GigaDevice Semiconductor Inc.

Arm® Cortex®-M3/4/23/33 32-bit MCU

Application Note
AN016
# Table of Contents

Table of Contents ........................................................................................................................................... 2
List of Figures ..................................................................................................................................................... 3
List of Tables ......................................................................................................................................................... 4

1. Introduction ..................................................................................................................................................... 5

2. Development environment constructione ..................................................................................................... 6
   2.1. Install cross-compilation tool ..................................................................................................................... 6
   2.2. Install C/C++ MinGW compiler .................................................................................................................. 9
   2.3. Install Cmake tool ...................................................................................................................................... 16
   2.4. Install Vscode and plug-ins ...................................................................................................................... 18
   2.5. Install Openocd ....................................................................................................................................... 20

3. CmakeLists file writing ................................................................................................................................... 21
   3.1. CMakeLists.txt and Cortex-M3.cmake files in the root directory ............................................................ 22
   3.2. CMakeLists.txt file in the gd_libs folder ............................................................................................... 23
   3.3. CMakeLists.txt file in the src folder ....................................................................................................... 24

4. Compile, download and debug ....................................................................................................................... 26
   4.1. Compile and download ............................................................................................................................. 26
   4.2. Debug ....................................................................................................................................................... 30

5. Revision history ............................................................................................................................................... 33
List of Figures

Figure 2-1. Select download of GUN Arm Embedded Toolchain .............................................. 6
Figure 2-2. GUN Arm Embedded Toolchain installation process 1 ........................................... 7
Figure 2-3. GUN Arm Embedded Toolchain installation process 2 ........................................... 7
Figure 2-4. GUN Arm Embedded Toolchain installation process 3 ......................................... 8
Figure 2-5. GUN Arm Embedded Toolchain installation process 4 ......................................... 8
Figure 2-6. GUN Arm Embedded Toolchain installation process 5 ......................................... 9
Figure 2-7. Test whether the GUN Arm Embedded Toolchain is installed successfully .......... 9
Figure 2-8. MinGW-W64 choose to download and install ......................................................... 10
Figure 2-9. MinGW-W64 installation process 1 ........................................................................ 10
Figure 2-10. MinGW-W64 installation process 2 ....................................................................... 11
Figure 2-11. MinGW-W64 installation process 3 ....................................................................... 11
Figure 2-12. MinGW-W64 installation process 4 ....................................................................... 11
Figure 2-13. MinGW-W64 installation process 5 ....................................................................... 12
Figure 2-14. MinGW-W64 installation process 6 ....................................................................... 12
Figure 2-15. Test whether MinGW-W64 is installed successfully 1 ........................................... 13
Figure 2-16. Copy MinGW-W64 bin folder path ........................................................................ 13
Figure 2-17. Add MinGW-W64 environment variable 1 .............................................................. 14
Figure 2-18. Add MinGW-W64 environment variable 2 .............................................................. 14
Figure 2-19. Add MinGW-W64 environment variable 3 .............................................................. 15
Figure 2-20. Test whether MinGW-W64 is installed successfully 2 ........................................... 16
Figure 2-21. Modify MinGW-W64 mingw32-make command to make command ....................... 16
Figure 2-22. Cmake installation process 1 ................................................................................ 16
Figure 2-23. Cmake installation process 2 ................................................................................ 17
Figure 2-24. Cmake installation process 3 ................................................................................ 17
Figure 2-25. Test whether Cmake is installed successfully ......................................................... 17
Figure 2-26. Vscode choose to download and install ................................................................. 18
Figure 2-27. Vscode installation process 1 ................................................................................. 18
Figure 2-28. Vscode installation process 2 ................................................................................. 19
Figure 2-29. Vscode installation process 3 ................................................................................. 19
Figure 2-30. Vscode installation process 4 ................................................................................ 19
Figure 2-31. Vscode plugin installation search interface ............................................................. 20
Figure 3-1. File organization chart ............................................................................................. 21
Figure 4-1. Running tasks in Vscode .......................................................................................... 28
Figure 4-2. The build process of Cmake to generate makefile .................................................... 28
Figure 4-3. Build process of makefile to generate executable file .............................................. 28
Figure 4-4. File organization structure in the build directory .................................................... 29
Figure 4-5. One-click compilation and download process .......................................................... 30
Figure 4-6. Vscode Debug interface ......................................................................................... 30
Figure 4-7. Add a breakpoint in the debugging interface and run ............................................... 31
Figure 4-8. View peripheral registers and variable values ........................................................ 31
List of Tables

Table 3-1. CMakeLists.txt code in the root directory ................................................................. 22
Table 3-2. Cortex-M3.cmake code in the root directory ............................................................. 22
Table 3-3. CMakeLists.txt code in gd_libs .................................................................................. 23
Table 3-4. CMakeLists.txt code in src ........................................................................................ 24
Table 4-1. tasks.json code ........................................................................................................... 26
Table 4-2. launch.json code ......................................................................................................... 27
Table 5-1. Revision history .......................................................................................................... 33
1. **Introduction**

In the process of project compilation, most of the make tools are used. The make tools include GNU Make, qmake, MS nmake and Makepp, etc. These make tools follow different specifications, standards and formate when writing makefiles on different platforms. Transplanting the project to a different platform for compilation during the development process will cause compilation failure. Using the CMAKE tool to generate Makefile will effectively solve the above problems. This application manual is based on the GD32F10x SDK development kit, and the CMAKE tool is used to build the compilation environment.
2. Development environment construction

The development environment is mainly introduced as follows:

- Hardware development board: GD32F103C-EVAL-V1.0 development board
- Cortex-M3: GD32F103C
- Operating system: Win10-64 bit
- Cross compilation tool chain: gcc-arm-none-eabi
- C/C++ compiler: MinGW
- Development environment: VS CODE+CMAKE
- Debug download tool: OPENOCD

2.1. Install cross-compilation tool


There are multiple version of GUN Arm Embedded Toolchain is available for downloading on the page. In this application manual, choose to download and install gcc-arm-none-eabi-10-2020-q4-major-win32.exe, as shown in Figure 2-1. Select download of GUN Arm Embedded Toolchain.

![Figure 2-1. Select download of GUN Arm Embedded Toolchain](attachment:image)

After the download is complete, double-click to install and select “OK”.

6
Build GD32MCU development environment based on cmake tool

Figure 2-2. GUN Arm Embedded Toolchain installation process 1

Click “Next”.

Figure 2-3. GUN Arm Embedded Toolchain installation process 2

Click “I accept”, select the installation path, select the default path and click install.
When the installation is complete, select the add path to environment variable option and click “Finish”.

Figure 2-4. GUN Arm Embedded Toolchain installation process 3

Figure 2-5. GUN Arm Embedded Toolchain installation process 4

Build GD32MCU development environment based on cmake tool
Build GD32MCU development environment based on cmake tool

Figure 2-6. GUN Arm Embedded Toolchain installation process

Check whether the installation is successful, enter cmd in the operation, click “OK”, enter “arm-none-eabi-gcc –v” in the command line, and the return result as shown in Figure 2-7. Test whether the GUN Arm Embedded Toolchain is installed successfully indicates that the installation is successful.

Figure 2-7. Test whether the GUN Arm Embedded Toolchain is installed successfully

2.2. Install C/C++ MinGW compiler

MinGW installation package download address: https://sourceforge.net/projects/mingw-w64/files/mingw-w64/mingw-w64-release/.

There are two installation methods, online installation and offline installation, this manual chooses online installation. Download MinGW-W64-install.exe.
Build GD32MCU development environment based on cmake tool

Figure 2-8. MinGW-W64 choose to download and install

After the download is complete, double-click to install. Select “Next”.

Figure 2-9. MinGW-W64 installation process 1

The configuration options are as follows, click “Next”. 
Select the installation path, select the default path.

Wait for the file download process.
Build GD32MCU development environment based on cmake tool

Click “Next”.

Figure 2-13. MinGW-W64 installation process 5

Click “Finish” to complete the installation.

Figure 2-14. MinGW-W64 installation process 6
Enter "gcc -v" in the cmd command line to output non-runnable programs. The solution to this problem is to manually add the downloaded file to the system environment variable.

**Figure 2-15. Test whether MinGW-W64 is installed successfully 1**

Open the MinGW installation path, copy the path in the /bin folder, and add the path to the system environment variables.

**Figure 2-16. Copy MinGW-W64 bin folder path**
Right-click “this computer” and select properties, click “advanced” to select environment variables.

Figure 2-17. Add MinGW-W64 environment variable 1

Select the system variable and click Path.

Figure 2-18. Add MinGW-W64 environment variable 2
Build GD32MCU development environment based on cmake tool

Click “New”, paste the copied path, and click “OK”.

Figure 2-19. Add MinGW-W64 environment variable 3

Enter “gcc -v” in cmd and output the gcc -v version number. The installation is successful.
2.3. Install Cmake tool

Cmake tool download and installation address: [https://cmake.org/download/](https://cmake.org/download/)

In this application manual, download and install version 3.20.1, and the installation process is as follows.

After the download is complete, double-click to install. Select “Next”.

Figure 2-22. Cmake installation process 1
Build GD32MCU development environment based on cmake tool

Select "Add Cmake to system PATH for all users", click “Next”, select the default path, and click “Next”.

Figure 2-23. Cmake installation process 2

Click “Install”, wait for the installation to complete, and click Finish to complete the installation.

Figure 2-24. Cmake installation process 3

Check whether the installation is successful, enter cmd in the run, click OK, enter cmake --version in the command line, the return result as shown in Figure 2-25. Test whether Cmake is installed successfully represents the successful installation.

Figure 2-25. Test whether Cmake is installed successfully
2.4. Install Vscode and plug-ins

Vscode download and installation address: https://code.visualstudio.com/

In this application manual, download and install version 1.53.0, the download and installation process is as follows.

Figure 2-26. Vscode choose to download and install

![Vscode download page]

After the download is complete, double-click to install. Select “Next”, select the installation path, and click “Next”.

Figure 2-27. Vscode installation process 1

![Vsode installation process image]
Build GD32MCU development environment based on cmake tool

Keep clicking “Next” to complete the installation.

Figure 2-28. Vscode installation process 2

Figure 2-29. Vscode installation process 3

Figure 2-30. Vscode installation process 4
In order to meet the development requirements, some plugins should also be installed, open the installer Vscode, select the plug-in center on the left, search for and install the following plug-ins respectively, and click to install. The installation plug-ins and version numbers are as follows:

- C/C++: V1.2.2
- Cortex-Debug: V0.3.12
- Chinese (simplified) Language for visual studio: V1.52.2

**Figure 2-31. Vscode plugin installation search interface**

2.5. **Install Openocd**

Download link of Openocd that supports GD32MCU: [https://github.com/GigaDevice-Semiconductor/openocd](https://github.com/GigaDevice-Semiconductor/openocd).

After the user completes the compilation of Openocd according to the requirements, the executable file openocd.exe is generated, and the file is added to the environment variable. For the method of adding the environment variable, please refer to [Install C/C++ MinGW compiler](https://github.com/GigaDevice-Semiconductor/openocd).
3. **CmakeLists file writing**

The file organization structure of this application manual is shown in **Figure 3-1. File organization chart**, and the contents of each folder are introduced below:

**Figure 3-1. File organization chart**

```
| - CMakeLists.txt
| - Ctiny-W3.make
| - opened_gdlink.cfg
| - readme.txt
| - build
| - gd_libs
| - GD32F10x
| - - Firmware
| - - CMSIS
| - - core_cm3.h
| - - core_cm0plus.h
| - - core_cm0.h
| - - gd32f10x.h
| - - system_gd32f10x.c
| - - system_gd32f10x.h
| - - gcc_startup
| - - startup_gd32f10x_md.S
| - - Peripherals
| - - Inc
| - - src
| - - inc
| - - gd32f10x Eval.h
| - - gd32f10x.l1h
| - - gd32f10x_libopt.h
| - - systick.h
| - - ldscripts
| - - gd32f10x_flash.ld
| - - src
| - CMakeLists.txt
| - gd32f10x Eval.c
| - gd32f10x_l1.c
| - main.o
| - systick.o
```

In the build folder, the intermediate files, library files and final executable files generated during the compilation process are stored;

In the gd_libs folder, store GD library files, mainly including peripheral library files, startup files and some header files;

In the inc folder, store the header files of the code written by the user and the header files related to the development board;

In the ldscripts folder, store the link script file;

In the src folder, the .c files written by the user and the .c files related to the development board are stored.

Among them, the CMakeLists.txt file is included in the root directory, gd_libs and src folders,
Build GD32MCU development environment based on cmake tool and the Cortex-M3.cmake file is included in the root directory. The content of CmakeLists.txt in each file is introduced below.

3.1. **CMakeLists.txt and Cortex-M3.cmake files in the root directory**

The content of the CMakeLists.txt file in the root directory is shown in **Table 3-1. CMakeLists.txt code in the root directory**.

<table>
<thead>
<tr>
<th>Table 3-1. CMakeLists.txt code in the root directory</th>
</tr>
</thead>
<tbody>
<tr>
<td># set the minimum supported version of CMake</td>
</tr>
<tr>
<td>cmake_minimum_required(VERSION 3.17)</td>
</tr>
<tr>
<td>SET(PRJ_NAME &quot;GD32F10x&quot;)</td>
</tr>
<tr>
<td># define project name</td>
</tr>
<tr>
<td>project(${PRJ_NAME})</td>
</tr>
<tr>
<td>#include Cortex-M3.cmake</td>
</tr>
<tr>
<td>include(Cortex-M3.cmake)</td>
</tr>
<tr>
<td># add directories gd_libs and src</td>
</tr>
<tr>
<td>add_subdirectory(gd_libs)</td>
</tr>
<tr>
<td>add_subdirectory(src)</td>
</tr>
</tbody>
</table>

The content of the Cortex-M3.cmake file is shown in **Table 3-2. Cortex-M3.cmake code in the root directory**. Users can modify the compilation options and add related macro definitions by modifying the parameters in the file.

<table>
<thead>
<tr>
<th>Table 3-2. Cortex-M3.cmake code in the root directory</th>
</tr>
</thead>
<tbody>
<tr>
<td># set the minimum supported version of CMake</td>
</tr>
<tr>
<td>cmake_minimum_required(VERSION 3.17)</td>
</tr>
<tr>
<td>#Cmake cross compilation configuration</td>
</tr>
<tr>
<td>SET(CMAKE_SYSTEM_NAME Generic)</td>
</tr>
<tr>
<td>#setup supportASM</td>
</tr>
<tr>
<td>ENABLE_LANGUAGE(ASM)</td>
</tr>
<tr>
<td>#debug mode</td>
</tr>
<tr>
<td>SET(CMAKE_BUILD_TYPE &quot;Debug&quot;)</td>
</tr>
<tr>
<td>#release mode</td>
</tr>
<tr>
<td>#SET(CMAKE_BUILD_TYPE &quot;Release&quot;)</td>
</tr>
<tr>
<td># set up C compilation tools</td>
</tr>
<tr>
<td>SET(CMAKE_C_COMPILER arm-none-eabi-gcc)</td>
</tr>
<tr>
<td># ELF to bin and hex file tool</td>
</tr>
<tr>
<td>SET(CMAKE_OBJCOPY arm-none-eabi-objcopy)</td>
</tr>
<tr>
<td># file size tool</td>
</tr>
<tr>
<td>SET(CMAKE_SIZE arm-none-eabi-size)</td>
</tr>
<tr>
<td># set floating point options</td>
</tr>
</tbody>
</table>
Build GD32MCU development environment based on cmake tool

```cmake
SET(MCU_FLAGS "-mcpu=cortex-m3 -mfloat-abi=softfp -mfpu=fpv4-sp-d16")
# set warning related information
SET(CMAKE_C_FLAGS "$\{MCU_FLAGS\} -w -Wno-unknown-pragmas")
# set debugging options
SET(CMAKE_C_FLAGS_DEBUG "-O0 -g2 -ggdb")
SET(CMAKE_C_FLAGS_RELEASE "-O3")
# add macro definition
ADD_DEFINITIONS(
    -DGD32F10X_HD
    -DUSE_STDPERIPH_DRIVER
)
```

### 3.2. CMakeLists.txt file in the gd_libs folder

The content of the CMakeLists.txt file in gd_libs is shown in Table 3-3. CMakeLists.txt code in gd_libs. This file mainly generates the gd32_lib library from the GD peripheral firmware library file and CMSIS related files and startup files, and specifies the location of the generated file.

#### Table 3-3. CMakeLists.txt code in gd_libs

```cmake
# set relevant path variables
SET(START_UP_DIR ${PROJECT_SOURCE_DIR}/gd_libs/GD32F10x/Firmware/CMSIS/gcc_startup)
SET(CORE_SUPPORT_DIR ${PROJECT_SOURCE_DIR}/gd_libs/GD32F10x/Firmware/CMSIS)
SET(PERIPHERALS_DIR ${PROJECT_SOURCE_DIR}/gd_libs/GD32F10x/Firmware/Peripherals)
# add header file search path
include_directories(
    ${CORE_SUPPORT_DIR}
    ${PERIPHERALS_DIR}/inc
    ${PROJECT_SOURCE_DIR}/inc
)
# set startup file variables
SET(START_UP_ASM startup_gd32f10x_md.S)
# set the C properties of the startup file
set_property(SOURCE ${START_UP_DIR}/${START_UP_ASM} PROPERTY LANGUAGE C)
# The GLOB option will generate a file list for all files matching the query expression, and store the list in the STD_LIB, SRC_CORE defined by the variable
file(GLOB STD_LIB ${PERIPHERALS_DIR}/src/*.c)
file(GLOB SRC_CORE ${CORE_SUPPORT_DIR}/*.c)
# generate library target gd32_lib
add_library(gd32_lib
    ${STD_LIB}
    ${SRC_CORE}
)```
Build GD32MCU development environment based on cmake tool

```
$(START_UP_DIR)/$(START_UP_ASM)

# set the name of the library output
set_target_properties(gd32_lib PROPERTIES OUTPUT_NAME "gd32_lib")

# set the default output path of library files
SET(LIBRARY_OUTPUT_PATH "${PROJECT_SOURCE_DIR}/build/lib")
```

### 3.3. CMakeLists.txt file in the src folder

The content of the CMakeLists.txt file in the src folder is shown in Table 3-4. CMakeLists.txt code in src. This file mainly realizes linking and compiling the .c file and gd32_lib library file written by the user to generate an executable file and specify the location of the generated executable file.

**Table 3-4. CMakeLists.txt code in src**

```
# all .c files in this path are defined as SRC_LIST
aux_source_directory( SRC_LIST )

# add header file search path
include_directories(
    ${PROJECT_SOURCE_DIR}/gd_libs/GD32F10x/Firmware/CMSIS
    ${PROJECT_SOURCE_DIR}/gd_libs/GD32F10x/Firmware/Peripherals/inc
    ${PROJECT_SOURCE_DIR}/inc)

# add non-standard shared library search path
link_directories(${PROJECT_SOURCE_DIR}/build/lib)

# set the relative path and variables of the linked file
SET(FLASH_LD_DIR ${PROJECT_SOURCE_DIR}/ldscripts)
SET(FLASH_LD_FILE gd32f10x_flash.ld)
SET(LINKER_SCRIPT ${FLASH_LD_DIR}/${FLASH_LD_FILE})

# set link options
SET(CMAKE_EXE_LINKER_FLAGS
    "-specs=nano.specs -specs=nosys.specs -T${LINKER_SCRIPT} -Wl,-Map=${PROJECT_BINARY_DIR}/${PRJ_NAME}.map,-cref -WI,-gc-sections")

# generate object file
add_executable(${PRJ_NAME}.elf ${SRC_LIST})

# link the object file with the library file
target_link_libraries(${PRJ_NAME}.elf gd32_lib)

# set executable file output path
set(EXECUTABLE_OUTPUT_PATH ${PROJECT_SOURCE_DIR}/build/bin)

# set ELF conversion path
SET(ELF_FILE ${PROJECT_SOURCE_DIR}/build/bin/${PRJ_NAME}.elf)
SET(HEX_FILE ${PROJECT_SOURCE_DIR}/build/bin/${PRJ_NAME}.hex)
SET(BIN_FILE ${PROJECT_SOURCE_DIR}/build/bin/${PRJ_NAME}.bin)
```
# add custom commands to realize ELF conversion of hex and bin files
add_custom_command(TARGET "${PRJ_NAME}.elf" POST_BUILD
          COMMAND ${CMAKE_OBJCOPY} -Obinary ${ELF_FILE} ${BIN_FILE}
          COMMAND ${CMAKE_OBJCOPY} -Oihex   ${ELF_FILE} ${HEX_FILE}
          COMMENT "Building ${PRJ_NAME}.bin and ${PRJ_NAME}.hex"
          COMMAND ${CMAKE_COMMAND} -E copy ${HEX_FILE} "${CMAKE_CURRENT_BINARY_DIR}/${PRJ_NAME}.hex"
          COMMAND ${CMAKE_COMMAND} -E copy ${BIN_FILE} "${CMAKE_CURRENT_BINARY_DIR}/${PRJ_NAME}.bin"
          COMMAND ${CMAKE_SIZE} --format=berkeley ${ELF_FILE} ${HEX_FILE}
          COMMENT "Invoking: Cross ARM GNU Print Size"
4. **Compile, download and debug**

4.1. **Compile and download**

Use vscode to open the project directory, a .vscode folder will be generated in the directory, and create tasks.json file and launch.json files in the folder. The tasks.json file mainly realizes compilation and download function through buttons instead of command lines. The launch.json file is mainly debugging configuration file. The specific file content and comments of the two files are shown in Table 4-1. `tasks.json` code and Table 4-2. `launch.json` code.

**Table 4-1. tasks.json code**

```json
{
  "version": "2.0.0",
  // specify the path where the command is executed
  "options": {
    "cwd": "${workspaceRoot}/build"
  }
  "tasks": [
    // execute cmake command to generate makefile
    "type": "shell",
    "label": "cmake",
    "command": "cmake",
    "args": [
      "-G",
      "MinGW Makefiles",
      "..
    ],
  },
  // execute the make command to generate an executable file
  "label": "make",
  "type": "shell",
  "command": "make",
  "args": [],
  "group": {
    "kind": "build",
    "isDefault": true
  },
  "dependsOn": [
    "cmake"
  ],
}
```

**Table 4-2. launch.json code**

```json
{
  // debug configuration
  "version": "0.2.3",
  "configurations": [
    {
      "name": "Debugging Launch Configuration",
      "type": "cppdbg",
      "request": "launch",
      "name": "Launch Program",
      "program": "${workspaceRoot}/build/GD32MCU.exe",
      "args": [],
      "cwd": "${workspaceRoot}/build",
      "miDebuggerPath": "${execPath}/mingw32\bin\gdb.exe",
      "preLaunchTask": "cmake", // specify the task to run before the debug process
      "postDebugTask": "clear" // specify the task to run after the debug process
    }
  ]
}
```
Build GD32MCU development environment based on cmake tool

```json
"problemMatcher": [],

// execute the openocd command to download the executable file to the target MCU
"type": "shell",
"label": "Build & Update",
"command": "openocd",
"args": [
  "-f",
  //absolute path of configuration file
  "E:/Work_Code/10.cmake/Cmake/Code/Example-windows/gd32f103C_example/openocd_gdlink_gd32f10x.cfg",
  "-c",
  // the absolute path of the compiled executable file
  "program E:/Work_Code/10.cmake/Cmake/Code/Example-windows/gd32f103C_example/build/bin/GD32F10x.elf verify reset exit"
],
"group": "build",
"dependsOn": "make"
}
```

Table 4-2. launch.json code

```json
/
// Use Intellisense to learn about possible attributes.
// Hover to view descriptions of existing attributes.
// For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
"version": "0.2.0",
"configurations": [
  {
    // Specify the path where the command is executed
    "cwd": "${workspaceRoot}"
    // Path to executable file
    "executable": "build/bin/GD32F10x.elf",
    "name": "Debug Microcontroller",
    "request": "launch",
    "type": "cortex-debug",
    // the path where the svd file is located
    "svdFile": "/GD32F10x_MD.svd",
    // openocd command
    "serverType": "openocd",
    // the relative path where the openocd configuration file is located
    "serverType": "openocd"
  }
]"```
Build GD32MCU development environment based on cmake tool

```json
"configFiles": [
  "./openocd_gdlink_gd32f10x.cfg",
]
```

Click "Terminal -> Run Task" in VScode, you can see the following options

**Figure 4-1. Running tasks in Vscode**

Select the cmake option, execute the cmake command and generate the makefile file. VScode terminal will print makefile build process information.

**Figure 4-2. The build process of Cmake to generate makefile**

Select the make option, execute the make command, realize the compilation function, and generate executable files .elf, .bin and .hex files. The vscode terminal will print the executable file generation process information. At the same time, it can be seen from the build directory that the generated elf, hex and bin files are stored in the bin directory.

**Figure 4-3. Build process of makefile to generate executable file**
Build GD32MCU development environment based on cmake tool

Figure 4-4. File organization structure in the build directory

The option Build&Update realizes one-click cmake build, compile and openocd download functions. The vscode terminal will print information about the build, compile and program
Build GD32MCU development environment based on cmake tool download process.

Figure 4-5. One-click compilation and download process

```
> Executing task: cmake -G "MinGW Makefiles" ...
-- Configuring done
-- Generating done
-- Build files have been written to: E:\Work_Code\10_cmake\Cmake\code\ExampleWindows/g
终端将被任务用到，按任意键关闭。

> Executing task: make
[ D: ] Built target gd32.lib
[ D: ] Built target GD32F40x.xlib
终端将被任务用到，按任意键关闭。

> Executing task: openOCD -f E:\Work_Code\10_cmake\Cmake\code\ExampleWindows/gd32f40x
 v0d32f40x.lib verify reset exit
```

4.2. Debug

Click "Run -> Start Debugging" in VScode to enter the debugging interface as shown in Figure 4-6. Vscode Debug interface. Run, single step, reset and terminate the program debugging in the upper right corner of the interface.

Figure 4-6. Vscode Debug interface
Add a breakpoint on the left side of the code, and execute Run to run to the breakpoint.

Figure 4-7. Add a breakpoint in the debugging interface and run

![Debugging interface]

View the peripheral register value on the left boundary surface, and add variables to view and other debugging operations.

Figure 4-8. View peripheral registers and variable values

![Source code]

```c
int main(void)
{
    gd_eval_led_init(LED2);
    gd_eval_led_init(LED3);
    gd_eval_led_init(LED4);
    gd_eval_led_init(LED5);
    systick_config();
    while(1)
    {
        /* turn on LED2 */
        gd_eval_led_on(LED2);
        /* insert 200 ms delay */
        delay_1ms(200);
        /* turn on LED3 */
        gd_eval_led_on(LED3);
        /* insert 200 ms delay */
        delay_1ms(200);
        /* turn on LED4 */
        gd_eval_led_on(LED4);
        /* insert 200 ms delay */
        delay_1ms(200);
    }
    return 0;
}
```
Build GD32MCU development environment based on cmake tool
5. Revision history

Table 5-1. 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>June 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