GigaDevice Semiconductor Inc.

Arm® Cortex®-M3/4/23/33 32-bit MCU

Application Note
AN041

Table of Contents

List of Figures

List of Tables

1. Development environment

- Development boards: GD32 MCU boards
- Hardware Debugger: J-Link V9/V10 or GD-Link
- Operating system: WIN7 64-bit OS
- IDE: eclipse-embedcpp-2021-03-R-win32-x86_64
- Cross toolchains: xpack-arm-none-eabi-gcc-10.2.1-1.1-win32-x64
- Build Tools: gnu-mcu-eclipse-windows-build-tools-2.12-20190422-1053-win64
- GDB server: OpenOCD / J-Link GDB Server CL V7.54b

2. Project development

2.1. 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 2-1. New ARM C project

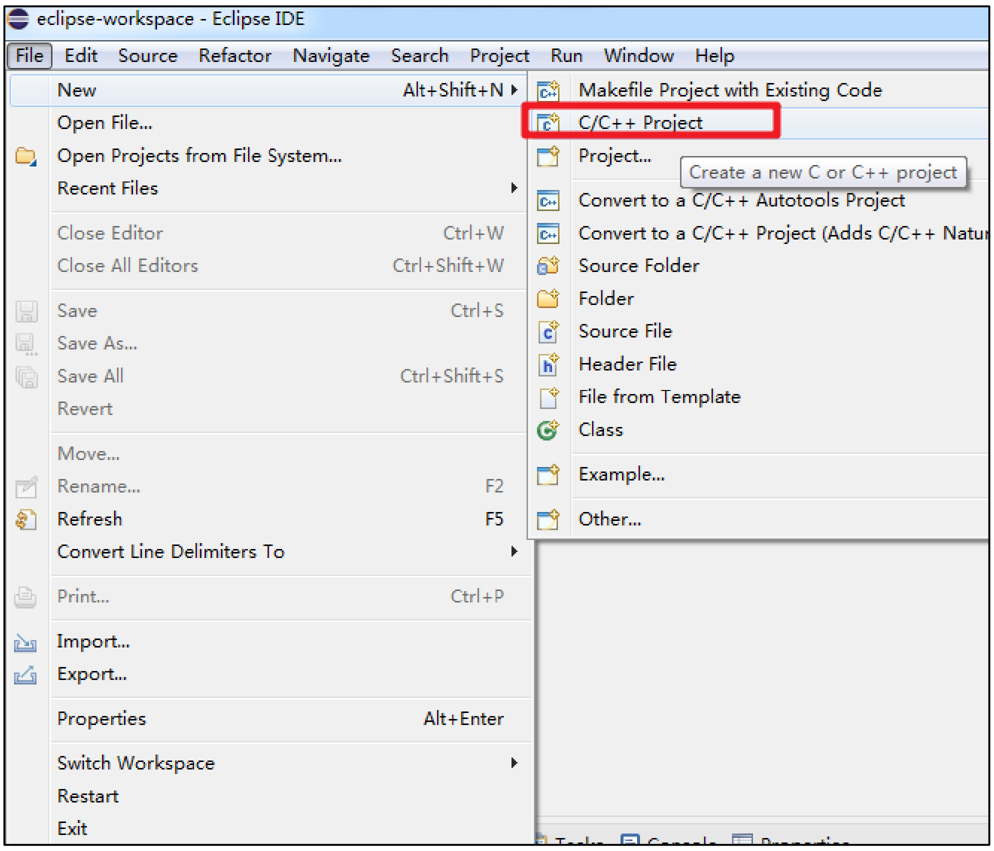
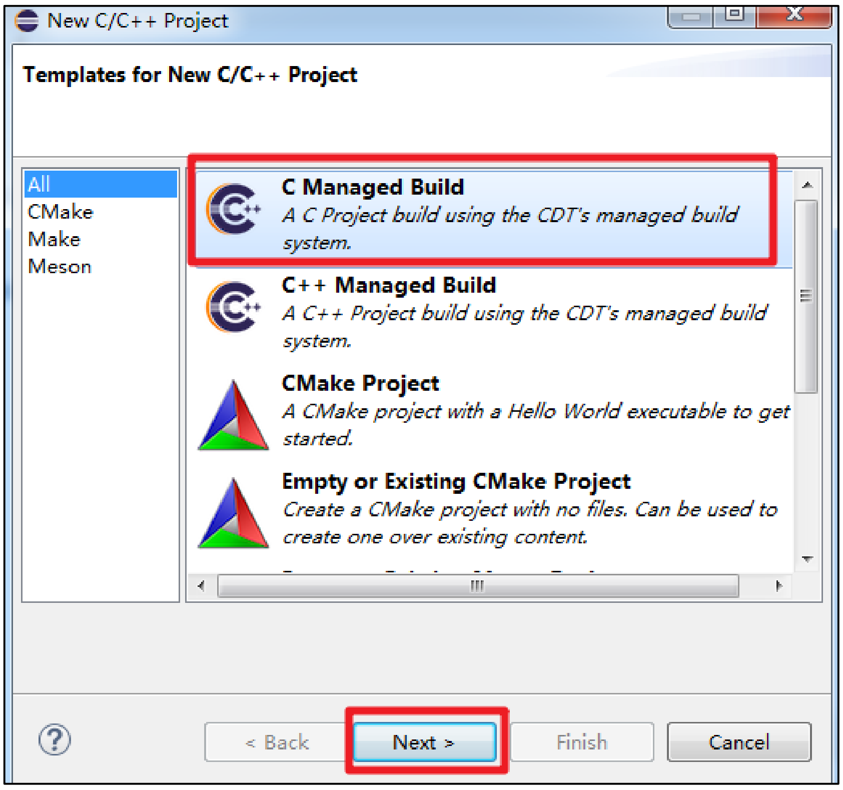
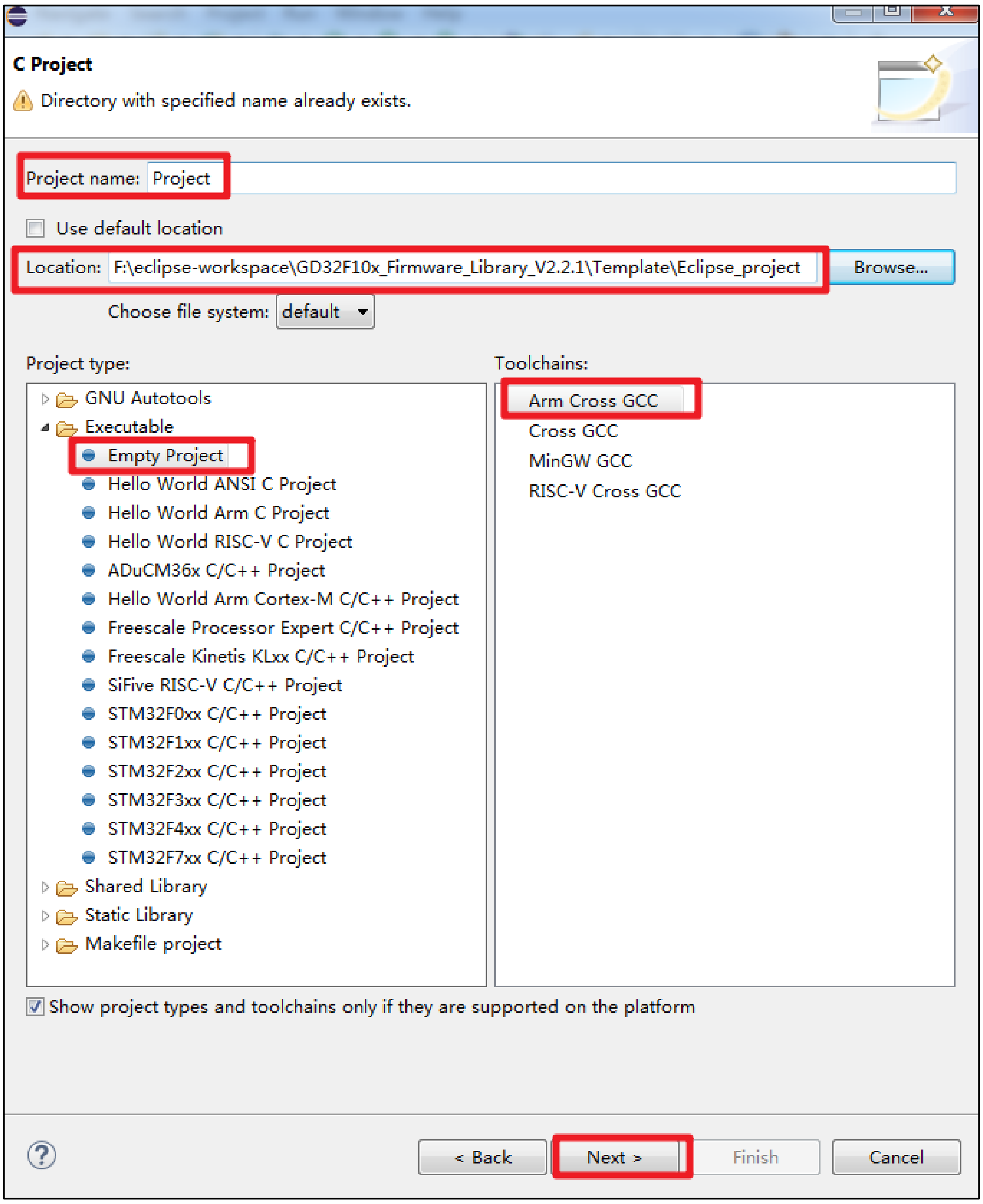


Figure 2-2. Select C Managed Build


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.
< Back
Next >
Finish
Cancel

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 ARM Cross GCC.

Figure 2-3. Create new ARM project name and select project storage path



C Project

⚠ Directory with specified name already exists.

Project name: Project

☐ Use default location

Location: F:\eclipse-workspace\GD32F10x_Firmware_Library_V2.2.1\Template\Eclipse_project

Choose file system: default

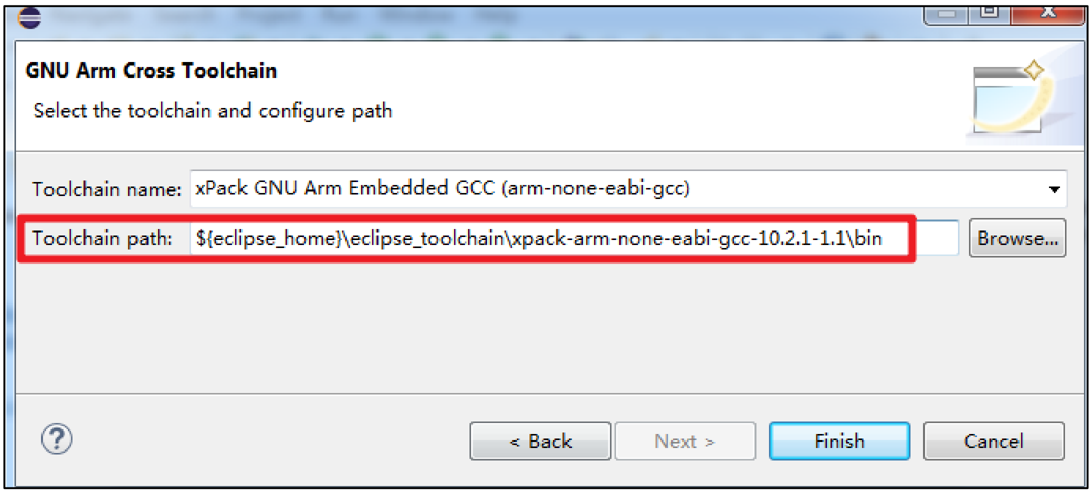
Project type:
- GNU Autotools
- Executable
  - Empty Project
  - Hello World ANSI C Project
  - Hello World Arm C Project
  - Hello World RISC-V C Project
  - ADuCM36x C/C++ Project
  - Hello World Arm Cortex-M C/C++ Project
  - Freescale Processor Expert C/C++ Project
  - Freescale Kinetis KLxx C/C++ 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 ARM Toolchains Path correctly, the path will be automatically selected here. If the Eclipse IDE has not set the ARM Toolchains Path, user can also select the absolute path to the ARM Toolchains here.

Figure 2-4. Select ARM cross toolchain path



GNU Arm Cross Toolchain

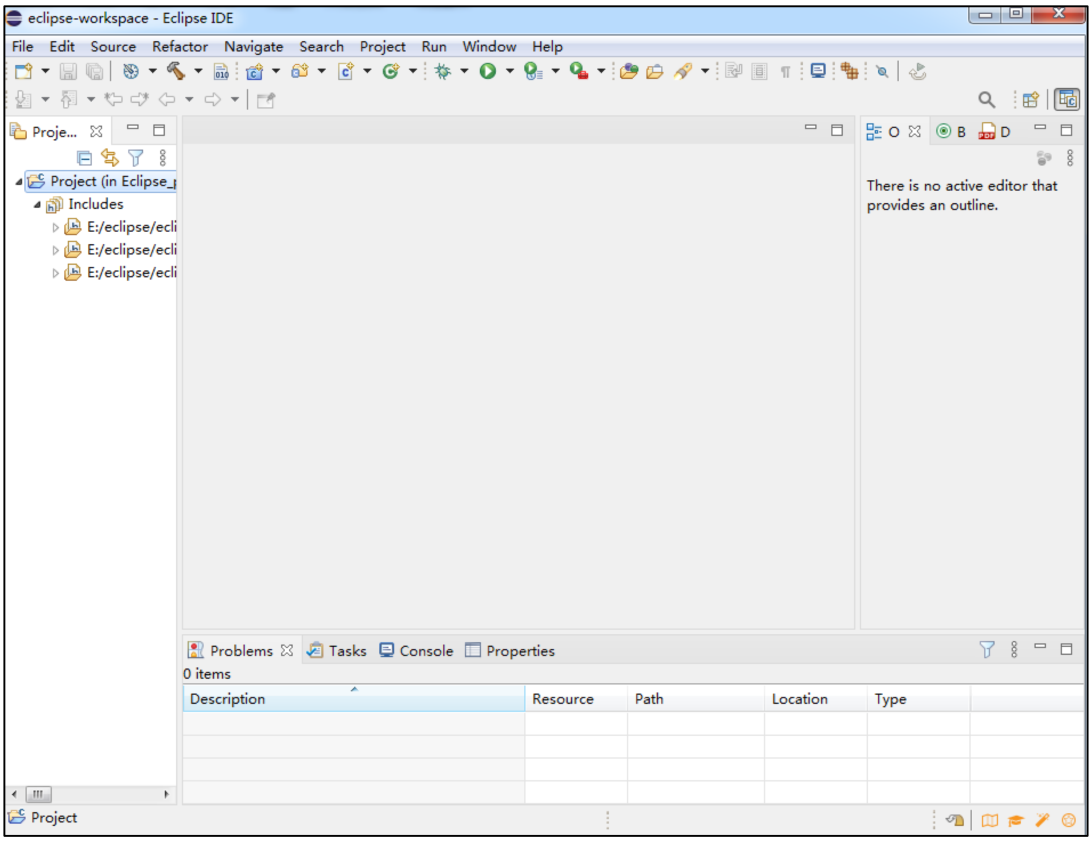
Select the toolchain and configure path

Toolchain name: xPack GNU Arm Embedded GCC (arm-none-eabi-gcc)

Toolchain path: ${eclipse_home}\eclipse_toolchain\xpack-arm-none-eabi-gcc-10.2.1-1.1\bin

Click "Finish" until the display interface is shown in ***Figure 2-5. Project perspective***. At this point, the establishment of the Project is completed.

Figure 2-5. Project perspective



There is no active editor that provides an outline.

0 items

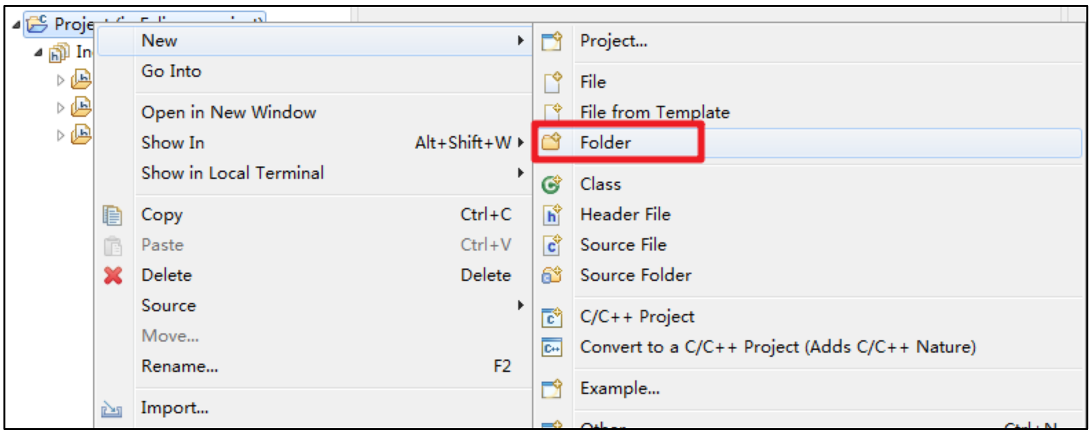
Description	Resource	Path	Location	Type

2.2. New project folder and add files

2.2.1. Ceate folders and add files manually

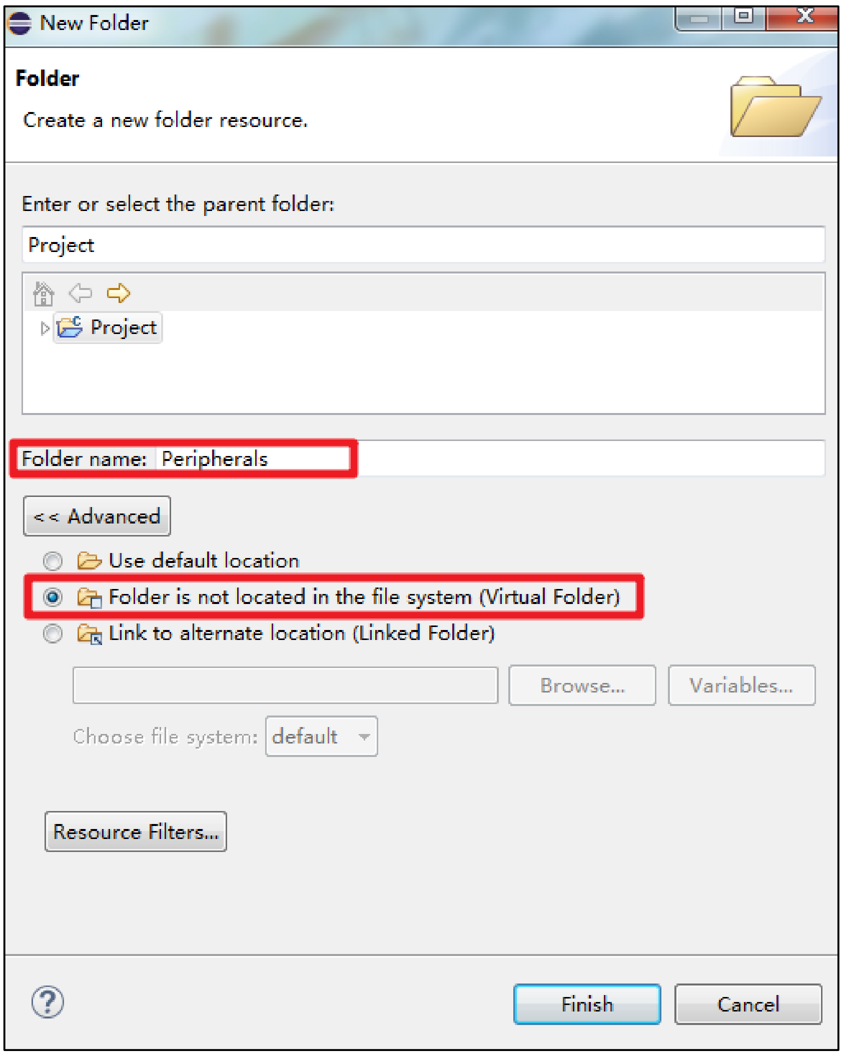
Right-click the project name and select “new->Folder”.

Figure 2-6. New project folder



Create a virtual folder “Peripherals”.

Figure 2-7. New virtual sub-folder



New Folder

Folder

Create a new folder resource.

Enter or select the parent folder:

Project

Folder name: Peripherals

<< Advanced

Use default location

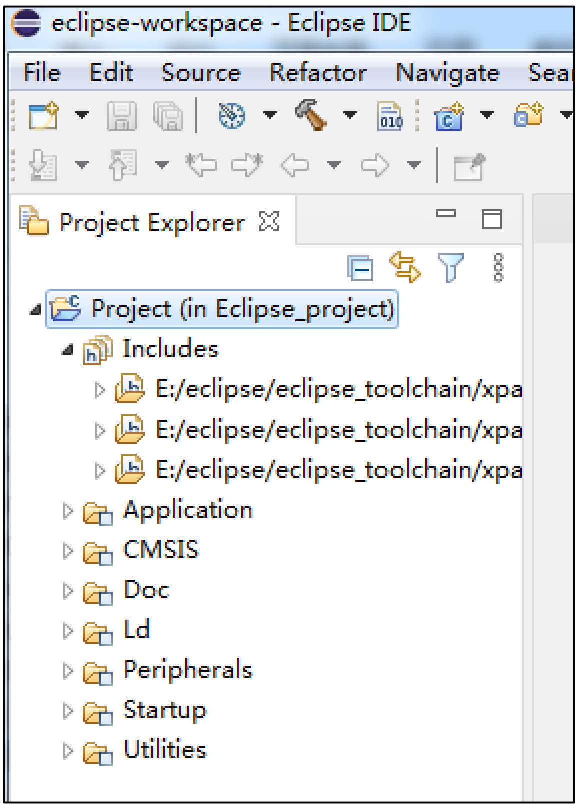
[✓ Folder is not located in the file system (Virtual Folder)]

Link to alternate location (Linked Folder)

Choose file system: default

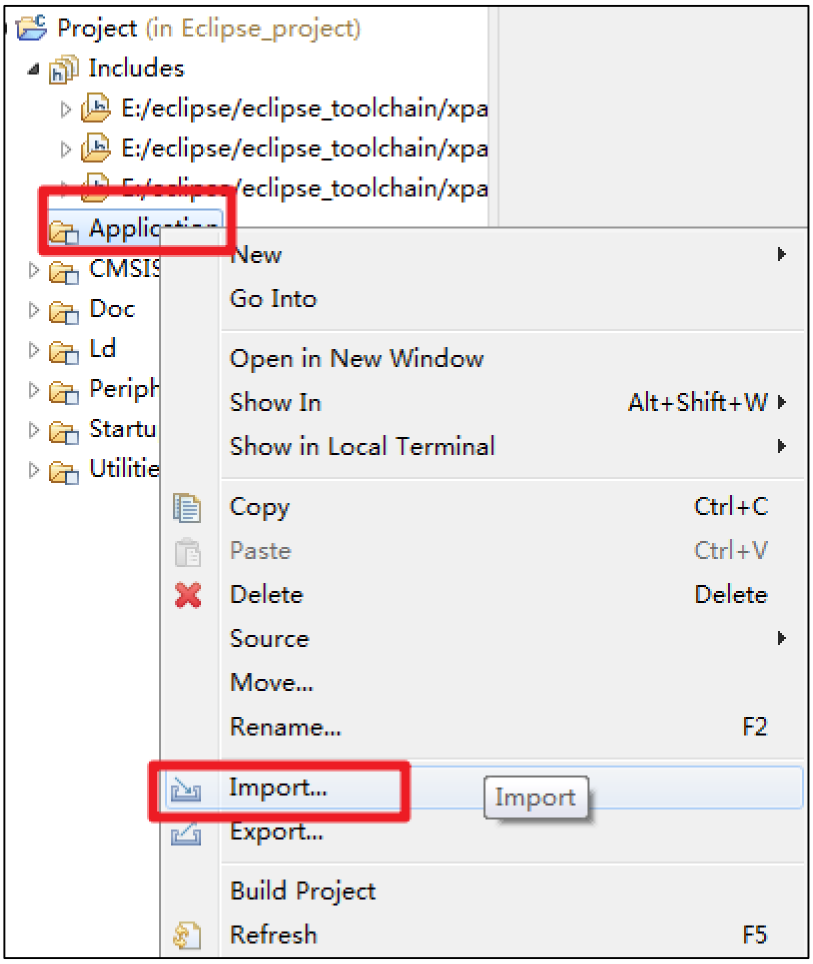
Create the Application, CMSIS, Doc, Ld, Startup and Utilities folders in the same way.

Figure 2-8. ARM project view


eclipse-workspace - Eclipse IDE
File Edit Source Refactor Navigate Sea
Project Explorer
Project (in Eclipse_project)
Includes
E:/eclipse/eclipse_toolchain/xpa
E:/eclipse/eclipse_toolchain/xpa
E:/eclipse/eclipse_toolchain/xpa
Application
CMSIS
Doc
Ld
Peripherals
Startup
Utilities

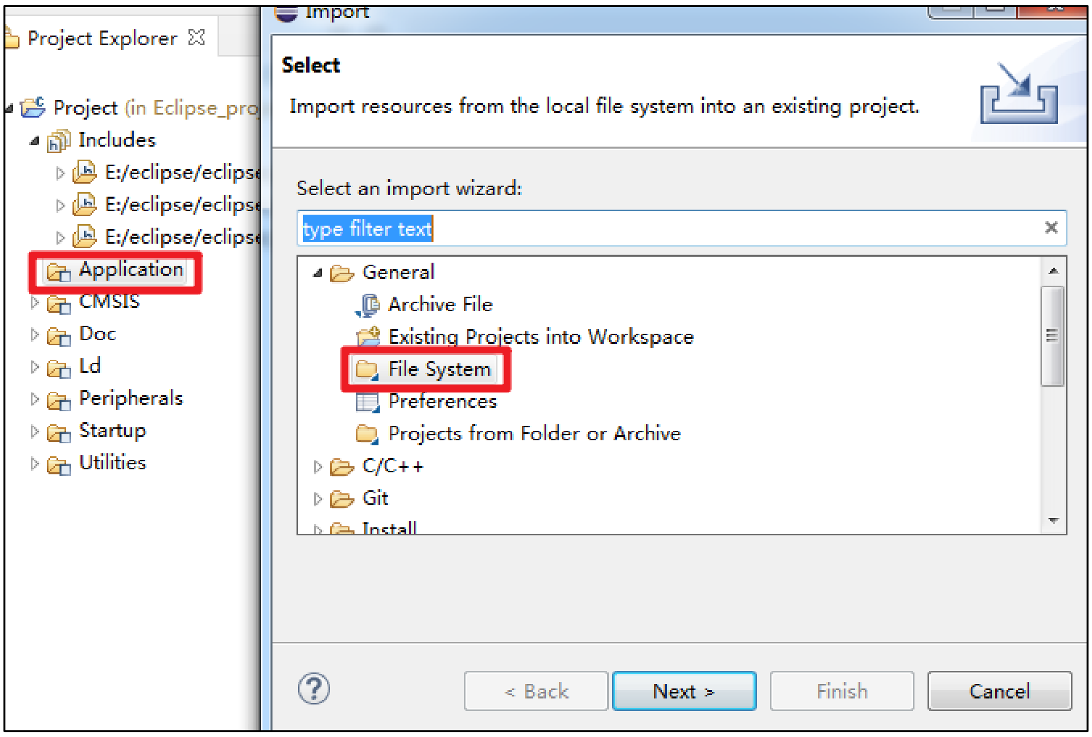
Right-click “Application” and select the “Import” option to import the file.

Figure 2-9. Add files



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

Figure 2-10. Select files to be imported



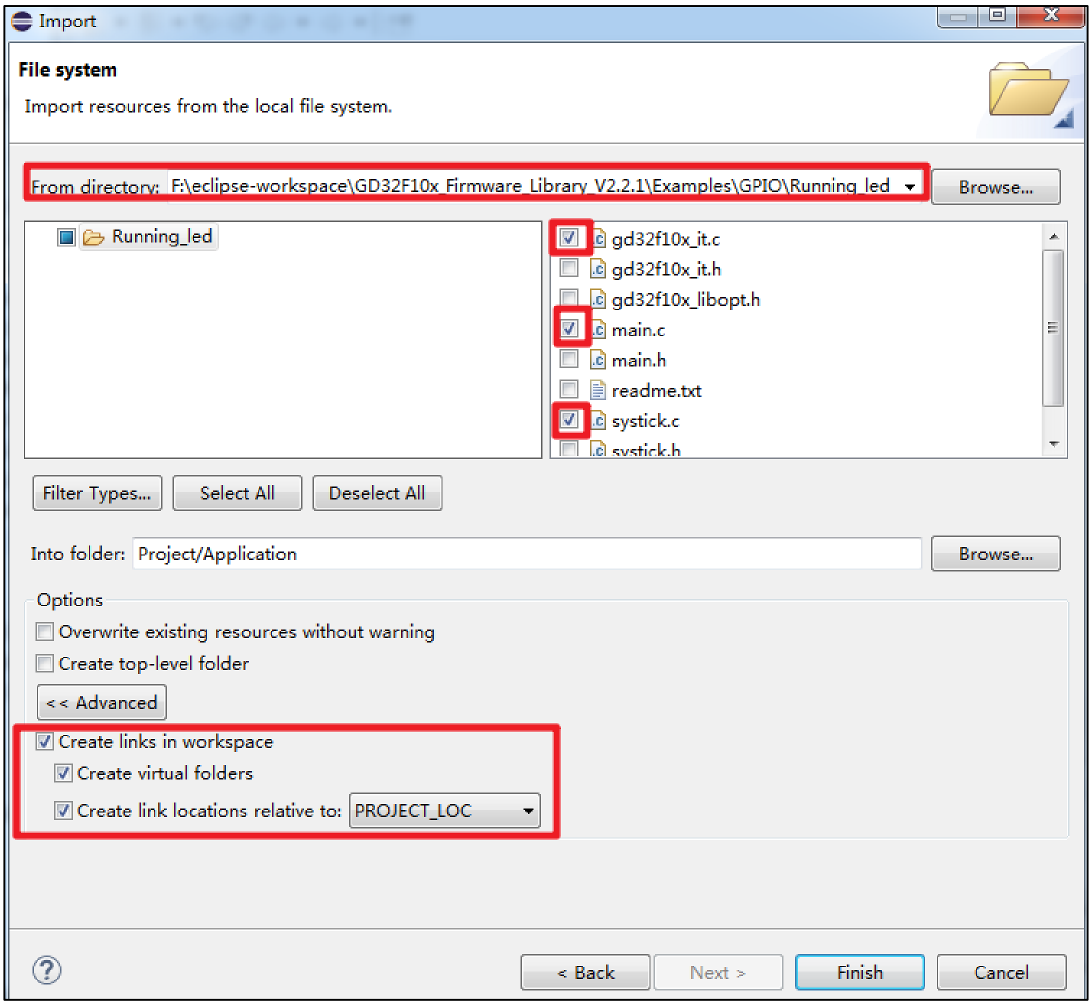
Select

Import resources from the local file system into an existing project.

Select an import wizard:

type filter text

Figure 2-11. Import files to the Application folder



Import

File system

Import resources from the local file system.

From directory: F:\eclipse-workspace\GD32F10x_Firmware_Library_V2.2.1\Examples\GPIO\Running_led

Running_led

- [✓] gd32f10x_it.c
- [ ] gd32f10x_it.h
- [ ] gd32f10x_libopt.h
- [✓] main.c
- [ ] main.h
- [ ] readme.txt
- [✓] systick.c
- [ ] systick.h

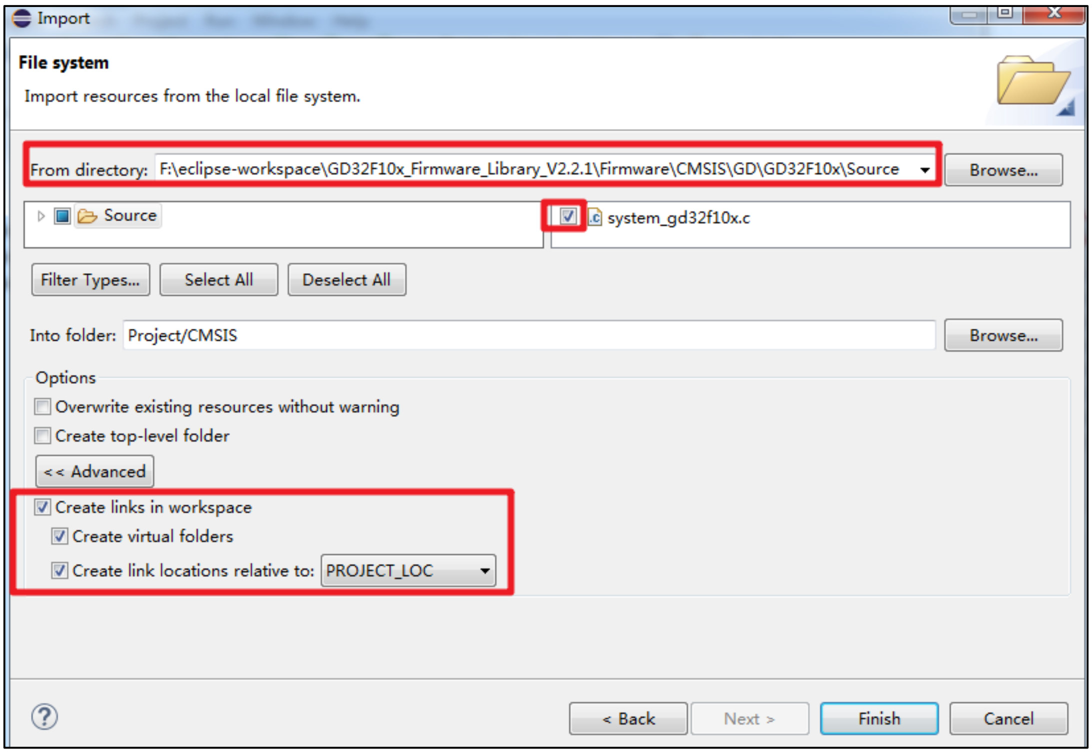
Into folder: Project/Application

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 “CMSIS”, “Doc”, “Ld”, “Peripherals”, “Startup” and “Utilities” folders.

Figure 2-12. Import files to the CMSIS folder



Import

File system

Import resources from the local file system.

From directory: F:\eclipse-workspace\GD32F10x_Firmware_Library_V2.2.1\Firmware\CMSIS\GD\GD32F10x\Source

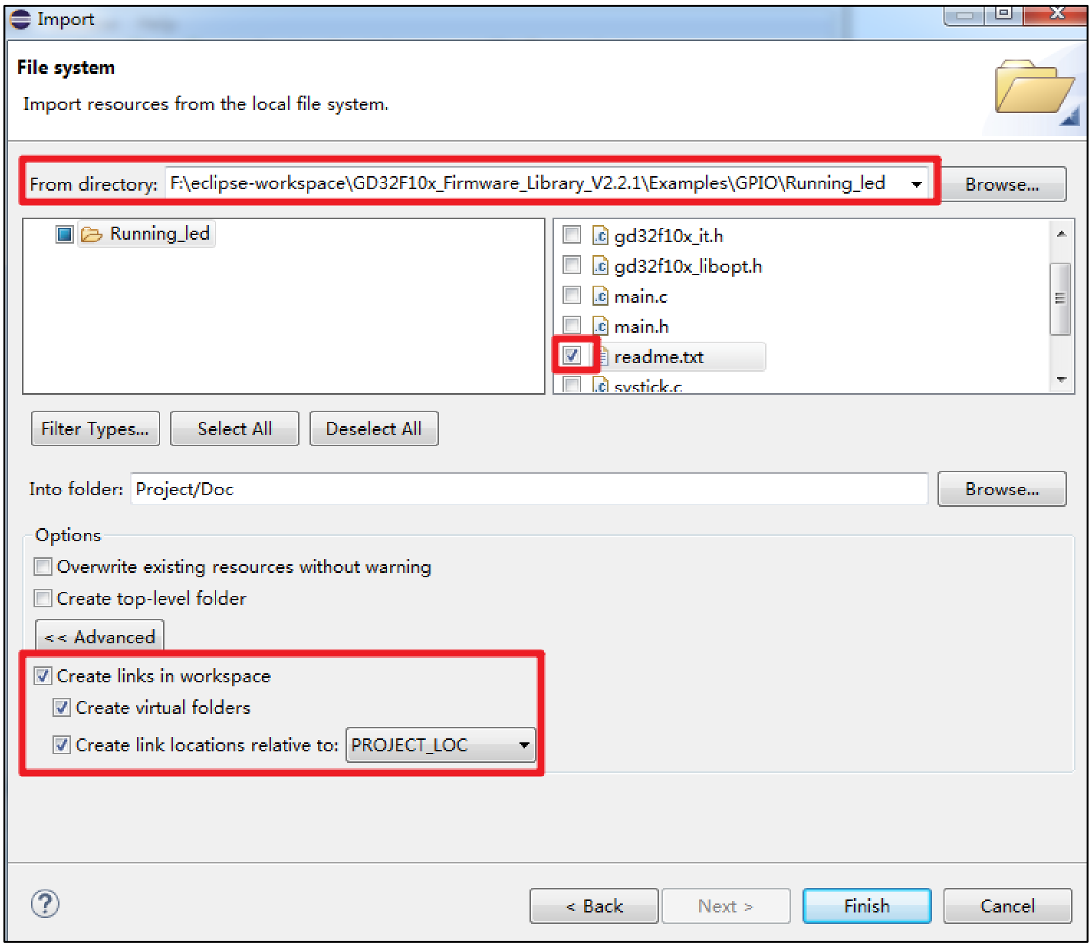
Source

[✓] system_gd32f10x.c

Into folder: Project/CMSIS

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 2-13. Import files to the Doc folder



Import

File system

Import resources from the local file system.

From directory: F:\eclipse-workspace\GD32F10x_Firmware_Library_V2.2.1\Examples\GPIO\Running_led

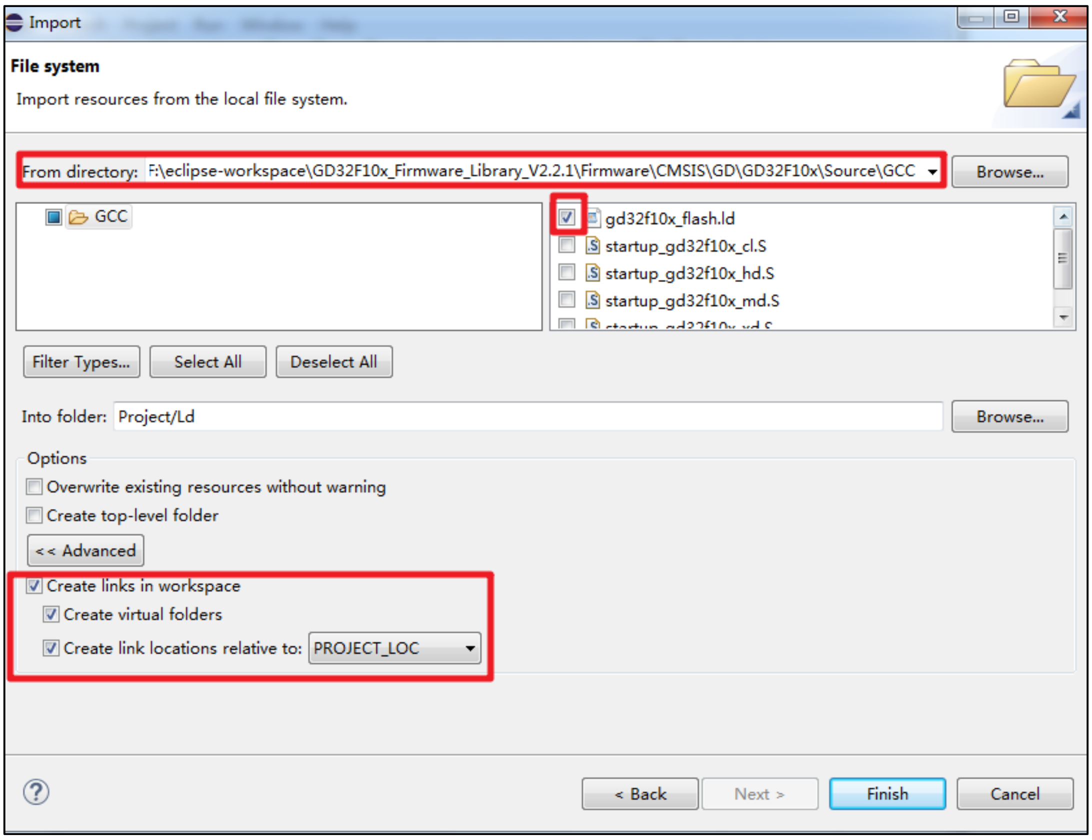
Running_led

- [ ] gd32f10x_it.h
- [ ] gd32f10x_libopt.h
- [ ] main.c
- [ ] main.h
- [✓] readme.txt
- [ ] systick.c

Into folder: Project/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 2-14. Import files to the LD folder



Import

File system

Import resources from the local file system.

From directory: F:\eclipse-workspace\GD32F10x_Firmware_Library_V2.2.1\Firmware\CMSIS\GD\GD32F10x\Source\GCC

GCC

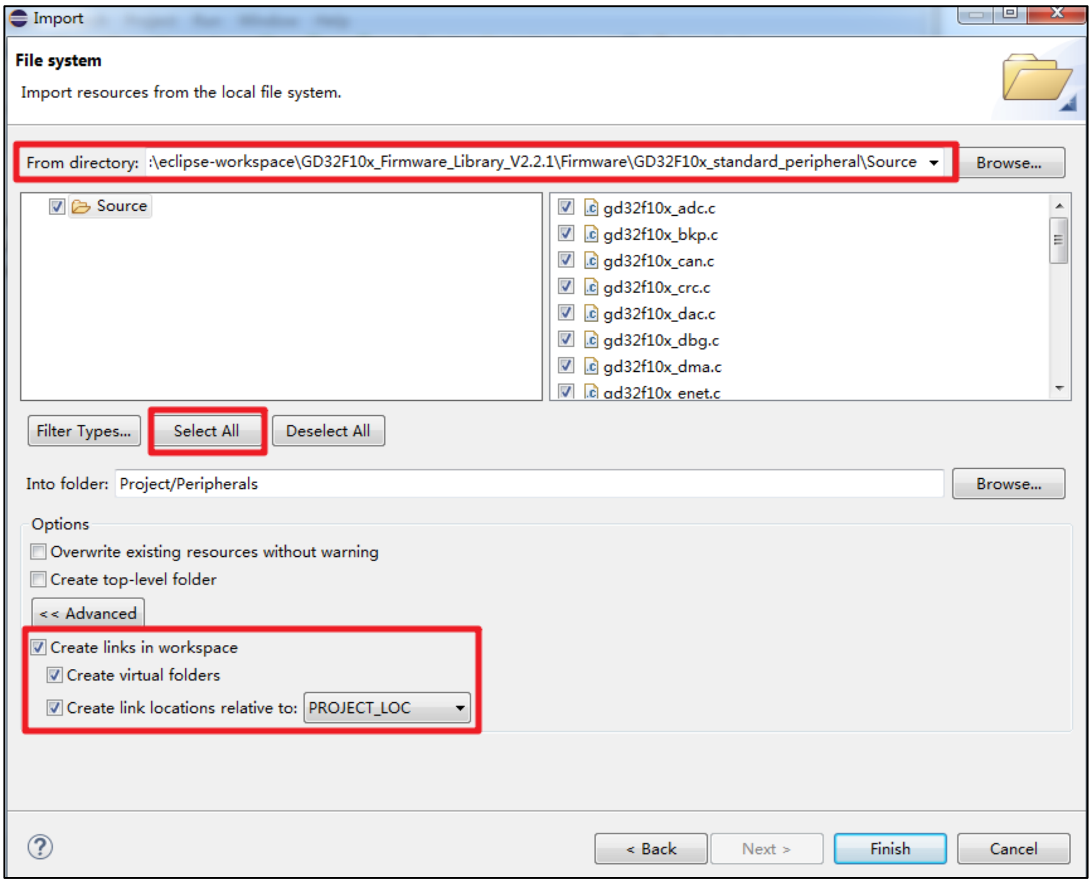
- [✓] gd32f10x_flash.ld
- [ ] startup_gd32f10x_cl.S
- [ ] startup_gd32f10x_hd.S
- [ ] startup_gd32f10x_md.S

Into folder: Project/Ld

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 2-15. Import files to the Peripherals folder


Import

File system

Import resources from the local file system.

From directory: :\eclipse-workspace\GD32F10x_Firmware_Library_V2.2.1\Firmware\GD32F10x_standard_peripheral\Source

☑ Source

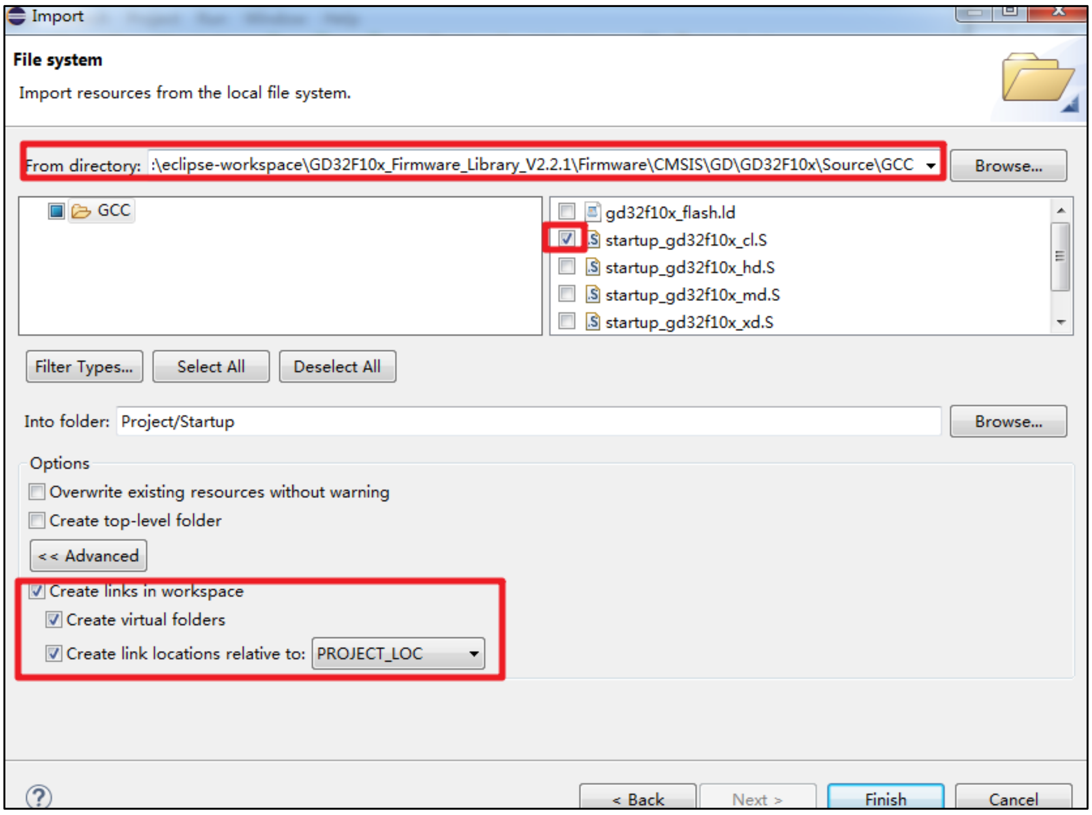
☑ gd32f10x_adc.c
☑ gd32f10x_bkp.c
☑ gd32f10x_can.c
☑ gd32f10x_crc.c
☑ gd32f10x_dac.c
☑ gd32f10x_dbg.c
☑ gd32f10x_dma.c
☑ gd32f10x_enet.c

Into folder: Project/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 2-16. Import files to the Startup folder



Import

File system

Import resources from the local file system.

From directory: :\eclipse-workspace\GD32F10x_Firmware_Library_V2.2.1\Firmware\CMSIS\GD\GD32F10x\Source\GCC

GCC

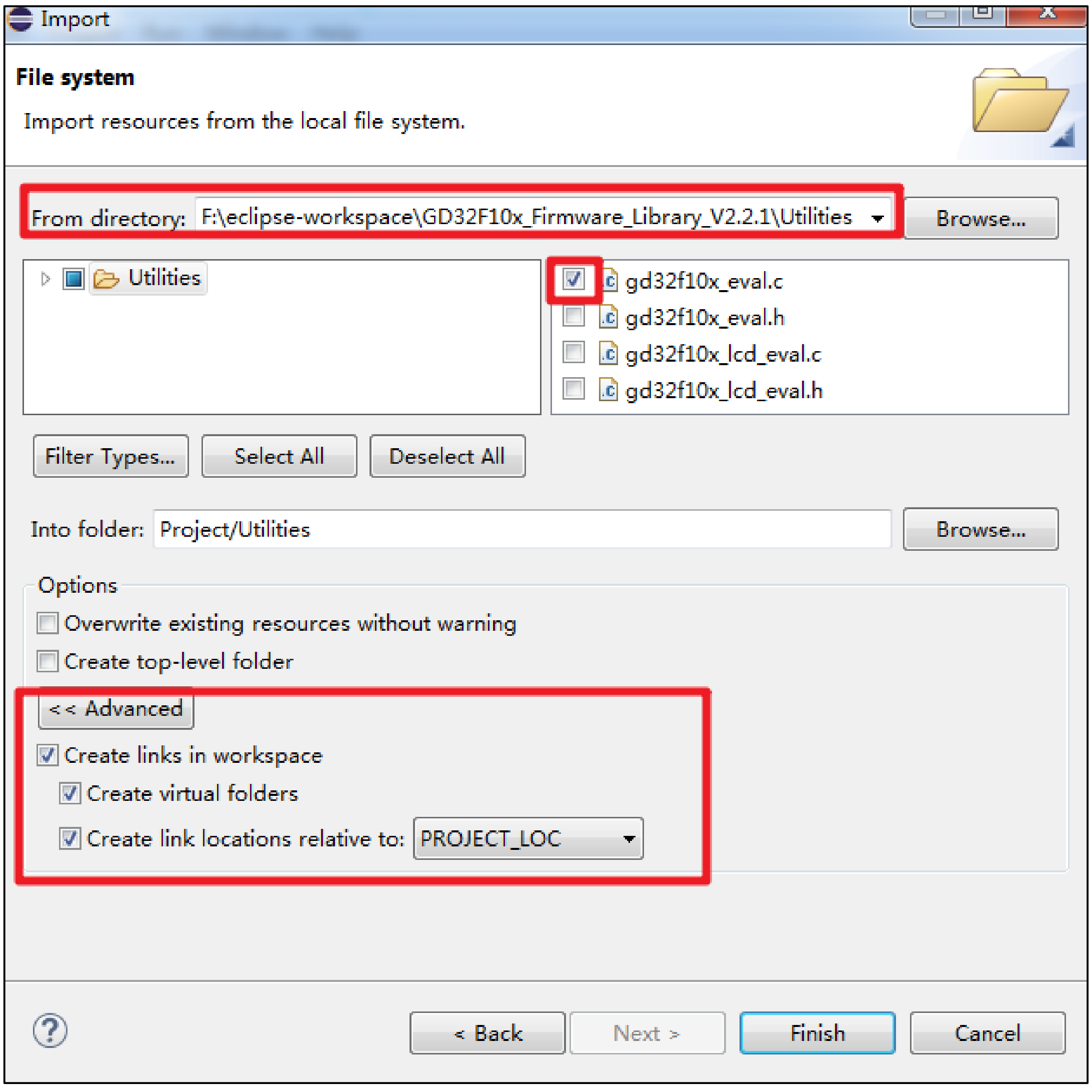
- [ ] gd32f10x_flash.ld
- [x] startup_gd32f10x_cl.S
- [ ] startup_gd32f10x_hd.S
- [ ] startup_gd32f10x_md.S
- [ ] startup_gd32f10x_xd.S

Into folder: Project/Startup

Options

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

Figure 2-17. Import files to the Utilities folder



Import

File system

Import resources from the local file system.

From directory: F:\eclipse-workspace\GD32F10x_Firmware_Library_V2.2.1\Utilities

Utilities

- [✓] gd32f10x_eval.c
- [ ] gd32f10x_eval.h
- [ ] gd32f10x_lcd_eval.c
- [ ] gd32f10x_lcd_eval.h

Into folder: Project/Utilities

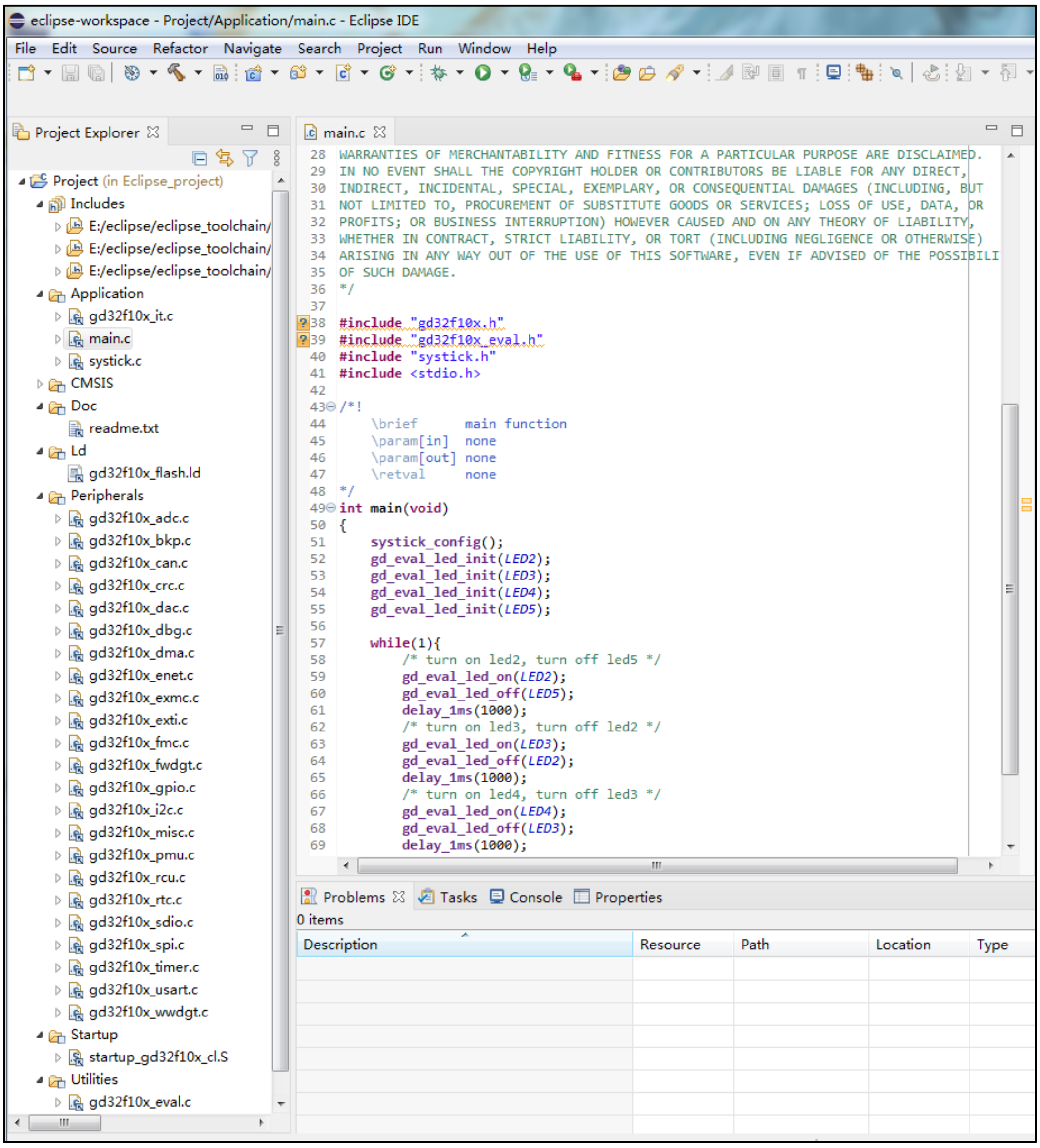
Options

- [ ] Overwrite existing resources without warning
- [ ] Create top-level folder

<< Advanced

- [✓] Create links in workspace
  - [✓] Create virtual folders
  - [✓] Create link locations relative to: PROJECT_LOC

Figure 2-18. Final ARM project view



```
Project (in Eclipse_project)
  Includes
    E:/eclipse/eclipse_toolchain/
    E:/eclipse/eclipse_toolchain/
    E:/eclipse/eclipse_toolchain/
  Application
    gd32f10x_it.c
    main.c
    systick.c
  CMSIS
  Doc
    readme.txt
  Ld
    gd32f10x_flash.ld
  Peripherals
    gd32f10x_adc.c
    gd32f10x_bkp.c
    gd32f10x_can.c
    gd32f10x_crc.c
    gd32f10x_dac.c
    gd32f10x_dbg.c
    gd32f10x_dma.c
    gd32f10x_enet.c
    gd32f10x_exmc.c
    gd32f10x_exti.c
    gd32f10x_fmc.c
    gd32f10x_fwdgt.c
    gd32f10x_gpio.c
    gd32f10x_i2c.c
    gd32f10x_misc.c
    gd32f10x_pmu.c
    gd32f10x_rcu.c
    gd32f10x_rtc.c
    gd32f10x_sdio.c
    gd32f10x_spi.c
    gd32f10x_timer.c
    gd32f10x_usart.c
    gd32f10x_wwdgt.c
  Startup
    startup_gd32f10x_cl.S
  Utilities
    gd32f10x_eval.c
```

```c
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 POSSIBILI
OF SUCH DAMAGE.
*/

#include "gd32f10x.h"
#include "gd32f10x_eval.h"
#include "systick.h"
#include <stdio.h>

/*!
    \brief      main function
    \param[in]  none
    \param[out] none
    \retval     none
*/
int main(void)
{
    systick_config();
    gd_eval_led_init(LED2);
    gd_eval_led_init(LED3);
    gd_eval_led_init(LED4);
    gd_eval_led_init(LED5);

    while(1){
        /* turn on led2, turn off led5 */
        gd_eval_led_on(LED2);
        gd_eval_led_off(LED5);
        delay_1ms(1000);
        /* turn on led3, turn off led2 */
        gd_eval_led_on(LED3);
        gd_eval_led_off(LED2);
        delay_1ms(1000);
        /* turn on led4, turn off led3 */
        gd_eval_led_on(LED4);
        gd_eval_led_off(LED3);
        delay_1ms(1000);
```

2.2.2. 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 2-19. Project folder structure

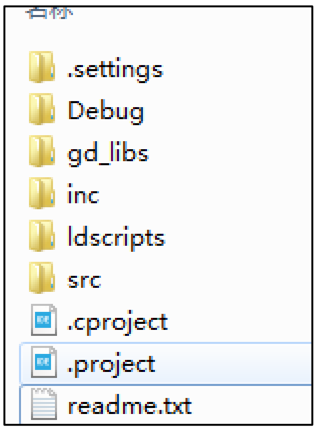

.settings
Debug
gd_libs
inc
ldscripts
src
.cproject
.project
readme.txt

Figure 2-20. Refresh the project

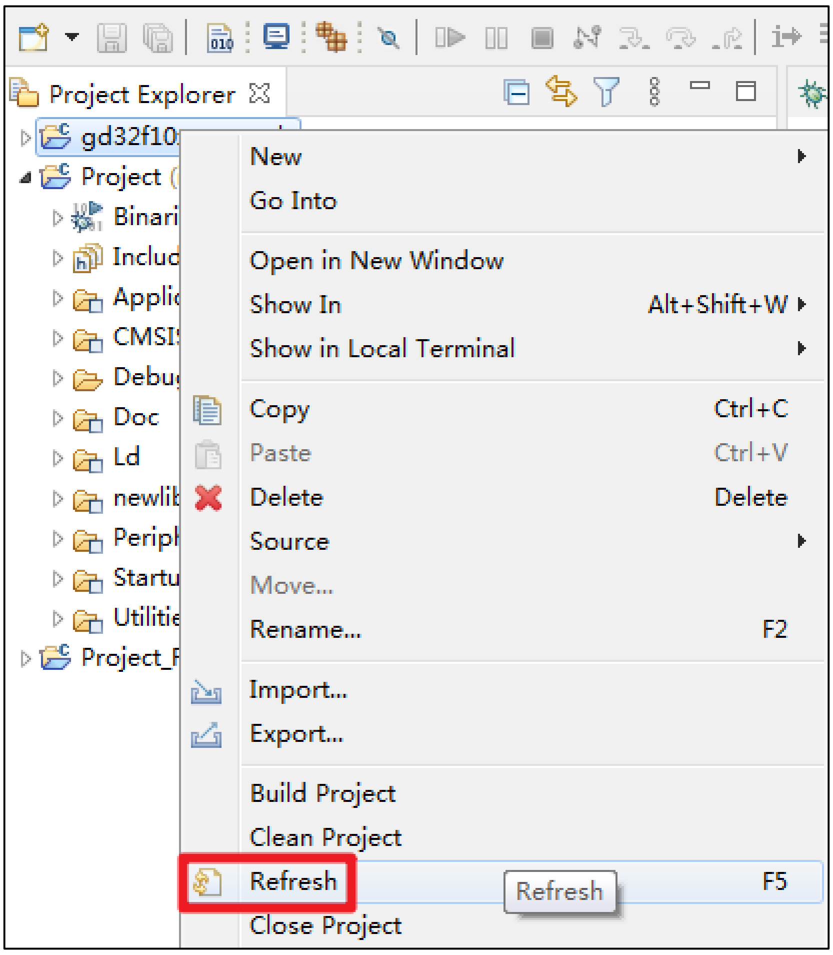
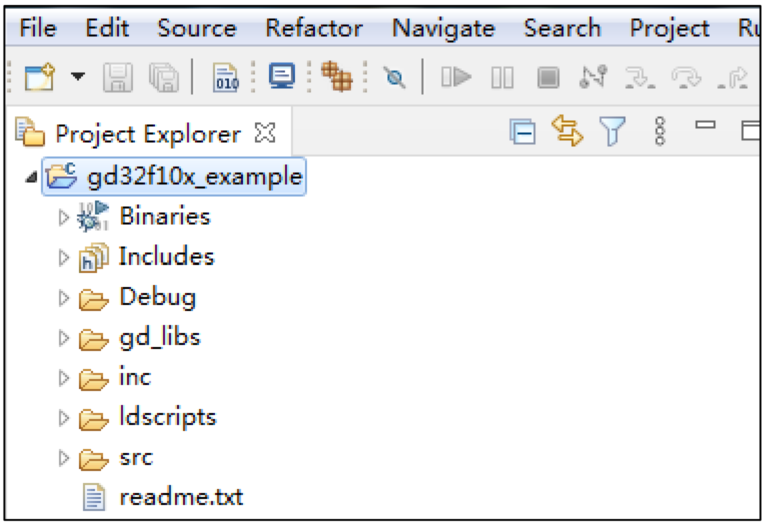


Figure 2-21. Project structure in Eclipse IDE

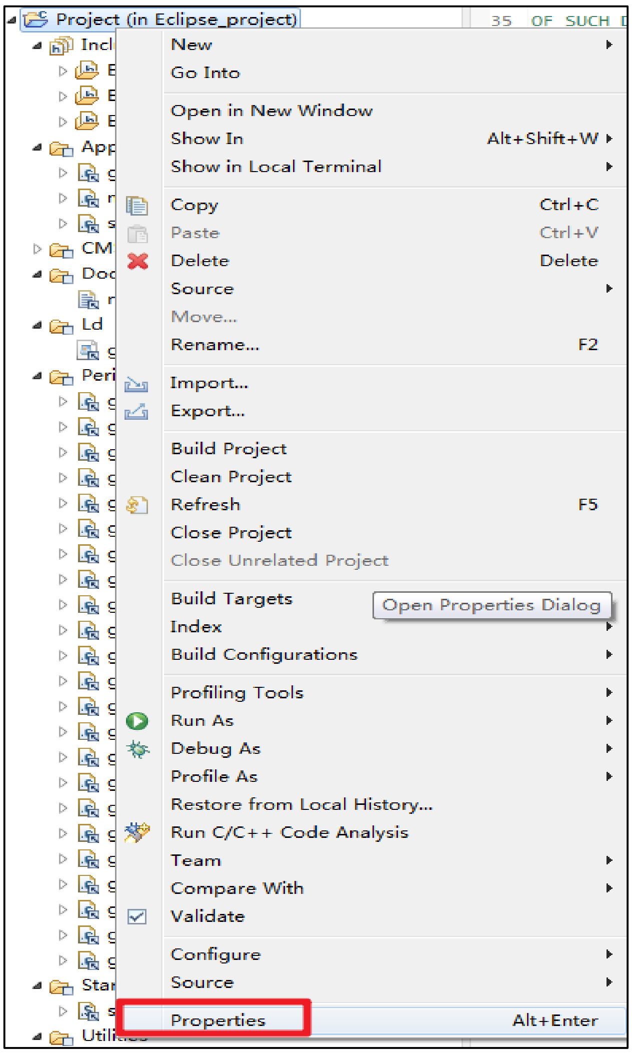


Note: The files and folders created in the "Refresh" method are all real, and once a file is deleted in the Eclipse IDE, the file will be deleted from the disk directly.

2.3. Project configurations

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

Figure 2-22. Project properties configurations

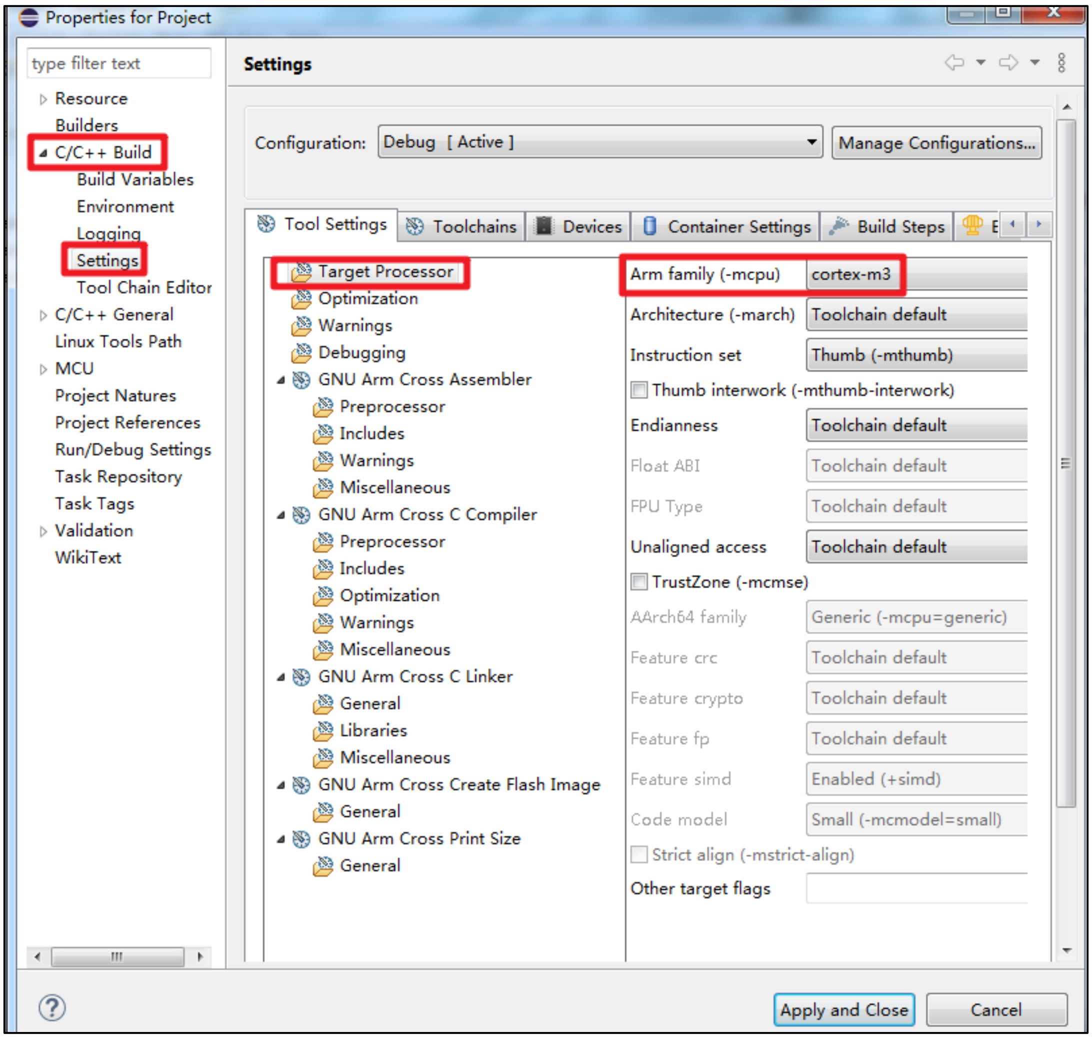


2.3.1. Target Processor option configuration

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

According to the core of the target chip, select cortex-m3, cortex-m4, cortex-m23 or cortex-m33. In this guide, select cortex-m3.

Figure 2-23. Target Processor configuration



Properties for Project

Settings

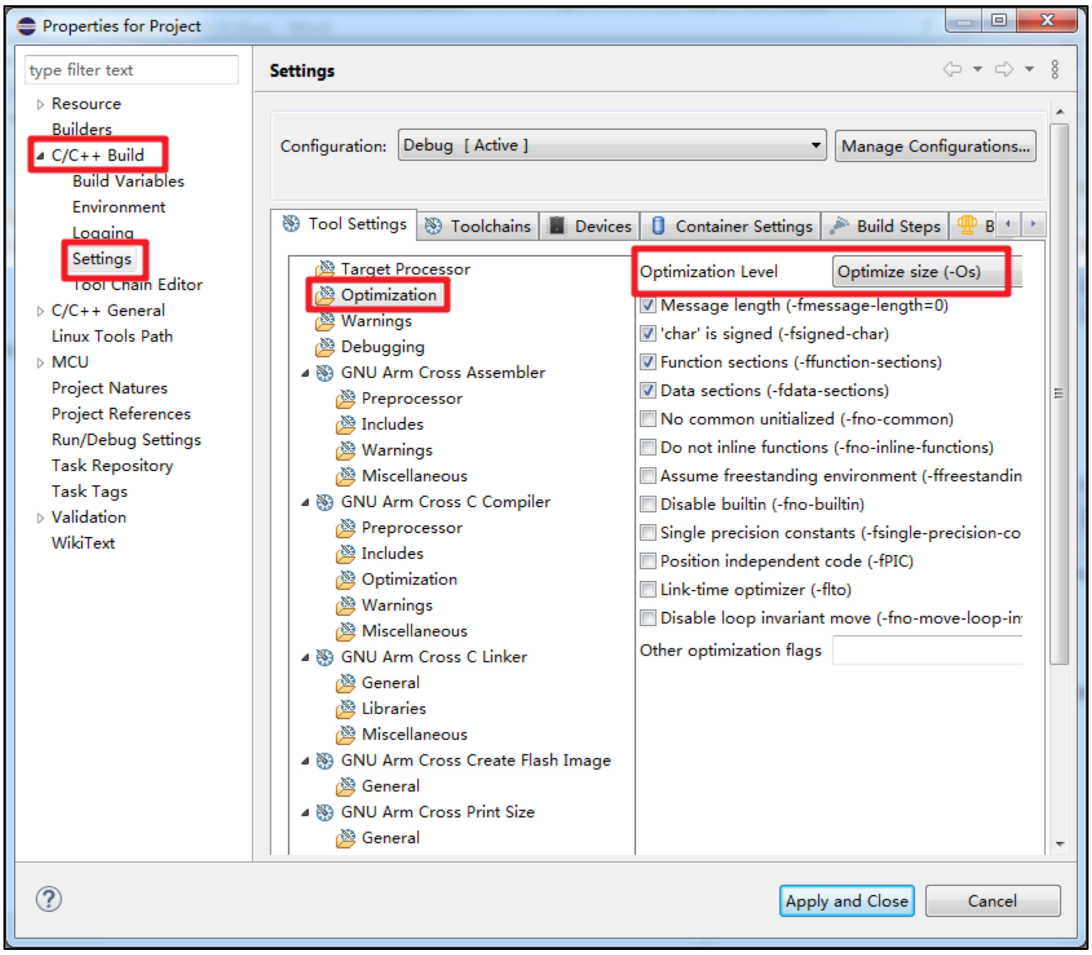
Configuration: Debug [Active]

Arm family (-mcpu)	cortex-m3
Architecture (-march)	Toolchain default
Instruction set	Thumb (-mthumb)
Thumb interwork (-mthumb-interwork)	☐
Endianness	Toolchain default
Float ABI	Toolchain default
FPU Type	Toolchain default
Unaligned access	Toolchain default
TrustZone (-mcmse)	☐
AArch64 family	Generic (-mcpu=generic)
Feature crc	Toolchain default
Feature crypto	Toolchain default
Feature fp	Toolchain default
Feature simd	Enabled (+simd)
Code model	Small (-mcmodel=small)
Strict align (-mstrict-align)	☐
Other target flags	

2.3.2. 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 2-24. 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 (-ffreestandin
- [ ] Disable builtin (-fno-builtin)
- [ ] Single precision constants (-fsingle-precision-co
- [ ] Position independent code (-fPIC)
- [ ] Link-time optimizer (-flto)
- [ ] Disable loop invariant move (-fno-move-loop-in

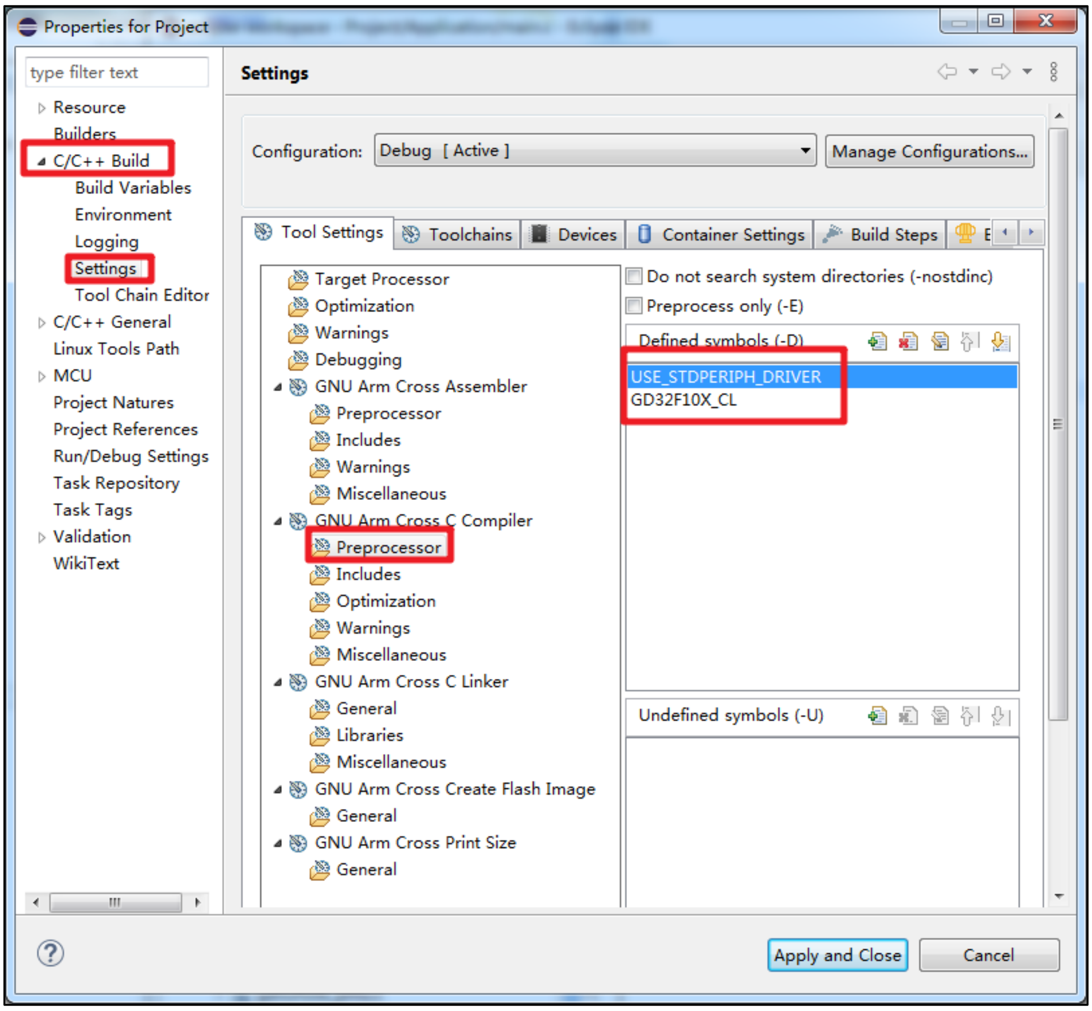
Other optimization flags

2.3.3. GNU Arm Cross C Compiler configuration

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

In this guide, add USE_STDPERIPH_DRIVER and GD32F10X_CL pre-compiled macros in the ‘’Preprocessor->Defined symbols’ option.

Figure 2-25. GNU Arm Cross C Compiler -> Preprocessor configuration



Properties for Project

Settings

Configuration: Debug [Active]

Tool Settings → GNU Arm Cross C Compiler → Preprocessor

Do not search system directories (-nostdinc)

Preprocess only (-E)

Defined symbols (-D)

USE_STDPERIPH_DRIVER
GD32F10X_CL

Undefined symbols (-U)

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

"${ProjDirPath}/../../Firmware/CMSIS/GD/GD32F10x/Include"

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

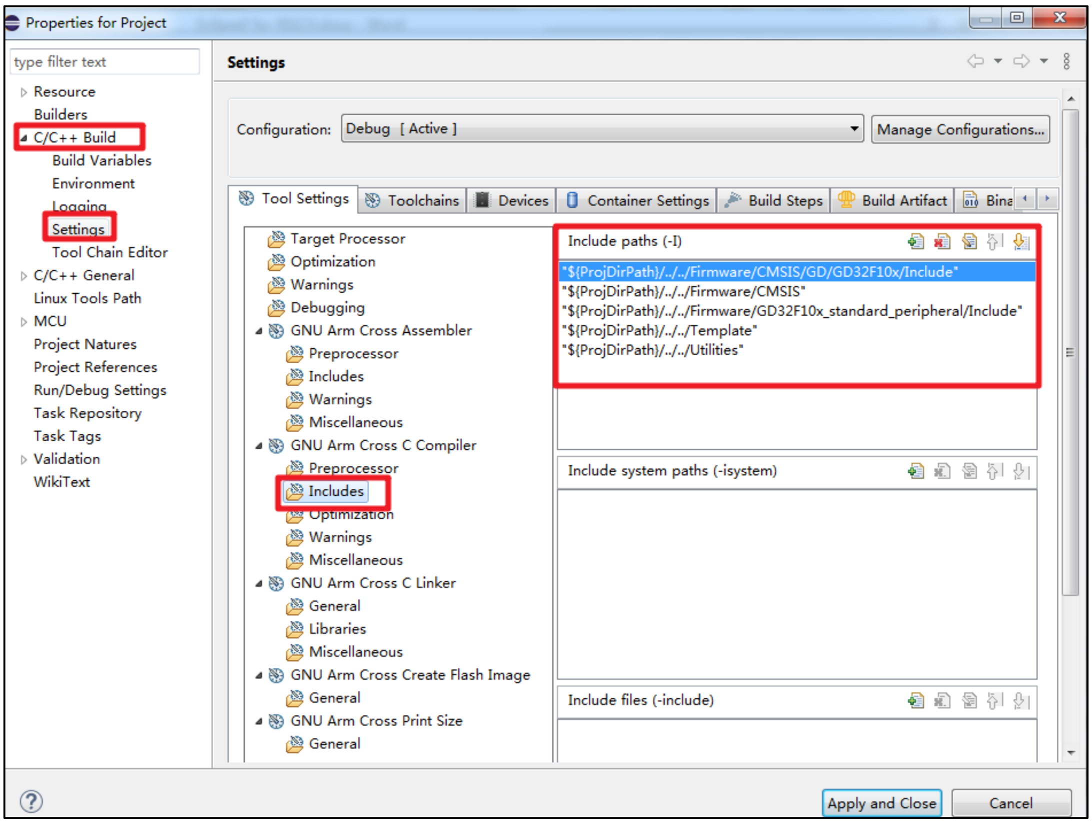
"${ProjDirPath}/../../Firmware/GD32F10x_standard_peripheral/Include"

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

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

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

Figure 2-26. GNU Arm Cross C Compiler -> Includes configuration


Properties for Project

type filter text

- Resource
- Builders
- C/C++ Build
  - Build Variables
  - Environment
  - Logging
  - Settings
  - Tool Chain Editor
- C/C++ General
- Linux Tools Path
- MCU
- Project Natures
- Project References
- Run/Debug Settings
- Task Repository
- Task Tags
- Validation
- WikiText

Settings

Configuration: Debug [Active] | Manage Configurations...

Tool Settings | Toolchains | Devices | Container Settings | Build Steps | Build Artifact | Bina

- Target Processor
- Optimization
- Warnings
- Debugging
- GNU Arm Cross Assembler
  - Preprocessor
  - Includes
  - Warnings
  - Miscellaneous
- GNU Arm Cross C Compiler
  - Preprocessor
  - Includes
  - Optimization
  - Warnings
  - Miscellaneous
- GNU Arm Cross C Linker
  - General
  - Libraries
  - Miscellaneous
- GNU Arm Cross Create Flash Image
  - General
- GNU Arm Cross Print Size
  - General

Include paths (-I)

```
"${ProjDirPath}/../../Firmware/CMSIS/GD/GD32F10x/Include"
"${ProjDirPath}/../../Firmware/CMSIS"
"${ProjDirPath}/../../Firmware/GD32F10x_standard_peripheral/Include"
"${ProjDirPath}/../../Template"
"${ProjDirPath}/../../Utilities"
```

Include system paths (-isystem)

Include files (-include)

Apply and Close | Cancel

2.3.4. GNU Arm Cross C Linker configuration

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

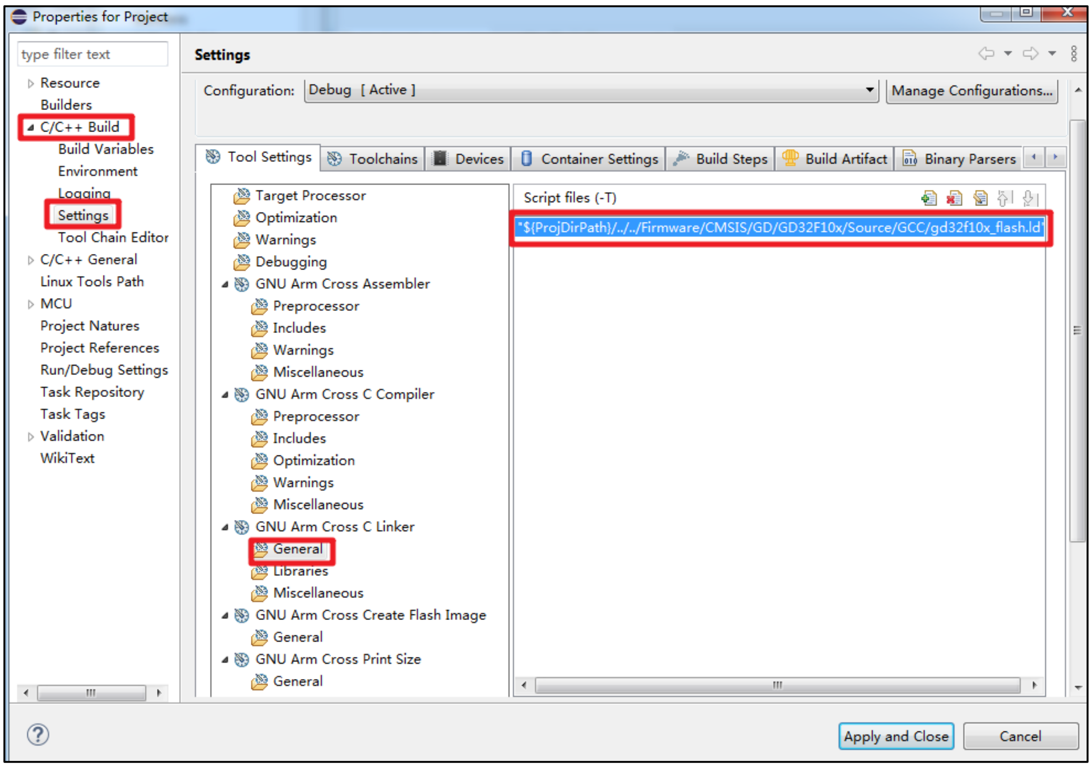
Add in the "General ->Script files" option:

"${ProjDirPath}/../../Firmware/CMSIS/GD/GD32F10x/Source/GCC/gd32f10x_flash.ld"

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.

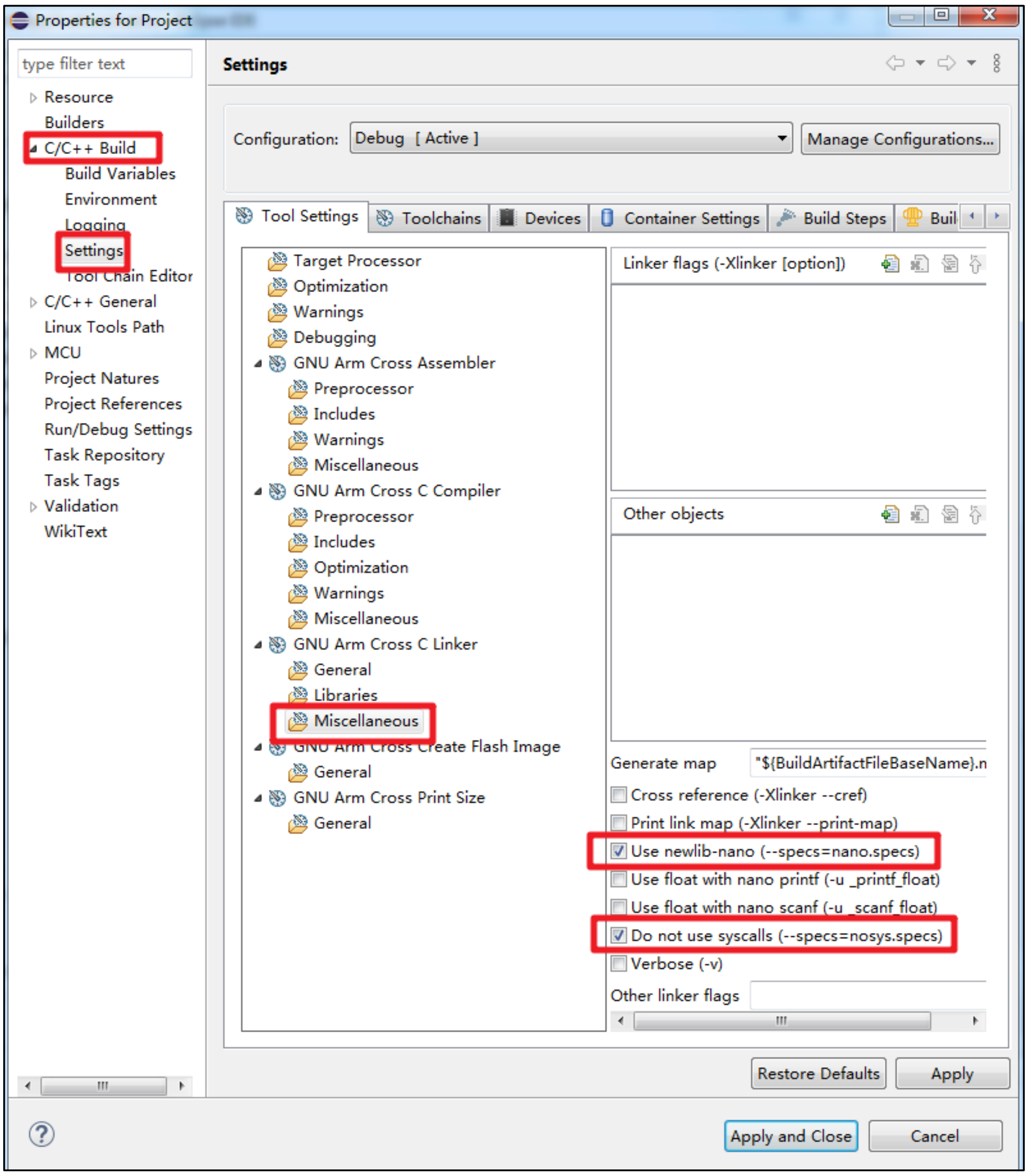
Figure 2-27. GNU Arm Cross C Linker -> General configuration



Script files (-T): "${ProjDirPath}/../../Firmware/CMSIS/GD/GD32F10x/Source/GCC/gd32f10x_flash.ld"

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

Figure 2-28. GNU Arm Cross C Linker -> Miscellaneous configuration



Configuration: Debug [Active]

Generate map "${BuildArtifactFileBaseName}.n

- [ ] Cross reference (-Xlinker --cref)
- [ ] Print link map (-Xlinker --print-map)
- [x] Use newlib-nano (--specs=nano.specs)
- [ ] Use float with nano printf (-u _printf_float)
- [ ] Use float with nano scanf (-u _scanf_float)
- [x] Do not use syscalls (--specs=nosys.specs)
- [ ] Verbose (-v)

Other linker flags

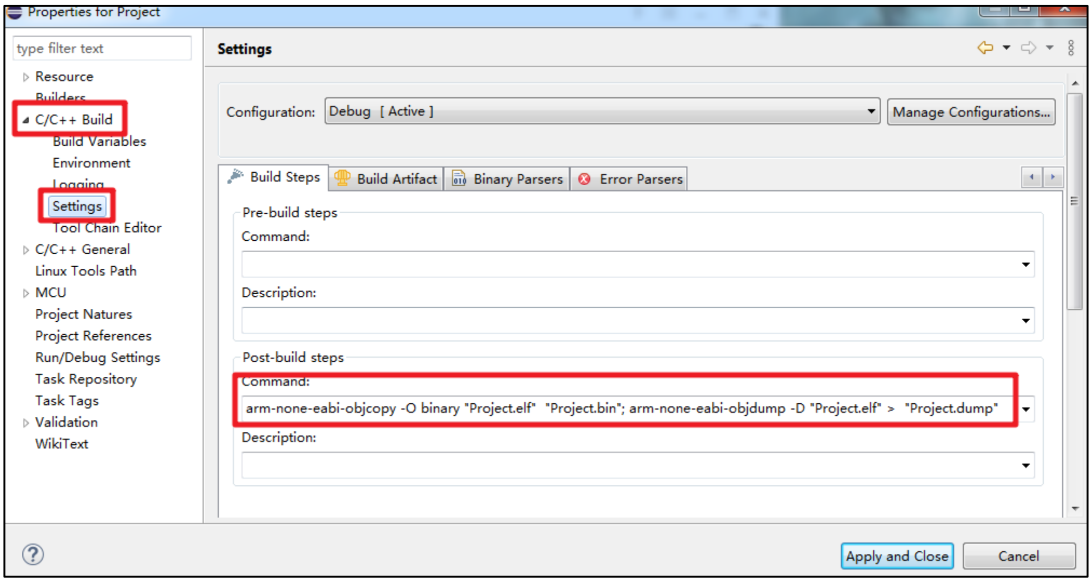
2.3.5. Build Steps configuration-generate bin file

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

Add in this guide:

arm-none-eabi-objcopy -O binary "Project.elf" "Project.bin"; arm-none-eabi-objdump -D "Project.elf"> "Project.dump"

Figure 2-29. Build Steps configuration



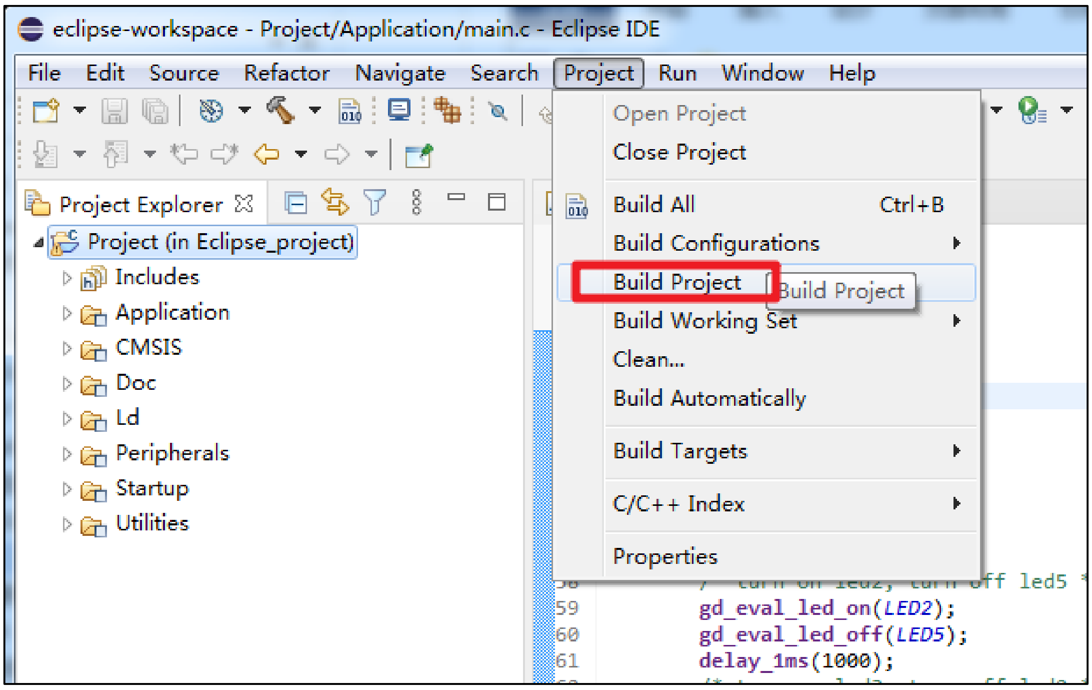
```
arm-none-eabi-objcopy -O binary "Project.elf"  "Project.bin"; arm-none-eabi-objdump -D "Project.elf" >  "Project.dump"
```

2.4. 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 2-30. Build project

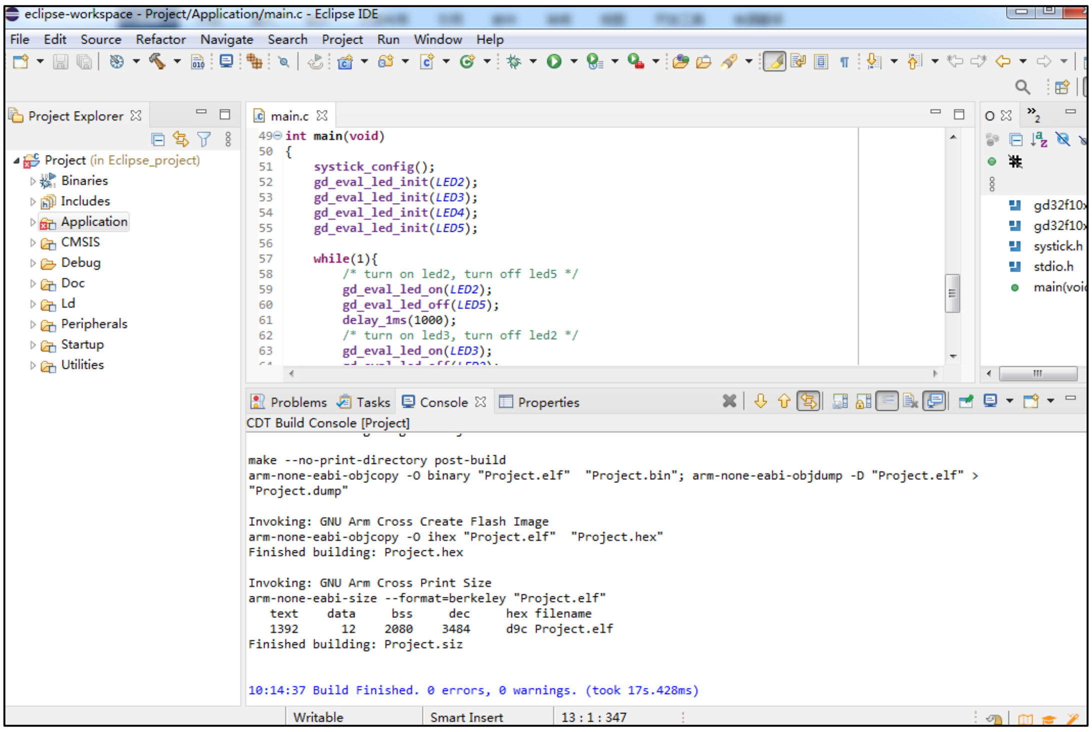


```
        gd_eval_led_on(LED2);
        gd_eval_led_off(LED5);
        delay_1ms(1000);
```

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 2-31. Build ARM project completed


eclipse-workspace - Project/Application/main.c - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
Project (in Eclipse_project)
Binaries
Includes
Application
CMSIS
Debug
Doc
Ld
Peripherals
Startup
Utilities
main.c
gd32f10x
gd32f10x
systick.h
stdio.h
main(void)
Problems Tasks Console Properties
CDT Build Console [Project]
Writable
Smart Insert
13 : 1 : 347

```
int main(void)
{
    systick_config();
    gd_eval_led_init(LED2);
    gd_eval_led_init(LED3);
    gd_eval_led_init(LED4);
    gd_eval_led_init(LED5);

    while(1){
        /* turn on led2, turn off led5 */
        gd_eval_led_on(LED2);
        gd_eval_led_off(LED5);
        delay_1ms(1000);
        /* turn on led3, turn off led2 */
        gd_eval_led_on(LED3);
```

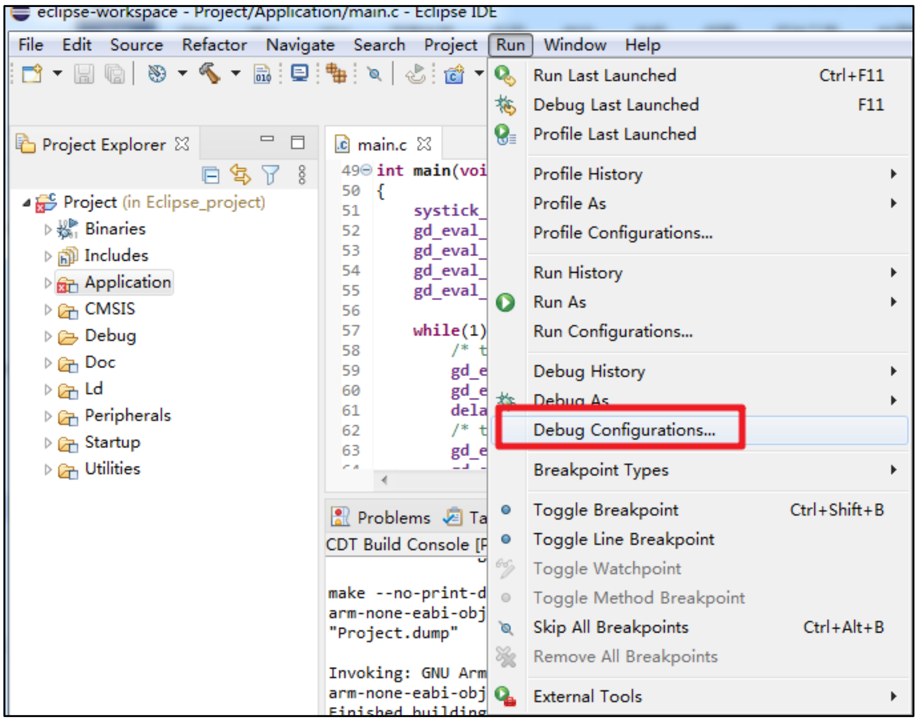
```
make --no-print-directory post-build
arm-none-eabi-objcopy -O binary "Project.elf"  "Project.bin"; arm-none-eabi-objdump -D "Project.elf" >
"Project.dump"

Invoking: GNU Arm Cross Create Flash Image
arm-none-eabi-objcopy -O ihex "Project.elf"  "Project.hex"
Finished building: Project.hex

Invoking: GNU Arm Cross Print Size
arm-none-eabi-size --format=berkeley "Project.elf"
   text    data     bss     dec     hex filename
   1392      12    2080    3484     d9c Project.elf
Finished building: Project.siz


10:14:37 Build Finished. 0 errors, 0 warnings. (took 17s.428ms)
```

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

2.5.1. Debug configuration interface

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

Figure 2-32. Enter Debug Configuratios interface



```
int main(voi
{
    systick_
    gd_eval_
    gd_eval_
    gd_eval_
    gd_eval_

    while(1)
        /* t
        gd_e
        gd_e
        dela
        /* t
        gd_e
```

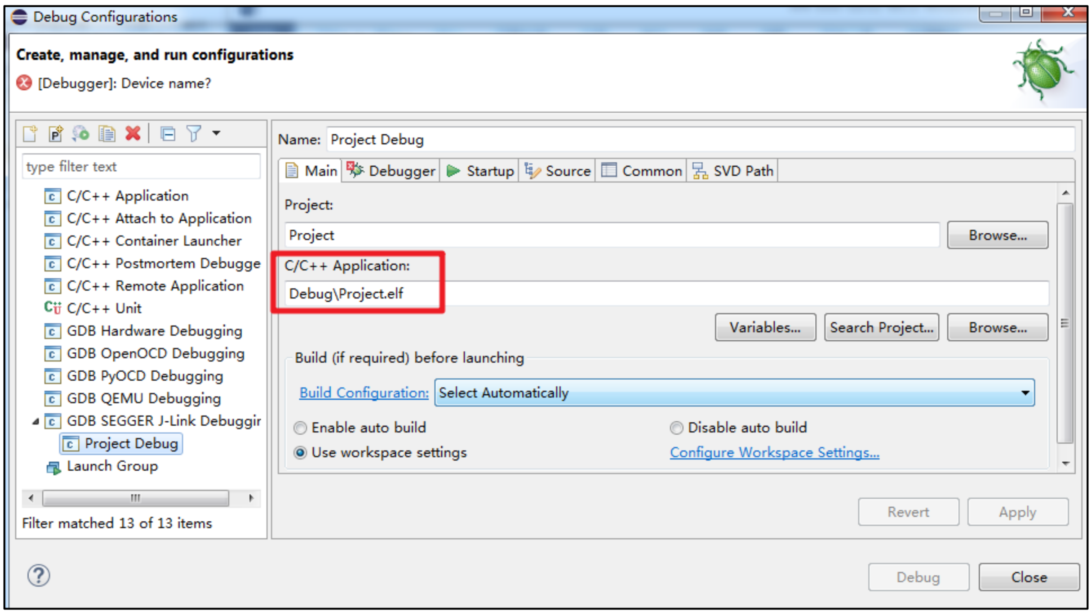
```
make --no-print-d
arm-none-eabi-obj
"Project.dump"

Invoking: GNU Arm
arm-none-eabi-obj
```

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.

2.5.2. Main tab

Figure 2-33. GDB SEGGER J-Link Debugging-Main tab



[Debugger]: Device name?

Name: Project Debug

Project: Project

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

Build Configuration: Select Automatically

Enable auto build Disable auto build [✓ 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.

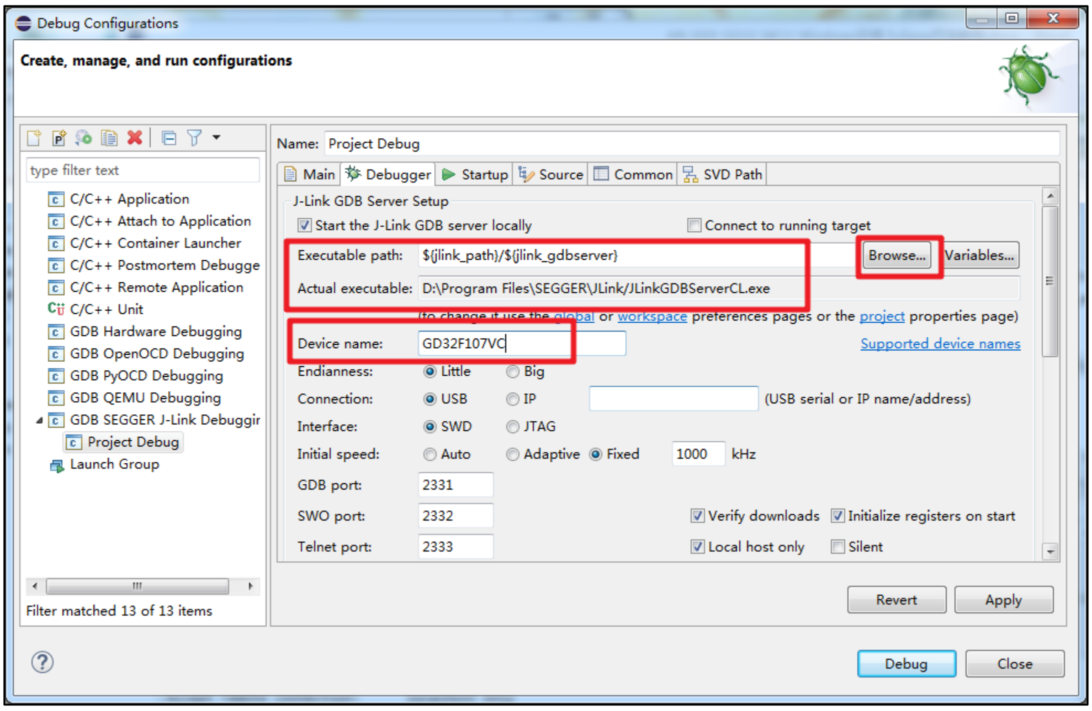
2.5.3. Debugger tab

In the “Debugger” tab, fill in the device name of the target chip model, which is GD32F107VC 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.

Figure 2-34. 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 Debugge
- C/C++ Remote Application
- C/C++ Unit
- GDB Hardware Debugging
- GDB OpenOCD Debugging
- GDB PyOCD Debugging
- GDB QEMU Debugging
- GDB SEGGER J-Link Debuggir
  - Project Debug
- Launch Group

Filter matched 13 of 13 items

Name: Project 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...   Variables...

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

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

Device name: GD32F107VC   Supported device names

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 start

Telnet port: 2333   [✓] Local host only   [ ] Silent

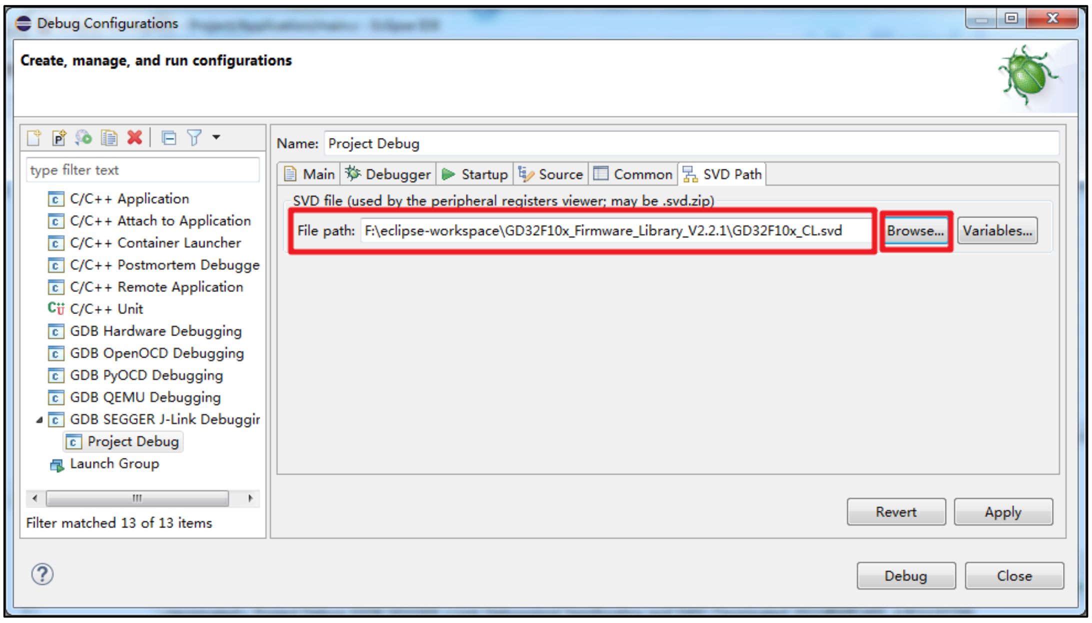
Revert   Apply

Debug   Close

2.5.4. SVD Path tab

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

Figure 2-35. GDB SEGGER J-Link Debugging-SVD Path tab



Name: Project Debug

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

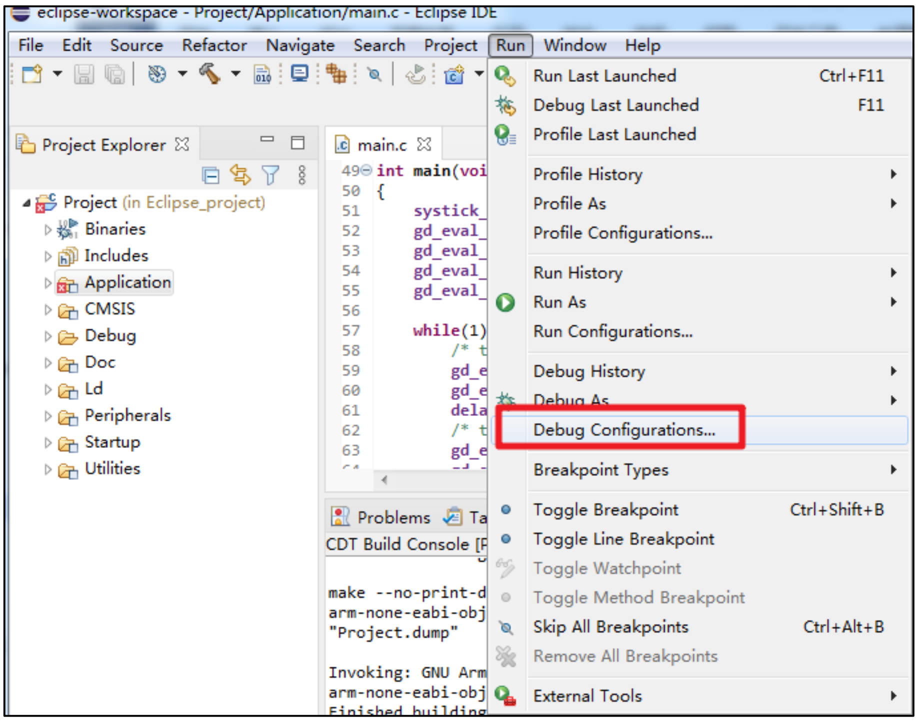
File path: F:\eclipse-workspace\GD32F10x_Firmware_Library_V2.2.1\GD32F10x_CL.svd

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

2.6.1. Debug configuration interface

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

Figure 2-36. Enter Debug Configuratios interface



```
int main(voi
{
    systick_
    gd_eval_
    gd_eval_
    gd_eval_
    gd_eval_

    while(1)
        /* t
        gd_e
        gd_e
        dela
        /* t
        gd_e
```

```
make --no-print-d
arm-none-eabi-obj
"Project.dump"

Invoking: GNU Arm
arm-none-eabi-obj
Finished building
```

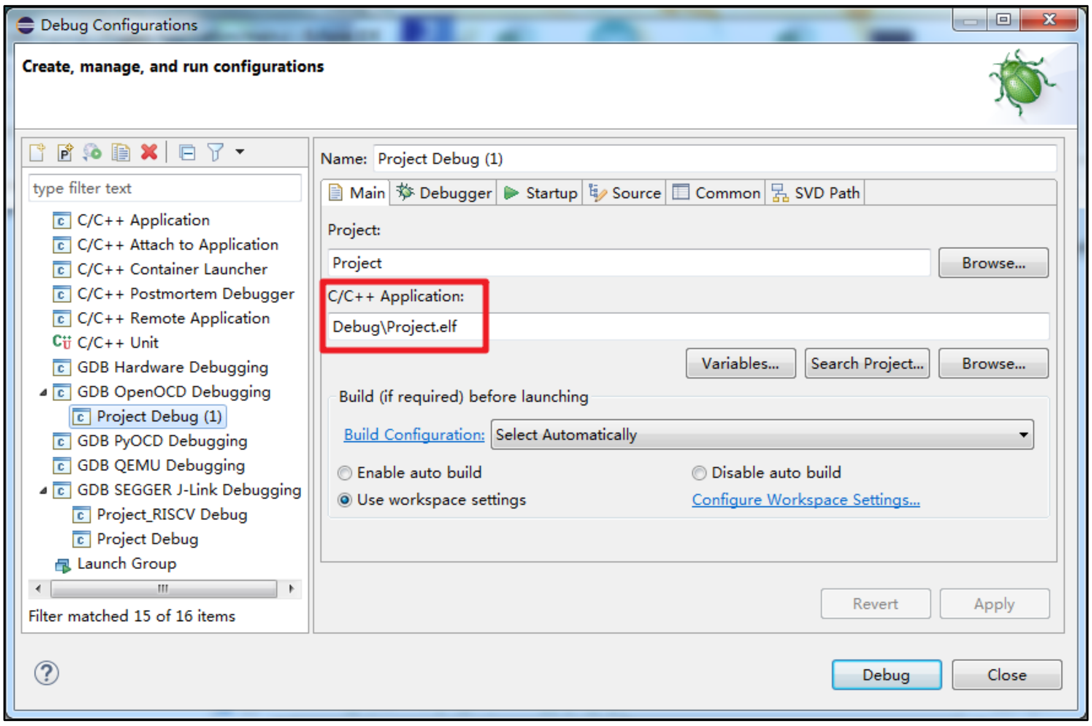
Use OpenOCD as the GDB Server, and use the GDB tool in the GCC tool chain as the GDB

Client.

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

2.6.2. Main tab

Figure 2-37. GDB OpenOCD Debugging-Main 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 Debugger
- C/C++ Remote Application
- C/C++ Unit
- GDB Hardware Debugging
- GDB OpenOCD Debugging
  - Project Debug (1)
- GDB PyOCD Debugging
- GDB QEMU Debugging
- GDB SEGGER J-Link Debugging
  - Project_RISCV Debug
  - Project Debug
- Launch Group

Filter matched 15 of 16 items

Name: Project Debug (1)

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

Project:

Project

C/C++ Application:

Debug\Project.elf

Build (if required) before launching

Build Configuration: Select Automatically

Enable auto build / Disable auto build

[✓ Use workspace settings] Configure 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.

2.6.3. Debugger tab

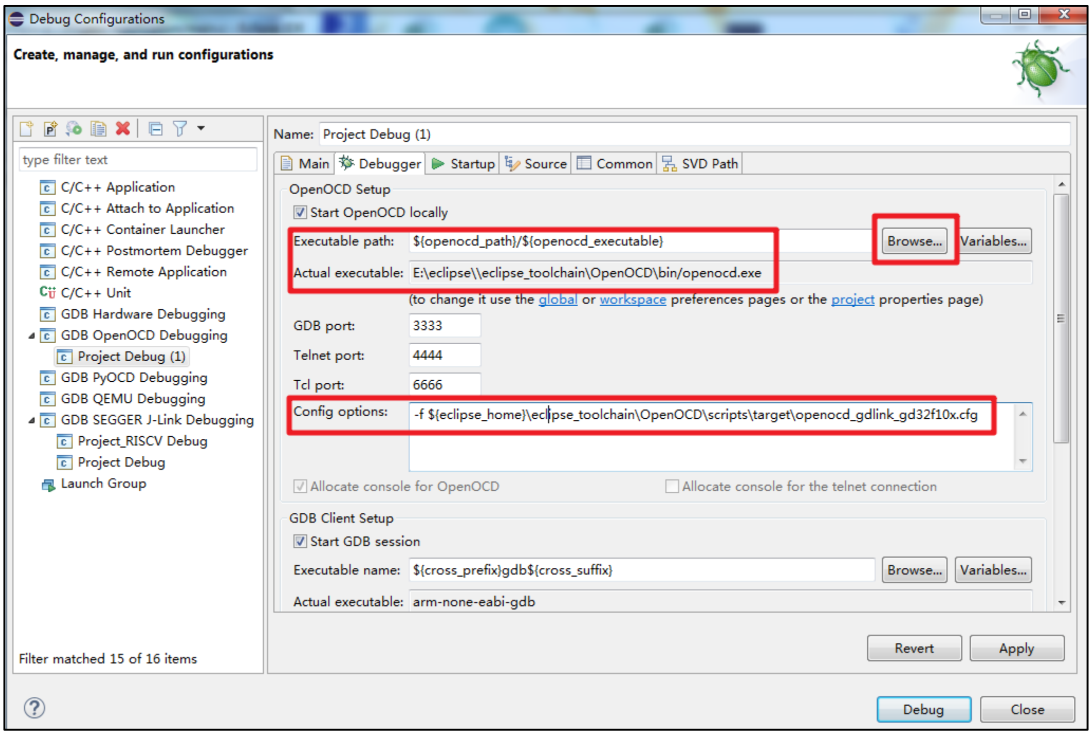
If the OpenOCD path has been configured correctly when setting up the Eclipse environment, it will be recognized automatically here. If user have not configured it correctly before, user can also select the absolute path of OpenOCD in the “Executable path” column.

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

-f ${eclipse_home}\eclipse_toolchain\OpenOCD\scripts\target\openocd_gdlink_gd32f10x.cfg

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

Figure 2-38. GDB OpenOCD Debugging-Debugger tab



Name: Project Debug (1)

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_gd32f10x.cfg

Allocate console for OpenOCD

Start GDB session

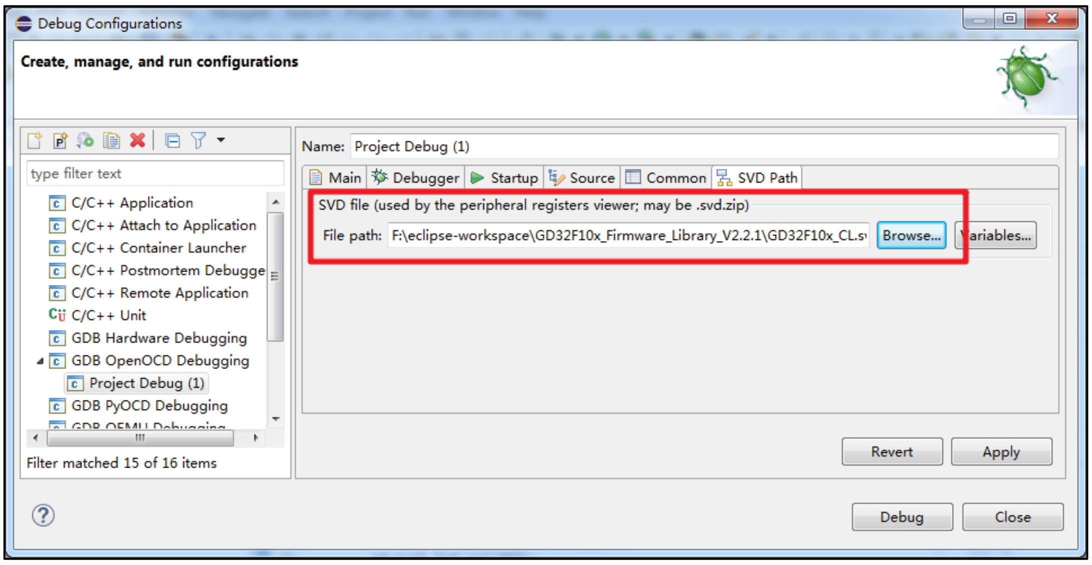
Executable name: ${cross_prefix}gdb${cross_suffix}

Actual executable: arm-none-eabi-gdb

2.6.4. SVD Path tab

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

Figure 2-39. GDB OpenOCD Debugging-SVD Path tab



Name: Project Debug (1)

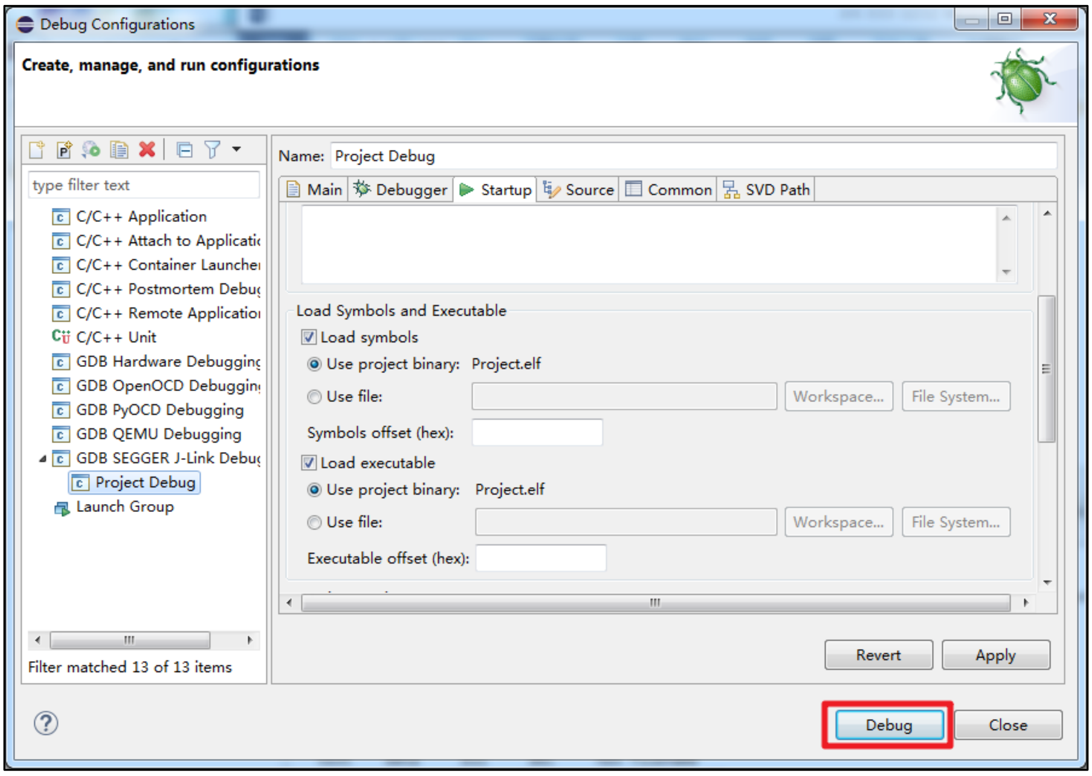
SVD file (used by the peripheral registers viewer; may be .svd.zip)

File path: F:\eclipse-workspace\GD32F10x_Firmware_Library_V2.2.1\GD32F10x_CL.s

2.7. Debug interface

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

Figure 2-40. Enter Debug perspective -1


Debug Configurations

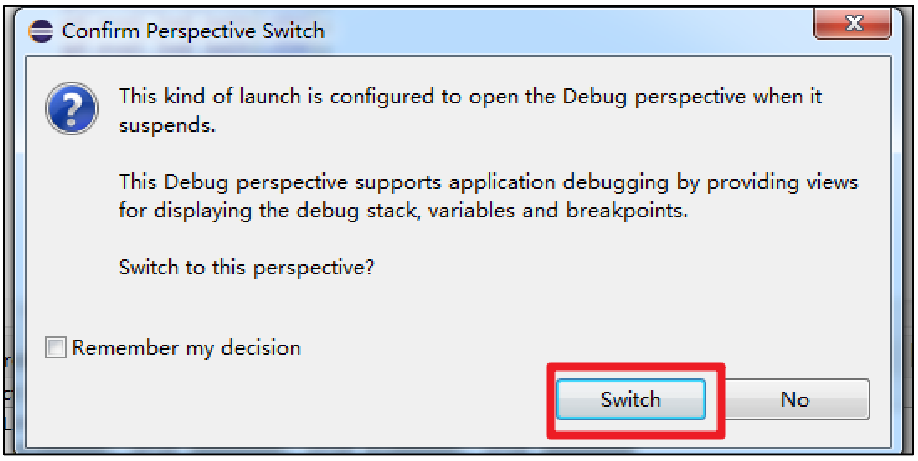
Create, manage, and run configurations

Name: Project Debug

Load Symbols and Executable
☑ Load symbols
● Use project binary: Project.elf
○ Use file:
Symbols offset (hex):
☑ Load executable
● Use project binary: Project.elf
○ Use file:
Executable offset (hex):

Switch to Debug perspective.

Figure 2-41. Enter Debug perspective -2


Confirm Perspective Switch

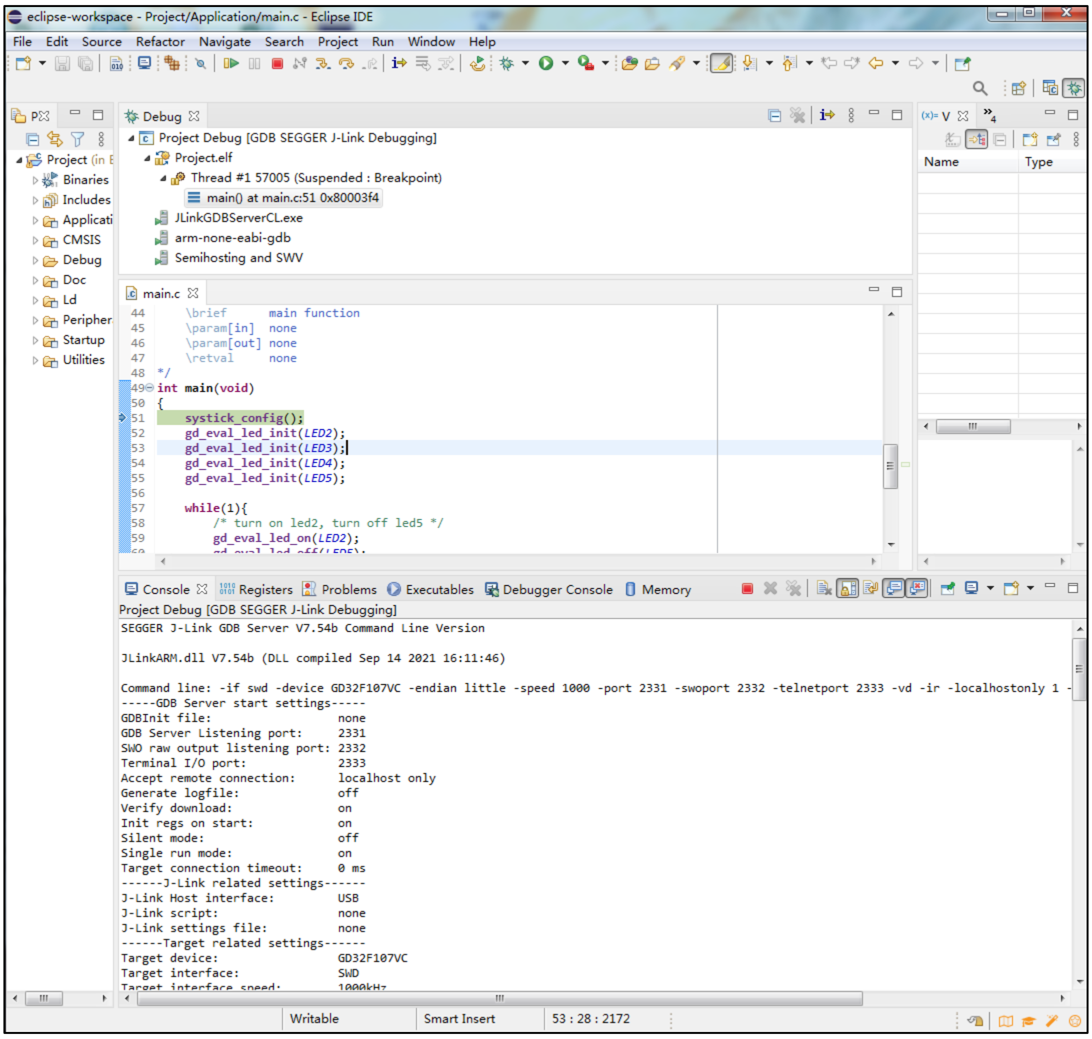
This kind of launch is configured to open the Debug perspective when it suspends.

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 2-42. Debug perspective



```
Project Debug [GDB SEGGER J-Link Debugging]
  Project.elf
    Thread #1 57005 (Suspended : Breakpoint)
      main() at main.c:51 0x80003f4
  JLinkGDBServerCL.exe
  arm-none-eabi-gdb
  Semihosting and SWV
```

```
    \brief      main function
    \param[in]  none
    \param[out] none
    \retval     none
*/
int main(void)
{
    systick_config();
    gd_eval_led_init(LED2);
    gd_eval_led_init(LED3);
    gd_eval_led_init(LED4);
    gd_eval_led_init(LED5);

    while(1){
        /* turn on led2, turn off led5 */
        gd_eval_led_on(LED2);
```

```
Project Debug [GDB SEGGER J-Link Debugging]
SEGGER J-Link GDB Server V7.54b Command Line Version

JLinkARM.dll V7.54b (DLL compiled Sep 14 2021 16:11:46)

Command line: -if swd -device GD32F107VC -endian little -speed 1000 -port 2331 -swoport 2332 -telnetport 2333 -vd -ir -localhostonly 1 -
-----GDB Server start settings-----
GDBInit file:                  none
GDB Server Listening port:     2331
SWO raw output listening port: 2332
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:                 GD32F107VC
Target interface:              SWD
Target interface speed:        1000kHz
```

2.7.1. Toolbar introduction

：resume

：suspend

：terminate

：step into

：step over

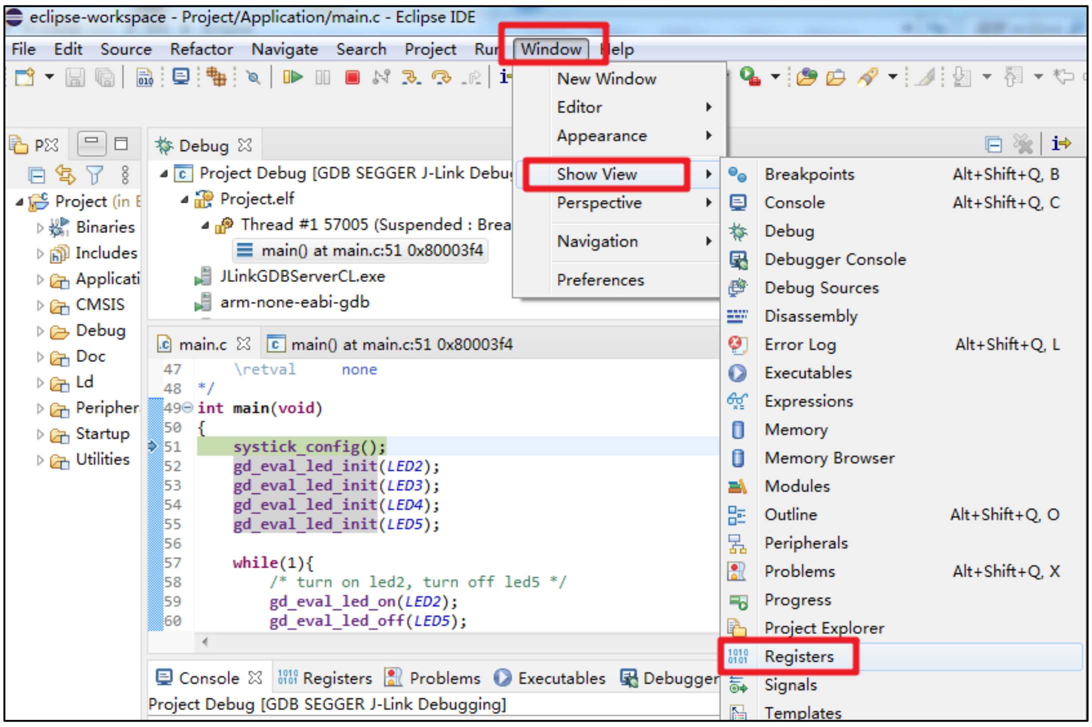
：step out

：reset

2.7.2. Registers view

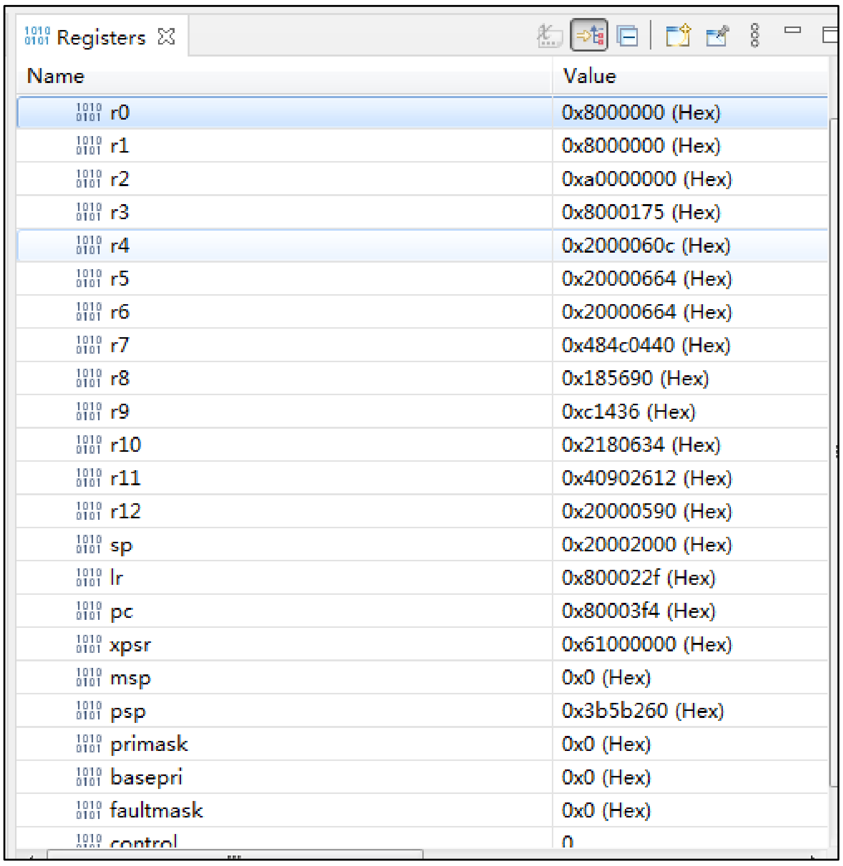
In the menu bar, select "Window->Show view->Registers" option, open it to view the value of general-purpose registers.

Figure 2-43. Open Registers view



```
    \retval     none
*/
int main(void)
{
    systick_config();
    gd_eval_led_init(LED2);
    gd_eval_led_init(LED3);
    gd_eval_led_init(LED4);
    gd_eval_led_init(LED5);

    while(1){
        /* turn on led2, turn off led5 */
        gd_eval_led_on(LED2);
        gd_eval_led_off(LED5);
```

Figure 2-44. Registers view



Registers

Name	Value
r0	0x8000000 (Hex)
r1	0x8000000 (Hex)
r2	0xa0000000 (Hex)
r3	0x8000175 (Hex)
r4	0x2000060c (Hex)
r5	0x20000664 (Hex)
r6	0x20000664 (Hex)
r7	0x484c0440 (Hex)
r8	0x185690 (Hex)
r9	0xc1436 (Hex)
r10	0x2180634 (Hex)
r11	0x40902612 (Hex)
r12	0x20000590 (Hex)
sp	0x20002000 (Hex)
lr	0x800022f (Hex)
pc	0x80003f4 (Hex)
xpsr	0x61000000 (Hex)
msp	0x0 (Hex)
psp	0x3b5b260 (Hex)
primask	0x0 (Hex)
basepri	0x0 (Hex)
faultmask	0x0 (Hex)
control	0

2.7.3. Peripherals view

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

Figure 2-45. Open Peripherals view

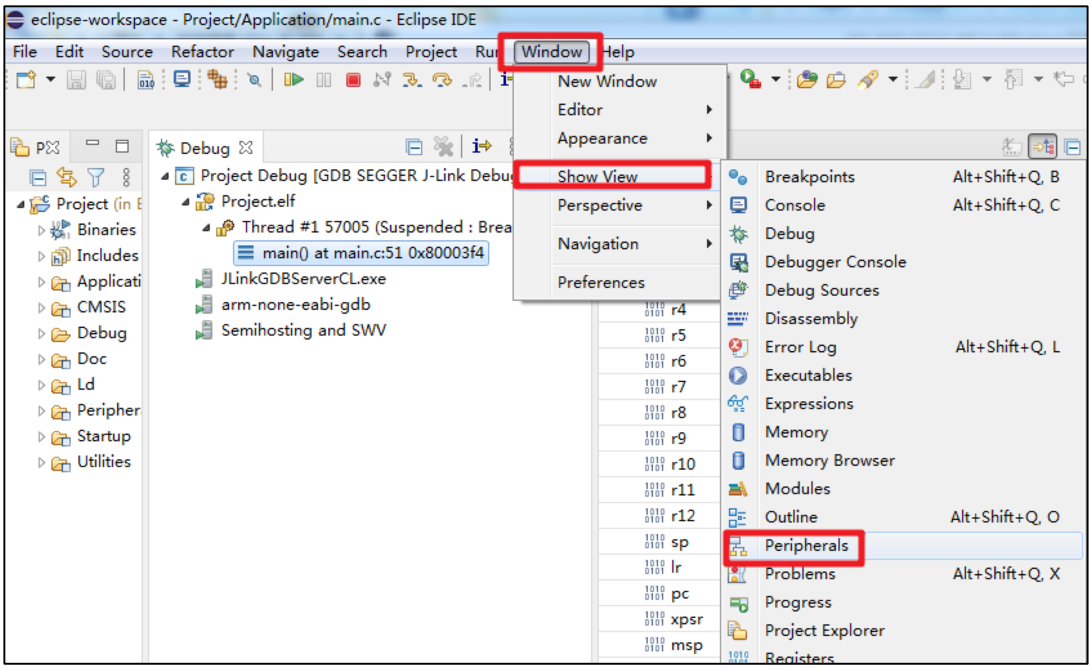
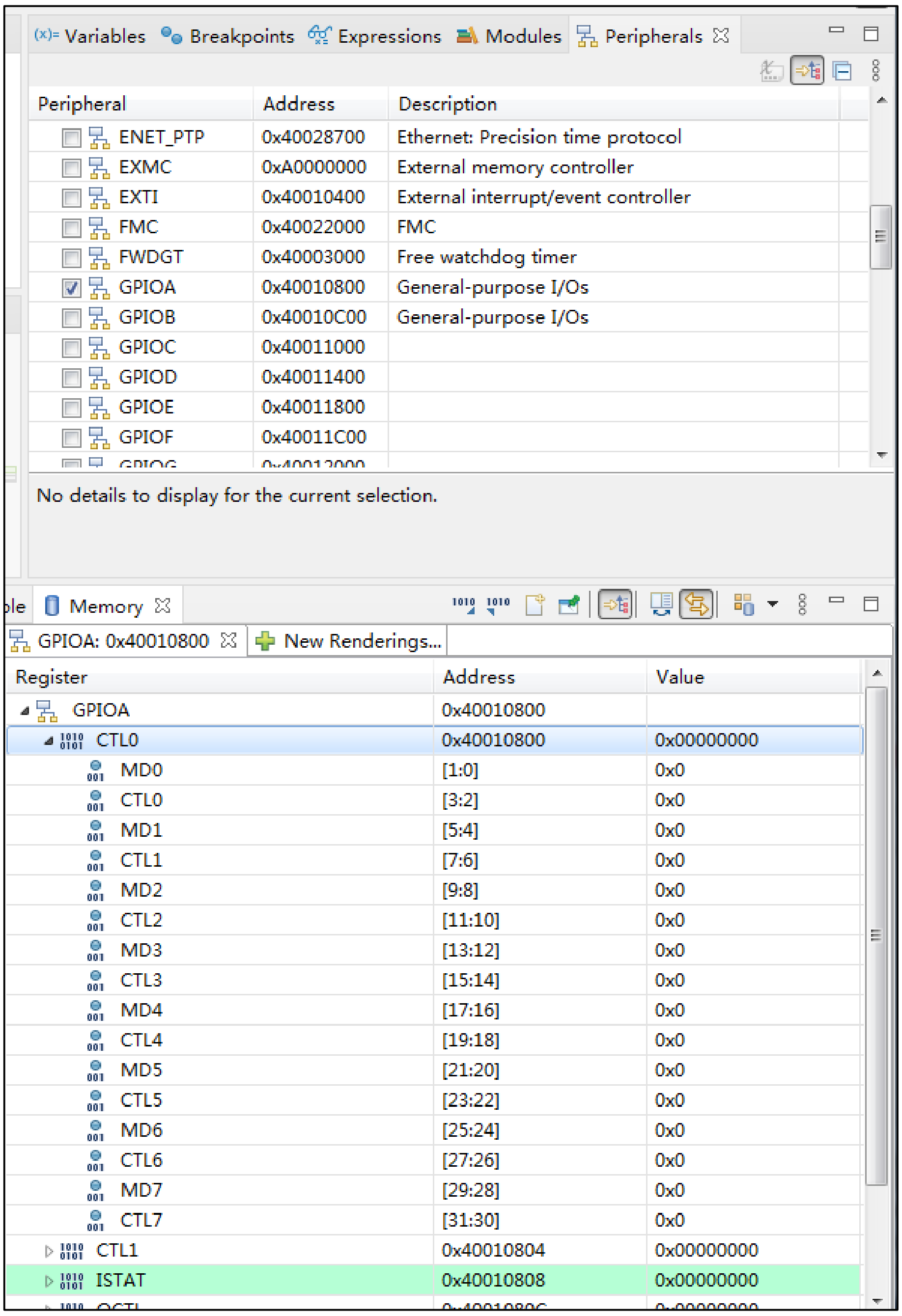


Figure 2-46. Peripherals view



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	

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

2.7.4. 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 2-47. Open Memory view

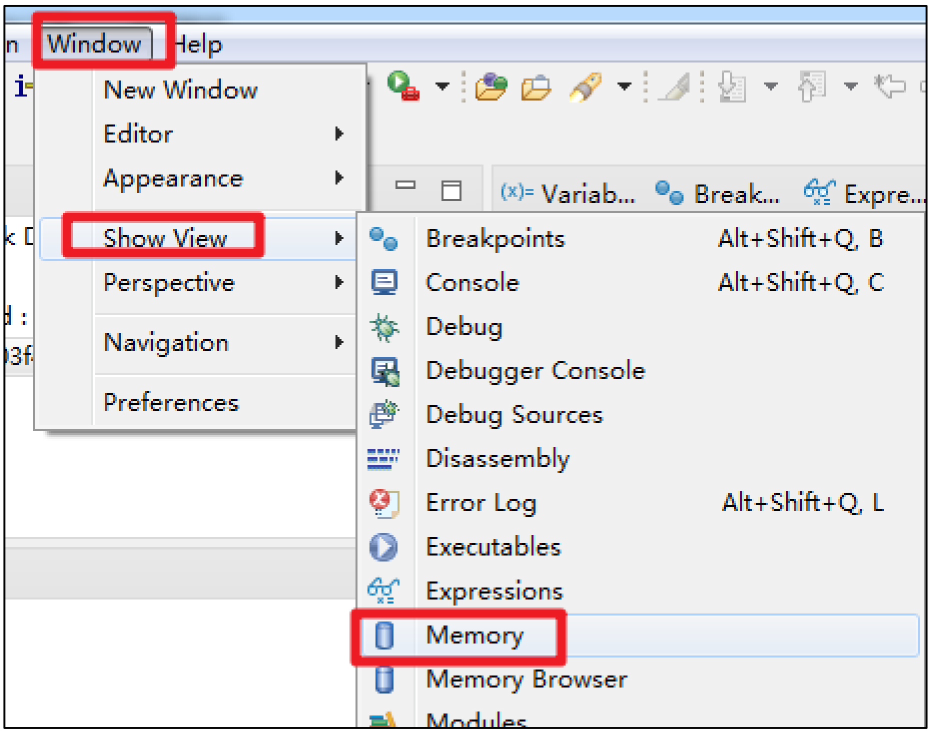
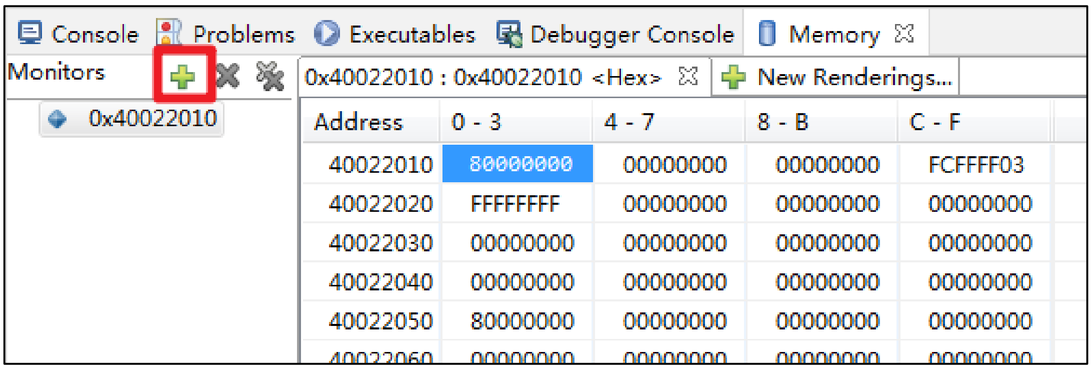


Figure 2-48. Memory view



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

2.7.5. 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 2-49. Open Expressions view

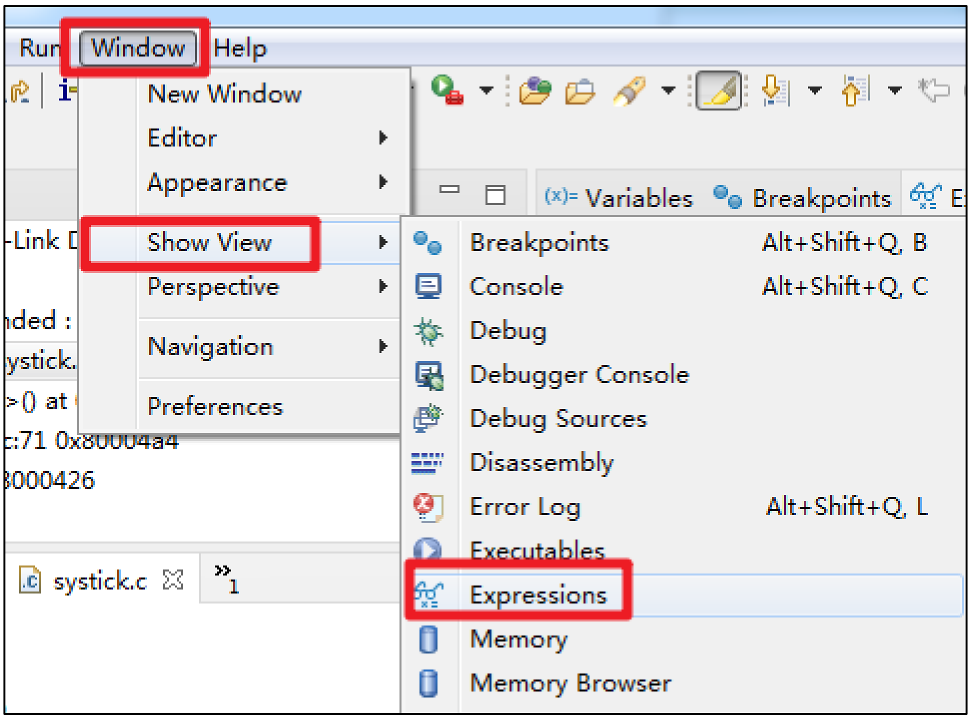
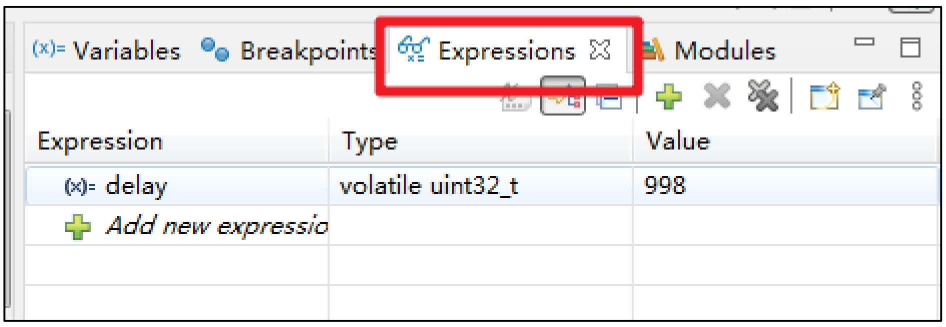


Figure 2-50. Expressions view



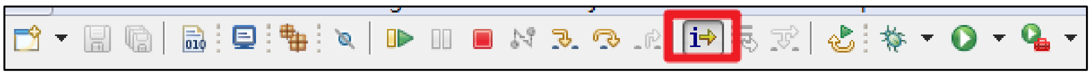
Expression	Type	Value
(x)= delay	volatile uint32_t	998
Add new expressio		

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.

2.7.6. Disassembly view

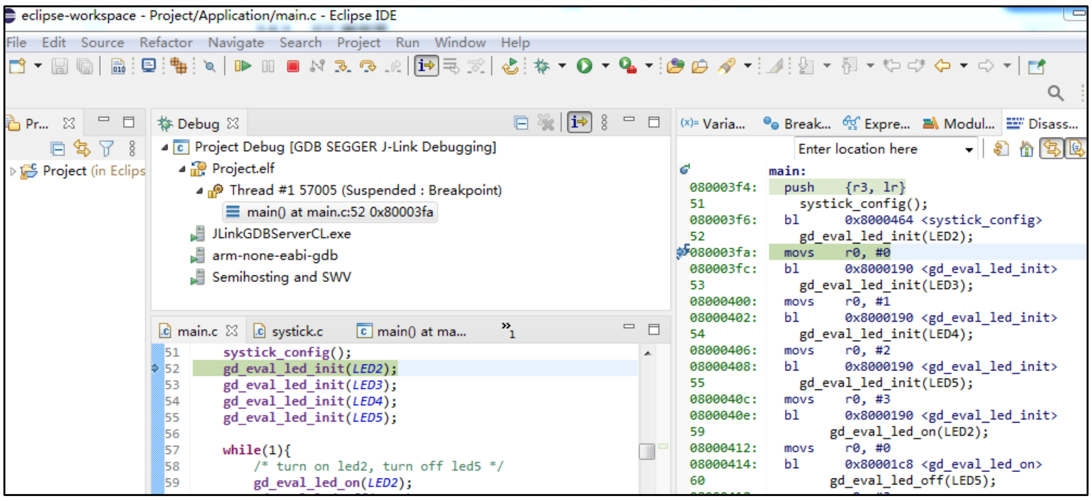
Select the “Instruction Stepping Mode button” in the debug toolbar to open the disassembly window.

Figure 2-51. Open Disassembly view



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

Figure 2-52. Disassembly view


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

```
Project Debug [GDB SEGGER J-Link Debugging]
  Project.elf
    Thread #1 57005 (Suspended : Breakpoint)
      main() at main.c:52 0x80003fa
  JLinkGDBServerCL.exe
  arm-none-eabi-gdb
  Semihosting and SWV
```

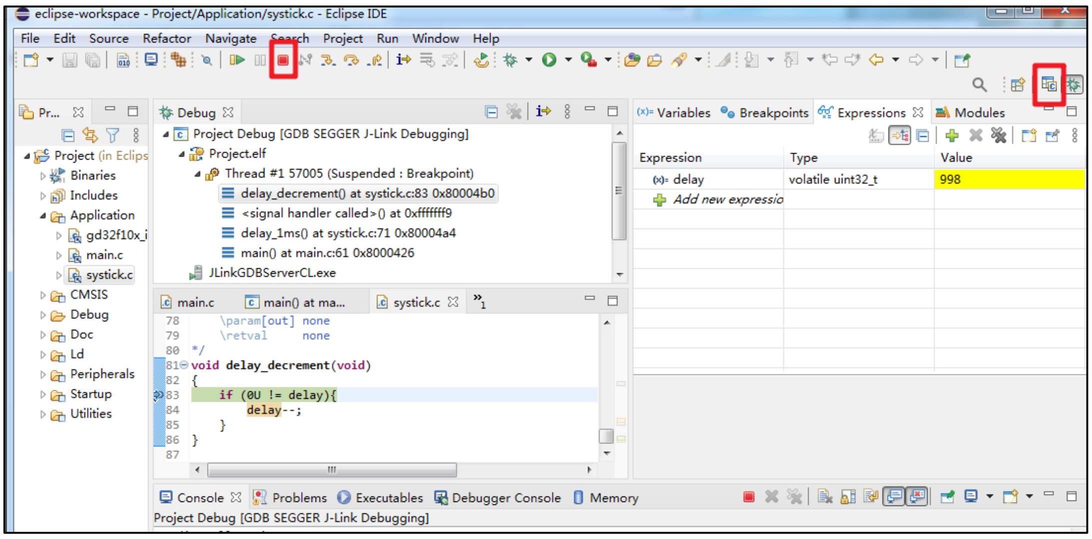
```
    systick_config();
    gd_eval_led_init(LED2);
    gd_eval_led_init(LED3);
    gd_eval_led_init(LED4);
    gd_eval_led_init(LED5);

    while(1){
        /* turn on led2, turn off led5 */
        gd_eval_led_on(LED2);
```

```
          main:
080003f4:   push    {r3, lr}
51            systick_config();
080003f6:   bl      0x8000464 <systick_config>
52            gd_eval_led_init(LED2);
080003fa:   movs    r0, #0
080003fc:   bl      0x8000190 <gd_eval_led_init>
53            gd_eval_led_init(LED3);
08000400:   movs    r0, #1
08000402:   bl      0x8000190 <gd_eval_led_init>
54            gd_eval_led_init(LED4);
08000406:   movs    r0, #2
08000408:   bl      0x8000190 <gd_eval_led_init>
55            gd_eval_led_init(LED5);
0800040c:   movs    r0, #3
0800040e:   bl      0x8000190 <gd_eval_led_init>
59                gd_eval_led_on(LED2);
08000412:   movs    r0, #0
08000414:   bl      0x80001c8 <gd_eval_led_on>
60                gd_eval_led_off(LED5);
```

2.7.7. Exit the Debug perspective

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

Figure 2-53. Exit the Debug perspective


eclipse-workspace - Project/Application/systick.c - Eclipse IDE

```
Project (in Eclipse)
  Binaries
  Includes
  Application
    gd32f10x_i
    main.c
    systick.c
  CMSIS
  Debug
  Doc
  Ld
  Peripherals
  Startup
  Utilities
```

```
Project Debug [GDB SEGGER J-Link Debugging]
  Project.elf
    Thread #1 57005 (Suspended : Breakpoint)
      delay_decrement() at systick.c:83 0x80004b0
      <signal handler called>() at 0xfffffff9
      delay_1ms() at systick.c:71 0x80004a4
      main() at main.c:61 0x8000426
  JLinkGDBServerCL.exe
```

```
    \param[out] none
    \retval     none
*/
void delay_decrement(void)
{
    if (0U != delay){
        delay--;
    }
}
```

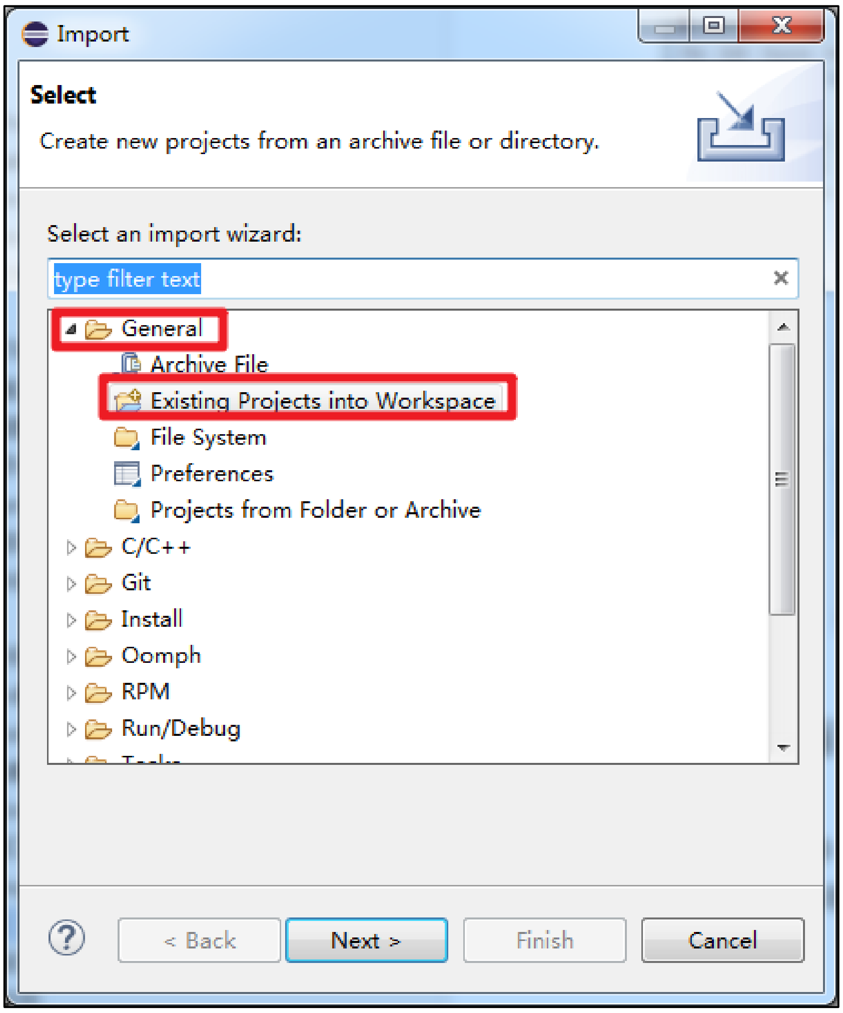
Expression	Type	Value
delay	volatile uint32_t	998
Add new expression		

Project Debug [GDB SEGGER J-Link Debugging]

3. Import an existing project

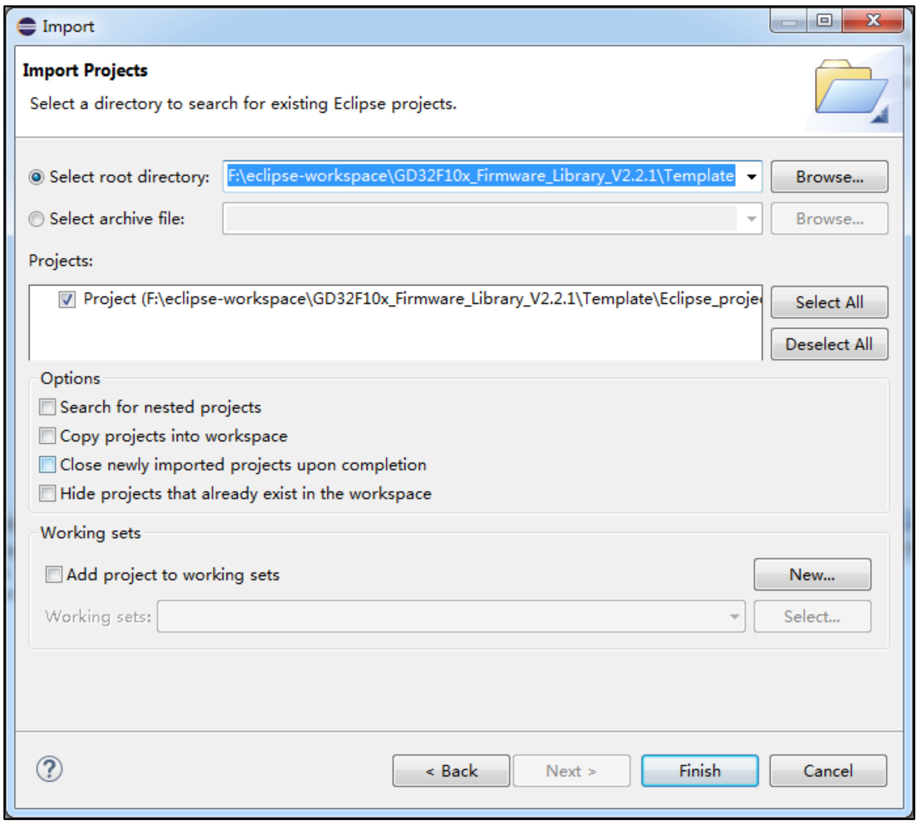
In addition to new projects, user can also import existing Eclipse projects directly. In the menu bar, click “File->Import”, select “General->Exisiting Projects into Workspace” to import the existing project, and click "Next".

Figure 3-1. 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 3-2. Import an existing project - 2



Import

Import Projects

Select a directory to search for existing Eclipse projects.

[✓ Select root directory:] F:\eclipse-workspace\GD32F10x_Firmware_Library_V2.2.1\Template

Select archive file:

Projects:

[✓] Project (F:\eclipse-workspace\GD32F10x_Firmware_Library_V2.2.1\Template\Eclipse_proje

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

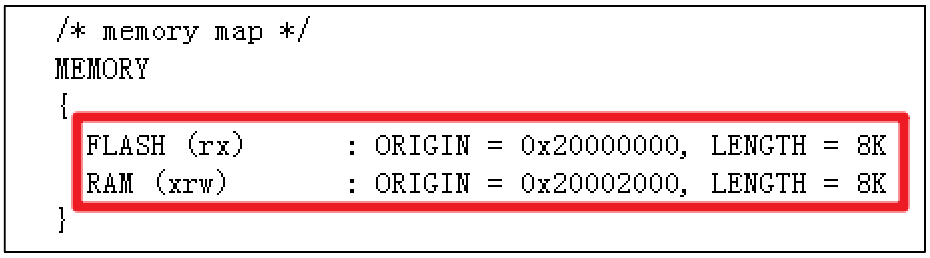
- [ ] Add project to working sets

Working sets:

4. Debug in RAM

Step 1: Modify the linker script, for example, as shown in ***Figure 4-1. Ld file memory map when debugging in RAM***.

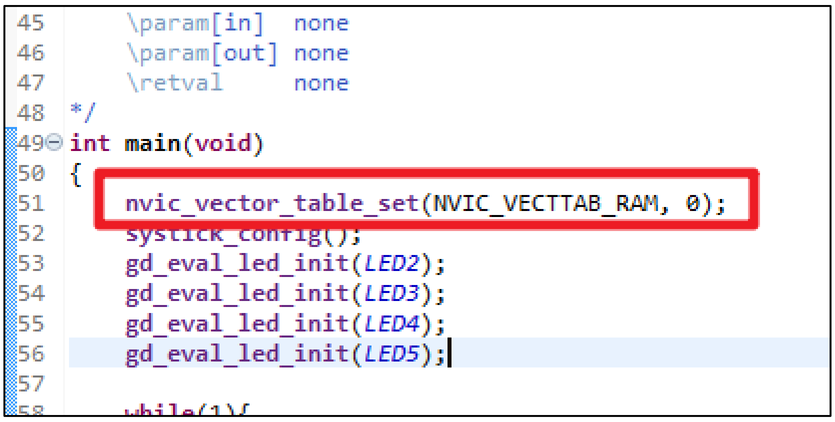
Figure 4-1. Ld file memory map when debugging in RAM



```
/* memory map */
MEMORY
{
  FLASH (rx)        : ORIGIN = 0x20000000, LENGTH = 8K
  RAM (xrw)         : ORIGIN = 0x20002000, LENGTH = 8K
}
```

Step 2: Relocate the interrupt vector table to SRAM. Recompile the project after completing steps 1 and 2.

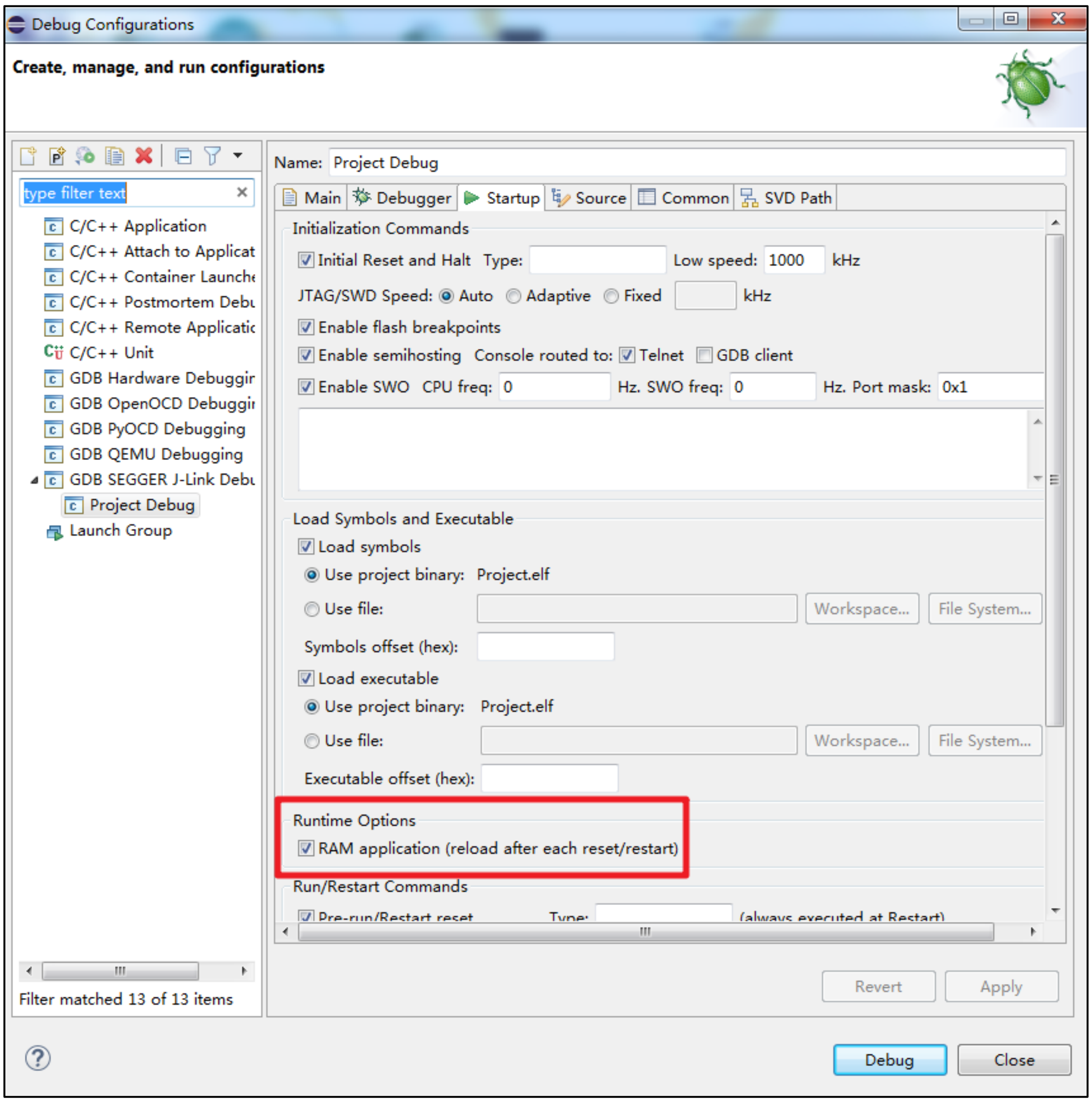
Figure 4-2. Relocate the interrupt vector table when debugging in RAM



```
    \param[in]  none
    \param[out] none
    \retval     none
*/
int main(void)
{
    nvic_vector_table_set(NVIC_VECTTAB_RAM, 0);
    systick_config();
    gd_eval_led_init(LED2);
    gd_eval_led_init(LED3);
    gd_eval_led_init(LED4);
    gd_eval_led_init(LED5);

    while(1){
```

Step 3: In the “Debug Configurations->Startup” option, check “RAM application”.

Figure 4-3. Debug configurations when debugging in RAM



Debug Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
- C/C++ Attach to Applicat
- C/C++ Container Launche
- C/C++ Postmortem Debu
- C/C++ Remote Applicatic
- C/C++ Unit
- GDB Hardware Debuggir
- GDB OpenOCD Debuggi
- GDB PyOCD Debugging
- GDB QEMU Debugging
- GDB SEGGER J-Link Debu
  - Project Debug
- Launch Group

Filter matched 13 of 13 items

Name: Project Debug

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

Initialization Commands

[✓] Initial Reset and Halt Type: Low speed: 1000 kHz

JTAG/SWD Speed: [● Auto] ○ Adaptive ○ Fixed kHz

[✓] Enable flash breakpoints

[✓] Enable semihosting Console routed to: [✓] Telnet [ ] GDB client

[✓] Enable SWO CPU freq: 0 Hz. SWO freq: 0 Hz. Port mask: 0x1

Load Symbols and Executable

[✓] Load symbols

[● Use project binary:] Project.elf

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

Symbols offset (hex):

[✓] Load executable

[● Use project binary:] Project.elf

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

Executable offset (hex):

Runtime Options

[✓] RAM application (reload after each reset/restart)

Run/Restart Commands

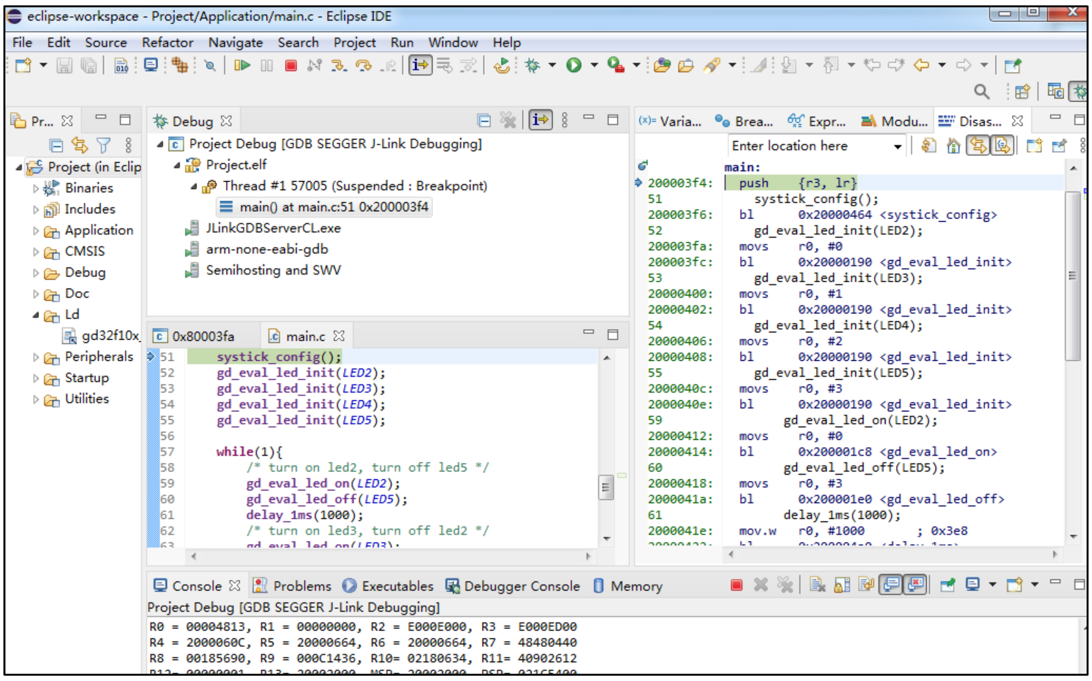
[✓] Pre-run/Restart reset Type: (always executed at Restart)

Revert Apply

Debug Close

Step 4: Enter the Debug perspective when debugging in RAM, as shown in the figure below.

Figure 4-4. Debug perspective when debugging in RAM



```
Project Debug [GDB SEGGER J-Link Debugging]
    Project.elf
        Thread #1 57005 (Suspended : Breakpoint)
            main() at main.c:51 0x200003f4
    JLinkGDBServerCL.exe
    arm-none-eabi-gdb
    Semihosting and SWV
```

```
    systick_config();
    gd_eval_led_init(LED2);
    gd_eval_led_init(LED3);
    gd_eval_led_init(LED4);
    gd_eval_led_init(LED5);

    while(1){
        /* turn on led2, turn off led5 */
        gd_eval_led_on(LED2);
        gd_eval_led_off(LED5);
        delay_1ms(1000);
        /* turn on led3, turn off led2 */
```

```
           main:
200003f4:    push     {r3, lr}
51             systick_config();
200003f6:    bl       0x20000464 <systick_config>
52             gd_eval_led_init(LED2);
200003fa:    movs     r0, #0
200003fc:    bl       0x20000190 <gd_eval_led_init>
53             gd_eval_led_init(LED3);
20000400:    movs     r0, #1
20000402:    bl       0x20000190 <gd_eval_led_init>
54             gd_eval_led_init(LED4);
20000406:    movs     r0, #2
20000408:    bl       0x20000190 <gd_eval_led_init>
55             gd_eval_led_init(LED5);
2000040c:    movs     r0, #3
2000040e:    bl       0x20000190 <gd_eval_led_init>
59                 gd_eval_led_on(LED2);
20000412:    movs     r0, #0
20000414:    bl       0x200001c8 <gd_eval_led_on>
60                 gd_eval_led_off(LED5);
20000418:    movs     r0, #3
2000041a:    bl       0x200001e0 <gd_eval_led_off>
61                 delay_1ms(1000);
2000041e:    mov.w    r0, #1000        ; 0x3e8
```

```
Project Debug [GDB SEGGER J-Link Debugging]
R0 = 00004813, R1 = 00000000, R2 = E000E000, R3 = E000ED00
R4 = 2000060C, R5 = 20000664, R6 = 20000664, R7 = 48480440
R8 = 00185690, R9 = 000C1436, R10= 02180634, R11= 40902612
```

5. Printing with printf

5.1. Use steps

Step 1: Add the syscall.c file, and add the following _write function definition to the file.

```
int _write(int file, char *ptr, int len)
{
        int DataIdx;

                for (DataIdx = 0; DataIdx < len; DataIdx++)
                {
                     __io_putchar( *ptr++ );
                }
        return len;

}
```

Step 2: Redirect usart to the __io_putchar function.

```
int __io_putchar(int ch)
{
       usart_data_transmit(EVAL_COM0, (uint8_t) ch );
       while(RESET == usart_flag_get(EVAL_COM0, USART_FLAG_TBE)){
       };

       return ch;

}
```

Step 3: Use the printf function to print normally.

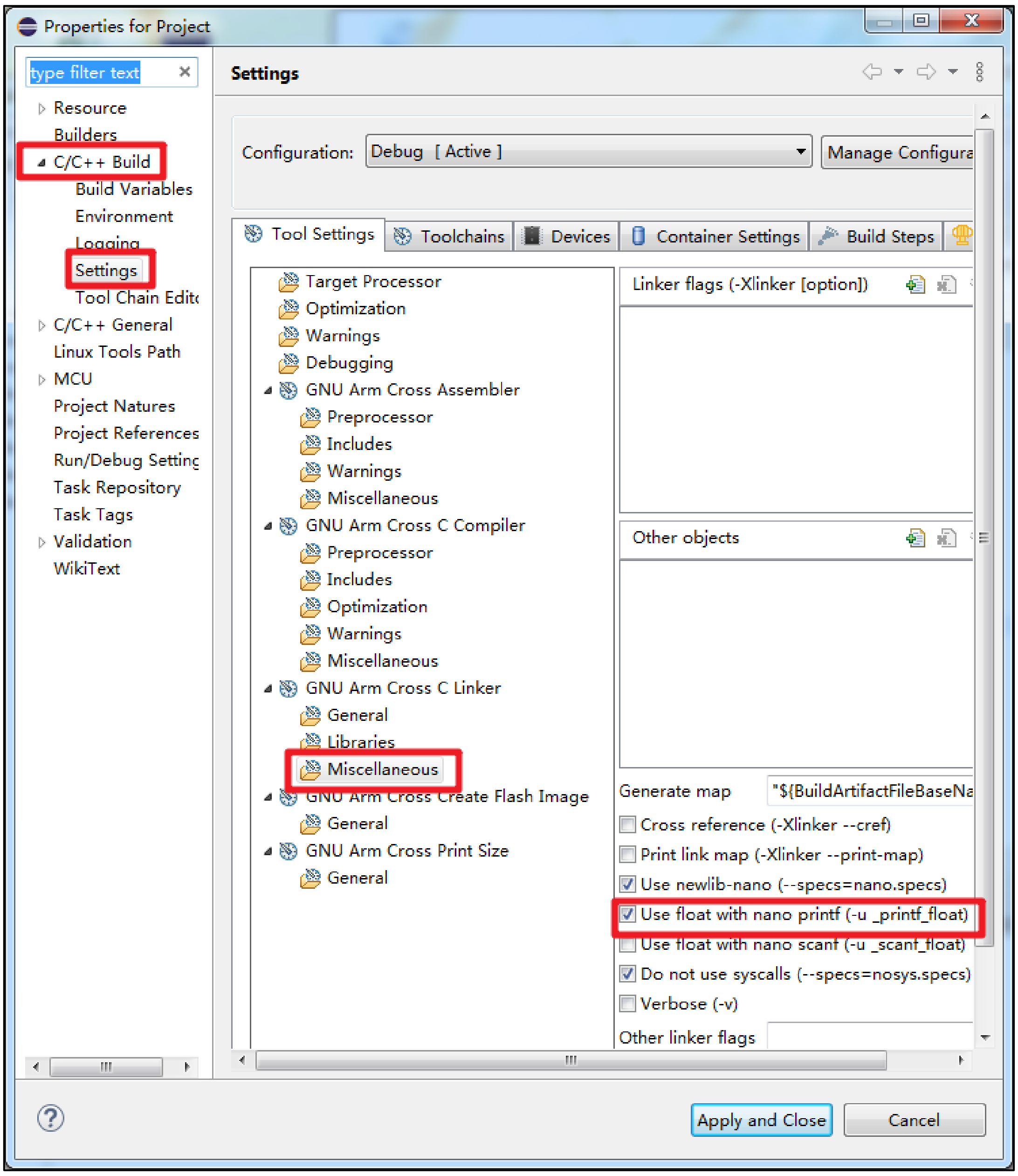
```
       printf("Running led test!\r\n");
```

5.2. 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 Arm Cross C Linker->Miscellaneous" option.

Figure 5-1. Print floating point data configuration


Properties for Project

Settings

Configuration: Debug [Active]

- 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) [✓]
- Cross reference (-Xlinker --cref) [ ]
- Print link map (-Xlinker --print-map) [ ]
- Verbose (-v) [ ]

Generate map: "${BuildArtifactFileBaseNa

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.

6. Revision history

Table 6-1. Revision history

Revision No.	Description	Date
1.0	Initial Release	Nov.30, 2021