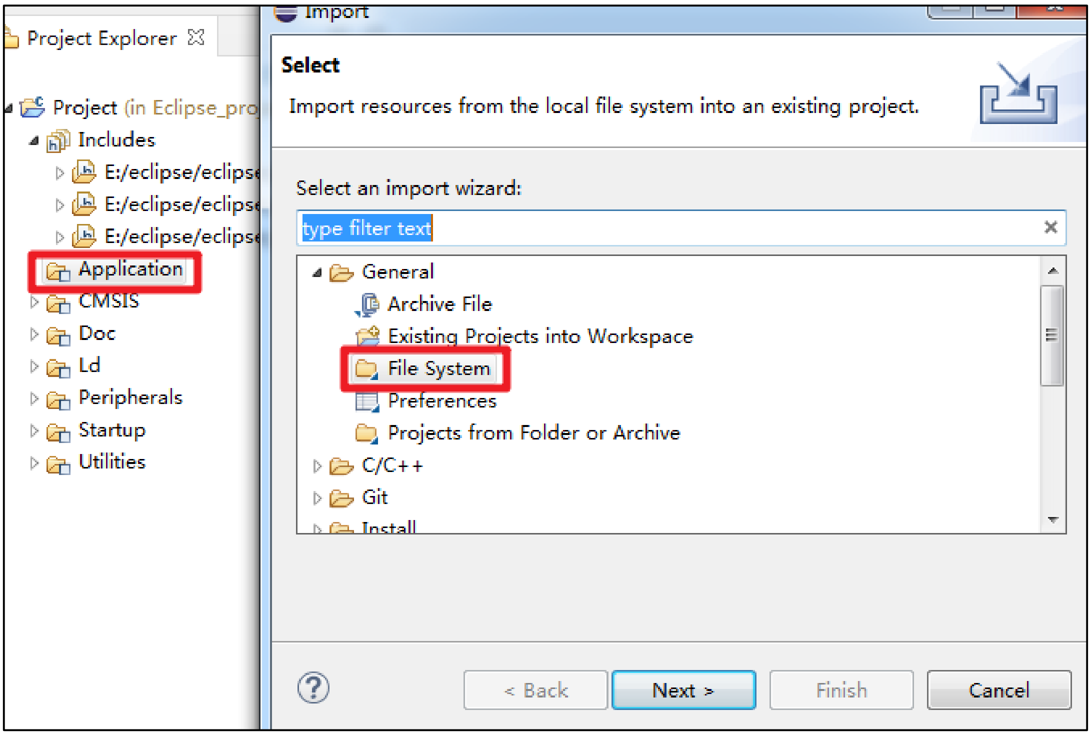
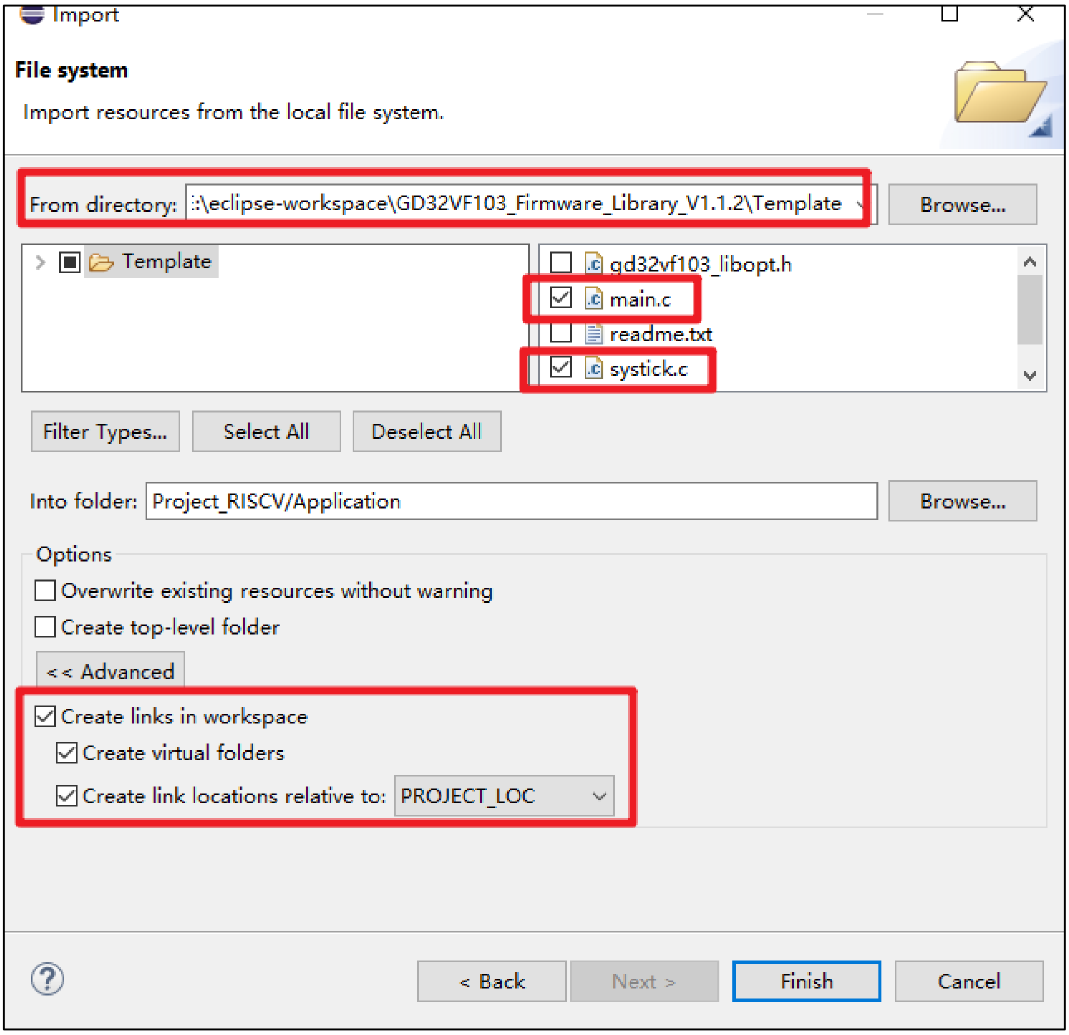


Figure 3-11. Import files to the Application folder



Import

File system

Import resources from the local file system.

From directory: :\eclipse-workspace\GD32VF103_Firmware_Library_V1.1.2\Template

- ■ Template
- □ gd32vf103_libopt.h
- ☑ main.c
- □ readme.txt
- ☑ systick.c

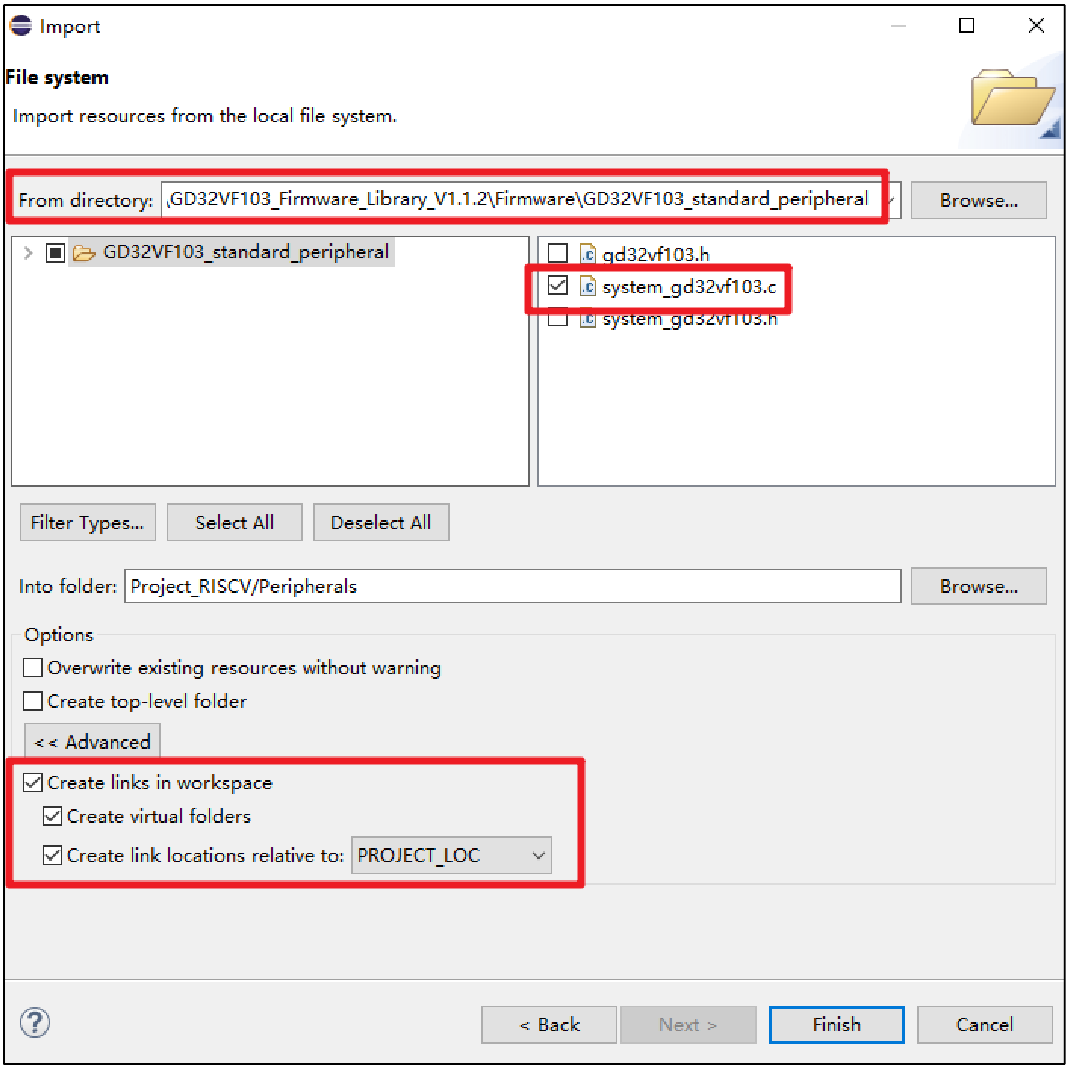
Into folder: Project_RISCV/Application

Options

- □ Overwrite existing resources without warning
- □ Create top-level folder
- ☑ Create links in workspace
- ☑ Create virtual folders
- ☑ Create link locations relative to: PROJECT_LOC

In the same way, import the required files into the “RISCV”, “Doc”, “Peripherals” and “Utilities” folders.

Figure 3-12. Import files to the Peripherals folder



Import

File system

Import resources from the local file system.

From directory: \GD32VF103_Firmware_Library_V1.1.2\Firmware\GD32VF103_standard_peripheral

■ GD32VF103_standard_peripheral

□ gd32vf103.h
☑ system_gd32vf103.c
□ system_gd32vf103.h

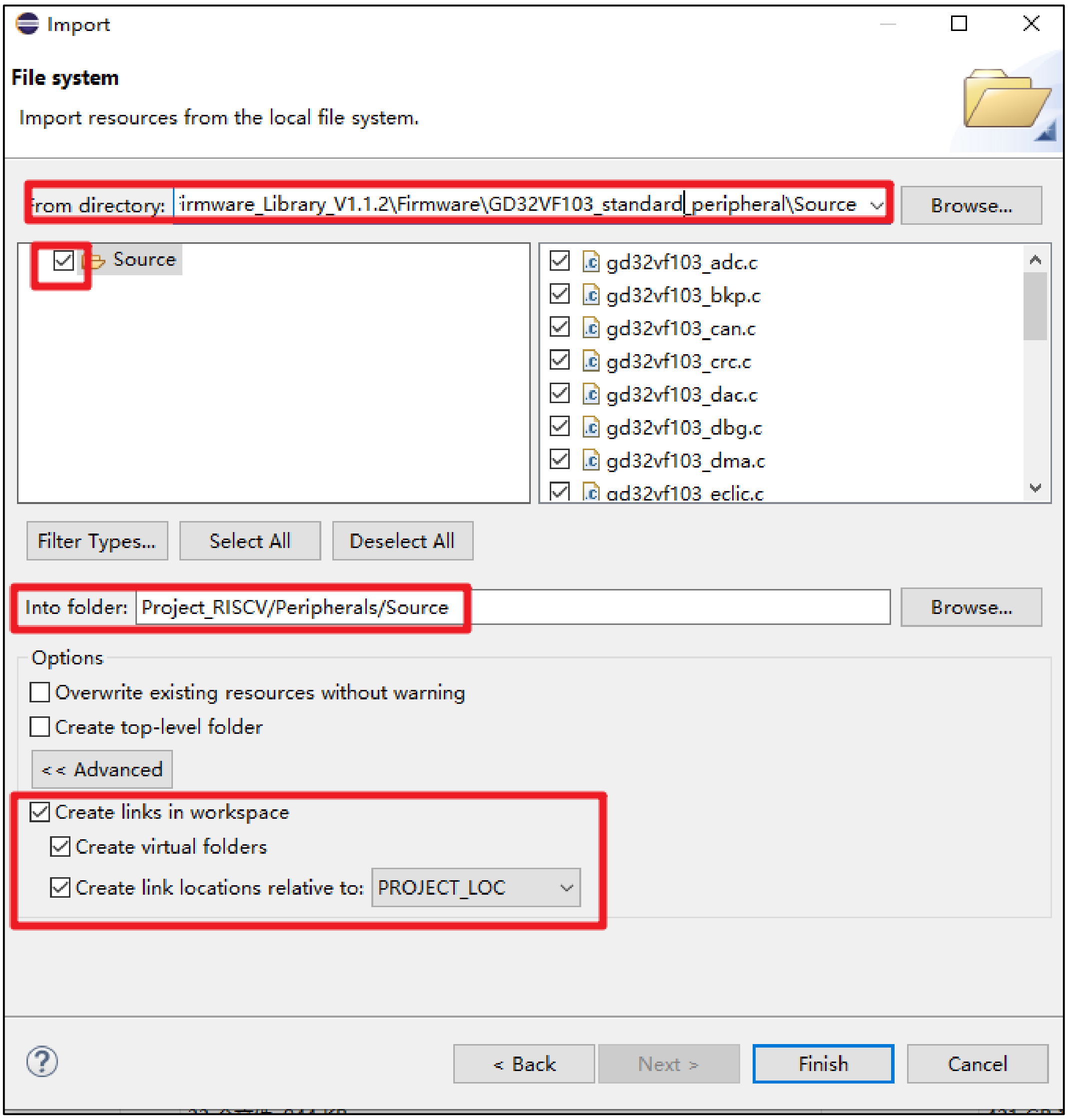
Into folder: Project_RISCV/Peripherals

Options

□ Overwrite existing resources without warning
□ Create top-level folder

☑ Create links in workspace
☑ Create virtual folders
☑ Create link locations relative to: PROJECT_LOC

Figure 3-13. Import files to the Peripheral-Source folder



Import

File system

Import resources from the local file system.

rom directory: irmware_Library_V1.1.2\Firmware\GD32VF103_standard_peripheral\Source

☑ Source

☑ gd32vf103_adc.c
☑ gd32vf103_bkp.c
☑ gd32vf103_can.c
☑ gd32vf103_crc.c
☑ gd32vf103_dac.c
☑ gd32vf103_dbg.c
☑ gd32vf103_dma.c
☑ gd32vf103_eclic.c

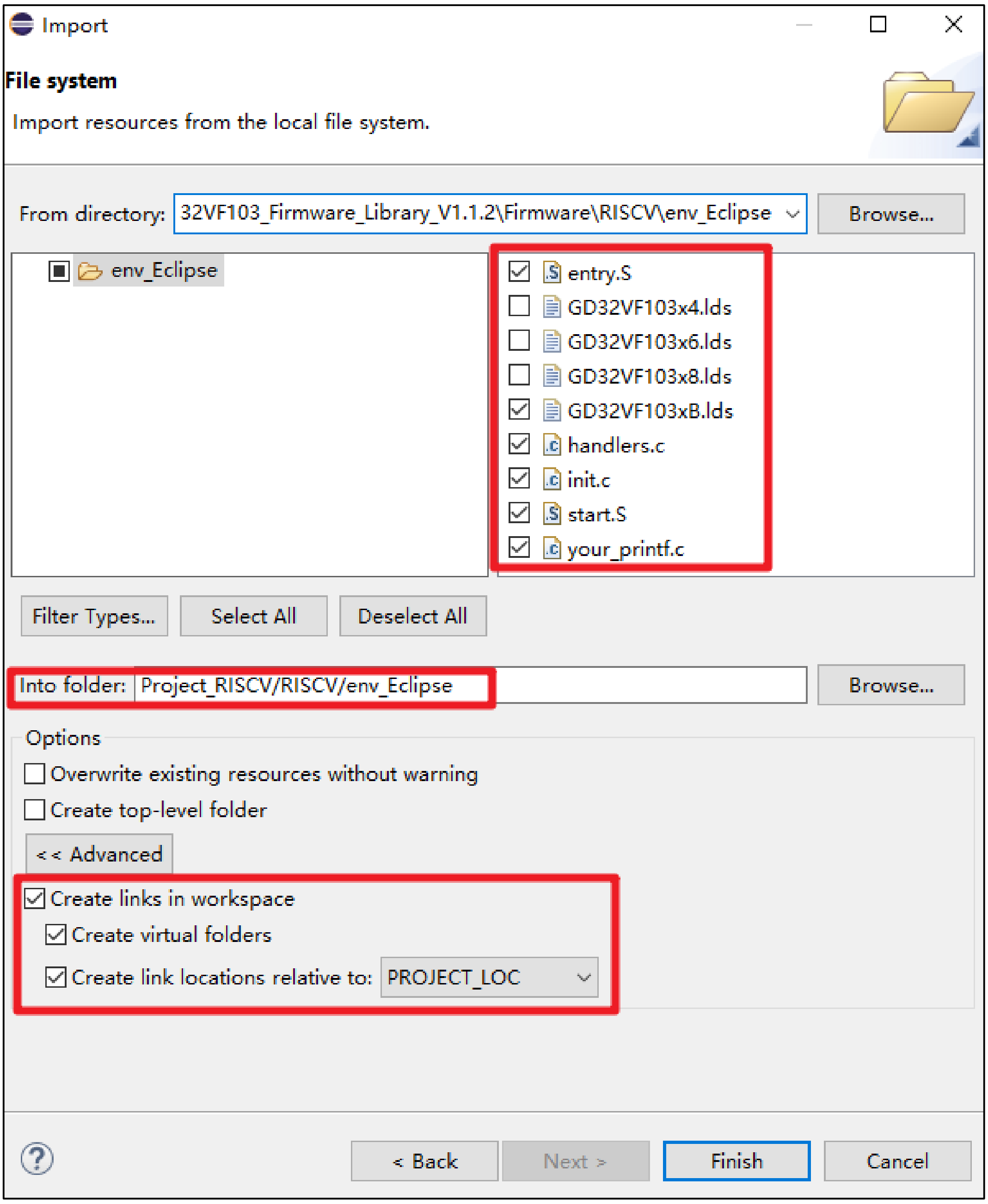
Into folder: Project_RISCV/Peripherals/Source

Options

☐ Overwrite existing resources without warning
☐ Create top-level folder

☑ Create links in workspace
☑ Create virtual folders
☑ Create link locations relative to: PROJECT_LOC

Figure 3-14. Import files to the RISCV-env_Eclipse folder



Import

File system

Import resources from the local file system.

From directory: 32VF103_Firmware_Library_V1.1.2\Firmware\RISCV\env_Eclipse

■ env_Eclipse

☑ entry.S
☐ GD32VF103x4.lds
☐ GD32VF103x6.lds
☐ GD32VF103x8.lds
☑ GD32VF103xB.lds
☑ handlers.c
☑ init.c
☑ start.S
☑ your_printf.c

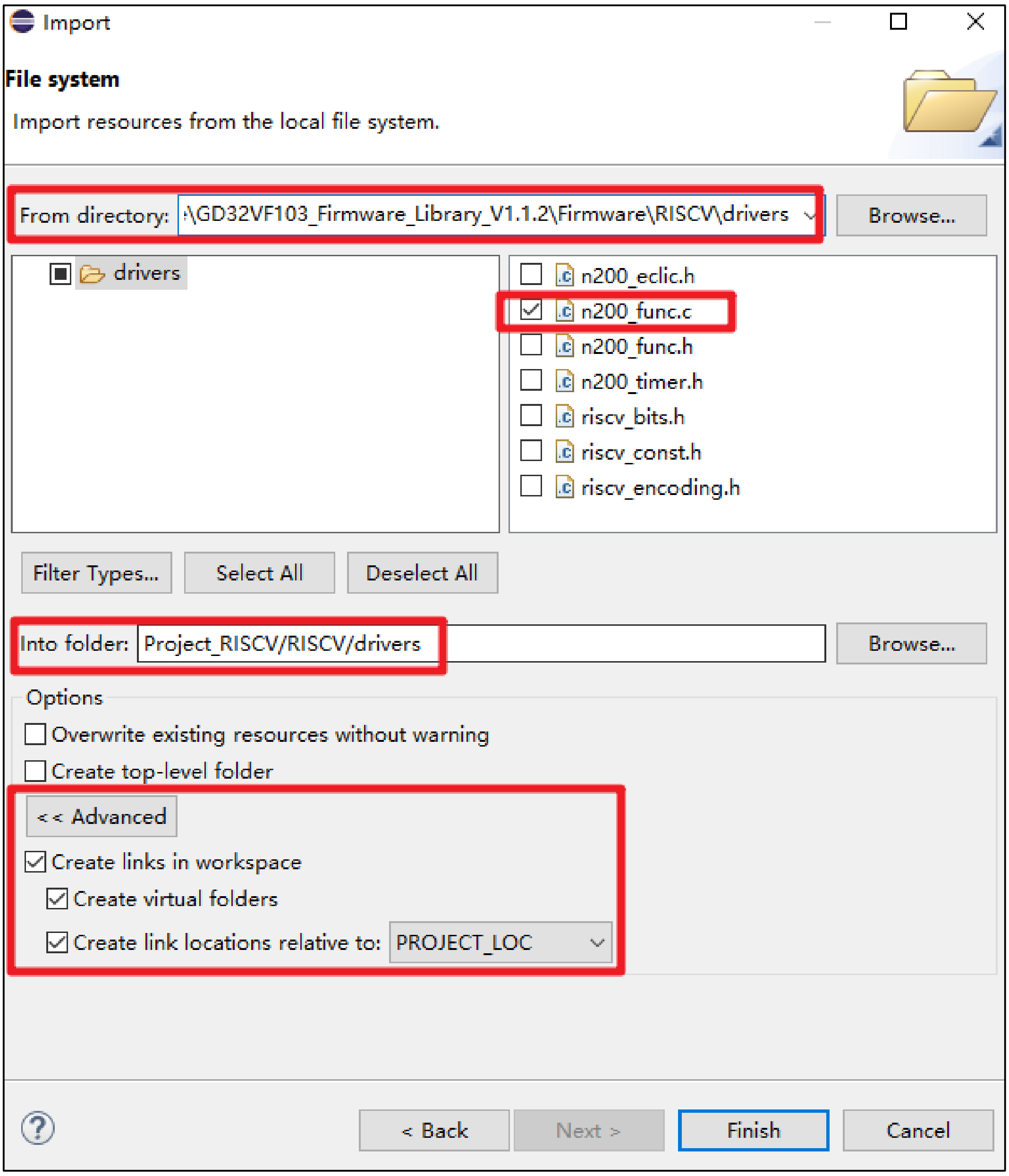
Into folder: Project_RISCV/RISCV/env_Eclipse

Options

☐ Overwrite existing resources without warning
☐ Create top-level folder

☑ Create links in workspace
☑ Create virtual folders
☑ Create link locations relative to: PROJECT_LOC

Figure 3-15. Import files to the RISCV-drivers folder



Import

File system

Import resources from the local file system.

From directory: :\GD32VF103_Firmware_Library_V1.1.2\Firmware\RISCV\drivers

■ drivers

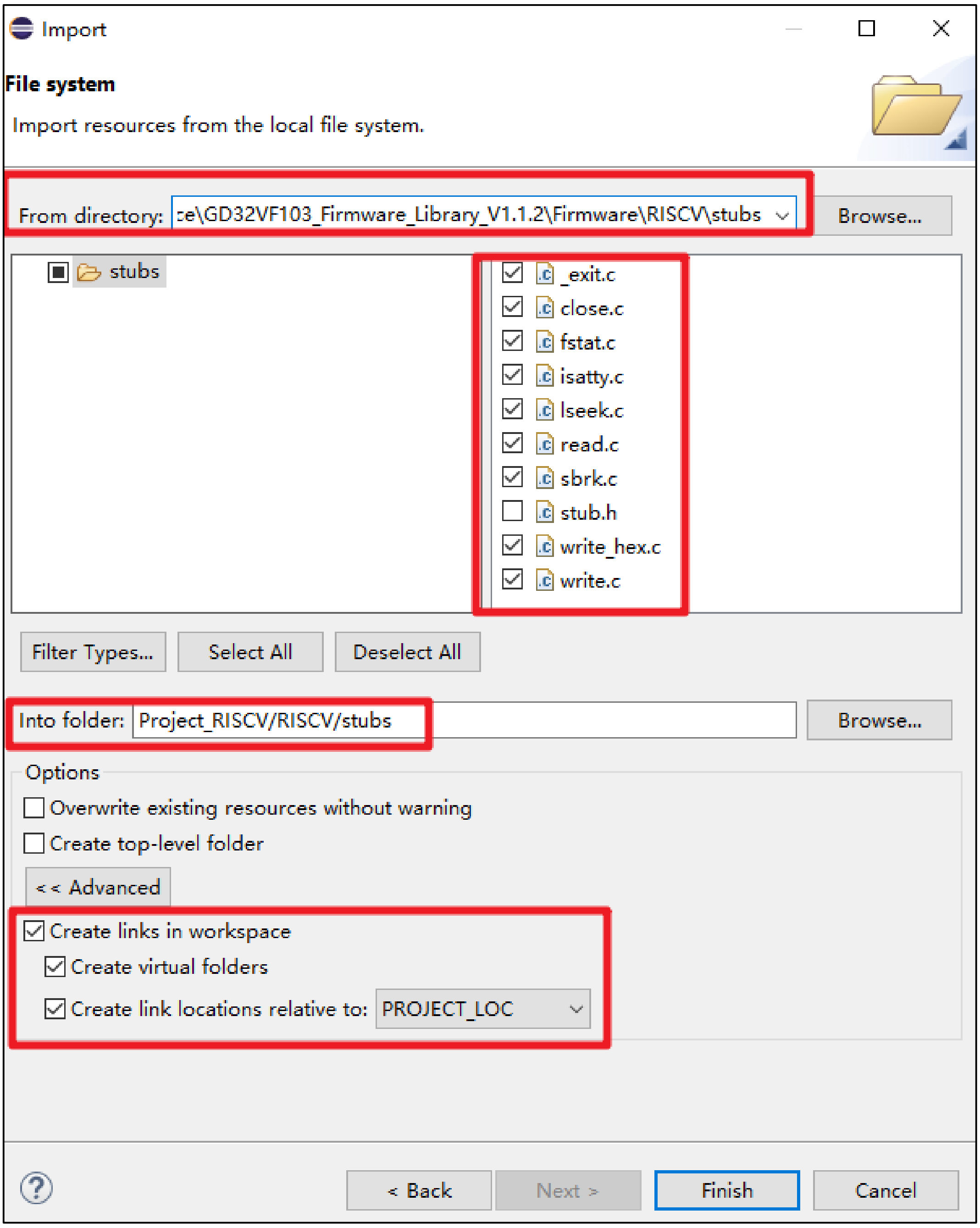
- □ n200_eclic.h
- ☑ n200_func.c
- □ n200_func.h
- □ n200_timer.h
- □ riscv_bits.h
- □ riscv_const.h
- □ riscv_encoding.h

Into folder: Project_RISCV/RISCV/drivers

Options

- □ Overwrite existing resources without warning
- □ Create top-level folder
- ☑ Create links in workspace
- ☑ Create virtual folders
- ☑ Create link locations relative to: PROJECT_LOC

Figure 3-16. Import files to the RISCV-stubs folder



Import

File system

Import resources from the local file system.

From directory: :e\GD32VF103_Firmware_Library_V1.1.2\Firmware\RISCV\stubs

■ stubs

☑ _exit.c
☑ close.c
☑ fstat.c
☑ isatty.c
☑ lseek.c
☑ read.c
☑ sbrk.c
☐ stub.h
☑ write_hex.c
☑ write.c

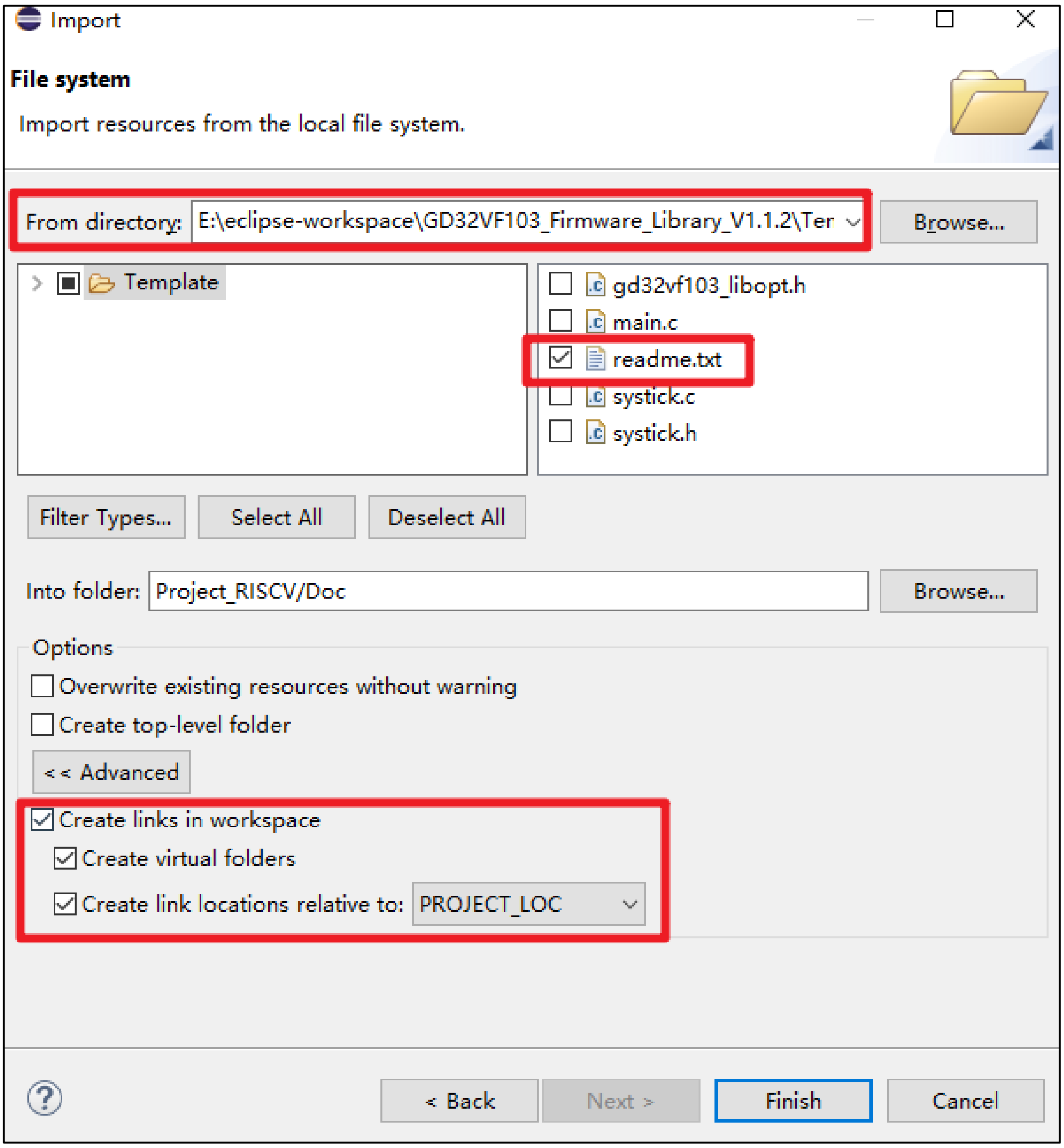
Into folder: Project_RISCV/RISCV/stubs

Options

☐ Overwrite existing resources without warning
☐ Create top-level folder

☑ Create links in workspace
☑ Create virtual folders
☑ Create link locations relative to: PROJECT_LOC

Figure 3-17. Import files to the Doc folder



Import

File system

Import resources from the local file system.

From directory: E:\eclipse-workspace\GD32VF103_Firmware_Library_V1.1.2\Ter

- ■ Template

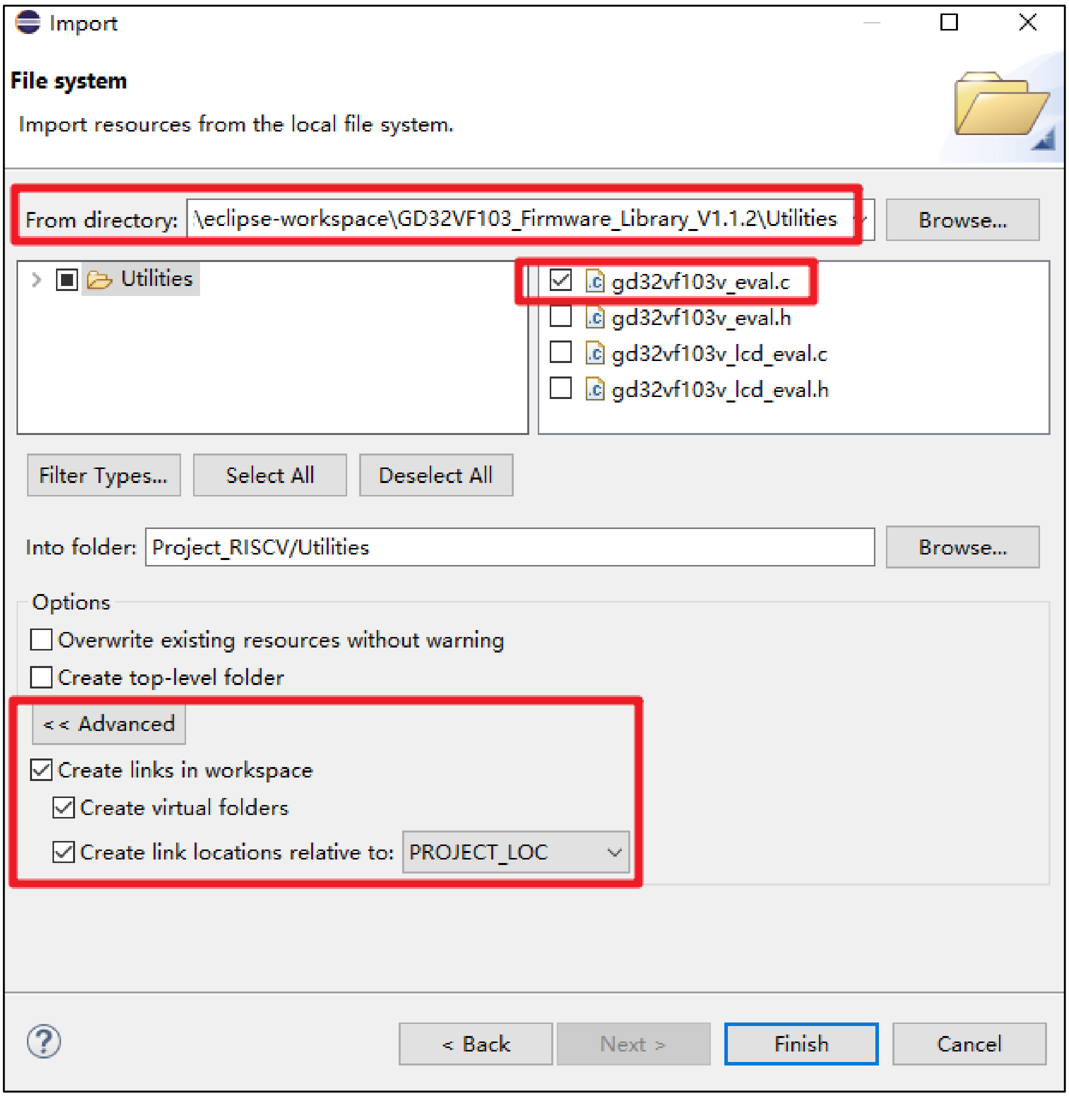
- ☐ gd32vf103_libopt.h
- ☐ main.c
- ☑ readme.txt
- ☐ systick.c
- ☐ systick.h

Into folder: Project_RISCV/Doc

Options

- ☐ Overwrite existing resources without warning
- ☐ Create top-level folder
- ☑ Create links in workspace
  - ☑ Create virtual folders
  - ☑ Create link locations relative to: PROJECT_LOC

Figure 3-18. Import files to the Utilities folder



Import

File system

Import resources from the local file system.

From directory: \eclipse-workspace\GD32VF103_Firmware_Library_V1.1.2\Utilities

■ Utilities

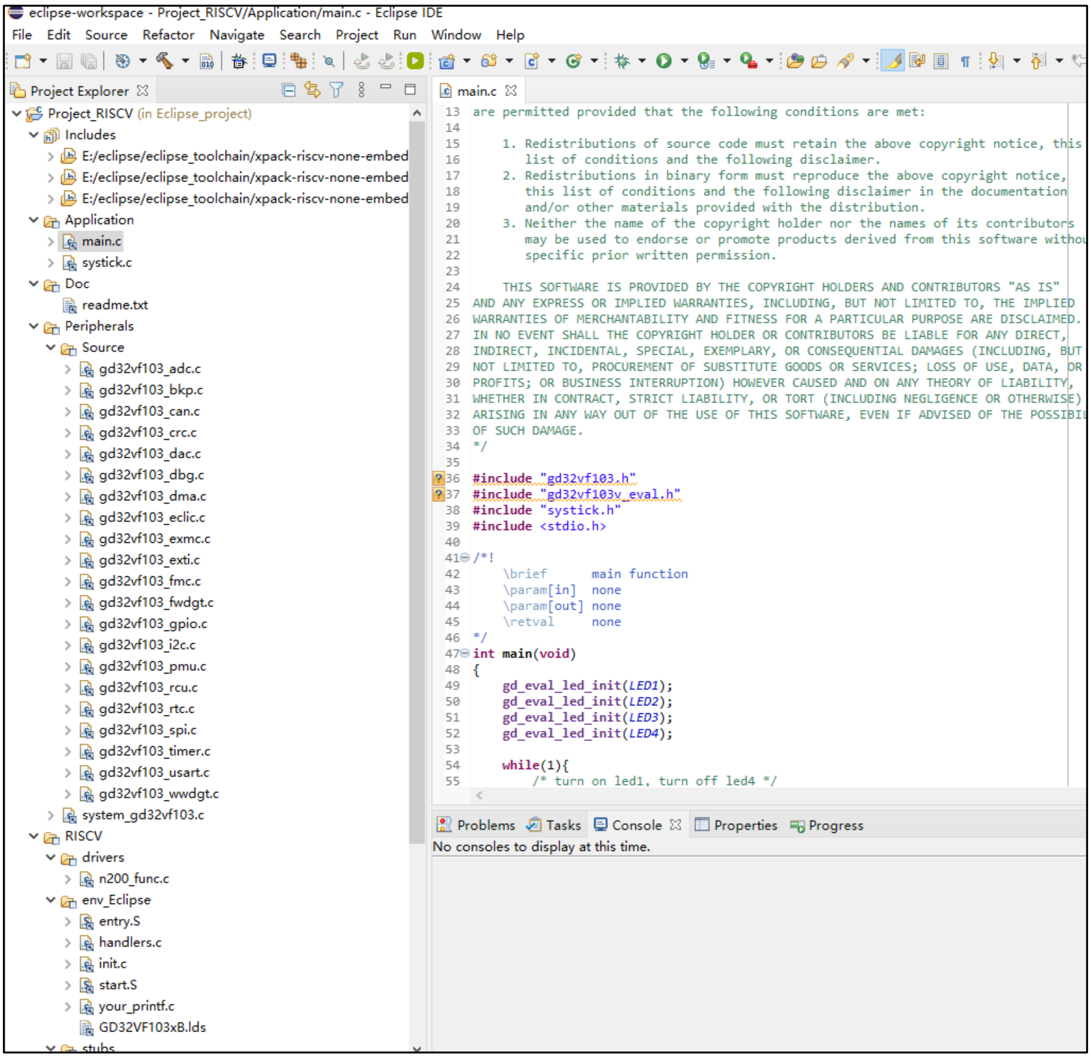
☑ gd32vf103v_eval.c
☐ gd32vf103v_eval.h
☐ gd32vf103v_lcd_eval.c
☐ gd32vf103v_lcd_eval.h

Into folder: Project_RISCV/Utilities

Options
☐ Overwrite existing resources without warning
☐ Create top-level folder

☑ Create links in workspace
☑ Create virtual folders
☑ Create link locations relative to: PROJECT_LOC

Figure 3-19. Final RISC-V project view


eclipse-workspace - Project_RISCV/Application/main.c - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
main.c
Problems
Tasks
Console
Properties
Progress

```
Project_RISCV (in Eclipse_project)
  Includes
    E:/eclipse/eclipse_toolchain/xpack-riscv-none-embed
    E:/eclipse/eclipse_toolchain/xpack-riscv-none-embed
    E:/eclipse/eclipse_toolchain/xpack-riscv-none-embed
  Application
    main.c
    systick.c
  Doc
    readme.txt
  Peripherals
    Source
      gd32vf103_adc.c
      gd32vf103_bkp.c
      gd32vf103_can.c
      gd32vf103_crc.c
      gd32vf103_dac.c
      gd32vf103_dbg.c
      gd32vf103_dma.c
      gd32vf103_eclic.c
      gd32vf103_exmc.c
      gd32vf103_exti.c
      gd32vf103_fmc.c
      gd32vf103_fwdgt.c
      gd32vf103_gpio.c
      gd32vf103_i2c.c
      gd32vf103_pmu.c
      gd32vf103_rcu.c
      gd32vf103_rtc.c
      gd32vf103_spi.c
      gd32vf103_timer.c
      gd32vf103_usart.c
      gd32vf103_wwdgt.c
    system_gd32vf103.c
  RISCV
    drivers
      n200_func.c
    env_Eclipse
      entry.S
      handlers.c
      init.c
      start.S
      your_printf.c
      GD32VF103xB.lds
    stubs
```

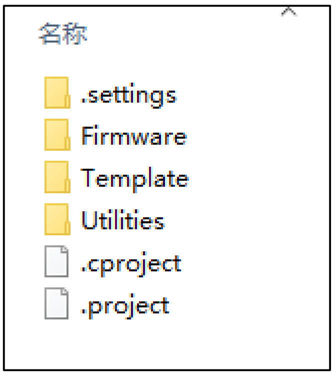
```
are permitted provided that the following conditions are met:

    1. Redistributions of source code must retain the above copyright notice, this
       list of conditions and the following disclaimer.
    2. Redistributions in binary form must reproduce the above copyright notice,
       this list of conditions and the following disclaimer in the documentation
       and/or other materials provided with the distribution.
    3. Neither the name of the copyright holder nor the names of its contributors
       may be used to endorse or promote products derived from this software withou
       specific prior written permission.

    THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBIL
OF SUCH DAMAGE.
*/

#include "gd32vf103.h"
#include "gd32vf103v_eval.h"
#include "systick.h"
#include <stdio.h>

/*!
    \brief      main function
    \param[in]  none
    \param[out] none
    \retval     none
*/
int main(void)
{
    gd_eval_led_init(LED1);
    gd_eval_led_init(LED2);
    gd_eval_led_init(LED3);
    gd_eval_led_init(LED4);

    while(1){
        /* turn on led1, turn off led4 */
```

No consoles to display at this time.

3.2.2. Ceate folders and add files by “Refresh”

In addition to the above-mentioned method of creating folders and importing corresponding files manually, user can also put the files that need to be imported together with its folders in the folder at the same level as the created .cproject file. In the Eclipse IDE, right-click the project name and select “Refresh” to import the folders and files into the project directly.

Figure 3-20. Project folder structure



名称

.settings
Firmware
Template
Utilities
.cproject
.project

Figure 3-21. Refresh the project

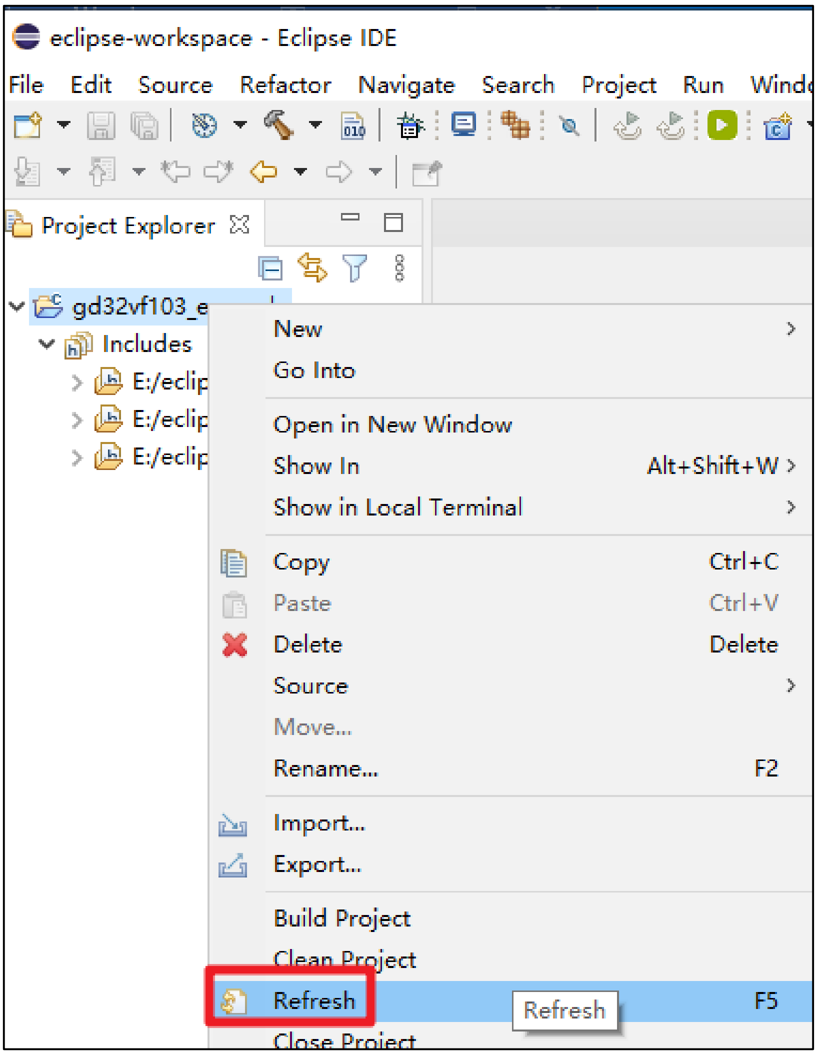
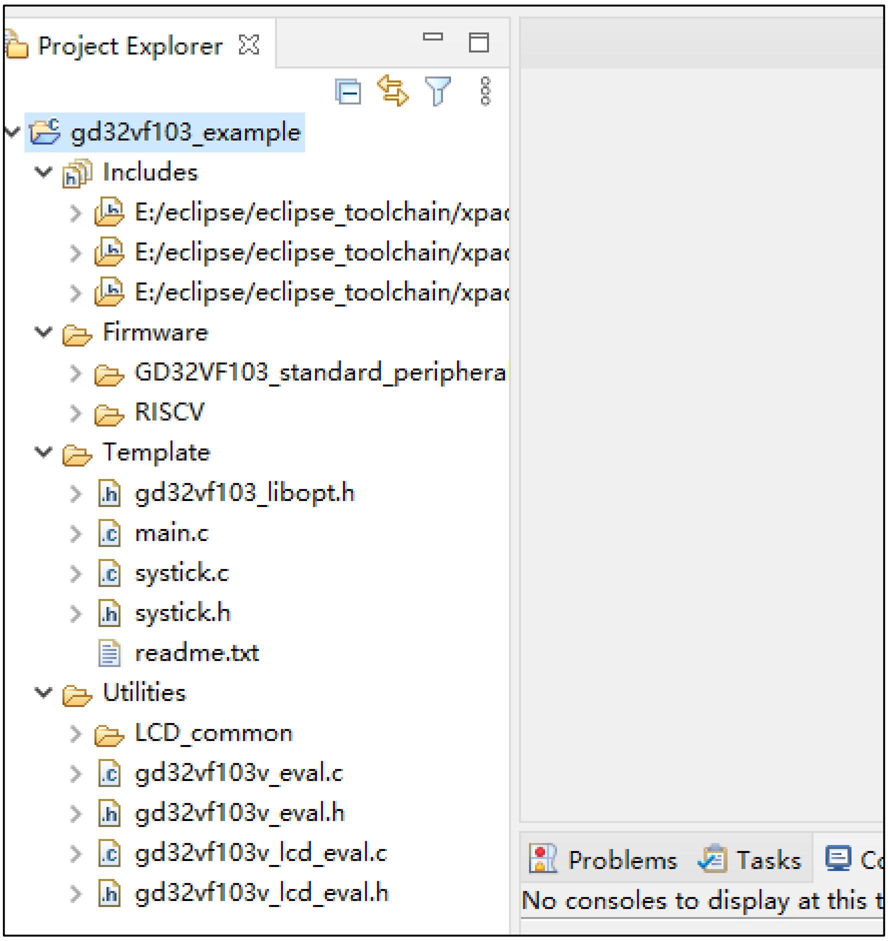


Figure 3-22. Project structure in Eclipse IDE



```
Project Explorer
gd32vf103_example
  Includes
    E:/eclipse/eclipse_toolchain/xpac
    E:/eclipse/eclipse_toolchain/xpac
    E:/eclipse/eclipse_toolchain/xpac
  Firmware
    GD32VF103_standard_periphera
    RISCV
  Template
    gd32vf103_libopt.h
    main.c
    systick.c
    systick.h
    readme.txt
  Utilities
    LCD_common
    gd32vf103v_eval.c
    gd32vf103v_eval.h
    gd32vf103v_lcd_eval.c
    gd32vf103v_lcd_eval.h
```

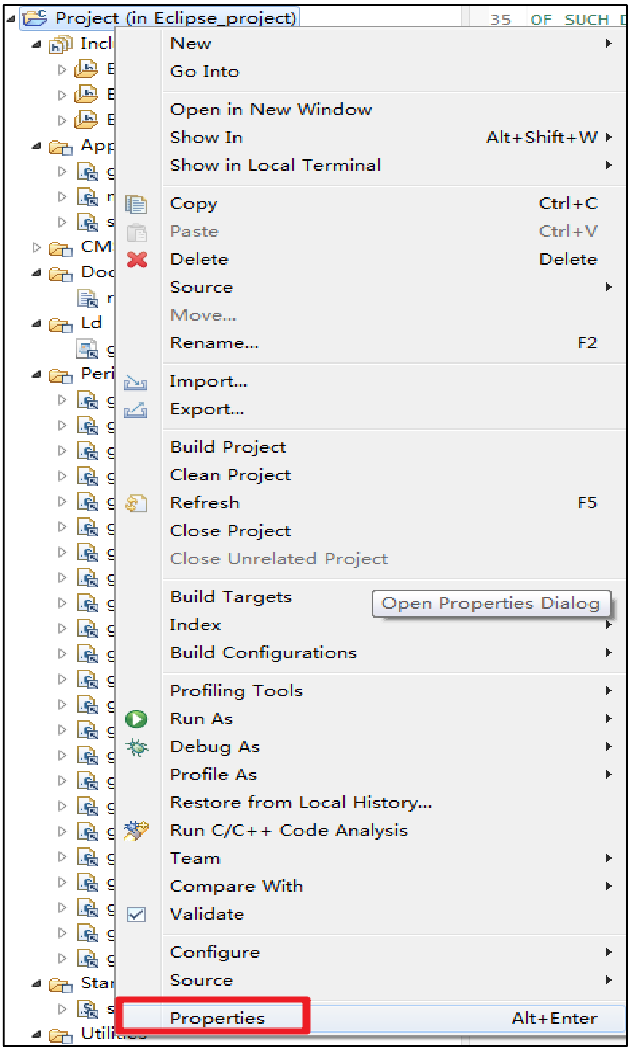
No consoles to display at this t

Note: The files and folders created in the "Refresh" method are all real, and once a file is deleted in the Eclipse IDE, the file will be deleted from the disk directly.

3.3. Project configurations

Right-click the project and select the “Properties” option to open it.

Figure 3-23. Project properties configurations

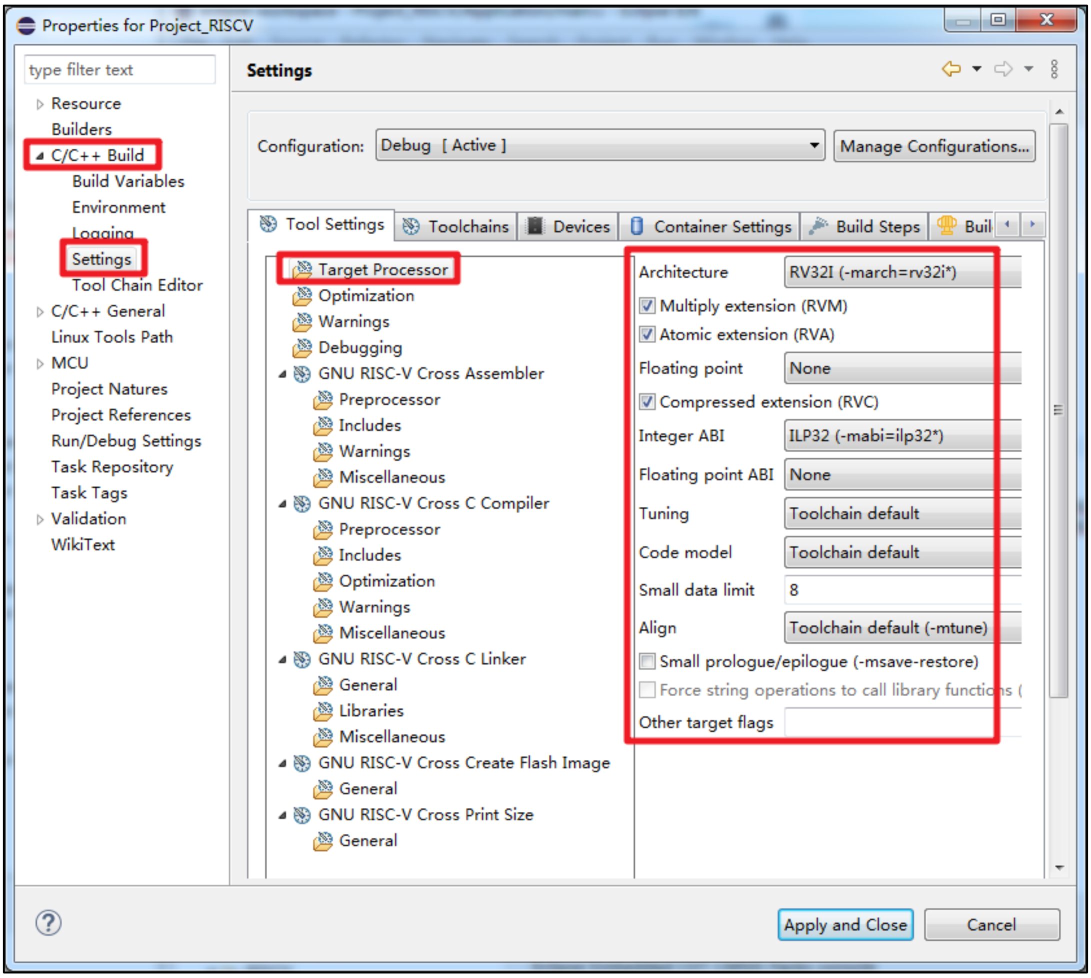


3.3.1. Target Processor option configuration

"C/C++ Build->Settings->Tool Settings->Target Processor" option configurations:

According to the core of the target chip. In this guide, select RV32I.

Figure 3-24. Target Processor configuration



Configuration: Debug [Active]

Architecture: RV32I (-march=rv32i*)

☑ Multiply extension (RVM)

☑ Atomic extension (RVA)

Floating point: None

☑ Compressed extension (RVC)

Integer ABI: ILP32 (-mabi=ilp32*)

Floating point ABI: None

Tuning: Toolchain default

Code model: Toolchain default

Small data limit: 8

Align: Toolchain default (-mtune)

☐ Small prologue/epilogue (-msave-restore)

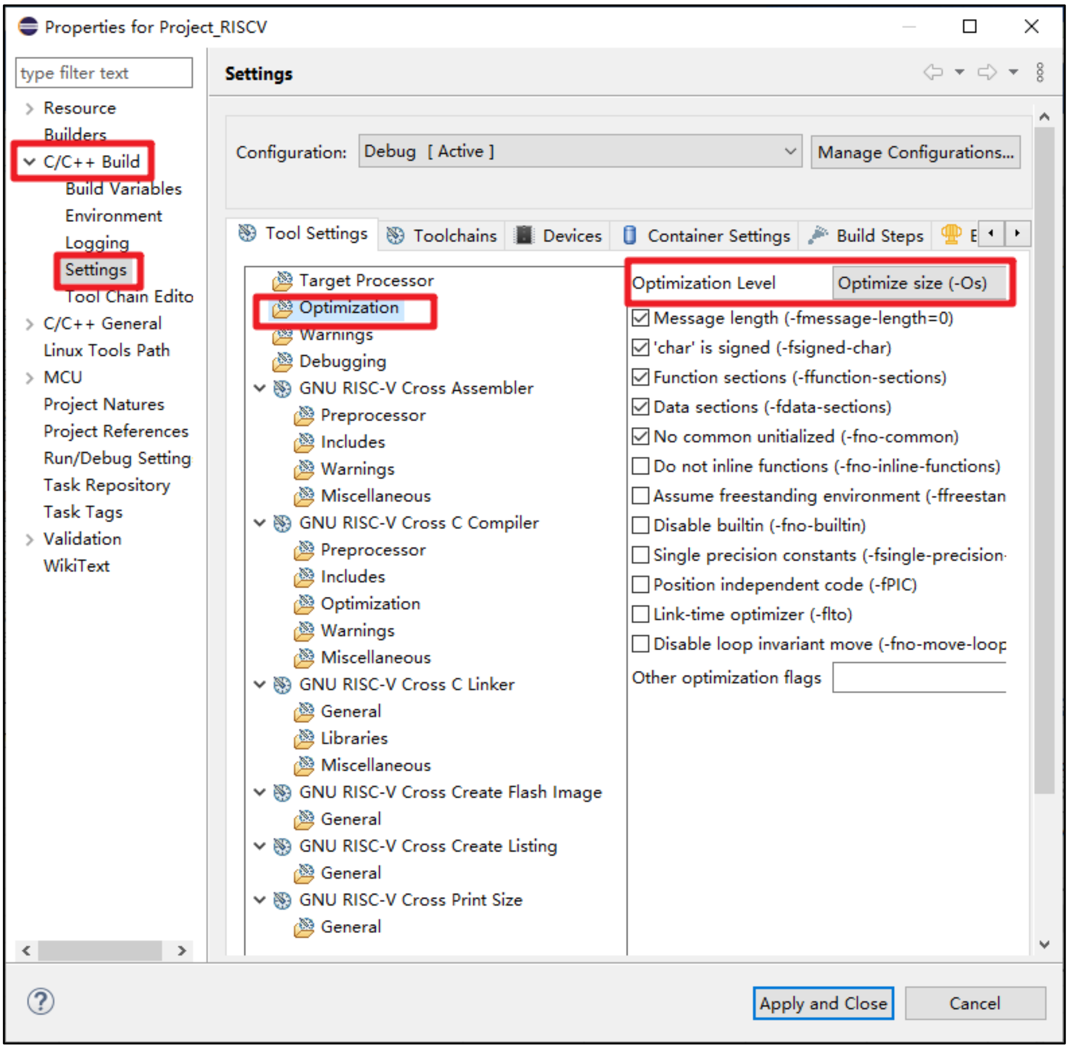
☐ Force string operations to call library functions (

Other target flags:

3.3.2. Optimization option configuration

Configure the optimization level in the "C/C++ Build->Settings->Tool Settings->Optimization" option, with options -O0, -O1, -O2, -O3, -Os, -Ofast, -Og.

Figure 3-25. Optimization configuration



Configuration: Debug [Active]

Optimization Level: Optimize size (-Os)

☑ Message length (-fmessage-length=0)
☑ 'char' is signed (-fsigned-char)
☑ Function sections (-ffunction-sections)
☑ Data sections (-fdata-sections)
☑ No common unitialized (-fno-common)
☐ Do not inline functions (-fno-inline-functions)
☐ Assume freestanding environment (-ffreestan
☐ Disable builtin (-fno-builtin)
☐ Single precision constants (-fsingle-precision-
☐ Position independent code (-fPIC)
☐ Link-time optimizer (-flto)
☐ Disable loop invariant move (-fno-move-loop

Other optimization flags:

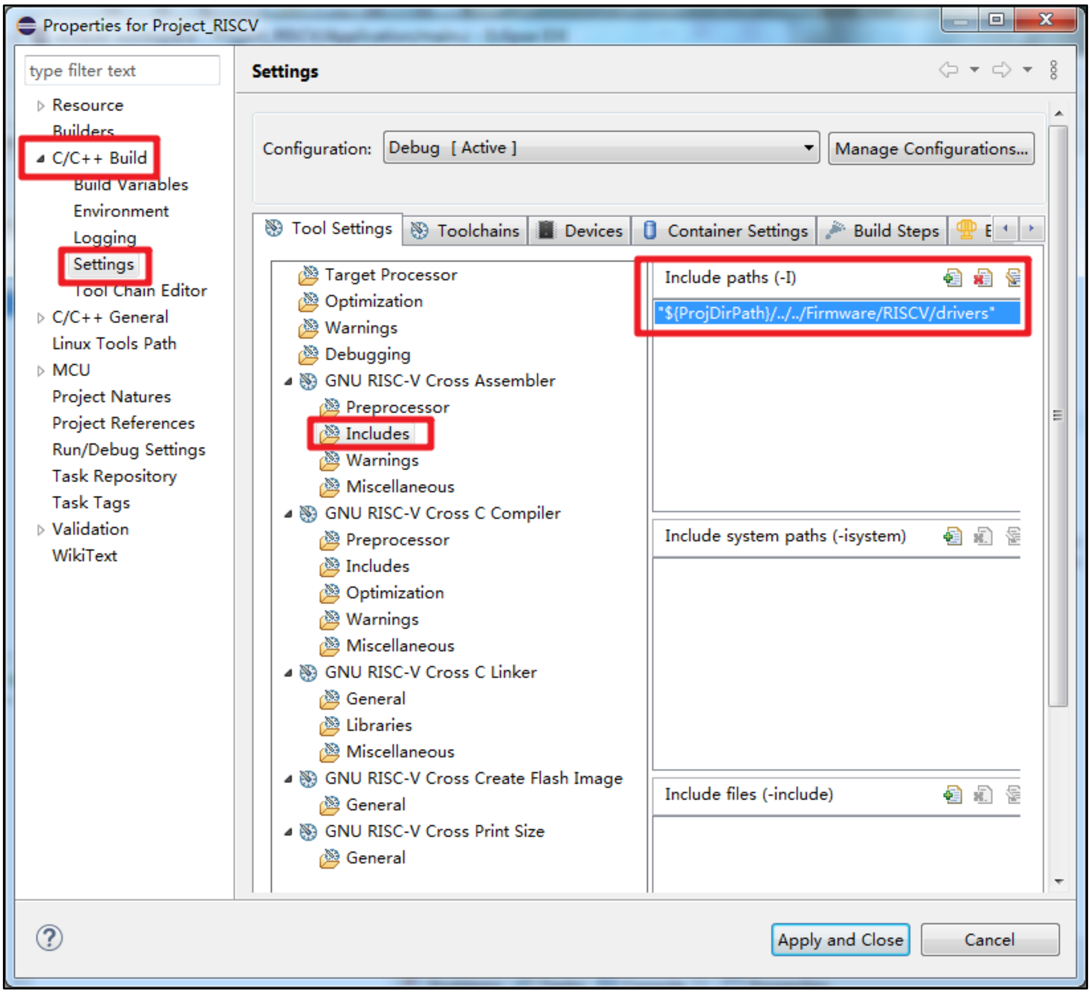
3.3.3. GNU RISC-V Cross Assembler configuration

Configure Cross C compilation options in the "C/C++ Build->Settings->Tool Settings->GNU RISC-V Cross Assembler" option.

Add the Assembler header file path required by the project in the "includes->Include paths" option. Add In this guide:

"${ProjDirPath}/../../Firmware/RISCV/drivers"

Figure 3-26. GNU RISC-V Cross Assembler -> Includes configuration



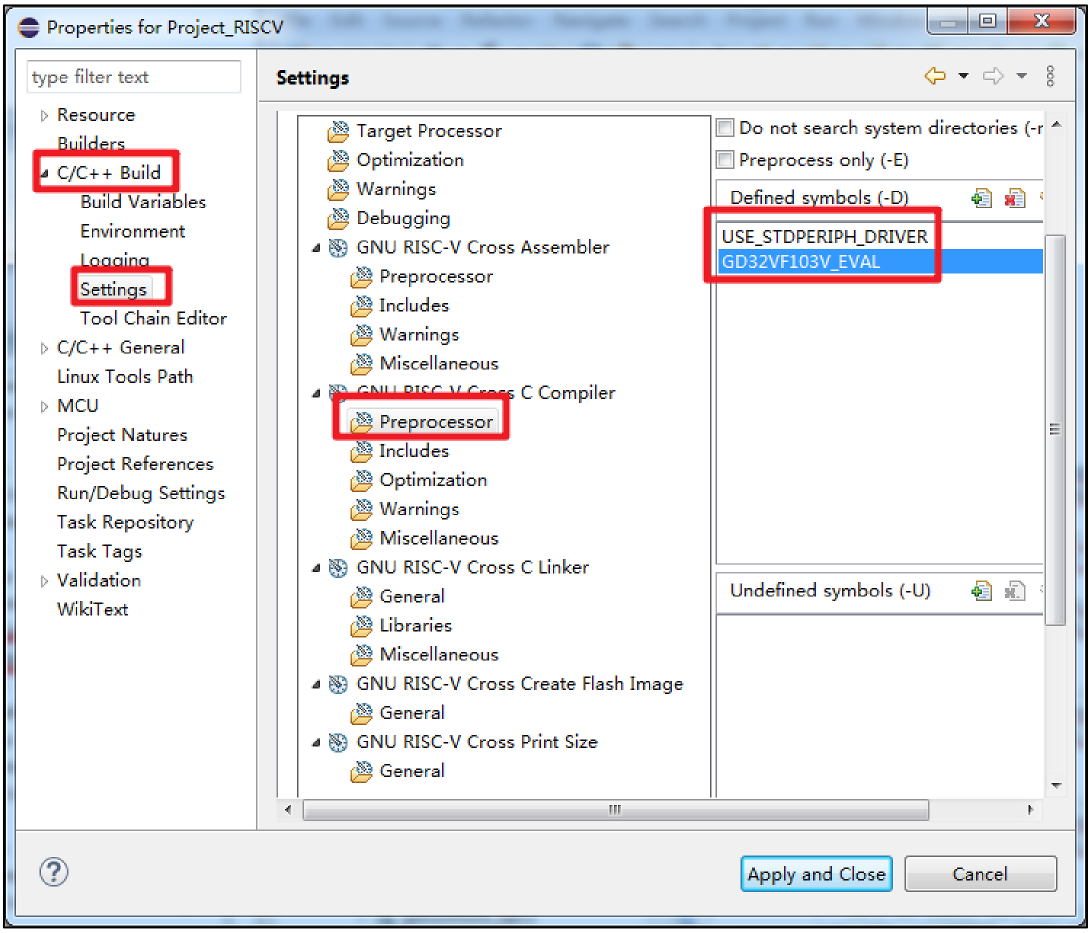
Include paths (-I): "${ProjDirPath}/../../Firmware/RISCV/drivers"

3.3.4. GNU RISC-V Cross C Compiler configuration

Configure Cross C compilation options in the "C/C++ Build->Settings->Tool Settings->GNU RISC-V Cross C Compiler" option.

In this guide, add USE_STDPERIPH_DRIVER and GD32VF103V_EVAL pre-compiled macros in the ‘’Preprocessor->Defined symbols’ option.

Figure 3-27. GNU RISC-V Cross C Compiler -> Preprocessor configuration



□ Do not search system directories (-r

□ Preprocess only (-E)

Defined symbols (-D)

USE_STDPERIPH_DRIVER
GD32VF103V_EVAL

Undefined symbols (-U)

Add the header file paths required by the project in the "includes->Include paths" option. Add in this guide:

"${ProjDirPath}/../../Firmware/RISCV/env_Eclipse"

"${ProjDirPath}/../../Firmware/RISCV/drivers"

"${ProjDirPath}/../../Firmware/GD32VF103_standard_peripheral/Include"

"${ProjDirPath}/../../Template"

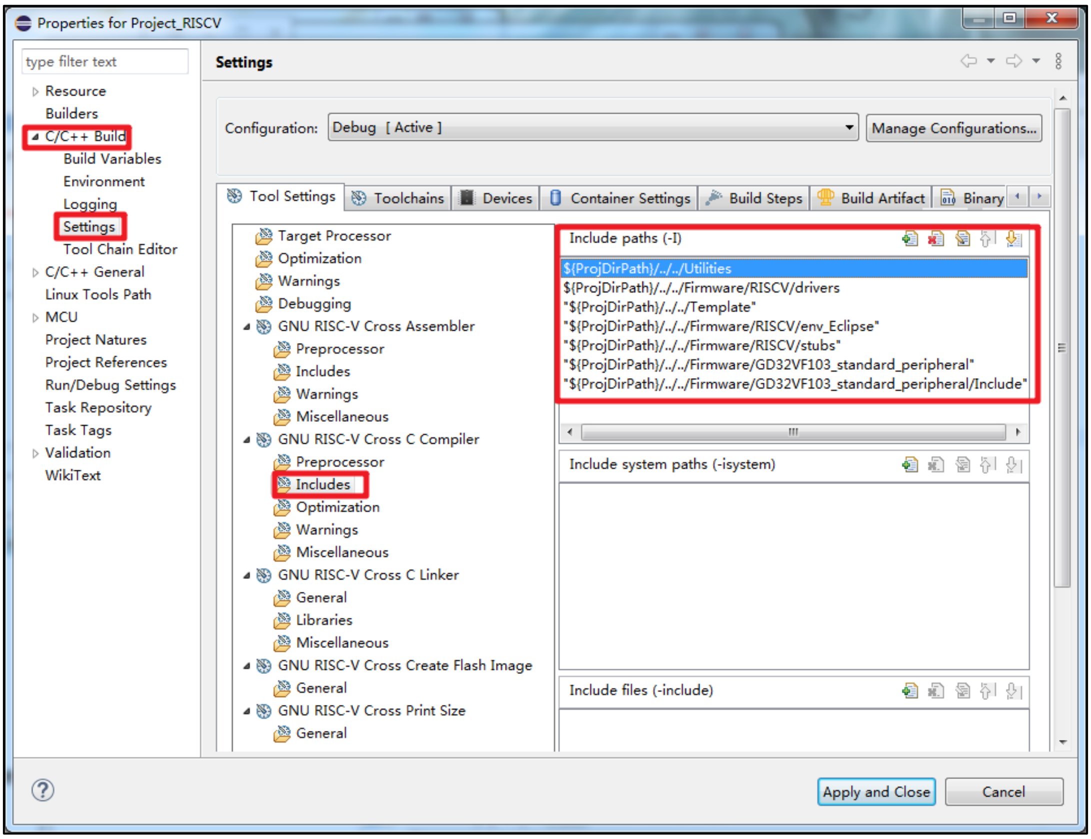
"${ProjDirPath}/../../Utilities"

"${ProjDirPath}/../../Firmware/RISCV/stubs"

"${ProjDirPath}/../../Firmware/GD32VF103_standard_peripheral"

Note: The header file path added in this guide is a relative path. User can also add the absolute path directly here.

Figure 3-28. GNU RISC-V Cross C Compiler -> Includes configuration



Configuration: Debug [Active]

Include paths (-I)

```
${ProjDirPath}/../../Utilities
${ProjDirPath}/../../Firmware/RISCV/drivers
"${ProjDirPath}/../../Template"
"${ProjDirPath}/../../Firmware/RISCV/env_Eclipse"
"${ProjDirPath}/../../Firmware/RISCV/stubs"
"${ProjDirPath}/../../Firmware/GD32VF103_standard_peripheral"
"${ProjDirPath}/../../Firmware/GD32VF103_standard_peripheral/Include"
```

3.3.5. GNU RISC-V Cross C Linker configuration

Configure Cross C link options in "C/C++ Build->Settings->Tool Settings->GNU RISC-V Cross C Linker".

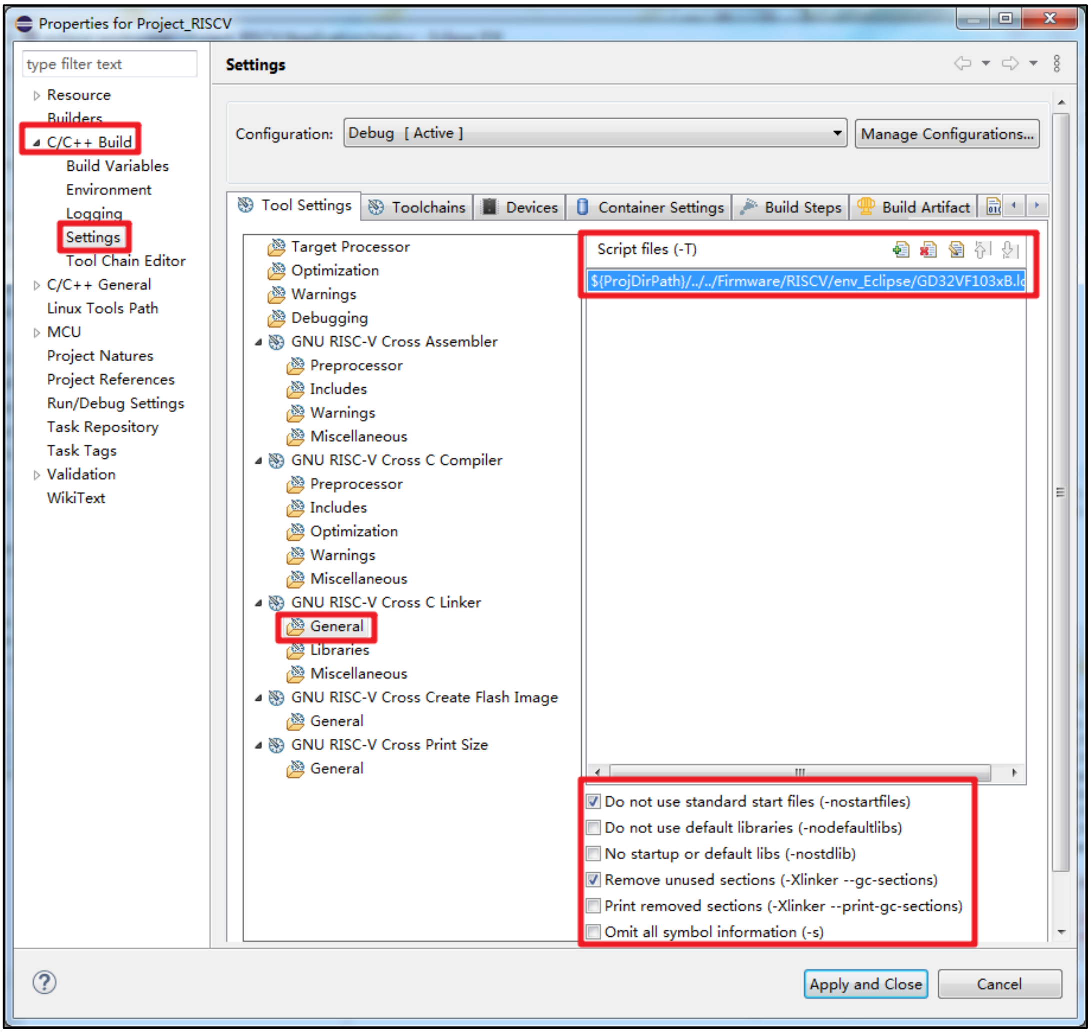
Add in the "General ->Script files" option:

"${ProjDirPath}/../../Firmware/RISCV/env_Eclipse/GD32VF103xB.lds"

The linker script is responsible for telling the linker how to configure memory for the compiled executable file. The ld script used should conform to the FLASH and SRAM size of the target chip and the memory configuration required by the customer.

Note: The ld file path added in this guide is a relative path. User can also add the absolute path directly here.

Figure 3-29. GNU RISC-V Cross C Linker -> General configuration



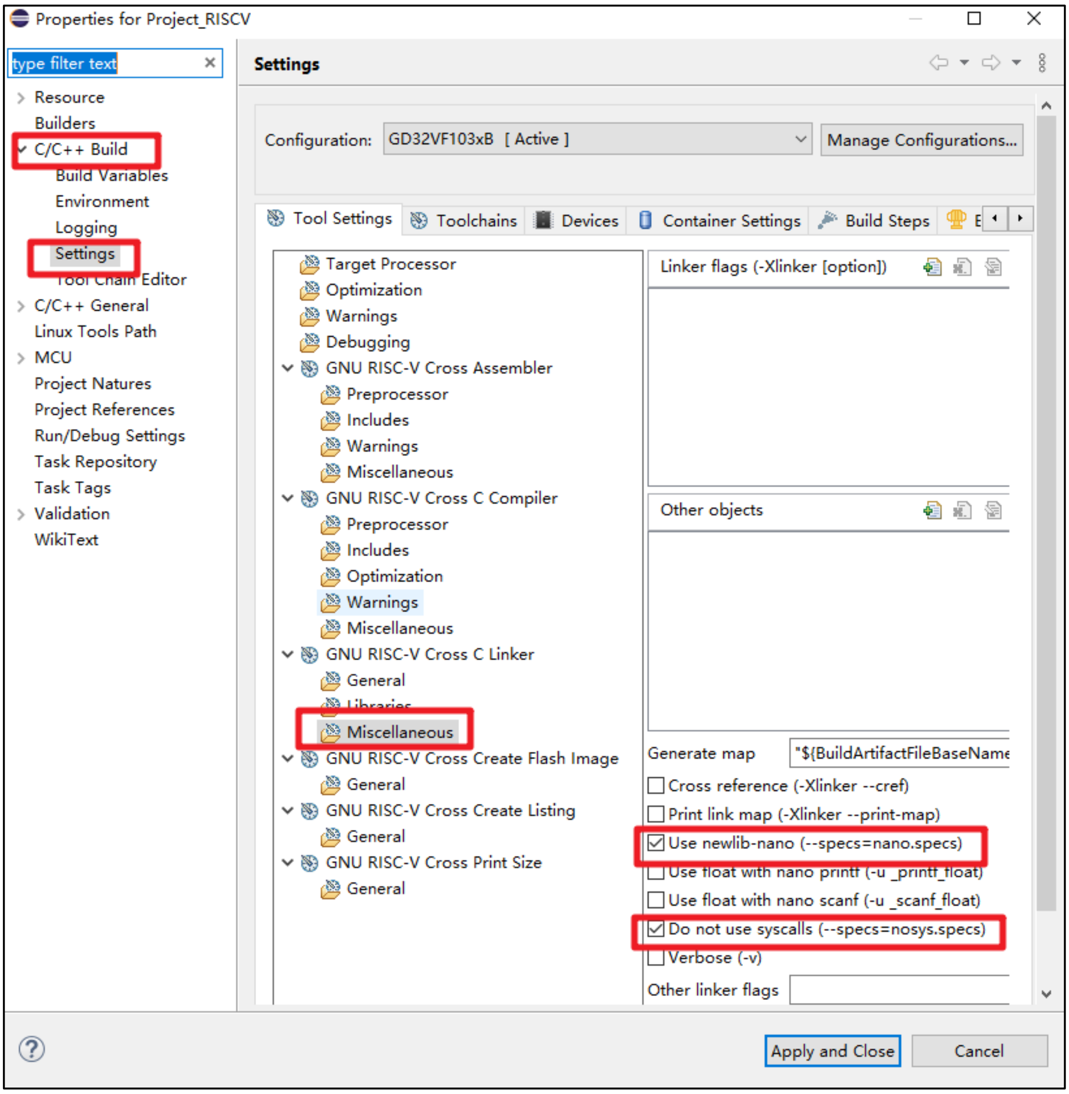
Configuration: Debug [Active]

Script files (-T)
${ProjDirPath}/../../Firmware/RISCV/env_Eclipse/GD32VF103xB.lc

☑ Do not use standard start files (-nostartfiles)
☐ Do not use default libraries (-nodefaultlibs)
☐ No startup or default libs (-nostdlib)
☑ Remove unused sections (-Xlinker --gc-sections)
☐ Print removed sections (-Xlinker --print-gc-sections)
☐ Omit all symbol information (-s)

In the “Miscellaneous” option, check “Use newlib-nano” and “Do not use syscalls”. (The code size can be optimized)

Figure 3-30. GNU RISC-V Cross C Linker -> Miscellaneous configuration



Configuration: GD32VF103xB [Active]

Generate map "${BuildArtifactFileBaseName

□ Cross reference (-Xlinker --cref)

□ Print link map (-Xlinker --print-map)

☑ Use newlib-nano (--specs=nano.specs)

□ Use float with nano printf (-u _printf_float)

□ Use float with nano scanf (-u _scanf_float)

☑ Do not use syscalls (--specs=nosys.specs)

□ Verbose (-v)

Other linker flags

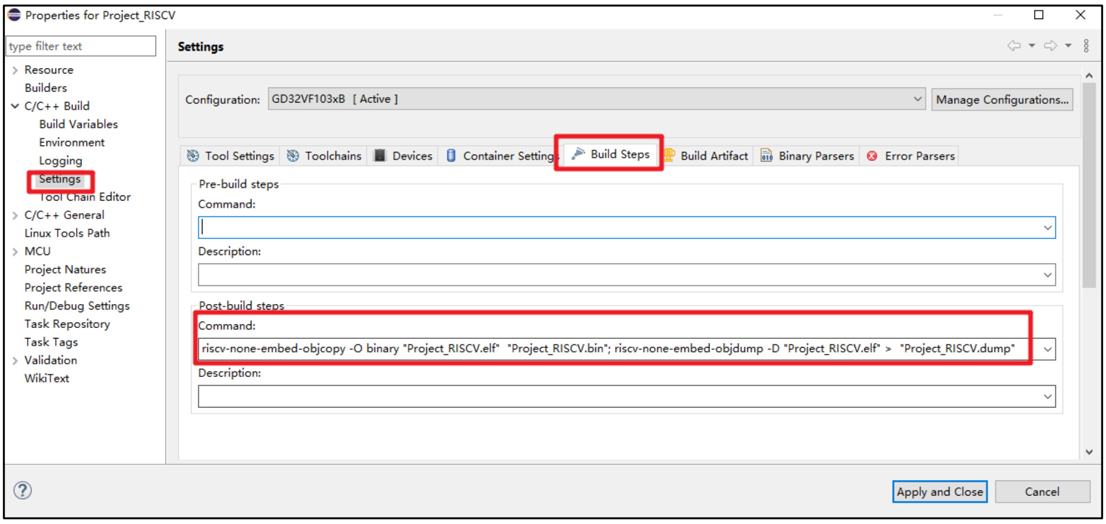
3.3.6. Build Steps configuration-generate bin file

In "C/C++ Build->Settings-> Build Steps", user can add commands to generate bin/hex files.

Add in this guide:

riscv-none-embed-objcopy -O binary "Project_RISCV.elf" "Project_RISCV.bin"; riscv-none-embed-objdump -D "Project_RISCV.elf" > "Project_RISCV.dump"

Figure 3-31. Build Steps configuration



Properties for Project_RISCV

Settings

Configuration: GD32VF103xB [Active]

Pre-build steps

Command:

Description:

Post-build steps

Command:

```
riscv-none-embed-objcopy -O binary "Project_RISCV.elf"  "Project_RISCV.bin"; riscv-none-embed-objdump -D "Project_RISCV.elf" >  "Project_RISCV.dump"
```

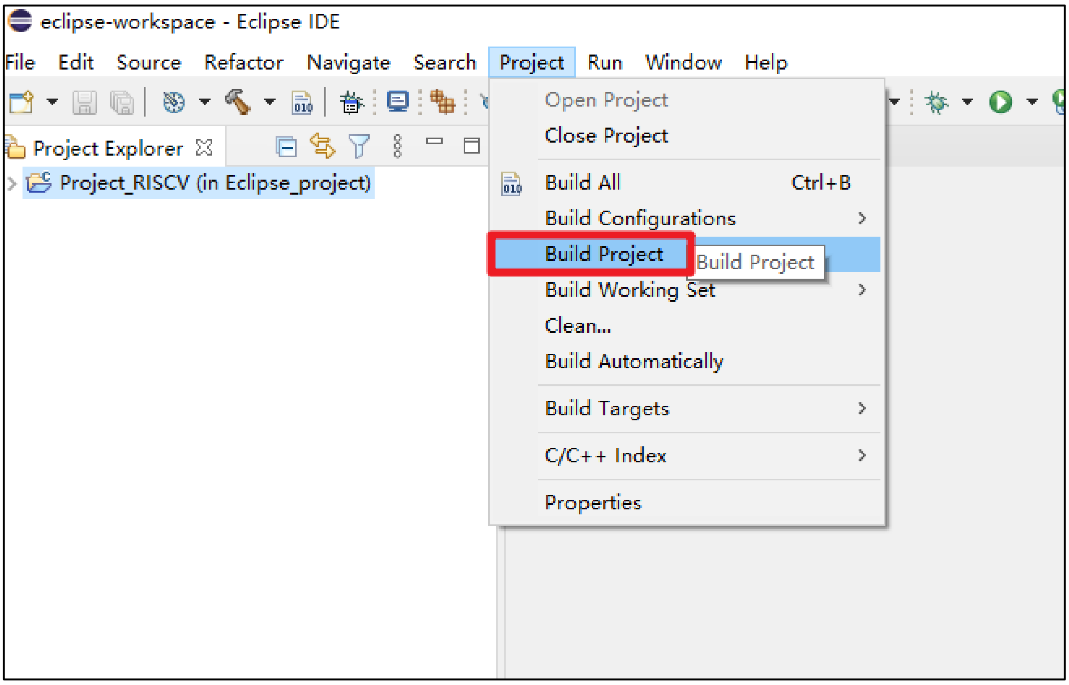
Description:

3.4. Build project

Select “Project->Build Project” to compile the current project.

Note: “Build Project” is to compile the current project, and “Build All” is to compile all the projects in the current workspace.

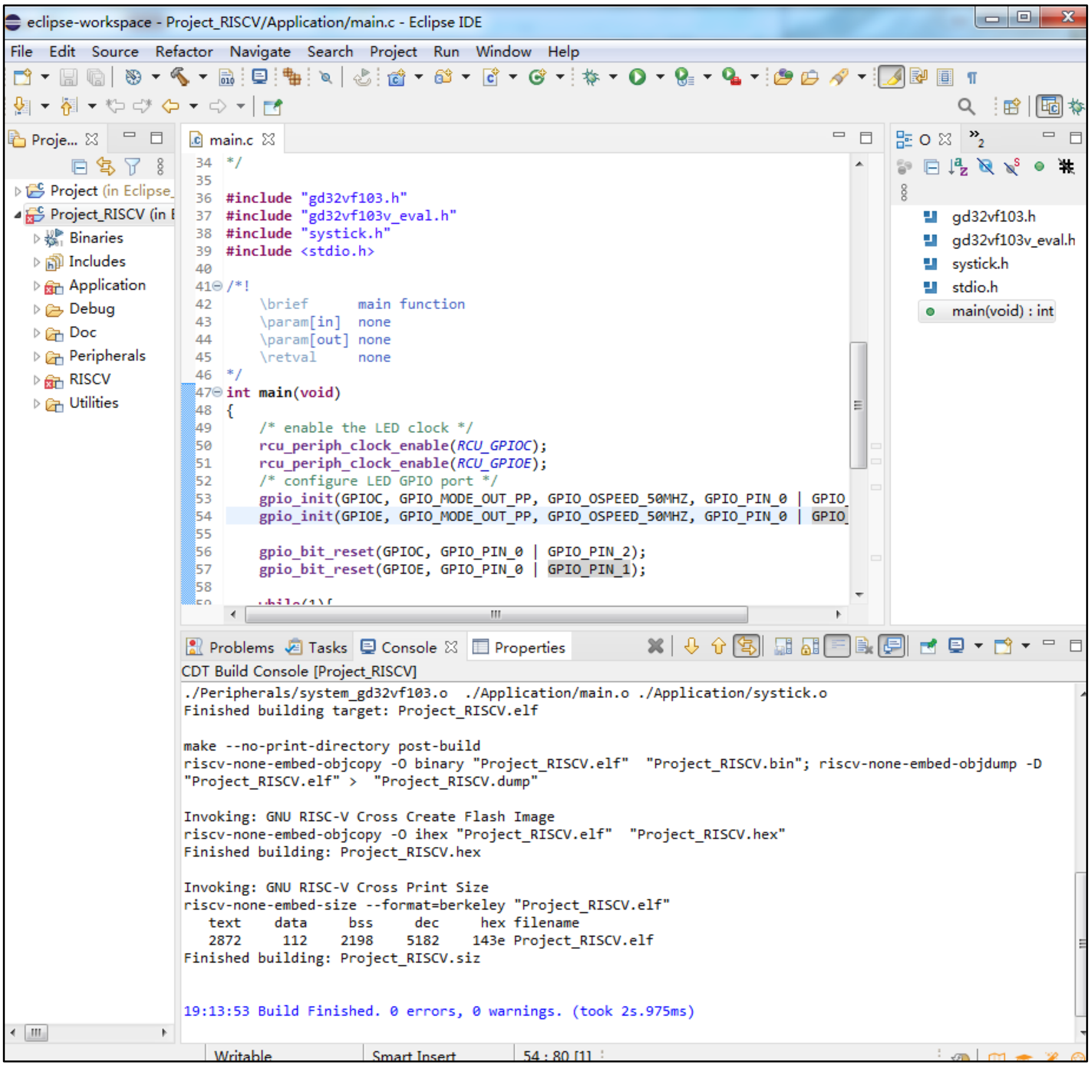
Figure 3-32. Build project



Note: User need to save the current project before compiling each time, otherwise the compiling is the last project. After modification, in order to ensure the correctness, please clean the project first and then build.

After compiling, it can be seen that the corresponding elf, hex and bin files have been generated.

Figure 3-33. Build RISC-V project completed



```
*/

#include "gd32vf103.h"
#include "gd32vf103v_eval.h"
#include "systick.h"
#include <stdio.h>

/*!
    \brief      main function
    \param[in]  none
    \param[out] none
    \retval     none
*/
int main(void)
{
    /* enable the LED clock */
    rcu_periph_clock_enable(RCU_GPIOC);
    rcu_periph_clock_enable(RCU_GPIOE);
    /* configure LED GPIO port */
    gpio_init(GPIOC, GPIO_MODE_OUT_PP, GPIO_OSPEED_50MHZ, GPIO_PIN_0 | GPIO_
    gpio_init(GPIOE, GPIO_MODE_OUT_PP, GPIO_OSPEED_50MHZ, GPIO_PIN_0 | GPIO_

    gpio_bit_reset(GPIOC, GPIO_PIN_0 | GPIO_PIN_2);
    gpio_bit_reset(GPIOE, GPIO_PIN_0 | GPIO_PIN_1);

```

CDT Build Console [Project_RISCV]

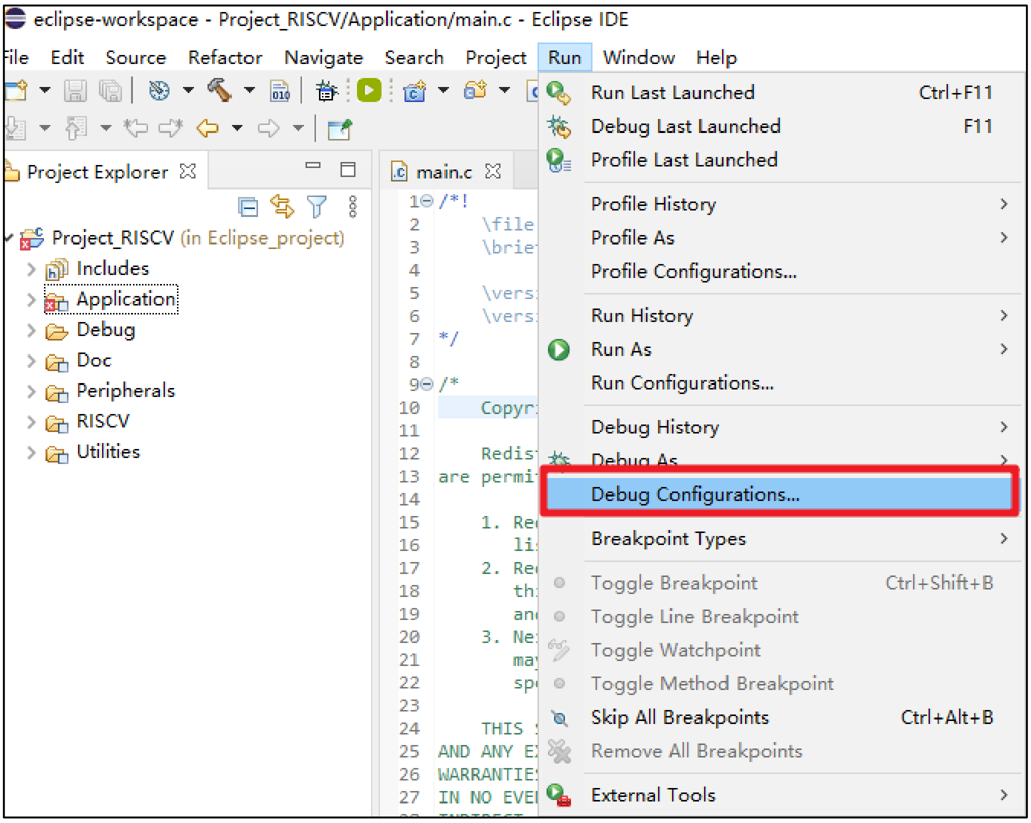
```
./Peripherals/system_gd32vf103.o  ./Application/main.o ./Application/systick.o
Finished building target: Project_RISCV.elf

make --no-print-directory post-build
riscv-none-embed-objcopy -O binary "Project_RISCV.elf"  "Project_RISCV.bin"; riscv-none-embed-objdump -D
"Project_RISCV.elf" >  "Project_RISCV.dump"

Invoking: GNU RISC-V Cross Create Flash Image
riscv-none-embed-objcopy -O ihex "Project_RISCV.elf"  "Project_RISCV.hex"
Finished building: Project_RISCV.hex

Invoking: GNU RISC-V Cross Print Size
riscv-none-embed-size --format=berkeley "Project_RISCV.elf"
   text    data     bss     dec     hex filename
   2872     112    2198    5182    143e Project_RISCV.elf
Finished building: Project_RISCV.siz


19:13:53 Build Finished. 0 errors, 0 warnings. (took 2s.975ms)
```

3.5. Use J-Link to download and debug the project

3.5.1. Debug configuration interface

In the menu bar, click “Run->Debug Configurations” to enter the Debug configuration interface.

Figure 3-34. Enter Debug Configuratios interface

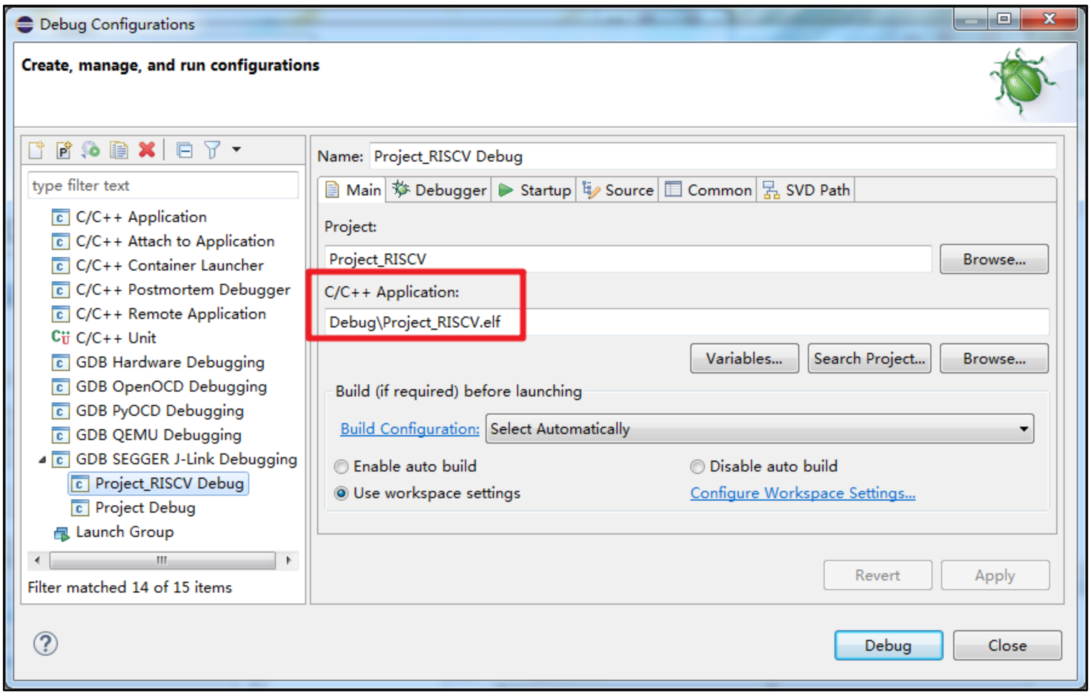


Use J-Link GDBServerCL as the GDB Server, and use the GDB tool in the GCC tool chain as the GDB Client.

Double-click GDB SEGGER J-Link Debugging to create a new set of J-Link configuration options.

3.5.2. Main tab

Figure 3-35. GDB SEGGER J-Link Debugging-Main tab



Name: Project_RISCV Debug

Project: Project_RISCV

C/C++ Application: Debug\Project_RISCV.elf

Build Configuration: Select Automatically

● Use workspace settings

In the “Main” tab, select the current project, usually the elf file under the current project will be added automatically. If not, user can click “Browse” to add the elf file manually.

Note: If user have compiled multiple models before, user need to select the corresponding executable elf file. For convenience, user can also create a new set of “Debug configuration” for each chip model.

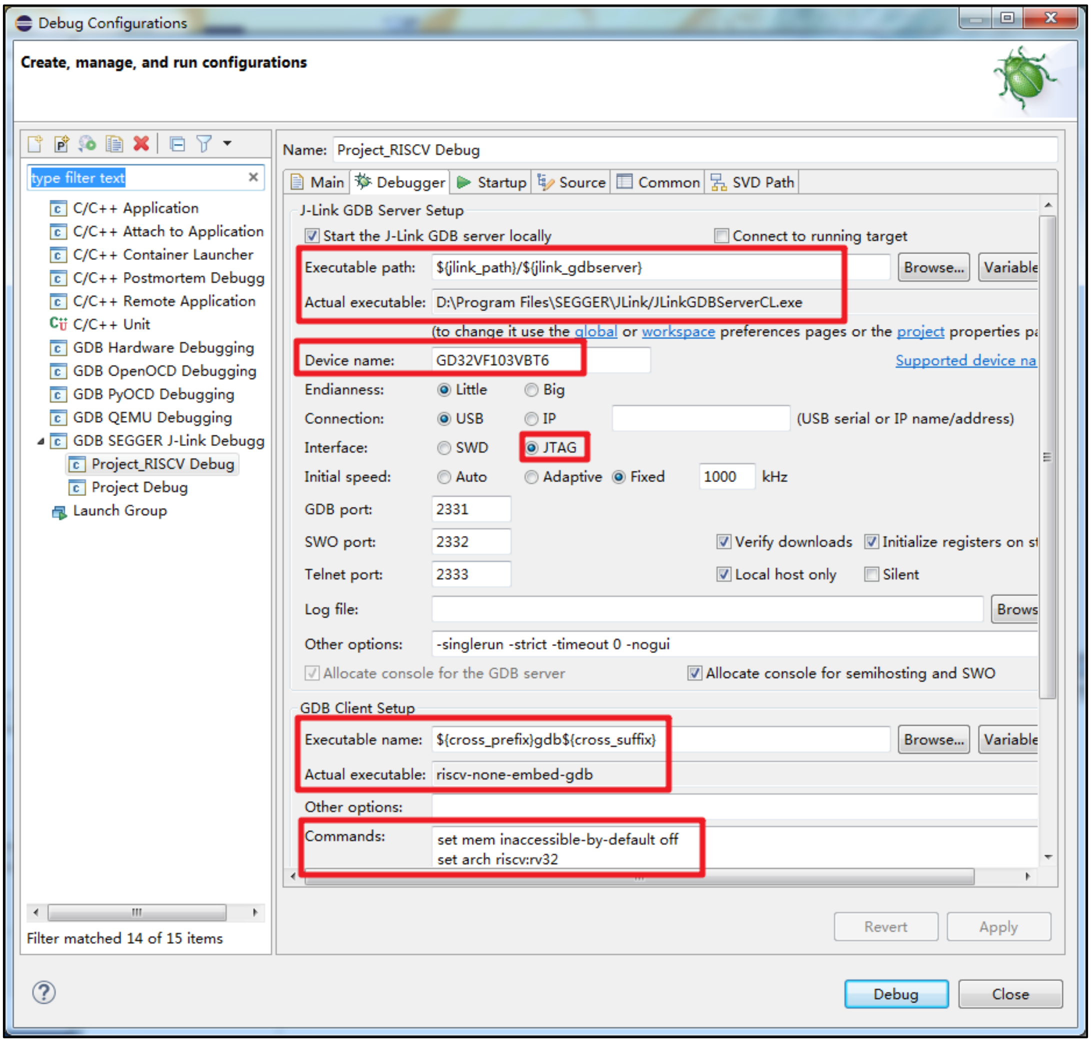
3.5.3. Debugger tab

In the “Debugger” tab, fill in the device name of the target chip model, which is GD32VF103VBT6 in this guide.

If the J-Link path has been configured correctly when setting up the Eclipse environment, it will be recognized automatically here. If user have not configured it correctly before, user can also select the absolute path of J-Link GDBServerCL in the “Executable path” column.

Note: The chip model filled in “Device name” column must be supported by the J-Link driver which is selected here. If it is not supported, please upgrade the J-Link driver version.

Figure 3-36. GDB SEGGER J-Link Debugging-Debugger tab



Debug Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
- C/C++ Attach to Application
- C/C++ Container Launcher
- C/C++ Postmortem Debugg
- C/C++ Remote Application
- C/C++ Unit
- GDB Hardware Debugging
- GDB OpenOCD Debugging
- GDB PyOCD Debugging
- GDB QEMU Debugging
- GDB SEGGER J-Link Debugg
  - Project_RISCV Debug
  - Project Debug
- Launch Group

Filter matched 14 of 15 items

Name: Project_RISCV Debug

Main | Debugger | Startup | Source | Common | SVD Path

J-Link GDB Server Setup

☑ Start the J-Link GDB server locally ☐ Connect to running target

Executable path: ${jlink_path}/${jlink_gdbserver} Browse... Variable

Actual executable: D:\Program Files\SEGGER\JLink/JLinkGDBServerCL.exe

(to change it use the global or workspace preferences pages or the project properties pa

Device name: GD32VF103VBT6 Supported device na

Endianness: ● Little ○ Big

Connection: ● USB ○ IP (USB serial or IP name/address)

Interface: ○ SWD ● JTAG

Initial speed: ○ Auto ○ Adaptive ● Fixed 1000 kHz

GDB port: 2331

SWO port: 2332 ☑ Verify downloads ☑ Initialize registers on st

Telnet port: 2333 ☑ Local host only ☐ Silent

Log file: Brows

Other options: -singlerun -strict -timeout 0 -nogui

☑ Allocate console for the GDB server ☑ Allocate console for semihosting and SWO

GDB Client Setup

Executable name: ${cross_prefix}gdb${cross_suffix} Browse... Variable

Actual executable: riscv-none-embed-gdb

Other options:

Commands:
```
set mem inaccessible-by-default off
set arch riscv:rv32
```

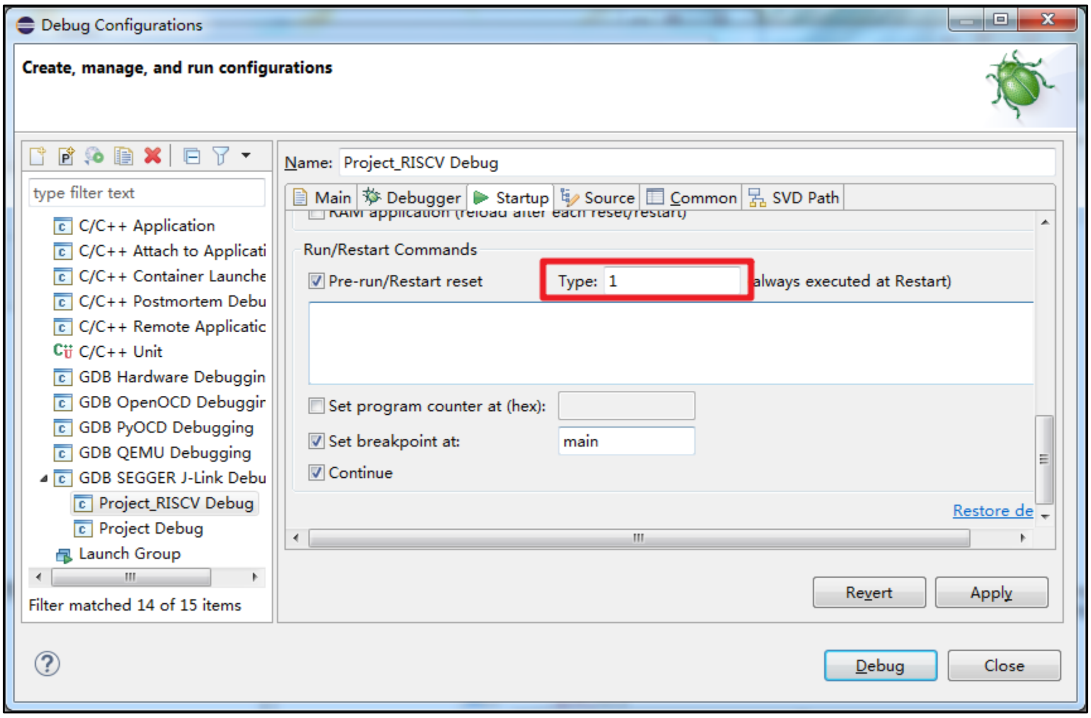
Revert Apply

Debug Close

3.5.4. Startup tab

In the “Startup” tab, configure type:

Figure 3-37. GDB SEGGER J-Link Debugging-Startup tab



Name: Project_RISCV Debug

RAM application (reload after each reset/restart)

Run/Restart Commands

☑ Pre-run/Restart reset Type: 1 (always executed at Restart)

□ Set program counter at (hex):

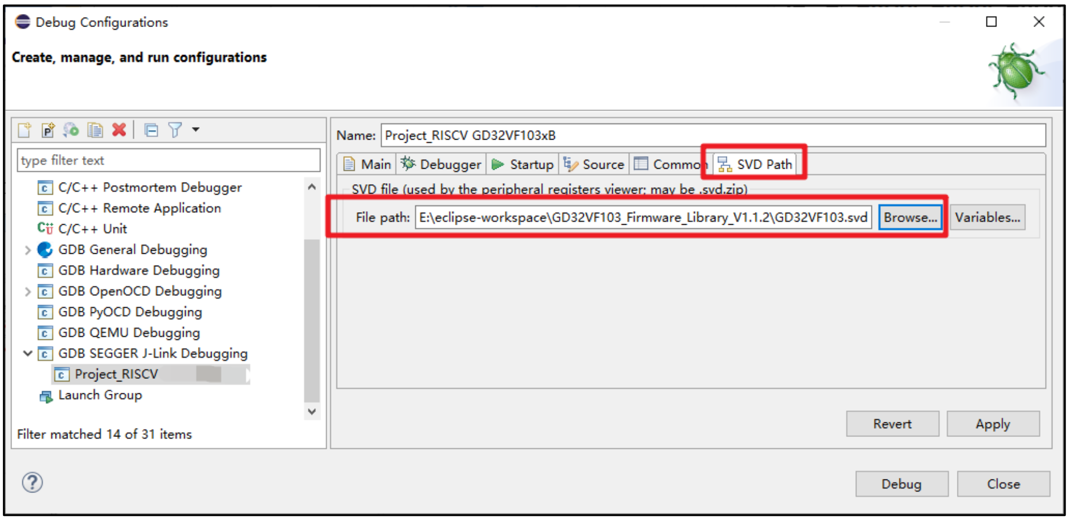
☑ Set breakpoint at: main

☑ Continue

3.5.5. SVD Path tab

In the “SVD Path” tab, select the SVD file required by the target chip.

Figure 3-38. GDB SEGGER J-Link Debugging-SVD Path tab



Name: Project_RISCV GD32VF103xB

SVD file (used by the peripheral registers viewer; may be .svd.zip)

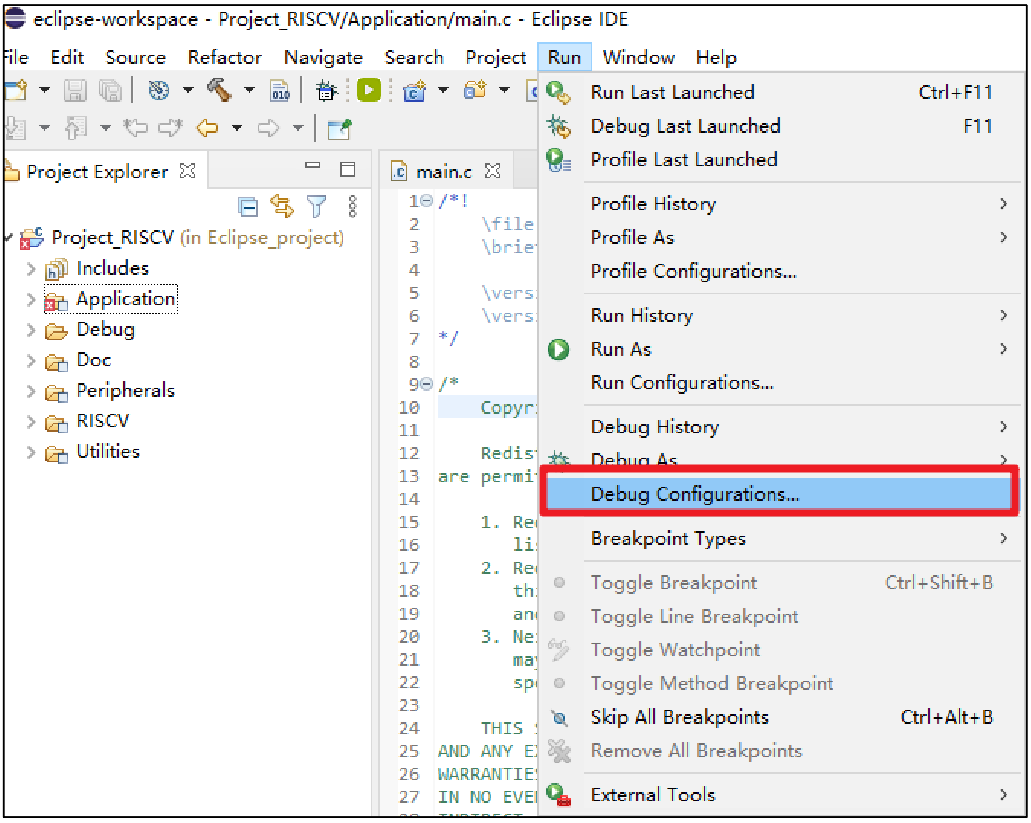
File path: E:\eclipse-workspace\GD32VF103_Firmware_Library_V1.1.2\GD32VF103.svd

3.6. Use GD-Link to download and debug the project

3.6.1. Debug configuration interface

In the menu bar, click “Run->Debug Configurations” to enter the Debug configuration interface.

Figure 3-39. Enter Debug Configuratios interface

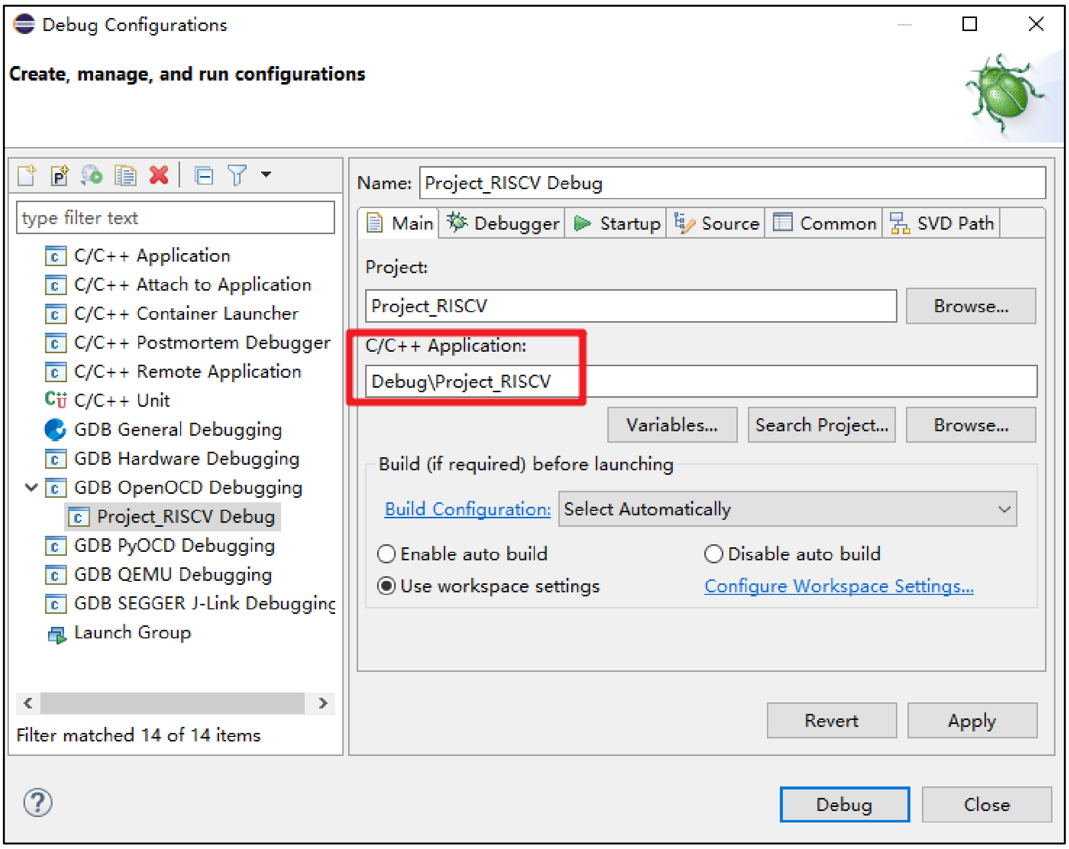


Use OpenOCD as the GDB Server, and use the GDB tool in the GCC tool chain as the GDB Client.

Double-click GDB OpenOCD Debugging to create a new set of OpenOCD configuration options.

3.6.2. Main tab

Figure 3-40. GDB OpenOCD Debugging-Main tab



Debug Configurations

Create, manage, and run configurations

Name: Project_RISCV Debug

Project: Project_RISCV

C/C++ Application: Debug\Project_RISCV

Build (if required) before launching

Build Configuration: Select Automatically

○ Enable auto build　○ Disable auto build
● Use workspace settings　Configure Workspace Settings...

Filter matched 14 of 14 items

In the “Main” tab, select the current project, usually the elf file under the current project will be added automatically. If not, user can click “Browse” to add the elf file manually.

Note: If user have compiled multiple models before, user need to select the corresponding executable elf file. For convenience, user can also create a new set of “Debug configuration” for each chip model.

3.6.3. Debugger tab

If the OpenOCD path has been configured correctly when setting up the Eclipse environment, it will be recognized automatically here. If user have not configured it correctly before, user can also select the absolute path of OpenOCD in the “Executable path” column.

In the “Config options” column, fill in the cfg file used. In this guide:

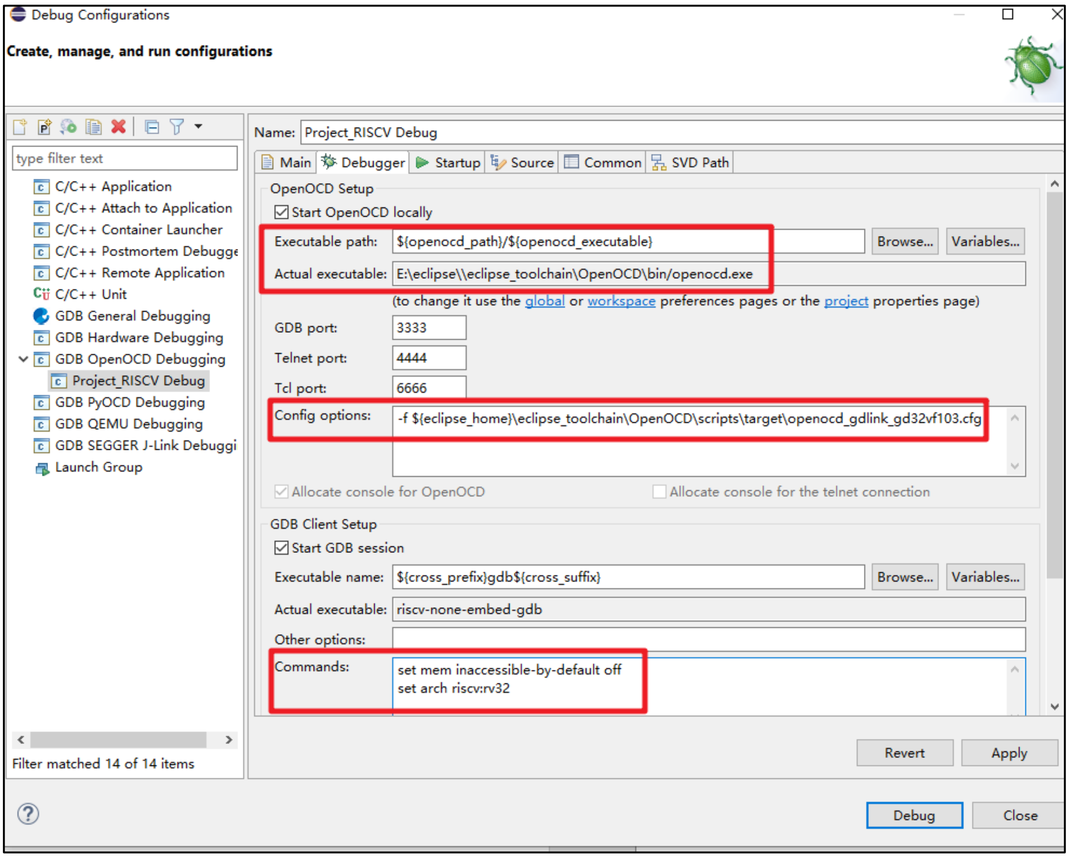
-f ${eclipse_home}\eclipse_toolchain\OpenOCD\scripts\target\openocd_gdlink_gd32vf103.cfg

The cfg file of OpenOCD provides information such as debugger, debugging protocol, target chip identification and target chip programming algorithm selection.

In the “Commands” column, fill:

set arch riscv:rv32

Figure 3-41. GDB OpenOCD Debugging-Debugger tab



Name: Project_RISCV Debug

OpenOCD Setup

☑ Start OpenOCD locally

Executable path: ${openocd_path}/${openocd_executable}

Actual executable: E:\eclipse\\eclipse_toolchain\OpenOCD\bin/openocd.exe

(to change it use the global or workspace preferences pages or the project properties page)

GDB port: 3333

Telnet port: 4444

Tcl port: 6666

Config options: -f ${eclipse_home}\eclipse_toolchain\OpenOCD\scripts\target\openocd_gdlink_gd32vf103.cfg

☑ Allocate console for OpenOCD □ Allocate console for the telnet connection

GDB Client Setup

☑ Start GDB session

Executable name: ${cross_prefix}gdb${cross_suffix}

Actual executable: riscv-none-embed-gdb

Other options:

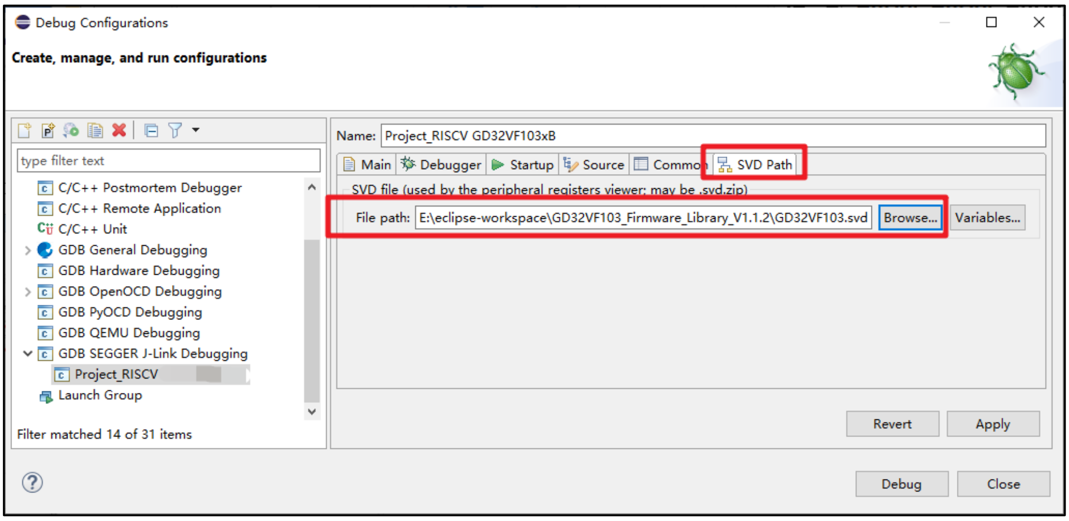
Commands:

```
set mem inaccessible-by-default off
set arch riscv:rv32
```

3.6.4. SVD Path tab

In the “SVD Path” tab, select the SVD file required by the target chip.

Figure 3-42. GDB OpenOCD Debugging-SVD Path tab



Name: Project_RISCV GD32VF103xB

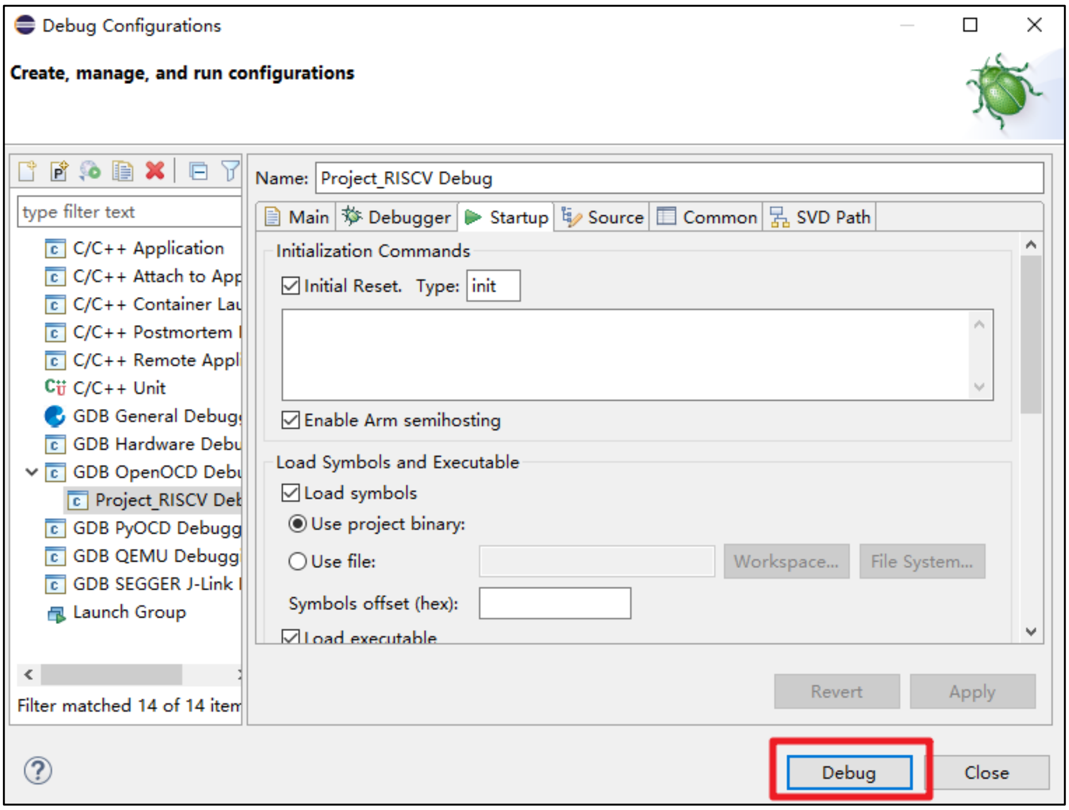
SVD file (used by the peripheral registers viewer; may be .svd.zip)

File path: E:\eclipse-workspace\GD32VF103_Firmware_Library_V1.1.2\GD32VF103.svd

3.7. Debug interface

After the debug configurations is completed, click “Debug” to enter the Debug perspective.

Figure 3-43. Enter Debug perspective -1



Debug Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
- C/C++ Attach to App
- C/C++ Container Lau
- C/C++ Postmortem I
- C/C++ Remote Appl
- C/C++ Unit
- GDB General Debug
- GDB Hardware Debu
- GDB OpenOCD Debu
  - Project_RISCV Deb
- GDB PyOCD Debugg
- GDB QEMU Debugg
- GDB SEGGER J-Link I
- Launch Group

Filter matched 14 of 14 item

Name: Project_RISCV Debug

Main | Debugger | Startup | Source | Common | SVD Path

Initialization Commands

☑ Initial Reset. Type: init

☑ Enable Arm semihosting

Load Symbols and Executable

☑ Load symbols

● Use project binary:

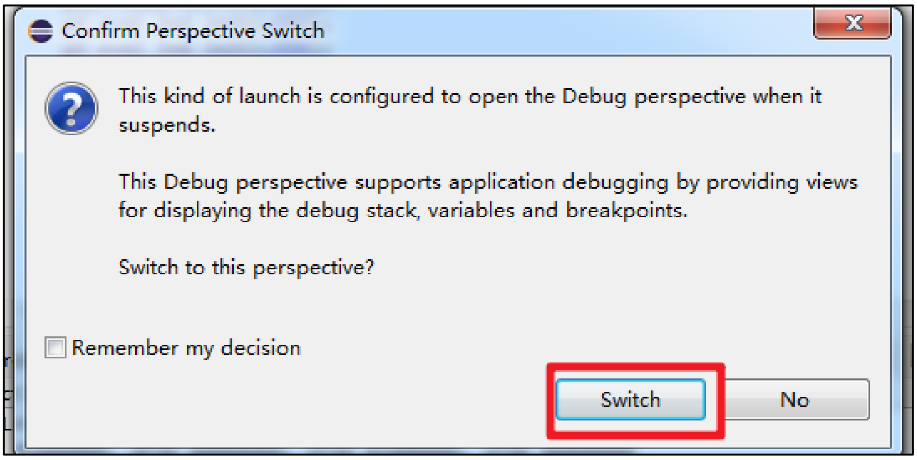
○ Use file: Workspace... File System...

Symbols offset (hex):

☑ Load executable

Switch to Debug perspective.

Figure 3-44. Enter Debug perspective -2



Confirm Perspective Switch

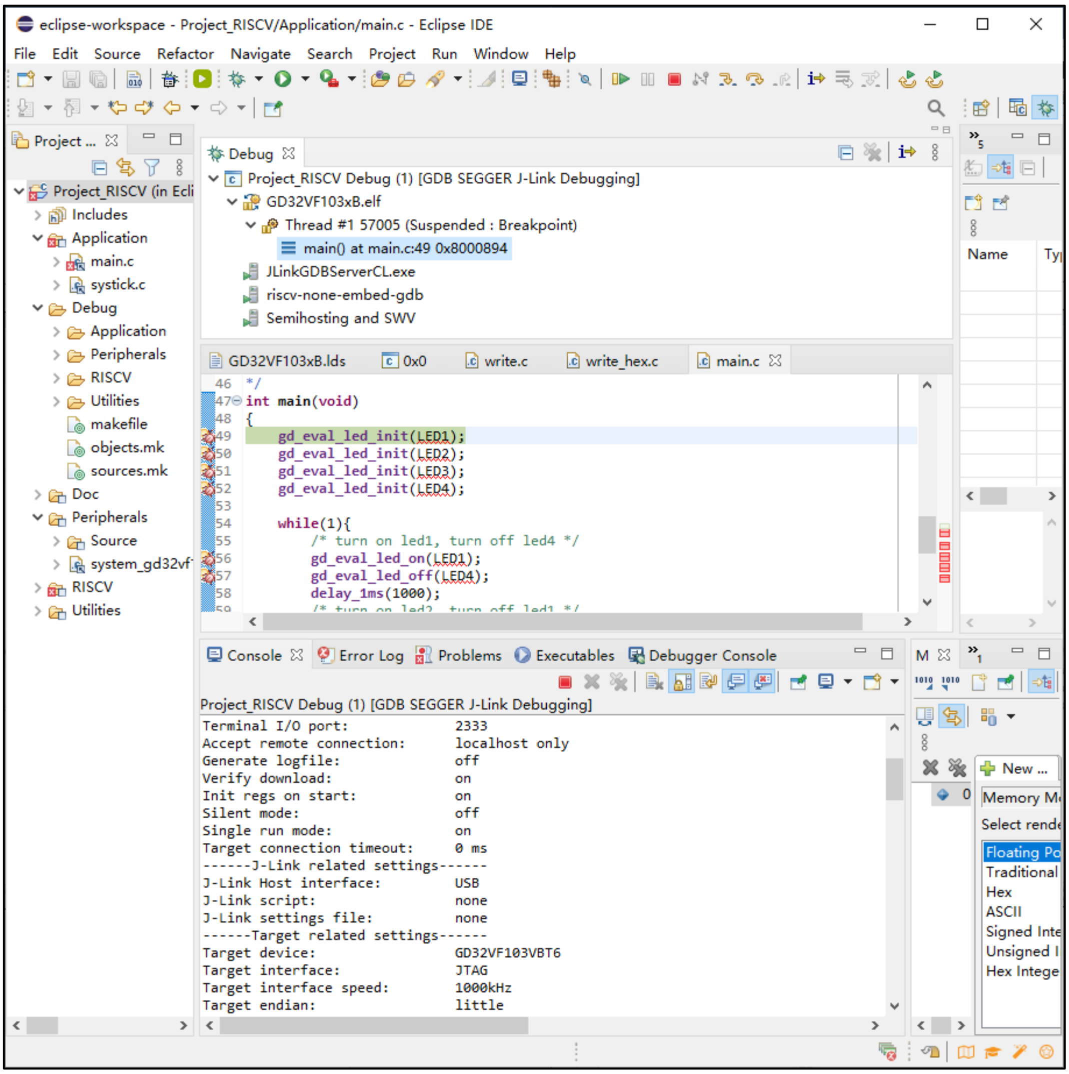
This kind of launch is configured to open the Debug perspective when it suspends.

This Debug perspective supports application debugging by providing views for displaying the debug stack, variables and breakpoints.

Switch to this perspective?

□ Remember my decision

Figure 3-45. Debug perspective



eclipse-workspace - Project_RISCV/Application/main.c - Eclipse IDE

Project_RISCV Debug (1) [GDB SEGGER J-Link Debugging]
GD32VF103xB.elf
Thread #1 57005 (Suspended : Breakpoint)
main() at main.c:49 0x8000894
JLinkGDBServerCL.exe
riscv-none-embed-gdb
Semihosting and SWV

```
*/
int main(void)
{
    gd_eval_led_init(LED1);
    gd_eval_led_init(LED2);
    gd_eval_led_init(LED3);
    gd_eval_led_init(LED4);

    while(1){
        /* turn on led1, turn off led4 */
        gd_eval_led_on(LED1);
        gd_eval_led_off(LED4);
        delay_1ms(1000);
```

Project_RISCV Debug (1) [GDB SEGGER J-Link Debugging]

```
Terminal I/O port:              2333
Accept remote connection:       localhost only
Generate logfile:               off
Verify download:                on
Init regs on start:             on
Silent mode:                    off
Single run mode:                on
Target connection timeout:      0 ms
------J-Link related settings------
J-Link Host interface:          USB
J-Link script:                  none
J-Link settings file:           none
------Target related settings------
Target device:                  GD32VF103VBT6
Target interface:               JTAG
Target interface speed:         1000kHz
Target endian:                  little
```

3.7.1. Toolbar introduction

： resume

： suspend

： terminate

： step into

： step over

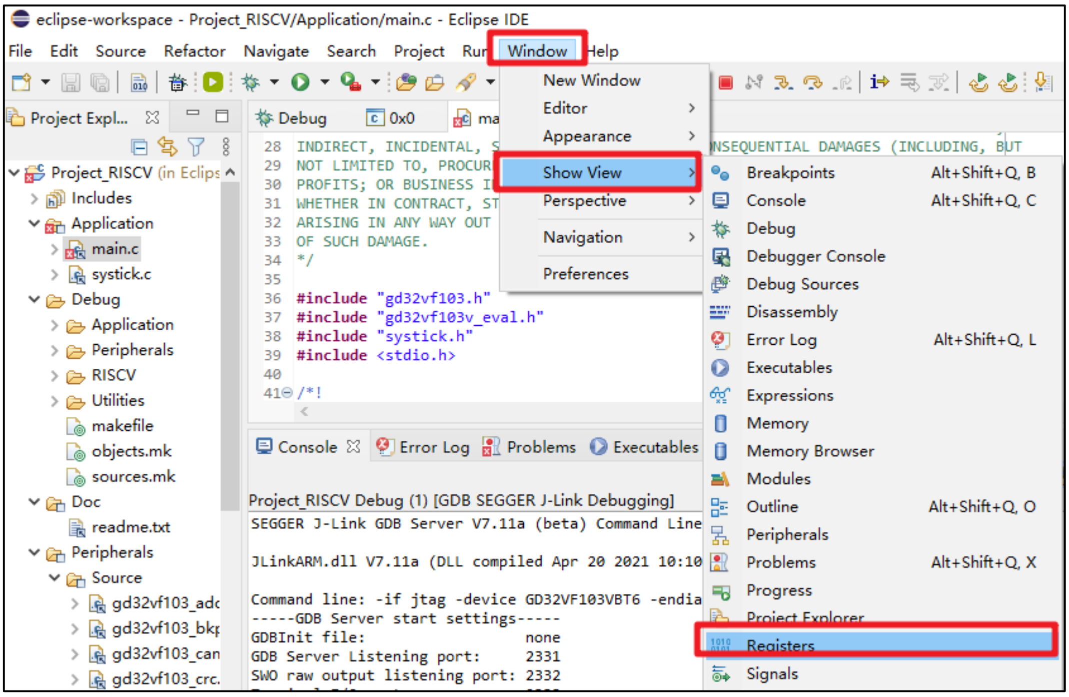
： step out

： reset

3.7.2. Registers view

In the menu bar, select “Window->Show view->Registers” option, open it to view the value of general-purpose registers.

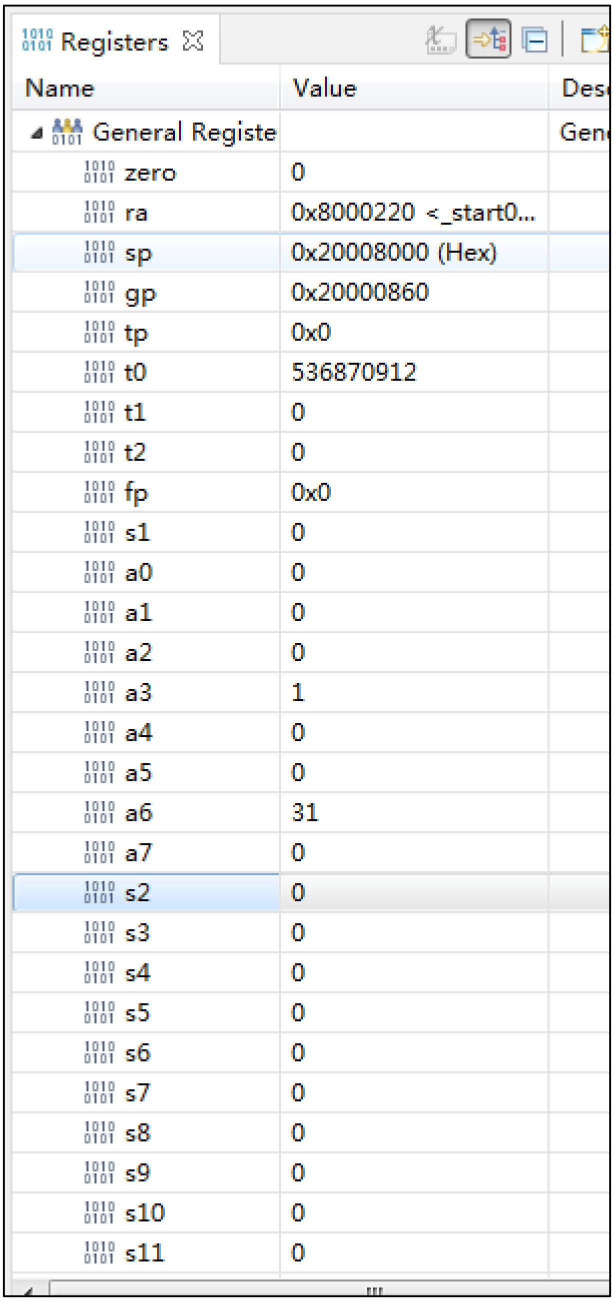
Figure 3-46. Open Registers view



```
INDIRECT, INCIDENTAL, S                    NSEQUENTIAL DAMAGES (INCLUDING, BUT
NOT LIMITED TO, PROCUR
PROFITS; OR BUSINESS I
WHETHER IN CONTRACT, ST
ARISING IN ANY WAY OUT
OF SUCH DAMAGE.
*/

#include "gd32vf103.h"
#include "gd32vf103v_eval.h"
#include "systick.h"
#include <stdio.h>

/*!
```

```
Project_RISCV Debug (1) [GDB SEGGER J-Link Debugging]
SEGGER J-Link GDB Server V7.11a (beta) Command Line

JLinkARM.dll V7.11a (DLL compiled Apr 20 2021 10:10

Command line: -if jtag -device GD32VF103VBT6 -endia
-----GDB Server start settings-----
GDBInit file:                  none
GDB Server Listening port:     2331
SWO raw output listening port: 2332
```

Figure 3-47. Registers view

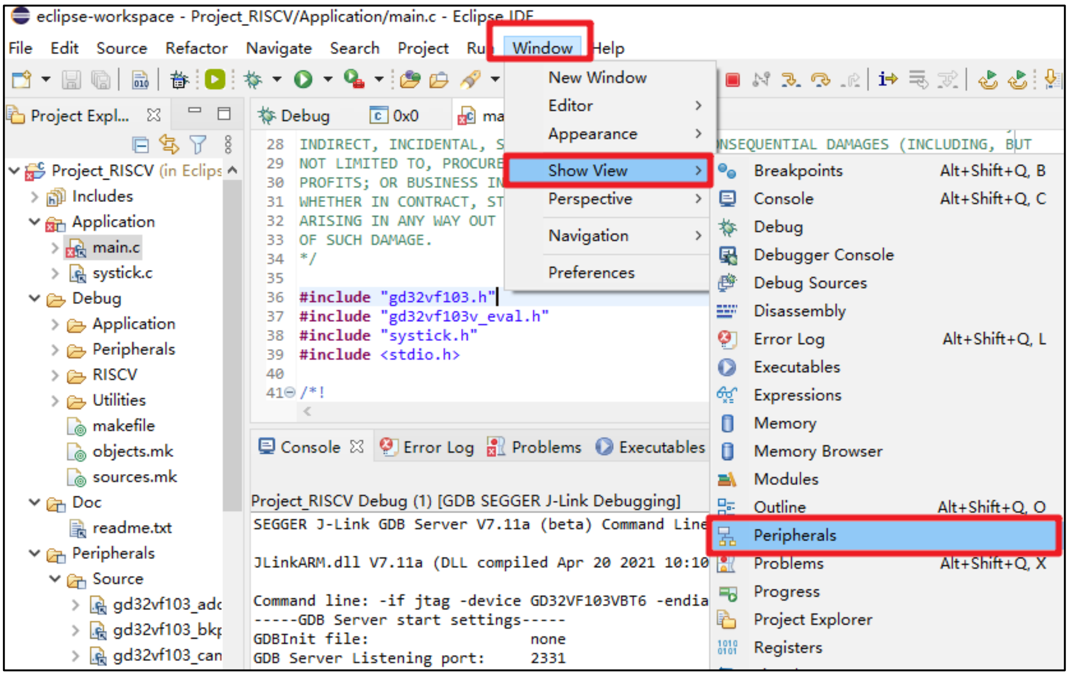


Name	Value	Des
General Registe		Gen
zero	0	
ra	0x8000220 <_start0...	
sp	0x20008000 (Hex)	
gp	0x20000860	
tp	0x0	
t0	536870912	
t1	0	
t2	0	
fp	0x0	
s1	0	
a0	0	
a1	0	
a2	0	
a3	1	
a4	0	
a5	0	
a6	31	
a7	0	
s2	0	
s3	0	
s4	0	
s5	0	
s6	0	
s7	0	
s8	0	
s9	0	
s10	0	
s11	0	

3.7.3. Peripherals view

In the menu bar, select “Window->Show view->Peripherals” option, open to view the value of the peripheral registers.

Figure 3-48. Open Peripherals view

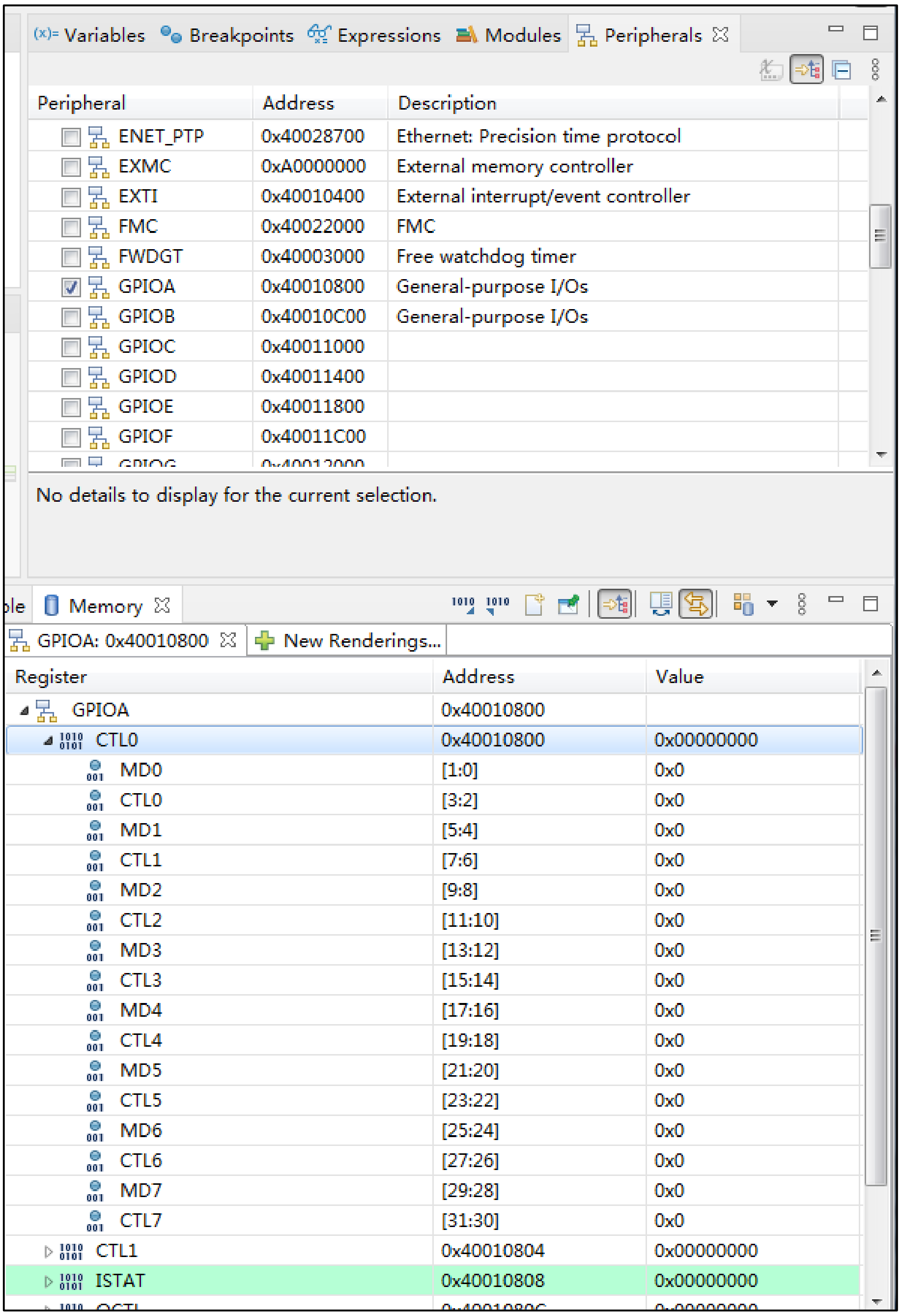


```
INDIRECT, INCIDENTAL, S
NOT LIMITED TO, PROCURE
PROFITS; OR BUSINESS IN
WHETHER IN CONTRACT, ST
ARISING IN ANY WAY OUT
OF SUCH DAMAGE.
*/

#include "gd32vf103.h"
#include "gd32vf103v_eval.h"
#include "systick.h"
#include <stdio.h>

/*!
```

```
NSEQUENTIAL DAMAGES (INCLUDING, BUT
```

```
Project_RISCV Debug (1) [GDB SEGGER J-Link Debugging]
SEGGER J-Link GDB Server V7.11a (beta) Command Line

JLinkARM.dll V7.11a (DLL compiled Apr 20 2021 10:10

Command line: -if jtag -device GD32VF103VBT6 -endia
-----GDB Server start settings-----
GDBInit file:                  none
GDB Server Listening port:     2331
```

Figure 3-49. Peripherals view



Peripheral	Address	Description
□ ENET_PTP	0x40028700	Ethernet: Precision time protocol
□ EXMC	0xA0000000	External memory controller
□ EXTI	0x40010400	External interrupt/event controller
□ FMC	0x40022000	FMC
□ FWDGT	0x40003000	Free watchdog timer
☑ GPIOA	0x40010800	General-purpose I/Os
□ GPIOB	0x40010C00	General-purpose I/Os
□ GPIOC	0x40011000	
□ GPIOD	0x40011400	
□ GPIOE	0x40011800	
□ GPIOF	0x40011C00	
□ GPIOG	0x40012000	

No details to display for the current selection.

Register	Address	Value
GPIOA	0x40010800	
CTL0	0x40010800	0x00000000
MD0	[1:0]	0x0
CTL0	[3:2]	0x0
MD1	[5:4]	0x0
CTL1	[7:6]	0x0
MD2	[9:8]	0x0
CTL2	[11:10]	0x0
MD3	[13:12]	0x0
CTL3	[15:14]	0x0
MD4	[17:16]	0x0
CTL4	[19:18]	0x0
MD5	[21:20]	0x0
CTL5	[23:22]	0x0
MD6	[25:24]	0x0
CTL6	[27:26]	0x0
MD7	[29:28]	0x0
CTL7	[31:30]	0x0
CTL1	0x40010804	0x00000000
ISTAT	0x40010808	0x00000000
OCTL	0x4001080C	0x00000000

3.7.4. Memory view

In the menu bar, select “Window->Show view->Memory” option, and click the "+" sign above the “Memory” window to open the corresponding memory address.

Figure 3-50. Open Memory view

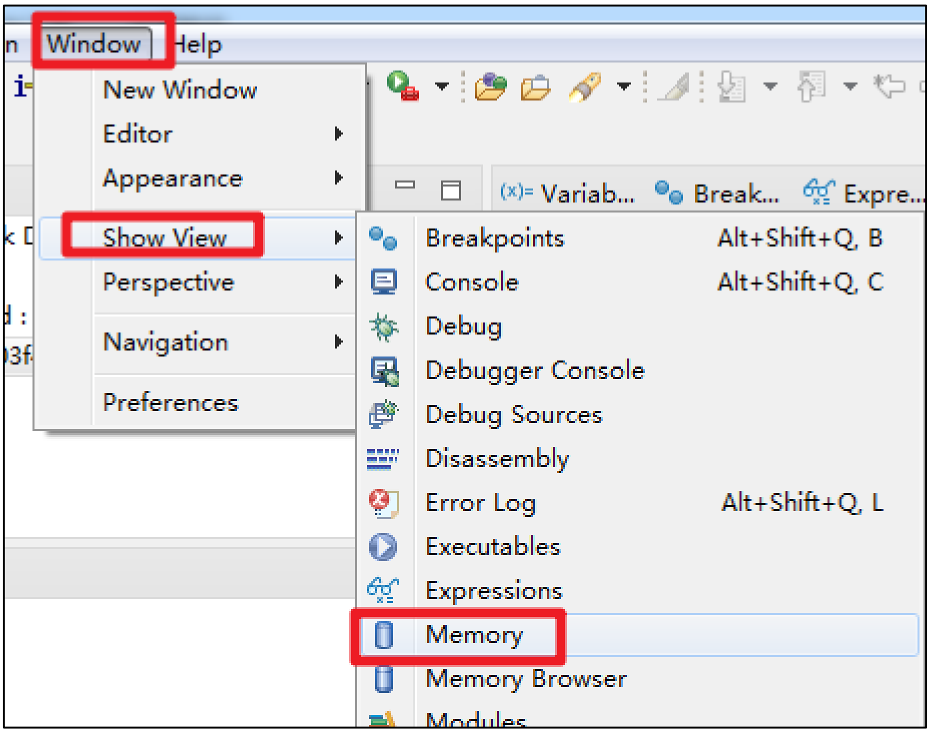
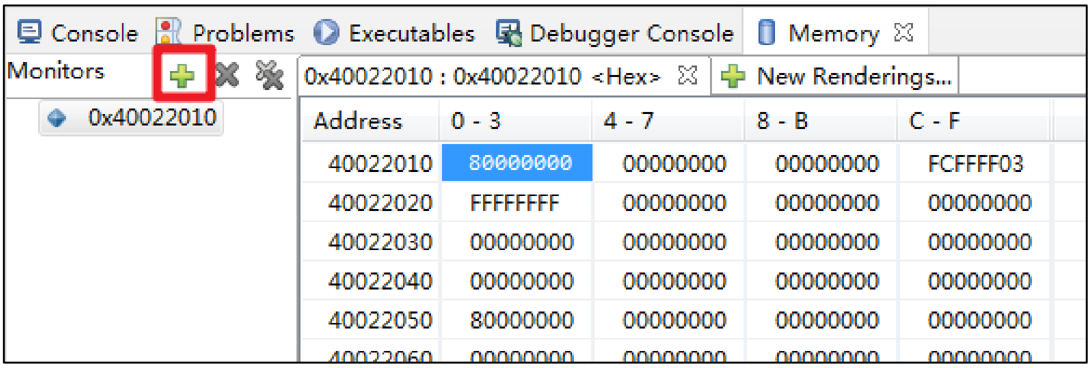


Figure 3-51. Memory view



Address	0 - 3	4 - 7	8 - B	C - F
40022010	80000000	00000000	00000000	FCFFFF03
40022020	FFFFFFFF	00000000	00000000	00000000
40022030	00000000	00000000	00000000	00000000
40022040	00000000	00000000	00000000	00000000
40022050	80000000	00000000	00000000	00000000
40022060	00000000	00000000	00000000	00000000

3.7.5. Expressions view

In the menu bar, select "Window->Show View->Expressions" and click the "+" sign in the “Expressions” window to add and view the value of the corresponding variable.

Figure 3-52. Open Expressions view

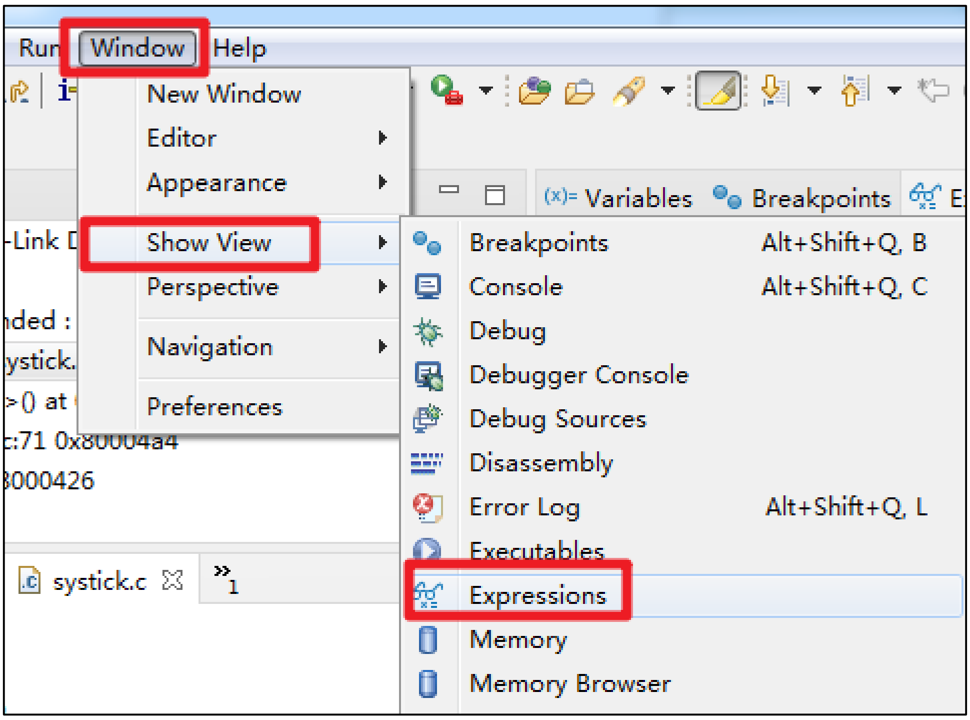
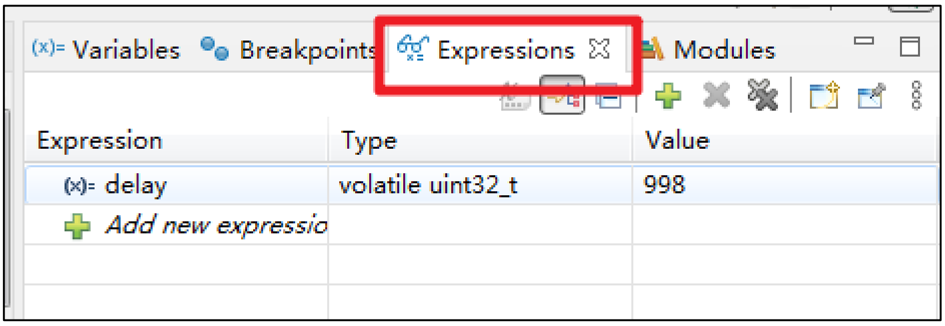


Figure 3-53. Expressions view



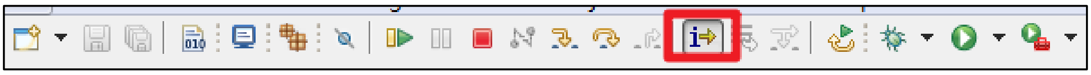
Expression	Type	Value
(x)= delay	volatile uint32_t	998
Add new expressio		

Note: Ecplise can only view the value of the variable when the code is not running. It is temporarily unable to update the value of the variable in real time.

3.7.6. Disassembly view

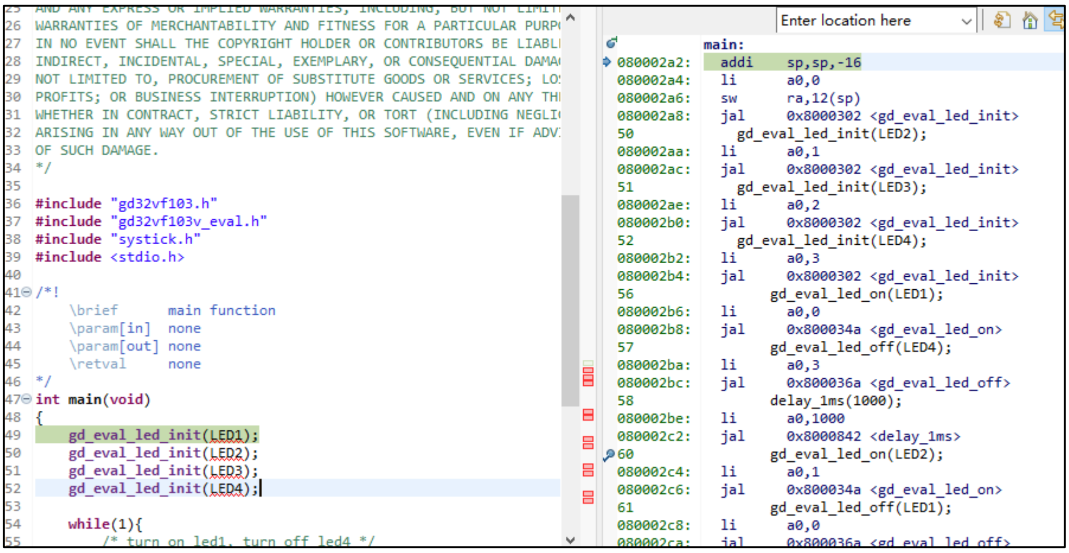
Select the “Instruction Stepping Mode button” in the debug toolbar to open the disassembly window.

Figure 3-54. Open Disassembly view



In the disassembly window, breakpoints can be enabled, assembly instructions can be executed in single step, etc.

Figure 3-55. Disassembly view



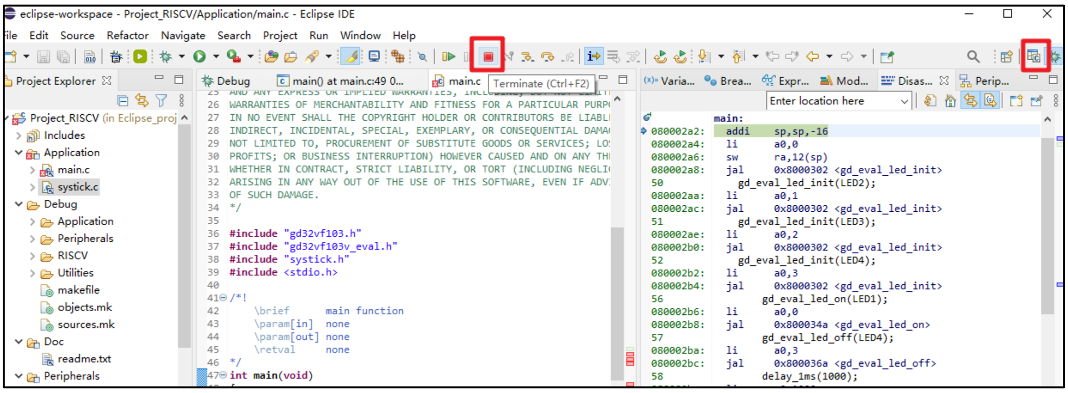
```
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMIT
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURP(
IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLI
INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMA(
NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LO!
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THI
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLI(
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADV:
OF SUCH DAMAGE.
*/

#include "gd32vf103.h"
#include "gd32vf103v_eval.h"
#include "systick.h"
#include <stdio.h>

/*!
    \brief      main function
    \param[in]  none
    \param[out] none
    \retval     none
*/
int main(void)
{
    gd_eval_led_init(LED1);
    gd_eval_led_init(LED2);
    gd_eval_led_init(LED3);
    gd_eval_led_init(LED4);

    while(1){
        /* turn on led1, turn off led4 */
```

```
Enter location here
          main:
080002a2:   addi    sp,sp,-16
080002a4:   li      a0,0
080002a6:   sw      ra,12(sp)
080002a8:   jal     0x8000302 <gd_eval_led_init>
50            gd_eval_led_init(LED2);
080002aa:   li      a0,1
080002ac:   jal     0x8000302 <gd_eval_led_init>
51            gd_eval_led_init(LED3);
080002ae:   li      a0,2
080002b0:   jal     0x8000302 <gd_eval_led_init>
52            gd_eval_led_init(LED4);
080002b2:   li      a0,3
080002b4:   jal     0x8000302 <gd_eval_led_init>
56                gd_eval_led_on(LED1);
080002b6:   li      a0,0
080002b8:   jal     0x800034a <gd_eval_led_on>
57                gd_eval_led_off(LED4);
080002ba:   li      a0,3
080002bc:   jal     0x800036a <gd_eval_led_off>
58                delay_1ms(1000);
080002be:   li      a0,1000
080002c2:   jal     0x8000842 <delay_1ms>
60                gd_eval_led_on(LED2);
080002c4:   li      a0,1
080002c6:   jal     0x800034a <gd_eval_led_on>
61                gd_eval_led_off(LED1);
080002c8:   li      a0,0
080002ca:   jal     0x800036a <gd_eval_led_off>
```

3.7.7. Exit the Debug perspective

Click the "Stop debugging" button, and then click "C/C++" to enter the project perspective.

Figure 3-56. Exit the Debug perspective



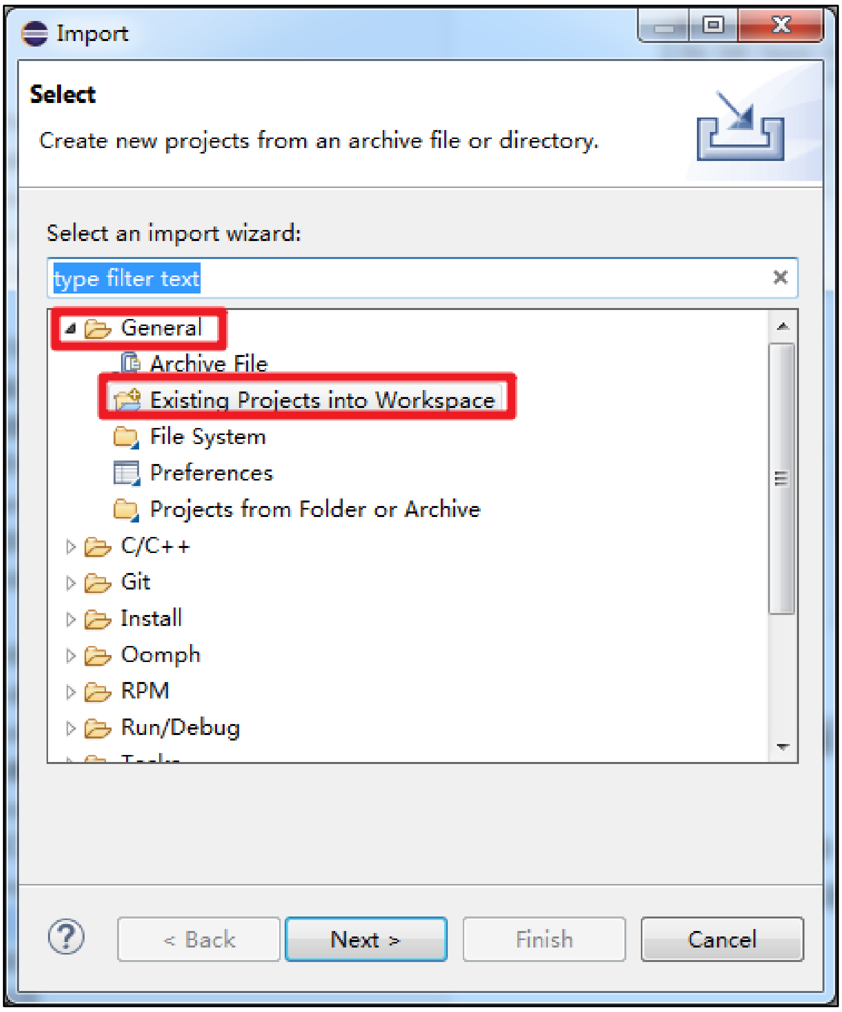
```
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMIT
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURP(
IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLI
INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMA(
NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LO!
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THI
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLI(
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADV:
OF SUCH DAMAGE.
*/

#include "gd32vf103.h"
#include "gd32vf103v_eval.h"
#include "systick.h"
#include <stdio.h>

/*!
    \brief      main function
    \param[in]  none
    \param[out] none
    \retval     none
*/
int main(void)
```

```
          main:
080002a2:   addi    sp,sp,-16
080002a4:   li      a0,0
080002a6:   sw      ra,12(sp)
080002a8:   jal     0x8000302 <gd_eval_led_init>
50            gd_eval_led_init(LED2);
080002aa:   li      a0,1
080002ac:   jal     0x8000302 <gd_eval_led_init>
51            gd_eval_led_init(LED3);
080002ae:   li      a0,2
080002b0:   jal     0x8000302 <gd_eval_led_init>
52            gd_eval_led_init(LED4);
080002b2:   li      a0,3
080002b4:   jal     0x8000302 <gd_eval_led_init>
56                gd_eval_led_on(LED1);
080002b6:   li      a0,0
080002b8:   jal     0x800034a <gd_eval_led_on>
57                gd_eval_led_off(LED4);
080002ba:   li      a0,3
080002bc:   jal     0x800036a <gd_eval_led_off>
58                delay_1ms(1000);
```

4. Import an existing project

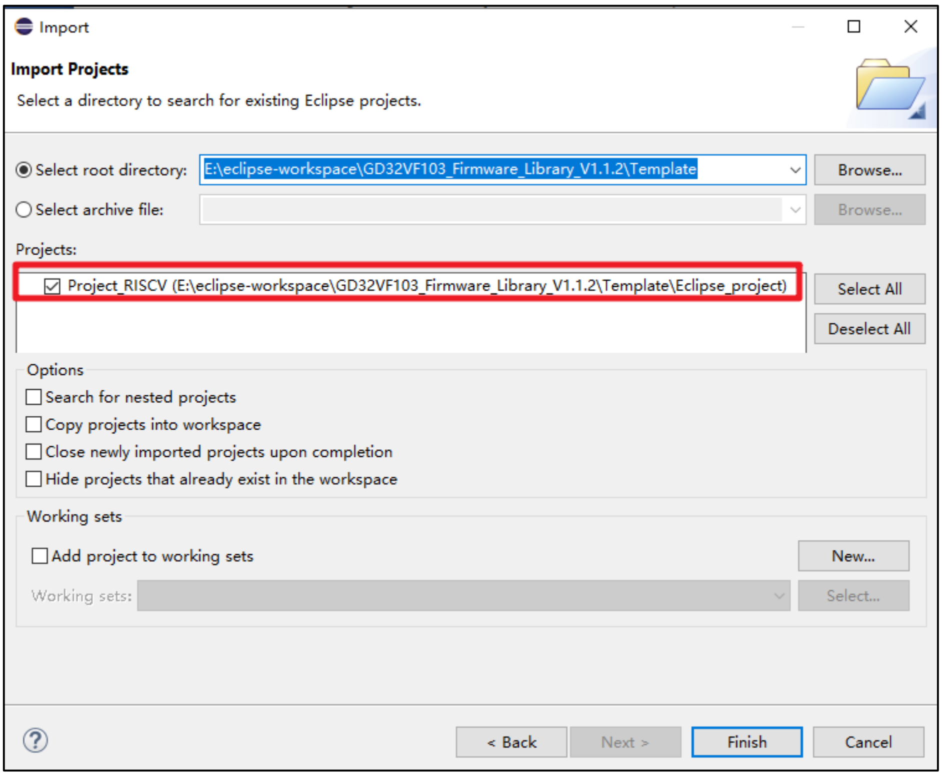
In addition to new projects, user can also import existing Eclipse projects directly. In the menu bar, click “File->Import”, select “General->Exisiting Projects into Workspace” to import the existing project, and click "Next".

Figure 4-1. Import an existing project - 1



Select the path of an existing project file, Eclipse will recognize all the projects under this path. Select the corresponding project, and click "Finish" to import the existing project.

Figure 4-2. Import an existing project - 2



Import

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory: E:\eclipse-workspace\GD32VF103_Firmware_Library_V1.1.2\Template

○ Select archive file:

Projects:

☑ Project_RISCV (E:\eclipse-workspace\GD32VF103_Firmware_Library_V1.1.2\Template\Eclipse_project)

Options

□ Search for nested projects
□ Copy projects into workspace
□ Close newly imported projects upon completion
□ Hide projects that already exist in the workspace

Working sets

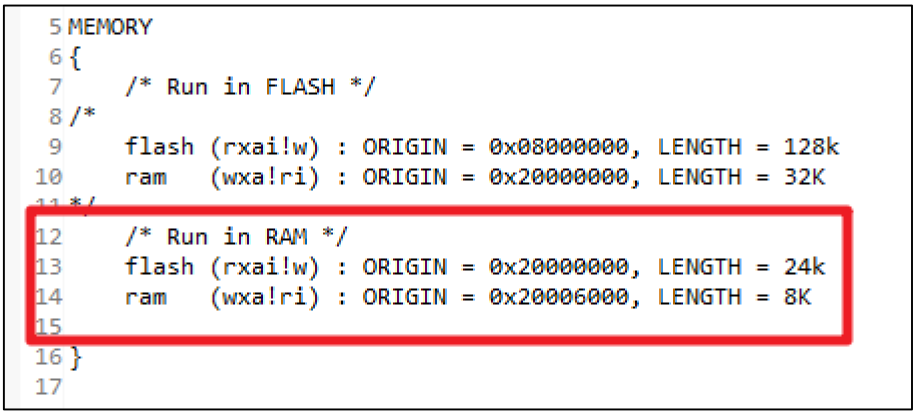
□ Add project to working sets

Working sets:

5. Debug in RAM

Step 1: Modify the link script and recompile the project (it is recommended to clean and then build) for example, as shown in *Figure 5-1. Ld file memory map when debugging in RAM*.

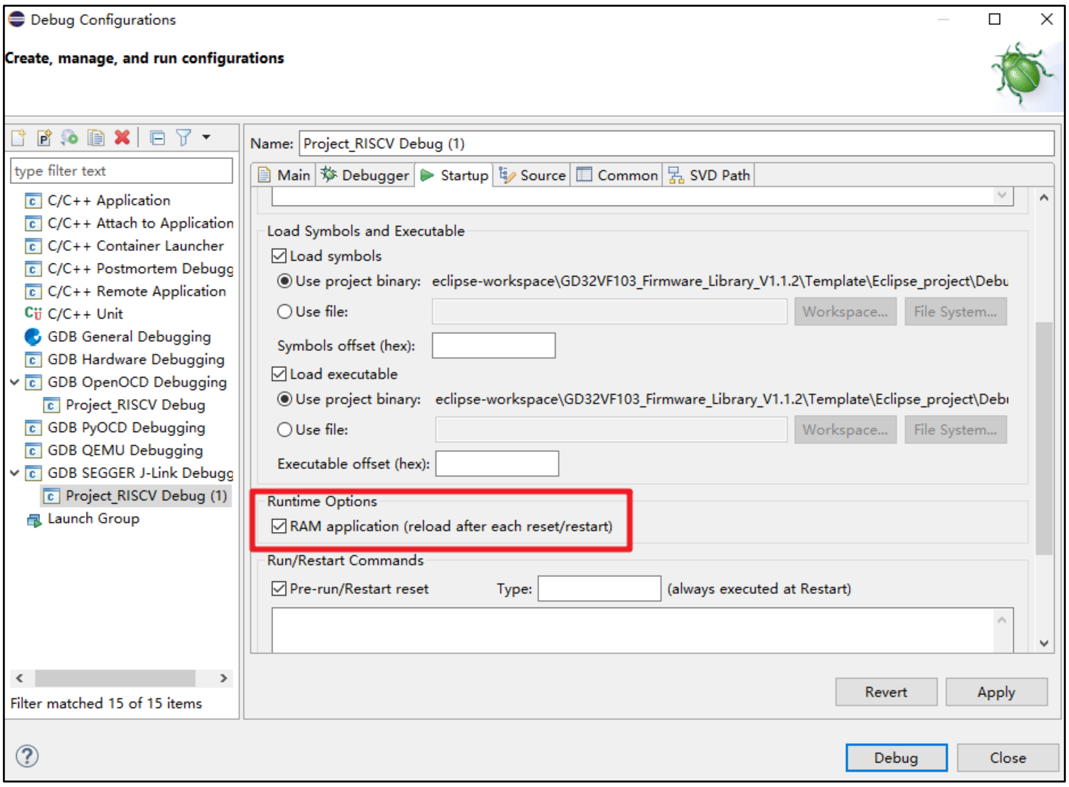
Figure 5-1. Ld file memory map when debugging in RAM



```
MEMORY
{
    /* Run in FLASH */
/*
    flash (rxai!w) : ORIGIN = 0x08000000, LENGTH = 128k
    ram   (wxa!ri) : ORIGIN = 0x20000000, LENGTH = 32K
*/
    /* Run in RAM */
    flash (rxai!w) : ORIGIN = 0x20000000, LENGTH = 24k
    ram   (wxa!ri) : ORIGIN = 0x20006000, LENGTH = 8K

}

```

Step 2: In the “Debug Configurations->Startup” option, check “RAM application”.

Figure 5-2. Debug configurations when debugging in RAM



Debug Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
- C/C++ Attach to Application
- C/C++ Container Launcher
- C/C++ Postmortem Debugg
- C/C++ Remote Application
- C/C++ Unit
- GDB General Debugging
- GDB Hardware Debugging
- GDB OpenOCD Debugging
  - Project_RISCV Debug
- GDB PyOCD Debugging
- GDB QEMU Debugging
- GDB SEGGER J-Link Debugg
  - Project_RISCV Debug (1)
- Launch Group

Filter matched 15 of 15 items

Name: Project_RISCV Debug (1)

Main | Debugger | Startup | Source | Common | SVD Path

Load Symbols and Executable

☑ Load symbols

● Use project binary: eclipse-workspace\GD32VF103_Firmware_Library_V1.1.2\Template\Eclipse_project\Debu

○ Use file: Workspace... File System...

Symbols offset (hex):

☑ Load executable

● Use project binary: eclipse-workspace\GD32VF103_Firmware_Library_V1.1.2\Template\Eclipse_project\Debu

○ Use file: Workspace... File System...

Executable offset (hex):

Runtime Options

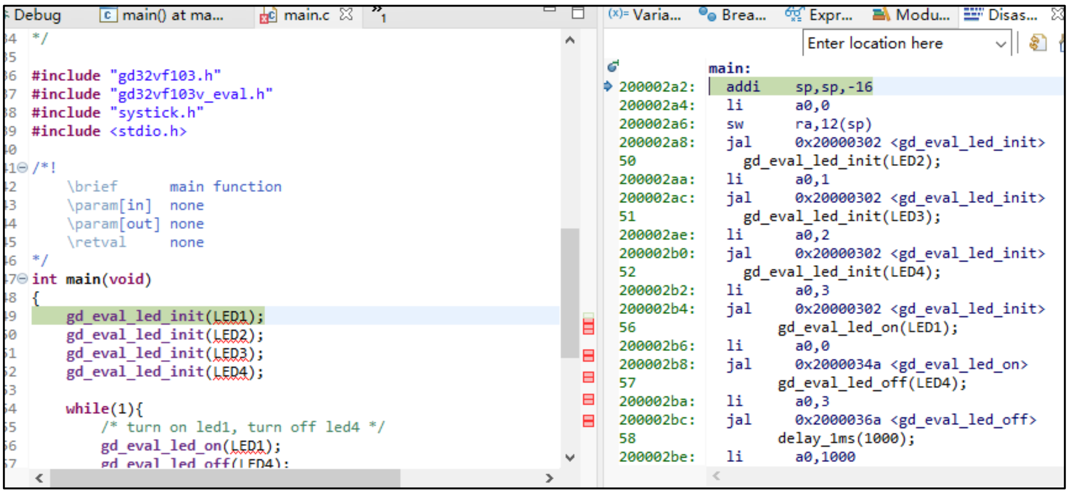
☑ RAM application (reload after each reset/restart)

Run/Restart Commands

☑ Pre-run/Restart reset Type: (always executed at Restart)

Step 3: Enter the Debug perspective when debugging in RAM, as shown in the figure below.

Figure 5-3. Debug perspective when debugging in RAM



```c
*/

#include "gd32vf103.h"
#include "gd32vf103v_eval.h"
#include "systick.h"
#include <stdio.h>

/*!
    \brief      main function
    \param[in]  none
    \param[out] none
    \retval     none
*/
int main(void)
{
    gd_eval_led_init(LED1);
    gd_eval_led_init(LED2);
    gd_eval_led_init(LED3);
    gd_eval_led_init(LED4);

    while(1){
        /* turn on led1, turn off led4 */
        gd_eval_led_on(LED1);
        gd_eval_led_off(LED4);
```

```
            main:
200002a2:   addi     sp,sp,-16
200002a4:   li       a0,0
200002a6:   sw       ra,12(sp)
200002a8:   jal      0x20000302 <gd_eval_led_init>
50            gd_eval_led_init(LED2);
200002aa:   li       a0,1
200002ac:   jal      0x20000302 <gd_eval_led_init>
51            gd_eval_led_init(LED3);
200002ae:   li       a0,2
200002b0:   jal      0x20000302 <gd_eval_led_init>
52            gd_eval_led_init(LED4);
200002b2:   li       a0,3
200002b4:   jal      0x20000302 <gd_eval_led_init>
56                gd_eval_led_on(LED1);
200002b6:   li       a0,0
200002b8:   jal      0x2000034a <gd_eval_led_on>
57                gd_eval_led_off(LED4);
200002ba:   li       a0,3
200002bc:   jal      0x2000036a <gd_eval_led_off>
58                delay_1ms(1000);
200002be:   li       a0,1000
```

6. Printing with printf

6.1. Use steps

Step 1: Add the following _write function definition to the file.

```
ssize_t _write(int fd, const void* ptr, size_t len) {
        const uint8_t * current = (const uint8_t *) ptr;
        {
                for (size_t jj = 0; jj < len; jj++) {
                        _put_char(current[jj]);

                        if (current[jj] == '\n') {
                                _put_char('\r');
                        }
                }
                return len;
        }

        return _stub(EBADF);

}
```

Step 2: Redirect usart to the put_char function.

```
int _put_char(int ch)
{
        usart_data_transmit(USART0, (uint8_t) ch );
        while ( usart_flag_get(USART0, USART_FLAG_TBE)== RESET){
        }

        return ch;

}
```

Step 3: Use the printf function to print normally. Note that "\n" needs to be added to printf function to flush the output stream.

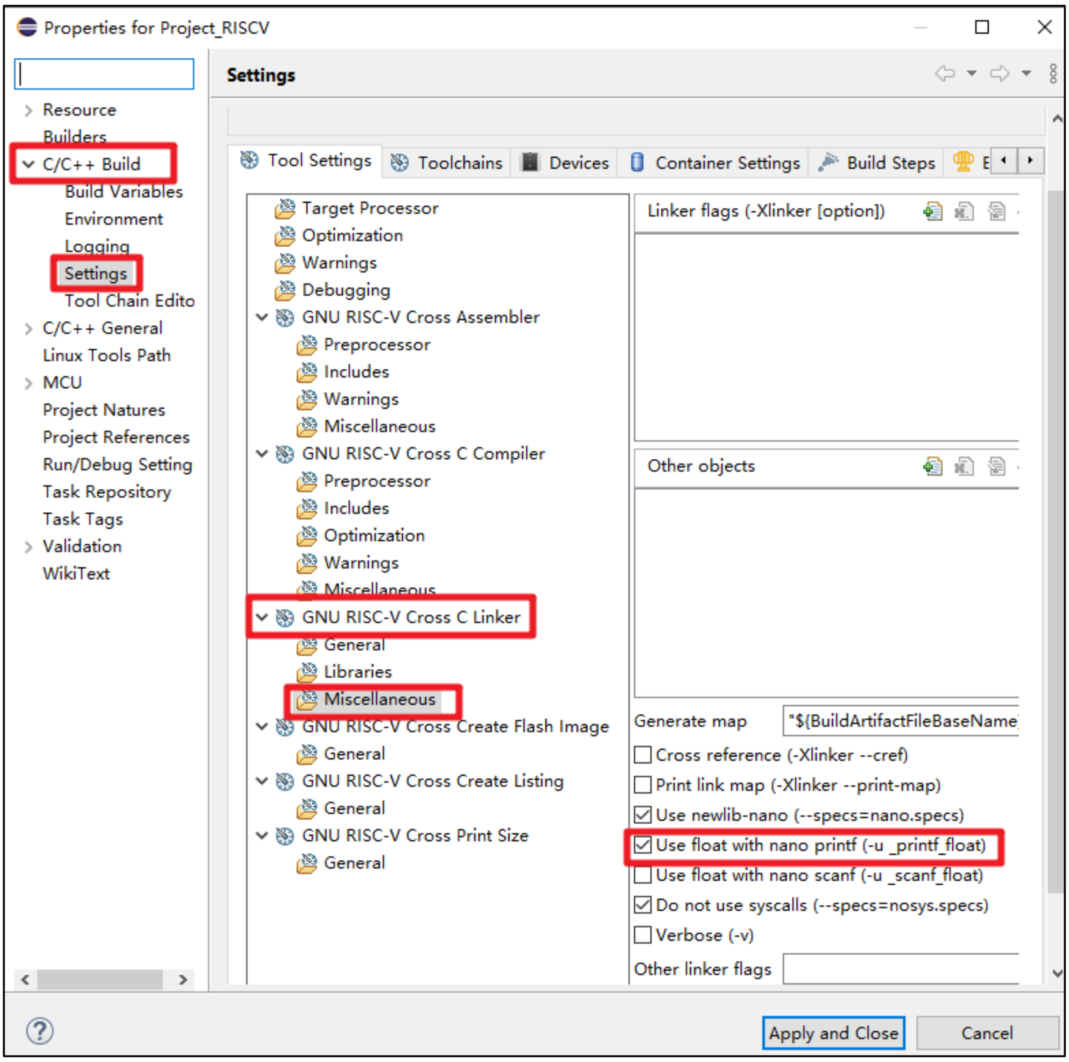
```
      printf("Running led test!\r\n");
```

6.2. Print floating point data configuration

Print floating point data configuration:

check the "-u _prinft_float" option in the project "Properties->C/C++ Build->Settings->Tool Settings->GNU RISC-V Cross C Linker->Miscellaneous" option.

Figure 6-1. Print floating point data configuration



Linker flags (-Xlinker [option])

Other objects

Generate map "${BuildArtifactFileBaseName

□ Cross reference (-Xlinker --cref)

□ Print link map (-Xlinker --print-map)

☑ Use newlib-nano (--specs=nano.specs)

☑ Use float with nano printf (-u _printf_float)

□ Use float with nano scanf (-u _scanf_float)

☑ Do not use syscalls (--specs=nosys.specs)

□ Verbose (-v)

Other linker flags

Note: 1. When using printf function, user need to add “\r\n” at the end of the printed content, for example, printf("Running led test!\r\n"). 2. Using printf function in GCC will greatly increase the size of the code. If it is an occasion that requires a high codesized size, printf function is not recommended.

7. Revision history

Table 7-1. Revision history

Revision No.	Description	Date
1.0	Initial Release	Jun.20, 2022

Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the “Product”), is owned by the Company under the intellectual property laws and treaties of the People’s Republic of China and other jurisdictions worldwide. The Company reserves all rights under such laws and treaties and does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

The Company makes no warranty of any kind, express or implied, with regard to this document or any Product, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Company does not assume any liability arising out of the application or use of any Product described in this document. Any information provided in this document is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Except for customized products which has been expressly identified in the applicable agreement, the Products are designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only. The Products are not designed, intended, or authorized for use as components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, atomic energy control instruments, combustion control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or Product could cause personal injury, death, property or environmental damage ("Unintended Uses"). Customers shall take any and all actions to ensure using and selling the Products in accordance with the applicable laws and regulations. The Company is not liable, in whole or in part, and customers shall and hereby do release the Company as well as it’s suppliers and/or distributors from any claim, damage, or other liability arising from or related to all Unintended Uses of the Products. Customers shall indemnify and hold the Company as well as it’s suppliers and/or distributors hRISC-Vless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Products.

Information in this document is provided solely in connection with the Products. The Company reserves the right to make changes, corrections, modifications or improvements to this document and Products and services described herein at any time, without notice.