GigaDevice Semiconductor Inc.

Arm® Cortex®- M3/M4/M23/M33 32-bit MCU

应用笔记 AN016



目录

目录		2
图索引.		3
表索引.	4	ł
1. 简介	5	5
2. 开发 ^现	不境搭建	5
2.1.	安装交叉编译工具	3
2.2.	安装 C/C++ MinGW 编译器	•
2.3.	安装 Cmake 工具16	3
2.4.	安装 Vscode 及插件18	3
2.5.	安装 Openocd)
3.Cmał	reLists 文件编写21	L
3.1.	根目录下 CMakeLists.txt 及 Cortex-M3.cmake 文件22	2
3.2.	gd_libs 文件夹中 CMakeLists.txt 文件23	3
3.3.	src 文件夹中 CMakeLists.txt 文件23	3
4.编译、	下载与调试	5
4.1.	编译与下载	3
4.2.	调试)
5.版本质	历史	3



图索引

图 2-1. GUN Arm Embedded Toolchain 选择下载	. 6
图 2-2. GUN Arm Embedded Toolchain 安装过程 1	. 6
图 2-3. GUN Arm Embedded Toolchain 安装过程 2	. 7
图 2-4. GUN Arm Embedded Toolchain 安装过程 3	. 7
图 2-5. GUN Arm Embedded Toolchain 安装过程 4	. 8
图 2-6. GUN Arm Embedded Toolchain 安装过程 5	. 8
图 2-7. 测试 GUN Arm Embedded Toolchain 是否安装成功	. 9
图 2-8. MinGW-W64 选择下载安装	. 9
图 2-9. MinGW-W64 安装过程 1	10
图 2-10. MinGW-W64 安装过程 2	10
图 2-11. MinGW-W64 安装过程 3	.11
图 2-12. MinGW-W64 安装过程 4	.11
图 2-13. MinGW-W64 安装过程 5	12
图 2-14. MinGW-W64 安装过程 6	12
图 2-15. 测试 MinGW-W64 是否安装成功 1	13
图 2-16. 复制 MinGW-W64 bin 文件夹路径	13
图 2-17. 添加 MinGW-W64 环境变量 1	14
图 2-18. 添加 MinGW-W64 环境变量 2	14
图 2-19. 添加 MinGW-W64 环境变量 3	15
图 2-20. 测试 MinGW-W64 是否安装成功 2	15
图 2-21. 修改 MinGW-W64 mingw32-make 命令为 make 命令	16
图 2-22. Cmake 安装过程 1	16
图 2-23. Cmake 安装过程 2	17
图 2-24. Cmake 安装过程 3	17
图 2-25. 测试 Cmake 是否安装成功	17
图 2-26. Vscode 选择下载安装	18
图 2-27. Vscode 安装过程 1	18
图 2-28. Vscode 安装过程 2	19
图 2-29. Vscode 安装过程 3	19
图 2-30. Vscode 安装过程 4	19
图 2-31. Vscode 插件安装搜索界面	20
图 3-1. 文件组织结构图	21
图 4-1. Vscode 中运行任务	28
图 4-2. Cmake 生成 makefile 的构建过程	28
图 4-3. makefile 生成可执行文件的构建过程	28
图 4-4. build 目录中文件组织结构	29
图 4-5. 一键编译和下载过程	29
图 4-6. Vscode 下调试界面	30
图 4-7. 调试界面中添加断点并运行	31
图 4-8. 查看外设寄存器及变量值	31



表索引

表 3-1. 根	目录下 CMakeLists.txt 代码 2	22
表 3-2. 根	目录下 Cortex-M3.cmake 代码2	22
表 3-3. gd_	_libs 中 CMakeLists.txt 代码	23
表 3-4. src	:中 CMakeLists.txt 代码	<u>2</u> 4
表 4-1. tas	ks.json 文件中代码	26
表 4-2. lau	inch.json 文件中代码	27
表 5-1. 版	本历史	33



1. 简介

在进行工程编译过程中,大多都会用到 make 工具,常用的 make 工具包括 GNU Make、qmake、 MS nmake 和 Makeepp 等,这些 make 工具在不同的平台上编写 makefile 时都遵循着不同的 规范、标准和格式,在开发过程中将工程移植到不同平台进行编译将会造成编译失败,使用 CMAKE 工具生成 Makefile 将可以有效解决上述问题,本应用手册基于 GD32F10x SDK 开发 包,使用 CMAKE 工具搭建编译环境。



2. 开发环境搭建

开发环境介绍主要如下:

- 硬件开发板: GD32F103C-EVAL-V1.0 开发板
- Cortex-M3: GD32F103C
- 操作系统: Win10-64 位
- 交叉编译工具链: gcc-arm-none-eabi
- C/C++编译器: MinGW
- 开发环境: VSCODE+CMAKE
- 调试下载工具: OPENOCD

2.1. 安装交叉编译工具

GNU Tools for Arm Embedded Processors 下载安装地址: <u>https://developer.arm.com/tools-</u> and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm/downloads

页面中有多个版本的 GUN Arm Embedded Toolchain 供选择下载,在本应用手册中选择下载 安装 gcc-arm-none-eabi-10-2020-q4-major-win32.exe,如*图 2-1. GUN Arm Embedded* <u>Toolchain 选择下载</u>所示。

arm Developer		IP Products \checkmark	Tools and Software \checkmark	Architectures ~	Solutions \sim	Community ~
	Overview GNU-A • GNU-RM • Architecture Support Specifications					
	What's new in 10-2020-q4-major In this release:					
	gcc-arm-none-eabi-10-2020-q4-major-win32.exe Windows 32-bit Installer (Signed for Windows 10 and later) (Formerly SHA2 signed binary) MD5: 41e9514904a1ee43d4f7882b47bc0294					
	2 gcc-arm-none-sabi-10-2020-q4-major-win32.zip Windows 32-bit ZIP package MDS: 5ee6542a2at847934177bc8fa1294cDd					
	gcc-arm-none-eabi-10-2020-g4-major-x86_64-linux.tar.bz2 Linux x86_64 Tarball MD5: 8312c4c9179988512226663fc81f9a31					
	gcc-arm-none-eabi-10-2020-q4-major-aarch64-linux.tar.bz2 Linux.AArch64 Tarball MDS: 1c3b8944c026d50362eef1f01f329a8e					

图 2-1. GUN Arm Embedded Toolchain 选择下载

下载完成后双击安装,选择"OK"。

图 2-2. GUN Arm Embedded Toolchain 安装过程 1



Please select a language.	
Chinese (Simplified)	~
OK Cancel	

点击下一步。

图 2-3. GUN Arm Embedded Toolchain 安装过程 2



点击"我接受",选择安装路径,选择默认路径即可点击安装。

图 2-4. GUN Arm Embedded Toolchain 安装过程 3





图 2-5. GUN Arm Embedded Toolchain 安装过程 4

🌍 GNU Arm Embedded Toolchain 10-2020-q4-major 🦳 🗌 🗙
选择安装位置 选择"GNU Arm Embedded Toolchain 10-2020-q4-major 10 2020"的安装文 🕡 件夹。
Setup 将安装 GNV Arm Embedded Toolchain 10-2020-q4-major 10 2020 在下列文件 夹。要安装到不同文件夹,单击[浏览(B)]并选择其他的文件夹。 单击[安装(I)] 开始安装进程。
目标文件夹
C:\Program Files (x86)\GNU Arm Embedded Toolchain\1(浏览(B)
所需空间: 697.9MB 可用空间: 39.8GB
Nullsoft Install System v2.51-1 < <u>上一步(P)</u> 安装(I) 取消(C)

在安装完成时,勾选"Add path to environment variable"选项,点击"完成"。

图 2-6. GUN Arm Embedded Toolchain 安装过程 5





检查是否安装成功,在运行中输入 cmd,点击确定,在命令行中输入 arm-none-eabi-gcc-v,返回结果如<u>图 2-7.测试 GUN Arm Embedded Toolchain 是否安装成功</u>所示代表安装成功。

图 2-7. 测试 GUN Arm Embedded Toolchain 是否安装成功

C:\Windows\system32\cmd.exe	-		×
Microsoft Vindows [版本 10.0.19042.631] (c) 2020 Microsoft Corporation, 保留所有权利。			
C:\Users\chran\arm-none-eabi-gcc -v Using built-imspecs. OLLECT_Commone-eabi-gcc			
COLLECT_LTO_WRAPPER=c:/program\ files\ (x86)/gnu\ arm\ embedded\ toolchain/10\ 2020-q4-major/bin//lib/gc i/10.2.1/lto-wrapper.exe	c/arm	-none-	eab
Target: arm-nome-eabi Configured with: /mnt/workspace/workspace/GCC-10-pipeline/jenkins-GCC-10-pipeline-48_20201124_1606180641/s re -build=x86_64-linux=gnu -host=i686-w64-mingw32target=arm-nome-eabi -r_prefix=/mnt/workspace/workspa line/jenkins=GCC-10-pipeline-48_20201124_1606180641/install-mingw/libinfodi==/mnt/workspace/workspace/GCC-10 ins=GCC-10-pipeline-48_20201124_1606180641/install-mingw/libinfodi==/mnt/workspace/workspace/GCC-10 ins=GCC-10-pipeline-48_20201124_1606180641/install=mingw/libinfodi==/mnt/workspace/workspace/GCC-10 ins=GCC-10-pipeline-48_20201124_1606180641/install=mingw/share/doc/gcc-arm-nome-eabi in=/mnt/workspace/workspace/GCC-10-pipeline-48_20201124_1606180641/install-mingw/share/doc/gcc-arm-nome-eabi in=/mnt/workspace/workspace/GCC-10-pipeline/jerkins=GCC-10-pipeline-48_20201124_1606180641/install-mingw/share/doc/gcc-arm-nome-eabi/dfenable-languages=c,c+tenable-lingw-dicage-1cdesable-libsqu	rc/gcc ce/GCC C-10-r)-pipel orkspac bi/mar share/c 5061806 11-f1cs xx-pch	c/conf: D-10-p: Dipelin line/j ce/worl nhtr doc/gco 041/in atd hdi	igu ne/ enk ksp mld c-a sta isa
le-nisdisable-shareddisable-threadsdisable-tlswith-gru-sswith-gru-idwith-headers=yesv ith-python-dirsbare/gcc-arm.none-eabiwith-systonef.mut/workspace/workspace/workspace/workspace/c0C-10- s=CC-10-pipeline-48_20201124_1606180641/build-mingw/host-libs/usrwith-gru-fmt/workspace/workspace/c0C enkins=CC-10-pipeline-48_20201124_1606180641/build-mingw/host-libs/usrwith-gru-fmt/workspace/workspace/c0C line/jenkins=CC-10-pipeline-48_20201124_1606180641/build-mingw/host-libs/usrwith-gru-fmt/workspace/workspace/workspace/c0C constraints=CC-10-pipeline-48_20201124_1606180641/build-mingw/host-libs/usrwith-gru-fmt/workspace/workspace/workspace/c0C constraints=CC-10-pipeline-48_20201124_1606180641/build-mingw/host-libs/usrwith-gru-fmt/workspace/workspace/ CC-10-pipeline/jenkins=CC-10-pipeline-48_20201124_1606180641/build-mingw/host-libs/usrwith-ibe/fmt/ kspace/GCC-10-pipeline/jenkins=GC-10-pipeline-48_20201124_1606180641/build-mingw/host-libs/usrwith-ibe/srwith-ibe/sr static-libecc -W.=Static1stdc++Ddynamic -lmwith-gruesion=GNU Arm Embedded Toolchain 10-2020-	ith-ne ipelir -10-pi ce/GCO rkspac ce/works t-1ibs q4-majo	ewlib .pelin ipelin C-10-p: ce/GCC ckspace space/ stdcxx or'	w e-4 kin e/j e/G wor ='- wit
h-multilib-list=rmprofile, aprofile Thread model: single			
Supported LTO compression algorithms: zlib gcc version 10.2.1 20201103 (release) (GNU Arm Embedded Toolchain 10-2020-q4-major)			
C:\Users\chran>_			

2.2. 安装 C/C++ MinGW 编译器

MinGW 安装包下载地址: <u>https://sourceforge.net/projects/mingw-w64/files/mingw-w64/mingw-w64-release/</u>,可以选择在线安装和离线安装两种方式,本手册选择在线安装,下载 MinGW-W64-install.exe。

图 2-8. MinGW-W64 选择下载安装



SOURCE FORGE	Open Source Software	Business Software	Resources
otals: 119 Items	572	.2 MB	32,550
MinGW-W64 Online Installer			
MinGW-W64-install.exe			
MinGW-W64 GCC-8.1.0			
• x86_64-posix-sjlj			
x86_64-posix-seh			
 x86_64-win32-sjlj 			
• x86_64-win32-seh			
IOSO-POSIX-SJIJ IGRA posix dworf			
1686-win32-sili			
• i686-win32-dwarf			
MinGW-W64 GCC-7.3.0			
• x86_64-posix-sjlj			
• x86_64-posix-seh			
• x86_64-win32-sjlj			
 x86_64-win32-seh 			
• i686-posix-sjlj			
1000-posix-dwart 1686-wip22-sili			
 i686-win32-dwarf 			
MinGW-W64 GCC-6.4.0			
• x86_64-posix-sjlj			

下载"完成"后,双击安装。选择"下一步"。

图 2-9. MinGW-W64 安装过程 1

遏 Installing MinGW	/-W64	_		×
	Welcome to the M Installation!	inGW-W64		
	Welcome to the MinGW-W64 o	nline installer.		
MinGW-W64		Next >	Cancel	

配置选择如下,点击"下一步"。

图 2-10. MinGW-W64 安装过程 2



🔏 Installing x86_64	4-8.1.0-win32-se	h-rt_v6-rev0		_		(
Settings Specify setup	settings.					
	Version Architecture Threads Exception Build revision	8.1.0 x86_64 win32 seh 0				
— MinGW-W64 ——		< Back	Next >		Cancel	

选择安装路径,选择默认路径。

图 2-11. MinGW-W64 安装过程 3

🖓 Installing x86_64-8.1.0-win32-seh-rt_v6-rev0 — 🗌 🗙	
Installation folder	
Select a destination folder where x86_64-8.1.0-win32-seh-rt_v6-rev0 will	
Setup will install files in the following folder.	
If you would like to install x86_64-8.1.0-win32-seh-rt_v6-rev0 into a different folder, click Browse and select another folder.	
Destination folder	
C:\Program Files\mingw-w64\x86_64-8.1.0-win32-seh-r Browse	
Space available: 38.87 GB	
✓ Create shortcuts in Start Menu	
- MinGW-W64	
< Back Next > Cancel	

等待下载文件过程。

图 2-12. MinGW-W64 安装过程 4



🛃 Installing x86_64-8.1.0-win32-seh-r	rt_vб-rev0	_			×
Installing Files Copying x86_64-8.1.0-win32-se	h-rt_v6-rev0 files to	o your compute	er.		
To stop or pause the installatio	on process, click Ca	ncel.			
Direcory: C:\Program F File: build-info.txt	iles\mingw-w64\\	mingw64			
– MinGW-W64 –		Next >		Cance	-

点击"Next"。

图 2-13. MinGW-W64 安装过程 5

🖓 Installing x86_64-8.1.0-win32-seh-rt_v6-rev0	_		Х
Installing Files			
Copying x86_64-8.1.0-win32-seh-rt_v6-rev0 files to your comp	uter.		◯,
Click Next to continue the installation.			
— MinGW-W64			
Next >		Cance	I

点击"Finish",完成"安装"。

图 2-14. MinGW-W64 安装过程 6



🔏 Installing x86_6	4-8.1.0-win32-seh-rt_v6-rev0	_		×
	x86_64-8.1.0-win32-seh-rt_ been successfully installed! Click Finish to complete the installation.	_v6-r	ev0 h	las
MinGW-W64			Finish	

在 cmd 命令行中输入 gcc-v, 输出不可运行程序, 需要将下载的文件加入到系统环境变量中。

图 2-15. 测试 MinGW-W64 是否安装成功 1



打开 MinGW 安装路径,复制/bin 文件夹下路径,将该路径添加到系统环境变量中。 图 2-16. 复制 MinGW-W64 bin 文件夹路径



Е

AN016 基于 Cmake 工具搭建 GD32MCU 开发环境

📙 🕑 📙 🖛 bii	n					
文件 主页	共享 3	臣看				
	010	ella ti ca ocica				_
< → × T	C:\Progr	am Files\mingw-w64\x86_64-	8.1.0-win32-seh-rt_v6-rev0\mingw64\bin			
L		名称	修改日期	类型	大小	
🖈 快速访问		📧 add-2line ave	2019 /5 /12 1.26	命用程序	1.021 KR	
三 桌面	*	ac or o	2010/5/13 1:20	应用程序	1,051 KB	
👆 下戴	*	ar.exe	2010/5/13 1:20		1,055 KB	
🔮 文档	*	as.exe	2010/5/13 1:20	应用程序	1,000 KD	
	*	C++.exe	2010/3/15 4:14	应用程序	1,914 ND	
	<u>_</u>	c++m.exe	2018/5/13 1:20	应用程序	1,030 KB	
SOIL	π	cpp.exe	2018/5/13 4:15	应用程序	1,912 KB	
🔡 视频		III diitool.exe	2018/5/13 1:20	应用程序	1,087 KB	
♪ 音乐		diiwrap.exe	2018/5/13 1:26	应用但序	52 KB	
		dwp.exe	2018/5/13 1:26	应用程序	3,049 KB	
		elfedit.exe	2018/5/13 1:26	№用程序	38 KB	
🛄 此电脑		III g++.exe	2018/5/13 4:14	应用程序	1,914 KB	
A		gcc.exe	2018/5/13 4:15	应用程序	1,911 KB	
📑 网络		gcc-ar.exe	2018/5/13 4:15	应用程序	58 KB	
		gcc-nm.exe	2018/5/13 4:15	应用程序	58 KB	
		📧 gcc-ranlib.exe	2018/5/13 4:15	应用程序	58 KB	
		📧 gcov.exe	2018/5/13 4:15	应用程序	1,503 KB	
		📧 gcov-dump.exe	2018/5/13 4:15	应用程序	1,370 KB	
		gcov-tool.exe	2018/5/13 4:15	应用程序	1,425 KB	
		📧 gdb.exe	2018/5/13 5:27	应用程序	59 KB	
		📧 gdborig.exe	2018/5/13 5:27	应用程序	9,372 KB	
		gdbserver.exe	2018/5/13 5:27	应用程序	495 KB	
		gendef.exe	2018/5/13 4:16	应用程序	55 KB	
		📧 genidl.exe	2018/5/13 4:16	应用程序	72 KB	
		📧 genpeimg.exe	2018/5/13 4:17	应用程序	28 KB	
		i gfortran.exe	2018/5/13 4:14	应用程序	1,913 KB	
		gprof.exe	2018/5/13 1:26	应用程序	1.094 KB	

此电脑右键选择属性,点击高级选择环境变量。

图 2-17. 添加 MinGW-W64 环境变量 1



选择系统变量,点击"Path"。

图 2-18. 添加 MinGW-W64 环境变量 2



文里	值			
OneDrive	C:\Users\chran\OneDrive			
OneDriveConsumer	C:\Users\chran\OneDrive			
Path	C:\Program Files (x86)\GNU Arm Embedded Toolchain\10 20			
TEMP	C:\Users\chran\AppData\Local\Temp			
ТМР	C:\Users\chran\AppData\Local\Temp			
	新建(N) 编辑(E) 删除(D)			
<u> </u>				
统变量(S)				
统变量(S) 变量	值			
统变量(S) 变量 ComSpec	值 C:\Windows\system32\cmd.exe			
統变量(S) 变量 ComSpec DriverData	值 C:\Windows\system32\cmd.exe C:\Windows\System32\Drivers\DriverData			
統变量(S) 变量 ComSpec DriverData NUMBER_OF_PROCESSORS	值 C:\Windows\system32\cmd.exe C:\Windows\System32\Drivers\DriverData 4			
統变量(S) 变量 ComSpec DriverData NUMBER_OF_PROCESSORS OS	值 C:\Windows\system32\cmd.exe C:\Windows\System32\Drivers\DriverData 4 Windows NT			
統变量(S) 变量 ComSpec DriverData NUMBER_OF_PROCESSORS OS Path	值 C:\Windows\system32\cmd.exe C:\Windows\System32\Drivers\DriverData 4 Windows NT C:\Windows\system32;C:\Windows;C:\Windows\System32\Wb			
統变量(S) 变量 ComSpec DriverData NUMBER_OF_PROCESSORS OS Path PATHEXT	值 C:\Windows\system32\cmd.exe C:\Windows\System32\Drivers\DriverData 4 Windows NT C:\Windows\system32;C:\Windows;C:\Windows\System32\Wb .COM;EXE;BAT;:CMD;:VBS;:VBE;JS;;JSE;:WSF;:WSH;:MSC			
統变量(S) 变量 ComSpec DriverData NUMBER_OF_PROCESSORS OS Path PATHEXT PROCESSOR_ARCHITECT	值 C:\Windows\system32\cmd.exe C:\Windows\System32\DriverS\DriverData 4 Windows NT C:\Windows\system32;C:\Windows;C:\Windows\System32\Wb .COM;EXE;BAT;CMD;:VBS;:VBE;JS;JSE;WSF;:WSH;IMSC AMD64			
統变量(S) 变量 ComSpec DriverData NUMBER_OF_PROCESSORS OS Path PATHEXT PROCESSOR_ARCHITECT	值 C:\Windows\system32\cmd.exe C:\Windows\System32\DriverS\DriverData 4 Windows NT C:\Windows\system32;C:\Windows;C:\Windows\System32\Wb .COM;EXE;BAT;CMD;VBS;VBE;JS;JSE;WSF;WSH;MSC AMD64			

点击"新建",粘贴复制的路径,点击"确定"。

图 2-19. 添加 MinGW-W64 环境变量 3

编辑环境变量	×
%SystemRoot%\system32	新建(N)
%SystemRoot%	
%SystemRoot%\System32\Wbem	编辑(E)
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\	
%SYSTEMROOT%\System32\OpenSSH\	浏览(B)
C:\Program Files\mingw-w64\x86_64-8.1.0-win32-seh-rt_v6-rev0	
	删除(D)
	上移(U)
	下移(O)
	编辑文本(T)
· · · · · · · · · · · · · · · · · · ·	取迷
URL URL	+tX/F3

在 cmd 中输入 gcc -v,输出 gcc -v 版本号,安装成功。 图 2-20. 测试 MinGW-W64 是否安装成功 2





在 MinGW 安装路径/bin 文件夹下将 mingw32-make.exe 文件复制拷贝并重命名为 make,后续使用该命令,直接输入 make 实现。

本地理豊 (C) > Program Files > mingw-w64 > x86_64-8.1.0-win32-seh-rt_v6-rev0 > mingw64 > bin > 本地理豊 (C) > Program Files > mingw-w64 > x86_64-8.1.0-win32-seh-rt_v6-rev0 > mingw64 > bin							
名称 ^	修改日期	类型	大小	名称	修改日期	英型	大小
	2010/5/12 416	where the tasks of	55 KD	Candef eve	2018/5/13 4:16	立田昭成	55 K
gender.exe	2018/5/13 4:10	山田田市	33 KB	I genidlexe	2018/5/13 4:16	应用程序	72 8
gendieze	2010/5/13 4:10	立ちます	22 KB	genpeima.exe	2018/5/13 4:17	应用程序	28 1
e genpening.exe	2010/5/13 4:17	立ちます	1.012 KP	ofortran.exe	2018/5/13 4:14	应用程序	1.913
giorianexe	2010/5/13 4.14	立ちます	1,913 KD	aprof.exe	2018/5/13 1:26	应用程序	1.094
I d hfd ava	2010/5/13 1.20	101/13/00/3*	1,054 KD	Id.bfd.exe	2018/5/13 1:26	应用程序	1.455
Id ave	2010/5/13 1.20	101/13/00/3*	1.455 KD	Id.exe	2018/5/13 1:26	应用程序	1.455
ld cold eve	2010/5/13 1:20	112.7734203*	5.027 KB	Id.gold.exe	2018/5/13 1:26	应用程序	5.037
a Ebstomic 1 dll	2010/5/13 1:20		22 KB	libatomic-1.dll	2018/5/13 4:15	应用程序扩展	321
Bhass s set 1 dll	2010/5/13 4:15		77 KB	libgcc s seh-1.dll	2018/5/13 4:15	应用程序扩展	77)
Bhaoma 1 dl	2010/5/12 4:15		152 KB	libgomp-1.dll	2018/5/13 4:15	应用程序扩展	153
libguadmath-0.dll	2010/5/12 4:15	山田田市が周	227 KB	libguadmath-0.dll	2018/5/13 4:15	应用程序扩展	327
Bibsso-0.dll	2010/5/12 4:15	山田辺市り周	17 KB	libssp-0.dll	2018/5/13 4:15	应用程序扩展	17
Bhetdess 6 dll	2010/5/12 4:15	山田田田市が開	1 200 KD	ibstdc++-6.dll	2018/5/13 4:15	应用程序扩展	1,388
libwingthread-1.dll	2018/5/13 4:15	应用程序扩展	51 KB	libwinpthread-1.dll	2018/5/13 4:15	应用程序扩展	51
mingw32-make.exe	2018/5/13 5:28	应用程序	216 KB	💽 make.exe	2018/5/13 5:28	应用程序	216
nm.exe	2018/5/13 1:26	应用程序	1,042 KB	i mingw32-make.exe	2018/5/13 5:28	应用程序	216 1
objcopy.exe	2018/5/13 1:26	应用程序	1,206 KB	III nm.exe	2018/5/13 1:26	应用程序	1,042
] objdump.exe	2018/5/13 1:26	应用程序	2,232 KB	objcopy.exe	2018/5/13 1:26	应用程序	1,206
anlib.exe	2018/5/13 1:26	应用程序	1,055 KB	📧 objdump.exe	2018/5/13 1:26	应用程序	2,232 1
readelf.exe	2018/5/13 1:26	应用程序	568 KB	📧 ranlib.exe	2018/5/13 1:26	应用程序	1,055
size.exe	2018/5/13 1:26	应用程序	1,032 KB	📧 readelf.exe	2018/5/13 1:26	应用程序	568
strings.exe	2018/5/13 1:26	应用程序	1,032 KB	📧 size.exe	2018/5/13 1:26	应用程序	1,032
🗉 strip.exe	2018/5/13 1:26	应用程序	1,206 KB	strings.exe	2018/5/13 1:26	应用程序	1,032
🗄 widl.exe	2018/5/13 4:17	应用程序	428 KB	📰 strip.exe	2018/5/13 1:26	应用程序	1,206
windmc.exe	2018/5/13 1:26	应用程序	1,053 KB	widLexe	2018/5/13 4:17	应用程序	428
windres.exe	2018/5/13 1:26	应用程序	1,140 KB	i windmc.exe	2018/5/13 1:26	应用程序	1,053
🗈 x86_64-w64-mingw32-c++.exe	2018/5/13 4:14	应用程序	1,914 KB	i windres.exe	2018/5/13 1:26	应用程序	1,140
x86_64-w64-mingw32-g++.exe	2018/5/13 4:14	应用程序	1,914 KB	1 x86_64-w64-mingw32-c++.exe	2018/5/13 4:14	应用程序	1,914
x86_64-w64-mingw32-gcc.exe	2018/5/13 4:15	应用程序	1,911 KB	1 x86_64-w64-mingw32-g++.exe	2018/5/13 4:14	应用程序	1,914
x86_64-w64-mingw32-gcc-8.1.0.exe	2018/5/13 4:15	应用程序	1,911 KB	E x86_64-w64-mingw32-gcc.exe	2018/5/13 4:15	应用程序	1,911
x86_64-w64-mingw32-gcc-ar.exe	2018/5/13 4:15	应用程序	58 KB	x86_64-w64-mingw32-gcc-8.1.0.exe	2018/5/13 4:15	应用程序	1,911 (
x86_64-w64-mingw32-gcc-nm.exe	2018/5/13 4:15	应用程序	58 KB	x86_64-w64-mingw32-gcc-ar.exe	2018/5/13 4:15	应用程序	58
x86_64-w64-mingw32-gcc-ranlib.exe	2018/5/13 4:15	应用程序	58 KB	x86_64-w64-mingw32-gcc-nm.exe	2018/5/13 4:15	应用程序	58 1
x86 64-w64-mingw32-gfortran.exe	2018/5/13 4:14	应用程序	1.913 KB	1 x80_04-w04-mingw32-gcc-ranlib.exe	2018/5/13 4:15	应用程序	58 K

图 2-21. 修改 MinGW-W64 mingw32-make 命令为 make 命令

2.3. 安装 Cmake 工具

Cmake 工具下载安装地址: <u>https://cmake.org/download/</u>。

在本应用手册中选择下载安装版本为 3.20.1,安装过程如下所示。

下载完成后,双击安装。选择下一步。

图 2-22. Cmake 安装过程 1





选择 "Add Cmake to system PATH for all users",点击 "Next",选择默认路径,点击 "Next"。

图 2-23. Cmake 安装过程 2

🛃 Install Options – 🗆 🗙	g∰ CMake Setup — □ ×
Install Options Choose options for installing CMake 3.20.1	Destination Folder Click Next to install to the default folder or click Change to choose another.
By default CMake does not add its directory to the system PATH.	Instal CMake to:
Do not add CMake to the system PATH Add CMake to the system PATH for all users Add CMake to the system PATH for the current user Create CMake Desktop Icon	C:\Program Files\CMake\ Change
Back Next Cancel	Back Next Cancel

点击"Install",等待安装完成,点击"Finish"完成安装。

图 2-24. Cmake 安装过程 3

	🛃 CMake Setup —	×	🖞 CMake Setup	- 🗆 🗙
	Ready to install CMake			Completed the CMake Setup Wizard
2	Click Install to begin the installation. Click Back to review or change any of you installation settings. Click Cancel to exit the wizard.	r		Click the Finish button to exit the Setup Wizard.
	Back Instal	Cancel		Back Finish Cancel

检查是否安装成功,在运行中输入 cmd,点击确定,在命令行中输入 cmake --version,返回结果如<u>图 2-25. 测试 Cmake 是否安装成功</u>所示代表安装成功。 图 2-25. 测试 Cmake 是否安装成功





2.4. 安装 Vscode 及插件

Vscode 下载安装地址: <u>https://code.visualstudio.com/</u>

在本应用手册中选择下载安装版本为 1.53.0, 下载安装过程如下所示。

图 2-26. Vscode 选择下载安装

💐 Visual Studio Code		pdates	Blog		Extensions					。 Search Docs		± I	Download	
			.57 is nov	w avai	lable! Read a	about the	e new featur	es and fixe	es from May					
Code ed	liting	j .		×1 File	Edit Selection EXTENSIONS: MARK		Debug Termi		viceWorkerjs - cre 35. index; viceWorkerjs > 6	ate-react-app - Visual Studic js <mark>JS serviceWorl</mark> dregister > @window.add	9 Code - In – kerjs × ÆventListener(1			
Redefir Free. Built on open source.	ned. Runs ever	ywhere.			Python Linting, I Microsof OftLens Superch Eric Amo	2019.6.24221 Debugging (n t — Git sup arge the Git d dio	¢54.9M ♦45 nulti-threaded, Instal 9.8.5 ¢23.1M ♦5 capabilities buil Instal		checkVal // Add d // servi navigate © product © productSub	LidServiceWorker(swU some additional logg ice worker/PWA docum pr.serviceWorken.rea	<pre>ing to loca entation. dy.then(()</pre>			
Download for Wind Stable Build	dows . Stable	Insiders		8 , ,	C/C++ I C/C++ I Microsof ESLint n Integrate Dirk Baer	124.0 ntelliSense, de t 30 es ESLint Java umer g er for Ch	\$ \$\phi_23M \$\$ 3.5 ebugging, and Instal \$ \$\phi_21.9M \$\$ 4.5 \$		 removeSite removeNebbi requestMec sendBeacor serviceNor storage storeSiteS storeNebbis 	<pre>SpecificTrackingExc LideTrackingExceptio LiaKeySystemAccess </pre>	eption n igator.servi ption			
Windows x64 User Inst Linux x64 .deb .rpm	aller ⊻ ⊻	∓ ∓ ₹			Debug y Microsof Languay Java Lint Red Hat Icons for VSCode	our JavaScrip ge Supp 0. ing, Intellisen icons 0.00 r Visual Studis icons Team	470 PIE/M \$45 ise, formatting, Pital PI/2M \$5 o Code	54 55 57 58 59 68	SuserAgent Vendor action regist avigator.sem .register(su .then(regist	erValidSW(swUrl, con viceWorken vUrl) ration => {				
Other dow	inloads			2003 P master	Vetur 0. Viet tool Pine Wu C# 1216 C# for V Microsof	11.1 ing for VS Co isual Studio C	⇔17M *45 ode Install ⇔154M *4 Code (powered Install	You can r Local: On Your Note that	ow view creat ht Network: ht	e-react-app in the b tp://localhost:3000, tp://l0.211.55.3:30 ent build is not opt Ln.43, Col 19 Space:2	browser. / 00/ timized. UTF-6 LF Ja	vaScript	•	

下载完成后,双击安装。选择"下一步",选择安装路径,点击"下一步"。

图 2-27. Vscode 安装过程 1

		_
💐 安装 - Microsoft Visual Studio Code (User) - 🗌 🗙	💐 安装 - Microsoft Visual Studio Code (User) - 🗌 🗙	
许可协议 维续安装前询阅读下列重要信息。	选择目标位置 您想将 Weual Studio Code 安装在什么地方?	
请仔细阅读下列许可协议。您在继续安装前必须同意这些协议条款。	安装程序将安装 Visual Studio Code 到下列文件夹中。	
<i>此件可進用于 Visual Studio Code 产品</i> 、Visual Studio Code 的源代码可根 据 虹T 许可协议 (<u>https://ithub.com/microsoft/vscode/blob/master/LICENSE.tr</u>) 在以下网 拉获取: <u>https://ithub.com/microsoft/vscode</u> 。有关其他许可信息,请查看 我们的求见问题解答, 网址为 <u>https://code.visualstudio.com/docs/supporting/fag</u> 。 Microsoft 软件许可条数	单击"下一步"建设。如果您想这探其它文件关,单击"消流"。 图 50mm 55mm 9450 014 500 1710 9 mm 96 mm	
MICROSOFT VISUAL STUDIO CODE 这些许可奈素是参与 Microsoft 公司(凱恩所在地的 Microsoft 公司的关联公司) >		
 ● 我问意此协议(A) ○ 我不同意此协议(D) 	至少需要有 294.3 MB 的可用磁盘空间。	
下一步(M)> 取消	<上一步(8) 下一步(9) 取消	



一直点击"下一步",即可完成安装。

图 2-28. Vscode 安装过程 2

🗙 安装 - Microsoft Visual Studio Code (User)	>	ズ 安装 - Microsoft Visual Studio Code (User) ー	• ×
这程开始菜单文件夹 您想在哪里放置程序的快捷方式?	×	法程用加任务 您想要安装程序执行哪些附加任务?	×
安美程序现在将在下列开始菜单文件夹中创建程序的快捷方式。		选择您想要安装程序在安装 Visual Studio Code 时执行的附加任务,然后单击"下一步"。	
单击"下一步"继续。如果您想选择其它文件夹,单击"浏览"。		附加快捷方式:	
Visual Studio Code	浏览(R)	□ 创建桌面快撞方式(D)	
		其他:	
		□ 将"通过 Code 打开"操作添加到 Windows 资源管理器文件上下文菜单	
		□ 将 通过 Code 打井 操作添加到 Windows 资源管理器目录上下又采单	
		□ 村 CODE 注册/J文文村III文件类型IIJ网络G ☑ 添加到 PATH (重启后生效)	
□不划建升始菜里又件夹(D)		_	
<上一步(8) 下一步	(N) > 取消	<上一步(6) 下一步(1) >	取消

图 2-29. Vscode 安装过程 3

💐 安装 - Microsoft Visual Studio Code (User) - 🗆 🗙	💐 安装 - Microsoft Visual Studio Code (User) - 🗆 🗙
准备安装 安装程序现在准备开始安装 Visual Studio Code 到您的纯脑中。	正在安装 安装程序正在安装 Visual Studio Code 到位的电脑中,请稍等。
单击 安装 "独缘此安装程序。如果念想要回顾或改变设置,请单击"上一步"。 目标位置: C. Users/bran/uppData/Local/Programs/Microsoft VS Code 开始至文性表: Visual Studio Code 附加任务: 著 " " 法加到 PATH (距启后主效)	正在第任第文件 C:\Users\dran\bpCData\Local\Programs\Hicrosoft V5 Code\Code.exe
<上一步(6) 安裝(1) 取消	取消

图 2-30. Vscode 安装过程 4

💐 安装 - Microsoft Visual Studio Code (User) - 🗆 🗙	
	Visual Studio Code 安装完成
	安装程序已在您的电脑中安装了 Visual Studio Code。此应用程序可以 通过选择安装的快捷方式运行。
	单击"完成"退出安装程序。
	☑ 运行 Visual Studio Code
	完成(F)

为了满足开发要求,还需要安装部分插件模块,打开安装程序 Vscode,选择左侧插件中心,分别搜索和安装如下插件,搜索到插件点击安装即可。本应用手册安装插件及版本号如下:

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



图 2-31. Vscode 插件安装搜索界面



2.5. 安装 Openocd

支持 GD32MCU 的 Openocd 地址为: <u>https://github.com/GigaDevice-Semiconductor/openocd</u>。

用户根据要求完成 Openocd 的编译后生成可执行文件 openocd.exe,并将该文件加入环境变量 中,加入环境变量的方法可参考 2.2 安装 C/C++ MinGW 编译器。



3. CmakeLists 文件编写

本应用手册文件组织结构如图 3-1. 文件组织结构图所示,下面分别介绍各文件夹中的内容:

CMakeLists.txt Cortex-M3. cmake openocd_gdlink.cfg readme.txt -huild -gd_libs CMakeLists.txt -GD32F10x L └—Firmware -CMSIS core_cm3.h core_cmFunc.h core_cmInstr.h gd32f10x.h system_gd32f10x.c system_gd32f10x.h -gcc_startup startup_gd32f10x_md.S Peripherals -inc L_src -inc gd32f103c_eval.h gd32f10x_it.h gd32f10x_libopt.h systick.h -ldscripts gd32f10x_flash.ld src CmakeLists.txt gd32f103c_eval.c gd32f10x_it.c main.c systick.c

图 3-1. 文件组织结构图

在 build 文件夹中存放编译过程中所产生的中间文件、库文件和最终的可执行文件;

gd_libs 文件夹中存放 GD 库文件,主要包括外设库文件、启动文件以及部分头文件;

inc 文件夹中存放用户编写代码的头文件以及开发板相关的头文件;

ldscripts 文件夹中存放链接脚本文件;

src 文件夹中存放用户编写代码的.c 文件以及开发板相关的.c 文件。

其中在根目录、gd_libs 和 src 文件夹下分别包含 CMakeLists.txt 文件,以及在根目录下包含 Cortex-M3.cmake 文件,下面分别介绍各文件中 CmakeLists.txt 的内容。



3.1. 根目录下 CMakeLists.txt 及 Cortex-M3.cmake 文件

根目录下 CMakeLists.txt 文件的内容如<u>表 3-1. 根目录下 CMakeLists.txt 代码</u>。 表 3-1. 根目录下 CMakeLists.txt 代码

#设置 CMake 最低支持版本

cmake_minimum_required(VERSION 3.17)

SET(PRJ_NAME "GD32F10x")

#定义工程名称

project(\${PRJ_NAME})

#引用 Cortex-M3.cmake

include(Cortex-M3.cmake)

#添加目录 gd_libs 和 src

add_subdirectory(gd_libs)

add_subdirectory(src)

Cortex-M3.cmake 文件的内容如<u>表 3-2. 根目录下 Cortex-M3.cmake 代码</u>,用户可通过修 改该文件中的参数来修改编译选项,以及添加相关宏定义:

表 3-2. 根目录下 Cortex-M3.cmake 代码

#设置 CMake 最低支持版本
cmake_minimum_required(VERSION 3.17)
#Cmake 交叉编译配置
SET(CMAKE_SYSTEM_NAME Generic)
#设置支持 ASM
ENABLE_LANGUAGE(ASM)
#Debug 模式
SET(CMAKE_BUILD_TYPE "Debug")
#Release 模式
#SET(CMAKE_BUILD_TYPE "Release")
#设置 C 编译工具
SET(CMAKE_C_COMPILER arm-none-eabi-gcc)
#ELF 转换为 bin 和 hex 文件工具
SET(CMAKE_OBJCOPY arm-none-eabi-objcopy)
#查看文件大小工具
SET(CMAKE_SIZE arm-none-eabi-size)
#设置浮点选项
SET(MCU_FLAGS "-mcpu=cortex-m3 -mfloat-abi=softfp -mfpu=fpv4-sp-d16")
#设置警告相关信息
SET(CMAKE_C_FLAGS "\${MCU_FLAGS} -w -Wno-unknown-pragmas")
#设置调试选项
SET(CMAKE_C_FLAGS_DEBUG "-O0 -g2 -ggdb")
SET(CMAKE_C_FLAGS_RELEASE "-O3")
#添加宏定义
ADD_DEFINITIONS(
-DGD32F10X HD



)

-DUSE_STDPERIPH_DRIVER

3.2. gd_libs 文件夹中 CMakeLists.txt 文件

gd_libs 中 CMakeLists.txt 文件的内容如 表 3-3. gd_libs 中 CMakeLists.txt 代码</u>所示,该文件主要实现将 GD 外设固件库文件以及 CMSIS 相关文件和启动文件生成 gd32_lib 库,并指定生成位置:

```
表 3-3. gd_libs 中 CMakeLists.txt 代码
#设置相关路径变量
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)
 #添加头文件搜索路径
 include_directories(
    ${CORE_SUPPORT_DIR}
    ${PERIPHERALS_DIR}/inc
    ${PROJECT_SOURCE_DIR}/inc
)
 #设置启动文件变量
SET(START_UP_ASM startup_gd32f10x_md.S)
#设置启动文件 C 属性
 set_property(SOURCE ${START_UP_DIR}/${START_UP_ASM} PROPERTY LANGUAGE C)
 # GLOB 选项将会为所有匹配查询表达式的文件生成一个文件 list,并将该 list 存储进变量所定义的
 STD_LIB,SRC_CORE 里
file(GLOB STD_LIB ${PERIPHERALS_DIR}/src/*.c)
 file(GLOB SRC_CORE ${CORE_SUPPORT_DIR}/*.c)
 #生成库目标 gd32_lib
 add_library(gd32_lib
    ${STD_LIB}
    ${SRC_CORE}
    ${START_UP_DIR}/${START_UP_ASM}
 )
 #设置库输出的名称
 set_target_properties(gd32_lib PROPERTIES OUTPUT_NAME "gd32_lib")
 #设置库文件的默认输出路径
 SET(LIBRARY_OUTPUT_PATH ${PROJECT_SOURCE_DIR}/build/lib)
```

3.3. src 文件夹中 CMakeLists.txt 文件

src 文件夹中 CMakeLists.txt 文件的内容如表 3-4. src 中 CMakeLists.txt 代码所示,该文件



AN016 基于 Cmake 工具搭建 GD32MCU 开发环境 主要实现将用户编写的.c 文件和 gd32_lib 库文件进行链接编译,生成可执行文件,并指定生

王要实现将用尸编与的.C 又件和 gd32_llb 库又件进行链接编译,生成可执行又件。	,开指定生			
 或可执行文件的位置。				
表 3-4. src 中 CMakeLists.txt 代码				
#该路径下所有.c 文件定义为 SRC_LIST				
aux_source_directory(. SRC_LIST)				
#添加头文件搜索路径				
include_directories(
\${PROJECT_SOURCE_DIR}/gd_libs/GD32F10x/Firmware/CMSIS				
\${PROJECT_SOURCE_DIR}/gd_libs/GD32F10x/Firmware/Peripherals/inc				
\${PROJECT_SOURCE_DIR}/inc				
)				
#添加非标准的共享库搜索路径				
link_directories(\${PROJECT_SOURCE_DIR}/build/lib)				
#设置链接文件相关路径和变量				
SET(FLASH_LD_DIR \${PROJECT_SOURCE_DIR}/Idscripts)				
SET(FLASH_LD_FILE gd32f10x_flash.ld)				
SET(LINKER_SCRIPT \${FLASH_LD_DIR}/\${FLASH_LD_FILE})				
#设置链接选项				
SET(CMAKE_EXE_LINKER_FLAGS				
"specs=nano.specs -specs=nosys.specs -T\${LINKER_SCRIPT}	-WI,-			
Map=\${PROJECT_BINARY_DIR}/\${PRJ_NAME}.map,cref -WI,gc-sections")				
#生成目标文件				
add_executable(\${PRJ_NAME}.elf \${SRC_LIST})				
#把目标文件与库文件进行链接				
target_link_libraries(\${PRJ_NAME}.elf gd32_lib)				
#设置可执行文件输出路径				
set(EXECUTABLE_OUTPUT_PATH \${PROJECT_SOURCE_DIR}/build/bin)				
#设置 ELF 转换路径				
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)				
#添加自定义命令实现 ELF 转换 hex 和 bin 文件				
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 \${F	IEX_FILE}			
"\${CMAKE_CURRENT_BINARY_DIR}/\${PRJ_NAME}.hex"				
COMMAND \${CMAKE_COMMAND} -E copy \${	3IN_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. 编译、下载与调试

4.1. 编译与下载

使用 vscode 打开该工程目录,在该目录下会产生.vscode 文件夹,在文件夹中新建 tasks.json 文件 launch.json 文件, tasks.json 文件主要实现不用通过命令行而通过按钮的方式来实现编译和下载功能, launch.json 文件主要是调试配置文件,具体文件内容和注释分别如<u>表 4-1.</u> tasks.json 文件中代码和表 4-2. launch.json 文件中代码所示。

```
表 4-1. tasks.json 文件中代码
 {
     "version": "2.0.0",
       //指定命令执行所在路径
     "options": {
         "cwd": "${workspaceRoot}/build"
     }
     "tasks": [
         {
         //执行 cmake 命令, 生成 makefile 文件
             "type": "shell",
             "label": "cmake",
             "command": "cmake",
             "args": [
                  "-G",
                  "MinGW Makefiles",
                  ".."
             ]
         },
         {
         //执行 make 命令, 生成可执行文件
             "label": "make",
             "type": "shell",
             "command": "make",
             "args": [],
              "group": {
                  "kind": "build",
                  "isDefault": true
             },
              "dependsOn": [
                "cmake"
             ],
             "problemMatcher": []
         },
```



```
{
             //执行 openocd 命令,下载可执行文件到目标 MCU
             "type": "shell",
             "label": "Build & Updatde",
             "command": "openocd",
             "args": [
                 "-f",
            //配置文件绝对路径
                 "E:/Work_Code/10.cmake/Cmake/Code/Example-
 windows/gd32f103C_example/openocd_gdlink_gd32f10x.cfg",
                 "-c",
              //编译生成的可执行文件绝对路径
                 "program E:/Work_Code/10.cmake/Cmake/Code/Example-
 windows/gd32f103C_example/build/bin/GD32F10x.elf verify reset exit"
             ],
             "group": "build",
             "dependsOn": "make"
         }
    ]
}
表 4-2. launch.json 文件中代码
 {
     // 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": [
         {
             //指定命令执行所在路径
             "cwd": "${workspaceRoot}",
             //可执行文件所在路径
             "executable": "build/bin/GD32F10x.elf",
             "name": "Debug Microcontroller",
             "request": "launch",
             "type": "cortex-debug",
             //svd 文件所在路径
             "svdFile": "./GD32F10x_MD.svd",
             //openocd 命令
             "servertype": "openocd",
             //openocd 配置文件所在相对路径
             "configFiles": [
                 "./openocd_gdlink_gd32f10x.cfg ",
             ]
```





在 VScode 中点击 "终端->运行任务",可以看到<u>图 4-1. Vscode 中运行任务</u>选项。

图 4-1. Vscode 中运行任务

选择要运行的任务			
Build & Updatde	最近使用过	63	×
make			
cmake			
🖻 cppbuild		已提	供
🔁 npm			
🖻 typescript			
🔁 grunt			
면 gulp			
🔁 jake			
显示所有任务			

选择 cmake 选项,执行 cmake 命令并生成 makefile 文件,从 vscode 终端打印信息可以看到 makefile 构建过程。

图 4-2. Cmake 生成 makefile 的构建过程



选择 make 选项,执行 make 命令,实现编译功能,并生成可执行文件.elf、.bin 和.hex 文件,从 vscode 终端打印信息可以看到可执行文件生成过程。同时从 build 目录下可以看出生成的 elf、hex 和 bin 文件存放在 bin 目录下。

图 4-3. makefile 生成可执行文件的构建过程



> Executing task. make <
Scanning dependencies of target gd32 lib
[3%] Building C object gd libs/CMakeFiles/gd32 lib.dir/GD32F10x/Firmware/Peripherals/src/gd32f10x adc.c.obj
6%] Building C object gd libs/CMakeFiles/gd32 lib.dir/GD32F10x/Firmware/Peripherals/src/gd32f10x bkp.c.obj
9%] Building C object gd libs/CMakeFiles/gd32 lib.dir/GD32F10x/Firmware/Peripherals/src/gd32f10x can.c.obj
[12%] Building C object gd libs/CMakeFiles/gd32 lib.dir/GD32F10x/Firmware/Peripherals/src/gd32f10x crc.c.obj
[16%] Building C object gd libs/CMakeFiles/gd32 lib.dir/GD32F10x/Firmware/Peripherals/src/gd32f10x dac.c.obj
[19%] Building C object gd libs/CMakeFiles/gd32 lib.dir/GD32F10x/Firmware/Peripherals/src/gd32f10x dbg.c.obj
[22%] Building C object gd libs/CMakeFiles/gd32 lib.dir/GD32F10x/Firmware/Peripherals/src/gd32f10x dma.c.obj
[25%] Building C object gd libs/CMakeFiles/gd32 lib.dir/GD32F10x/Firmware/Peripherals/src/gd32f10x enet.c.obj
[29%] Building C object gd libs/CMakeFiles/gd32 lib.dir/GD32F10x/Firmware/Peripherals/src/gd32f10x exmc.c.obj
[32%] Building C object gd libs/CMakeFiles/gd32 lib.dir/GD32F10x/Firmware/Peripherals/src/gd32f10x exti.c.obj
[35%] Building C object gd libs/CMakeFiles/gd32 lib.dir/GD32F10x/Firmware/Peripherals/src/gd32f10x fmc.c.obj
[38%] Building C object gd libs/CMakeFiles/gd32 lib.dir/GD32F10x/Firmware/Peripherals/src/gd32f10x fwdgt.c.obj
[41%] Building C object gd libs/CMakeFiles/gd32 lib.dir/GD32F10x/Firmware/Peripherals/src/gd32f10x gpio.c.obj
[45%] Building C object gd libs/CMakeFiles/gd32 lib.dir/GD32F10x/Firmware/Peripherals/src/gd32f10x i2c.c.obj
[48%] Building C object gd libs/CMakeFiles/gd32 lib.dir/GD32F10x/Firmware/Peripherals/src/gd32f10x misc.c.obj
51% Building C object gd libs/CMakeFiles/gd32 lib.dir/GD32F10x/Firmware/Peripherals/src/gd32f10x pmu.c.obj
54%] Building C object gd libs/CMakeFiles/gd32_lib.dir/GD32F10x/Firmware/Peripherals/src/gd32f10x rcu.c.obj
[58%] Building C object gd_libs/CMakeFiles/gd32_lib.dir/GD32F10x/Firmware/Peripherals/src/gd32f10x_rtc.c.obj
[61%] Building C object gd_libs/CMakeFiles/gd32_lib.dir/GD32F10x/Firmware/Peripherals/src/gd32f10x_sdio.c.obj
[64%] Building C object gd libs/CMakeFiles/gd32 lib.dir/GD32F10x/Firmware/Peripherals/src/gd32f10x spi.c.obj
[67%] Building C object gd_libs/CMakeFiles/gd32_lib.dir/GD32F10x/Firmware/Peripherals/src/gd32f10x_timer.c.obj
[70%] Building C object gd_libs/CMakeFiles/gd32_lib.dir/GD32F10x/Firmware/Peripherals/src/gd32f10x_usart.c.obj
[74%] Building C object gd_libs/CMakeFiles/gd32_lib.dir/GD32F10x/Firmware/Peripherals/src/gd32f10x_wwdgt.c.obj
[77%] Building C object gd_libs/CMakeFiles/gd32_lib.dir/GD32F10x/Firmware/CMSIS/system_gd32f10x.c.obj
[80%] Building C object gd_libs/CMakeFiles/gd32_lib.dir/GD32F10x/Firmware/CMSIS/gcc_startup/startup_gd32f10x_md.S.obj
[83%] Linking C static library\lib\libgd32_lib.a
[83%] Built target gd32_lib
Scanning dependencies of target GD32F10x.elf
[87%] Building C object src/CMakeFiles/GD32F10x.elf.dir/gd32f103c_eval.c.obj
[96%] Building C object src/CMakeFiles/GD32F10x.elf.dir/gd32f10x_it.c.obj
[93%] Building C object src/CMakeFiles/GD32F10x.elf.dir/main.c.obj
[96%] Building C object src/CMakeFiles/GD32F10x.elf.dir/systick.c.obj
[100%] Linking C executable\bin\GD32F10x.elf
Invoking: Cross ARM GNU Print Size
text data bss dec hex-filename
10552 140 2080 12772 31e4 E:/Work_Code/10.cmake/Cmake/Code/Example-windows/gd32f103C_example/build/bin/GD32F10x
0 10692 0 10692 29c4 E:/Work_Code/10.cmake/Cmake/Code/Example-windows/gd32f103C_example/build/bin/GD32F10x
[100%] Built target GD32F10x.elf

图 4-4. build 目录中文件组织结构



选项 Build&Updata,实现一键 cmake 构建、编译以及 openocd 下载功能,从 vscode 终端打印信息可以看到构建、编译和程序下载过程。

图 4-5. 一键编译和下载过程



> Executing task: cmake -G 'MinGW Makefiles' <
Configuring dono
Compating done
Ruild files have been written to: E:/Work Code/10 cmake/Cmake/Code/Evample_windows/
build files have been writeen to. E./work_code/io.cmake/cmake/code/Example-windows/
终端将被任务重用,按任意键关闭。
> Executing task: make <
[83%] Built target gd32_lib [100%] Built target GD32F10x.elf
终 <u>端将被任务重用,按任意键关</u> 闭。
<pre>> Executing task: openocd -f E:/Work_Code/10.cmake/Cmake/Code/Example-windows/gd32f1030 n/GD32F10x.elf verify reset exit' <</pre>
xPack OpenOCD _ x86_64_Open_On-Chin_Debugger_0_10_0+dev_dirty (2021-03-04-06:52)
Licensed under GNI GPL v2
For bug reports, read
http://openocd.org/doc/doxygen/bugs.html
cortex_m reset_config sysresetreq
Info : CMSIS-DAP: SWD Supported
Info : CMSIS-DAP: JTAG Supported
Info : CMSIS-DAP: FW Version = 2.0.0
Info : CMSIS-DAP: Interface Initialised (SWD)
Info : SWCLK/TCK = 1 SWDIO/TMS = 1 TDI = 1 TDO = 1 nTRST = 0 nRESET = 1
Info : CMSIS-DAP: Interface ready
Info : clock speed 1000 kHz
Info : SWD DPIDR 0x1ba01477
Info : gd32f10x.cpu: hardware has 6 breakpoints, 4 watchpoints
Info : starting gdb server for gd32f10x.cpu on 3333
Info : Listening on port 3333 for gdb connections
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x08002890 msp: 0x2000c000
** Programming Started **
Into : device id = 0x13090414
Info : flash size = 256kbytes
Info : bank-≻num_sectors = 128
Info : bank-≻size = 262144
Info : GD32: Flash erasing
Info : GD32: Flash erase sector erase(0 to 5)
Info : GD32: Flash erasefinished
Info : GD32: Flash writing
Info : bank=0000000028a0da0 buffer=0000000002905580 offset=000000000 count=00003000
Info : GD32: Flash write not words to write, padding with 0xff
Into : GD32: Elash write
** Programming Finished **
** Verity Started **
** Verified OK **
** Resetting Target **
snutaown commana invokea

4.2. 调试

在 VScode 中点击"运行->启动调试",进入调试界面如<u>图 4-6. Vscode 下调试界面</u>所示,可 通过右上角对程序调试进行运行、单步、复位以及终止操作。 图 4-6. Vscode 下调试界面





可以通过在代码左侧加入断点,并执行 Run 运行到断点处。

图 4-7. 调试界面中添加断点并运行



在左边界面可以查看外设寄存器值,可添加变量查看等调试操作。

图 4-8. 查看外设寄存器及变量值







5. 版本历史

表 5-1. 版本历史

版本号.	描述	日期
1.0	首次发布	2021年06月30日



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