## Table of Contents

Table of Contents ................................................................................................................. 2
List of Figures ....................................................................................................................... 4
List of Tables ......................................................................................................................... 6

1. Development environment ............................................................................................... 7

2. Project development .......................................................................................................... 8

   2.1. New project ................................................................................................................ 8

   2.2. New project folder and add files ................................................................................ 12
       2.2.1. Create folders and add files manually ................................................................. 12
       2.2.2. Create folders and add files by “Refresh” ............................................................ 23

   2.3. Project configurations ............................................................................................... 25
       2.3.1. Target Processor option configuration ............................................................... 26
       2.3.2. Optimization option configuration .................................................................... 27
       2.3.3. GNU Arm Cross C Compiler configuration ....................................................... 28
       2.3.4. GNU Arm Cross C Linker configuration ............................................................ 30
       2.3.5. Build Steps configuration-generate bin file ....................................................... 32

   2.4. Build project ................................................................................................................ 33

   2.5. Use J-Link to download and debug the project ......................................................... 34
       2.5.1. Debug configuration interface .......................................................................... 34
       2.5.2. Main tab ........................................................................................................... 35
       2.5.3. Debugger tab .................................................................................................... 36
       2.5.4. SVD Path tab .................................................................................................. 36

   2.6. Use GD-Link to download and debug the project ...................................................... 37
       2.6.1. Debug configuration interface .......................................................................... 37
       2.6.2. Main tab ........................................................................................................... 38
       2.6.3. Debugger tab .................................................................................................... 38
       2.6.4. SVD Path tab .................................................................................................. 39

   2.7. Debug interface .......................................................................................................... 39
       2.7.1. Toolbar introduction ......................................................................................... 41
       2.7.2. Registers view .................................................................................................. 42
       2.7.3. Peripherals view ............................................................................................... 43
       2.7.4. Memory view .................................................................................................... 45
       2.7.5. Expressions view .............................................................................................. 46
       2.7.6. Disassembly view .............................................................................................. 47
       2.7.7. Exit the Debug perspective .............................................................................. 48

3. Import an existing project ................................................................................................. 49
4. Debug in RAM ........................................................................................................... 51
5. Printing with printf .................................................................................................. 54
  5.1. Use steps ................................................................................................................. 54
  5.2. Print floating point data configuration ................................................................. 54
6. Revision history ......................................................................................................... 56
## List of Figures

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

Table 6-1. Revision history ........................................................................................................... 56
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, LAUNCH eclipse-workspace. Under "File->New" option, user can choose to create a new C/C++ Project and select C Managed Build option.

**Figure 2-1. New ARM 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 ARM Cross GCC.
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

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
2.2. New project folder and add files

2.2.1. Create 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”.

---

AN041

GD32 ARM MCU Eclipse development tutorial for Windows
Create the Application, CMSIS, Doc, Ld, Startup and Utilities folders in the same way.
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
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

Figure 2-13. Import files to the Doc folder
Figure 2-14. Import files to the LD folder
Figure 2-15. Import files to the Peripherals folder
Figure 2-16. Import files to the Startup folder

Import resources from the local file system.

From directory: \(\text{eclipse workspace}\)\GD32F10x\Firmware\Library\V2.2.1\Firmware\CM3\GD\GD32F10x\Source\GCC

Into folder: Project/Startup

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 2-17. Import files to the Utilities folder
2.2.2. 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 2-19. Project folder structure

Figure 2-20. Refresh the project
2.3. Project configurations

Right-click the project and select the "Properties" option to open it.
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.
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.
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.
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.
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.
In the “Miscellaneous” option, check “Use newlib-nano” and “Do not use syscalls”. (The code size can be optimized)
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"
```
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**

**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**

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.
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**
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**

### 2.5.4. SVD Path tab

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

![Debug configuration interface](image)

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

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.
2.6.4. SVD Path tab

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

2.7. Debug interface

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

Figure 2-41. Enter Debug perspective -2
2.7.1. Toolbar introduction

- : resume
- : suspend
- : terminate
- : step into
- : step over
- : step out
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
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
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.
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

![Expressions view](image)

**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.

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**

![Disassembly view](image)

In the disassembly window, breakpoints can be enabled, assembly instructions can be executed in single step, etc.
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
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->Existing Projects into Workspace" to import the existing project, and click "Next".

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

![Import existing project dialog]

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
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

```c
/* memory map */
MEMORY
{
  FLASH (rx) : ORIGIN = 0x20000000, LENGTH = 8K
  RAM (xrw)  : ORIGIN = 0x20020000, 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

```c
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);
}
```

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

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
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.

```c
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.

```c
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.

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

5.2. Print floating point data configuration

Print floating point data configuration:

check the “-u _printf_float” option in the project “Properties->C/C++ Build->Settings->Tool Settings->GNU Arm Cross C Linker->Miscellaneous” option.
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>
<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>Nov.30, 2021</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.

© 2021 GigaDevice – All rights reserved