GigaDevice Semiconductor Inc.

GD32 RISC-V MCU Eclipse development tutorial for Windows

Application Note
AN067
# Table of Contents

Table of Contents ................................................................................................. 2
List of Figures ........................................................................................................ 4
List of Tables ........................................................................................................ 6
1. Introduction ....................................................................................................... 7
2. Development environment ............................................................................. 8
3. Project development ....................................................................................... 9

3.1. New project .................................................................................................. 9
3.2. New project folder and add files ................................................................. 12
  3.2.1. Create folders and add files manually .................................................. 12
  3.2.2. Create folders and add files by “Refresh” .......................................... 24
3.3. Project configurations ............................................................................... 26
  3.3.1. Target Processor option configuration ............................................. 27
  3.3.2. Optimization option configuration .................................................... 28
  3.3.3. GNU RISC-V Cross Assembler configuration .................................. 29
  3.3.4. GNU RISC-V Cross C Compiler configuration .................................. 30
  3.3.5. GNU RISC-V Cross C Linker configuration ....................................... 32
  3.3.6. Build Steps configuration-generate bin file ....................................... 34
3.4. Build project ............................................................................................... 35
3.5. Use J-Link to download and debug the project ......................................... 36
  3.5.1. Debug configuration interface ............................................................. 36
  3.5.2. Main tab ............................................................................................. 38
  3.5.3. Debugger tab ...................................................................................... 38
  3.5.4. Startup tab ........................................................................................ 39
  3.5.5. SVD Path tab .................................................................................... 40
3.6. Use GD-Link to download and debug the project ..................................... 41
  3.6.1. Debug configuration interface ............................................................. 41
  3.6.2. Main tab ............................................................................................. 42
  3.6.3. Debugger tab ...................................................................................... 42
  3.6.4. SVD Path tab .................................................................................... 43
3.7. Debug interface .......................................................................................... 44
  3.7.1. Toolbar introduction ........................................................................... 45
  3.7.2. Registers view ................................................................................... 46
  3.7.3. Peripherals view ............................................................................... 47
  3.7.4. Memory view ..................................................................................... 49
  3.7.5. Expressions view .............................................................................. 50
  3.7.6. Disassembly view ............................................................................... 51
4. Import an existing project .......................................................... 53
5. Debug in RAM ........................................................................ 55
6. Printing with printf .................................................................. 57
   6.1. Use steps ........................................................................... 57
   6.2. Print floating point data configuration ................................ 57
7. Revision history ........................................................................ 59
List of Figures

Figure 2-1. New RISC-V C project ........................................................................................................ 9
Figure 2-2. Select C Managed Build .................................................................................................... 10
Figure 2-3. Create new RISC-V project name and select project storage path ................................. 11
Figure 2-4. Select RISC-V cross toolchain path ................................................................................ 12
Figure 2-5. Project perspective .......................................................................................................... 12
Figure 2-6. New project folder .......................................................................................................... 13
Figure 2-7. New virtual sub-folder ...................................................................................................... 13
Figure 2-8. RISC-V project view ........................................................................................................ 14
Figure 2-9. Add files .......................................................................................................................... 15
Figure 2-10. Select files to be imported .............................................................................................. 15
Figure 2-11. Import files to the Application folder ............................................................................ 16
Figure 2-12. Import files to the Peripherals folder ............................................................................. 17
Figure 2-13. Import files to the Peripheral-Source folder .................................................................. 18
Figure 2-14. Import files to the RISCV-env_Eclipse folder ............................................................... 19
Figure 2-15. Import files to the RISCV-drivers folder ...................................................................... 20
Figure 2-16. Import files to the RISCV-stubs folder ....................................................................... 21
Figure 2-17. Import files to the Doc folder ......................................................................................... 22
Figure 2-18. Import files to the Utilities folder ................................................................................. 23
Figure 2-19. Final RISC-V project view ............................................................................................ 24
Figure 2-20. Project folder structure .................................................................................................. 25
Figure 2-21. Refresh the project ........................................................................................................ 25
Figure 2-22. Project structure in Eclipse IDE ..................................................................................... 26
Figure 2-23. Project properties configurations .................................................................................. 27
Figure 2-24. Target Processor configuration ..................................................................................... 28
Figure 2-25. Optimization configuration ............................................................................................ 29
Figure 2-26. GNU RISC-V Cross Assembler -> Includes configuration ........................................... 30
Figure 2-27. GNU RISC-V Cross C Compiler -> Preprocessor configuration .................................... 31
Figure 2-28. GNU RISC-V Cross C Compiler -> Includes configuration .......................................... 32
Figure 2-29. GNU RISC-V Cross C Linker -> General configuration ............................................... 33
Figure 2-30. GNU RISC-V Cross C Linker -> Miscellaneous configuration ..................................... 34
Figure 2-31. Build Steps configuration .............................................................................................. 35
Figure 2-32. Build project .................................................................................................................. 35
Figure 2-33. Build RISC-V project completed .................................................................................... 36
Figure 2-34. Enter Debug Configurations interface .......................................................................... 37
Figure 2-35. GDB SEGGER J-Link Debugging-Main tab ................................................................. 38
Figure 2-36. GDB SEGGER J-Link Debugging-Debugger tab ............................................................ 39
Figure 2-37. GDB SEGGER J-Link Debugging-Startup tab ............................................................... 40
Figure 2-38. GDB SEGGER J-Link Debugging-SVD Path tab ............................................................ 40
Figure 2-39. Enter Debug Configurations interface .......................................................................... 41
Figure 2-40. GDB OpenOCD Debugging-Main tab ............................................................................ 42
Figure 2-41. GDB OpenOCD Debugging-Debugger tab ................................................................. 43
Figure 2-42. GDB OpenOCD Debugging-SVD Path tab ............................................................... 43
Figure 2-43. Enter Debug perspective -1 ...................................................................................... 44
Figure 2-44. Enter Debug perspective -2 ...................................................................................... 44
Figure 2-45. Debug perspective ................................................................................................... 45
Figure 2-46. Open Registers view ............................................................................................... 46
Figure 2-47. Registers view ......................................................................................................... 47
Figure 2-48. Open Peripherals view ............................................................................................ 48
Figure 2-49. Peripherals view ....................................................................................................... 49
Figure 2-50. Open Memory view ................................................................................................. 50
Figure 2-51. Memory view .......................................................................................................... 50
Figure 2-52. Open Expressions view ........................................................................................... 51
Figure 2-53. Expressions view .................................................................................................... 51
Figure 2-54. Open Disassembly view .......................................................................................... 51
Figure 2-55. Disassembly view .................................................................................................... 52
Figure 2-56. Exit the Debug perspective .................................................................................... 52
Figure 3-1. Import an existing project - 1 .................................................................................... 53
Figure 3-2. Import an existing project - 2 .................................................................................... 54
Figure 4-1. Ld file memory map when debugging in RAM ......................................................... 55
Figure 4-2. Debug configurations when debugging in RAM .................................................... 55
Figure 4-3. Debug perspective when debugging in RAM ............................................................ 56
Figure 5-1. Print floating point data configuration ...................................................................... 58
List of Tables

Table 6-1. Revision history ....................................................................................................................... 59
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, user can choose to create a new C/C++ Project and select C Managed Build option.

**Figure 3-1. New RISC-V C project**
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

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.
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

3.2. New project folder and add files

3.2.1. Create folders and add files manually

Right-click the project name and select “new->Folder”.
Create a virtual folder “Peripherals”.

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

Right-click “Application” and select the “Import” option to import the file.
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**
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 resources from the local file system.

From directory: GD32VF103_Firmware_Library_V1.2.0\Firmware\GD32VF103_standard_peripheral

- system_gd32vf103.c

Into folder: Project_RISCV/Peripherals

Options:
- Create links in workspace
  - Create virtual folders
  - Create link locations relative to: PROJECT LOC

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

From directory: firmware/library_v1.1.2/firmware/GD32VF103_standard_peripheral/Source

Into folder: Project_RISCV/Peripherals/Source

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

[Diagram of the Eclipse import settings window with highlighted options]
Figure 3-14. Import files to the RISCV-env_Eclipse folder

![Image of the Eclipse import dialog]

- Import files to the RISCV-env_Eclipse folder.
- From directory: `32VF103_Firmware_Library_V1.1.2\Firmware\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

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

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 RISC-V-stubs folder

- From directory: C:\GD32VF103_Firmware_Library_V1.1.2\Firmware\RISCV\stubs
- Into folder: Project_RISCV/RISCV/stubs
- Options:
  - Create links in workspace
    - Create virtual folders
    - Create link locations relative to: PROJECT_LOC
Figure 3-17. Import files to the Doc folder

Options
- Create links in workspace
- Create virtual folders
- Create link locations relative to: PROJECT_LOC
Figure 3-18. Import files to the Utilities folder

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

Into folder: Project_RISCV\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
3.2.2. Create 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

- .settings
- Firmware
- Template
- Utilities
- .cproject
- .project

Figure 3-21. Refresh the project
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.
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.
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.
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 it in this guide:

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

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.
3.5.4. **Startup tab**

In the “Startup” tab, configure type:
3.5.5. **SVD Path tab**

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

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**

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**
3.7. Debug interface

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

Figure 3-43. Enter Debug perspective -1

Switch to Debug perspective.

Figure 3-44. Enter Debug perspective -2
3.7.1. Toolbar introduction

- ![Resume icon] : resume
- ![Suspend icon] : suspend
- ![Terminate icon] : terminate
- ![Step into icon] : step into
- ![Step over icon] : step over
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
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
### 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.
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

![Image of Expressions view]

Figure 3-53. Expressions view

![Image of Expressions view]

Note: Eclipse 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

![Image of Disassembly view]

In the disassembly window, breakpoints can be enabled, assembly instructions can be executed in single step, etc.
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
4. **Import an existing project**

In addition to new projects, users can also import existing Eclipse projects directly. In the menu bar, click “File -> Import”, select “General -> Existing 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
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](image)

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

![Figure 5-2. Debug configurations when debugging in RAM](image)

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
6. Printing with printf

6.1. Use steps

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

```c
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.

```c
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.

```c
printf("Running led test!\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

Note: 1. When using printf function, user need to add "\n\n" at the end of the printed content, for example, printf("Running led test!\n\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

<table>
<thead>
<tr>
<th>Revision No.</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>Initial Release</td>
<td>Jun.20, 2022</td>
</tr>
</tbody>
</table>
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 harmless 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.