

GMD (GOWIN MCU Designer)软件 **用户指南**

SUG549-2.0, 2025-06-13

版权所有 © 2025 广东高云半导体科技股份有限公司

GO 《 N 高云、 《 、Gowin、GOWIN、gowin、GowinSynthesis、高云、云源、小蜜蜂以 及晨熙均为广东高云半导体科技股份有限公司注册商标,本手册中提到的其他任何商标,其 所有权利属其拥有者所有。未经本公司书面许可,任何单位和个人都不得擅自摘抄、复制、 翻译本文档内容的部分或全部,并不得以任何形式传播。

免责声明

本文档并未授予任何知识产权的许可,并未以明示或暗示,或以禁止反言或其它方式授予任 何知识产权许可。除高云半导体在其产品的销售条款和条件中声明的责任之外,高云半导体 概不承担任何法律或非法律责任。高云半导体对高云半导体产品的销售和/或使用不作任何 明示或暗示的担保,包括对产品的特定用途适用性、适销性或对任何专利权、版权或其它知 识产权的侵权责任等,均不作担保。高云半导体对文档中包含的文字、图片及其它内容的准 确性和完整性不承担任何法律或非法律责任,高云半导体保留修改文档中任何内容的权利, 恕不另行通知。高云半导体不承诺对这些文档进行适时的更新。

版本信息

日期	版本	说明
2019/07/29	1.0	初始版本。
2019/09/12	1.1	 集成 GNU ARM 和 RISC-V MCU 编译工具链; 更新 Eclipse 插件为 Eclipse 网站最新版本; 更新 GNU ARM MCU 编译工具链为 GNU 网站最新版本; 更新优化界面配置。
2019/11/12	1.2	 更新 GNU RISC-V MCU 编译工具链为 GNU 网站最新版本; IDE 内部集成 Java Development Kit,简化 IDE 安装流程; IDE 内部集成高云下载软件 Programmer; IDE 界面完全高云定制化,简化界面选项配置,提高 IDE 易用性; 解决已知的 IDE 安装和使用问题。
2019/12/10	1.3	 License 管理; IDE 界面优化。
2020/06/01	1.4	 支持 Gowin_EMPU(GW1NS-4C)和 Gowin_EMPU_M3 编程设 计、编译、下载和在线调试; 支持 Gowin_PicoRV32 软件在线调试; 集成调试软件 OpenOCD; 集成 Olimex 调试仿真器驱动软件; 更新下载工具 Programmer,支持 Gowin_EMPU(GW1NS-4C)下 载; 更新 Gowin_EMPU(GW1NS-2C)、Gowin_EMPU_M1、 Gowin_EMPU(GW1NS-4C)、Gowin_EMPU_M3、 Gowin_PicoRV32 软件编程参考设计示例; 支持 Help 帮助系统,在线查看 GMD 和 Gowin MCU 用户手册。
2022/10/28	1.4.1	删除 Gowin_EMPU(GW1NS-2C)。
2023/04/20	1.5	 新增B版GW5AT-138/GW5AST-138器件支持; 更新帮助文档; 更新下载工具 Programmer; 新增安装路径配置和软件工具链配置; 新增MCU软件开发库、应用程序案例、用户手册; 新增RTOS应用程序案例。
2025/06/13	2.0	 升级 Eclipse 基础版本到 2024-12,并基于 Eclipse CDT 2024- 12 基础版本开发此版本; 升级 Arm GCC/GDB Toolchain 基础版本到 13.3.1-1.1; 升级 RISC-V GCC/GDB Toolchain 基础版本到 14.2.0-3; 升级 OpenOCD 基础版本到 0.12.0-4;

日期	版本	说明
		● OpenOCD 支持 GWU2X、GWUSB、WinUSB JLink 等调试器;
		● OpenOCD 支持 Gowin_EMPU_M1、Gowin_EMPU(GW1NS-
		4C)、Gowin_PicoRV32 等 MCU 调试;
		● 升级 Windows build tools 基础版本到 4.4.1-2;
		● 集成 Arm QEMU 仿真调试工具,基础版本 8.2.6-1;
		● 集成 RISC-V QEMU 仿真调试工具,基础版本 8.2.6-1;
		● Arm QEMU 支持 Gowin_EMPU(GW1NS-4C) MCU 仿真调试;
		● RISC-V QEMU 支持 Gowin_PicoRV32 MCU 仿真调试;
		● 集成 Gprof 软件工具,支持 Profiling 功能;
		● 集成 Gcov 软件工具,支持 Code Coverage 功能;
		● 支持 Parse HexDump Data 功能, 配合 Gprof 和 Gcov 工具, 收
		集和解析运行数据;
		● 升级 Programmer 软件工具基础版本到 1.9.11.02;
		● 集成 GoBridgeDriver 软件工具,支持基于 WinUSB 驱动的下载和调试。

目录

目录i
图目录v
表目录xi
1 关于本手册1
1.1 手册内容1
1.2 相关文档1
1.3 术语、缩略语2
1.4 技术支持与反馈2
2 概述3
2.1 参考文档3
2.2 MCU 支持
3 平台支持5
3.1 平台支持5
3.2 运行环境5
4 下载与安装6
4.1 软件下载6
4.2 IDE 安装6
4.3 驱动安装7
4.3.1 GWU2X
4.3.2 GWUSB V3.3
4.3.3 GWUSB V4.1
4.3.4 Olimex
4.3.5 J-Link WinUSB 11
4.3.6 Segger J-Link 11

i

	5.1 标题栏13
	5.2 菜单栏13
	5.2.1 File 菜单项
	5.2.2 Edit 菜单项
	5.2.3 Source 菜单项19
	5.2.4 Refactor 菜单项
	5.2.5 Navigate 菜单项20
	5.2.6 Search 菜单项22
	5.2.7 Project 菜单项
	5.2.8 Run 菜单项
	5.2.9 Gowin 菜单项
	5.2.10 Window 菜单项
	5.2.11 Help 菜单项
	5.3 工具栏
	5.4 项目资源管理器
	5.5 代码编辑区
	5.6 控制台
	5.6.1 控制台视图
	5.6.2 问题视图
	5.6.3 终端视图
6	创建工程
	6.1 新建工程
	6.1.1 新建工程(Arm MCU)
	6.1.2 新建工程(RISC-V MCU)40
	6.2 导入己有工程
7	导入旧版本 GMD 创建的工程46
	7.1 修改导入的 Arm MCU 工程46
	7.2 修改导入的 RISC-V MCU 工程
8	编译工程

ii

	8.1 编译工具	
	8.2 设置工程	
	8.2.1 Arm MCU 工程设置	
	8.2.2 RISC-V MCU 工程设置	75
	8.3 编译工程	
	8.3.1 编译方法	
	8.3.2 编译工程	
9	下载工程	99
	9.1 下载工具	
	9.2 下载方法	100
	9.3 下载工程	100
10	调试运行工程	106
	10.1 调试工具	106
	10.2 注意事项	106
	10.3 调试模式	106
	10.4 调试方法	107
	10.5 调试设置	108
	10.5.1 Arm MCU 调试设置	109
	10.5.2 RISC-V MCU 调试设置	116
	10.6 运行工程	119
	10.6.1 启动调试视图	119
	10.6.2 调试功能选项	120
	10.7 调试视图	121
	10.7.1 内存视图	122
	10.7.2 内存浏览器视图	129
	10.7.3 寄存器视图	132
	10.7.4 变量视图	133
	10.7.5 表达式视图	133
	10.7.6 断点视图	134
	10.7.7 反汇编视图	134

10.7.8 控制台视图
11 QEMU 仿真调试136
11.1 QEMU 工具
11.2 QEMU 使用方法136
11.2.1 Arm MCU QEMU 设置137
11.2.2 RISC-V MCU QEMU 141
11.3 仿真调试工程145
12 Code Coverage 功能146
12.1 关于 Code Coverage 功能 146
12.2 使用 Code Coverage 功能 146
12.2.1 创建工程146
12.2.2 设置相关选项 147
12.2.3 编译工程151
12.2.4 运行工程及使用 Code Coverage 功能151
13 Profiling 功能159
13.1 关于 Profiling 功能159
13.2 使用 Profiling 功能159
13.2.1 创建工程159
13.2.2 设置相关选项 160
13.2.3 运行工程及使用 Profiling 功能162

图目录

图 4-1 选择 Gowin > GoBridgeDriver	7
图 4-2 GoBridgeDriver 驱动软件	8
图 4-3 选择 Options > List All Devices	8
图 4-4 GWU2X 软件驱动安装	9
图 4-5 GWUSB V3.3 软件驱动安装	9
图 4-6 GWUSB V4.1 软件驱动安装	10
图 4-7 Olimex 软件驱动安装	11
图 4-8 J-Link WinUSB 软件驱动安装	11
图 5-1 软件用户界面	12
图 5-2 New C/C++ Project	15
图 5-3 C Project	15
图 5-4 Import	16
图 5-5 Import Projects	17
图 5-6 Debug Configurations	25
图 5-7 Programmer	26
图 5-8 GoBridgeDriver	27
图 5-9 About GMD	30
图 5-10 项目资源管理器	32
图 5-11 代码编辑区	33
图 5-12 控制台视图	33
图 5-13 问题视图	34
图 5-14 启动终端	35
图 5-15 串口终端	35
图 5-16 串口终端输出	36
图 6-1 新建工程	37

图 6-2 选择项目类型和软件工具链	38
图 6-3 选择平台和配置	39
图 6-4 选择工具链和路径	40
图 6-5 选择项目类型和软件工具链	41
图 6-6 选择平台和配置	42
图 6-7 选择工具链和路径	43
图 6-8 选择导入的方式	44
图 6-9 选择需要导入的工程	45
图 7-1 Arm 工程导入后编译报错	47
图 7-2 软件工具链设置	48
图 7-3 RISC-V 工程导入后编译报错	49
图 7-4 RISC-V 工具链设置	50
图 7-5 编译选项设置	51
图 8-1 Arm 软件工具链	53
图 8-2 RISC-V 软件工具链	54
图 8-3 Arm MCU 工程设置	55
图 8-4 RISC-V MCU 工程设置	56
图 8-5 Target Processor	57
图 8-6 Optimization	58
图 8-7 Warnings	60
图 8-8 Debugging	61
图 8-9 GNU ARM Cross Assembler > Preprocessor	62
图 8-10 GNU ARM Cross Assembler > Includes	63
图 8-11 GNU ARM Cross Assembler > Warnings	64
图 8-12 GNU ARM Cross Assembler > Miscellaneous	65
图 8-13 GNU ARM Cross C Compiler > Preprocessor	66
图 8-14 GNU ARM Cross C Compiler > Includes	67
图 8-15 GNU ARM Cross C Compiler > Optimization	68
图 8-16 GNU ARM Cross C Compiler > Warnings	69
图 8-17 GNU ARM Cross C Compiler > Miscellaneous	70

图 8-18 GNU ARM Cross C Linker > General	71
图 8-19 GNU ARM Cross C Linker > Libraries	72
图 8-20 GNU ARM Cross C Linker > Miscellaneous	73
图 8-21 GNU ARM Cross Create Flash Image	74
图 8-22 GNU ARM Cross Print Size	75
图 8-23 Target Processor	76
图 8-24 Optimization	77
图 8-25 Warnings	79
图 8-26 Debugging	80
图 8-27 GNU RISC-V Cross Assembler > Preprocessor	81
图 8-28 GNU RISC-V Cross Assembler > Includes	82
图 8-29 GNU RISC-V Cross Assembler > Warnings	83
图 8-30 GNU RISC-V Cross Assembler > Miscellaneous	84
图 8-31 GNU RISC-V Cross C Compiler > Preprocessor	85
图 8-32 GNU RISC-V Cross C Compiler > Includes	86
图 8-33 GNU RISC-V Cross C Compiler > Optimization	87
图 8-34 GNU RISC-V Cross C Compiler > Warnings	88
图 8-35 GNU RISC-V Cross C Compiler > Miscellaneous	89
图 8-36 GNU RISC-V Cross C Linker > General	90
图 8-37 GNU RISC-V Cross C Linker > Libraries	91
图 8-38 GNU RISC-V Cross C Linker > Miscellaneous	92
图 8-39 GNU RISC-V Cross Create Flash Image	93
图 8-40 GNU RISC-V Cross Create Listing	94
图 8-41 GNU RISC-V Cross Print Size	95
图 8-42 Build	96
图 8-43 Arm MCU 工程编译示例	97
图 8-44 RISC-V MCU 工程编译示例	98
图 9-1 下载工具	99
图 9-2 Programmer	100
图 9-3 下载设置	101

图 9-4 下载设置.	
图 9-5 下载设置.	
图 9-6 下载设置.	
图 9-7 下载设置.	
图 10-1 Debug	
图 10-2 创建调试	视图108
图 10-3 Main 视图	图109
图 10-4 Debugge	er 视图 110
图 10-5 Startup 礼	观图111
图 10-6 Main 视图	ধ্ব 112
图 10-7 Debugge	er 视图 113
图 10-8 Startup 社	观图115
图 10-9 Main 视图	图116
图 10-10 Debugg	ger 视图
图 10-11 Startup	视图118
图 10-12 启动调试	试视图120
图 10-13 Window	v > Show View > Other
图 10-14 Debug \	View
图 10-15 内存视图	图
图 10-16 选择要出	监视的线程或堆栈123
图 10-17 Add Me	emory Monitor 123
图 10-18 Monitor	Memory
图 10-18 传统格式	式的内存渲染器
图 10-19 创建浮,	点格式的内存渲染器125
图 10-20 浮点格式	式的内存渲染器
图 10-21 设置字	节顺序126
图 10-22 存储内容	容显示126
图 10-23 设置渲述	边效里 197
	本从木
图 10-24 设置内石	来众朱存渲染效果
图 10-24 设置内在 图 10-25 Import I	存渲染效果

图 10-26 Export Memory	129
图 10-27 内存浏览器视图	129
图 10-28 传统格式的内存渲染器	130
图 10-29 建立浮点格式的内存渲染器	130
图 10-30 浮点格式的内存渲染器	131
图 10-31 设置字节顺序	131
图 10-32 Find/Replace	132
图 10-33 寄存器视图	132
图 10-34 变量视图	133
图 10-35 表达式视图	133
图 10-36 断点视图	134
图 10-37 反汇编视图	134
图 10-38 控制台视图	135
图 11-1 GDB QEMU arm Debugging	137
图 11-2 Main 视图	138
图 11-3 Debugger 视图	139
图 11-4 Startup 视图	140
图 11-5 GDB QEMU riscv32 Debugging	141
图 11-6 Main 视图	142
图 11-7 Debugger 视图	143
图 11-8 Startup 视图	144
图 11-9 启动 QEMU 仿真调试	145
图 12-1 指定所有源代码	148
图 12-2 指定某个源代码	149
图 12-3 设置 Linux Tools Path	150
图 12-4 设置 Linux Tools Path	150
图 12-5 编译工程	151
图 12-6 运行工程并查看 Profiling 信息	152
图 12-7 Select All	153
图 12-8 Parse HexDump Data	154

图 12-9 解析 Profiling 信息并生成对应文件15-	4
图 12-10 查看生成的对应文件 15	5
图 12-11 打开.gcda 文件15	5
图 12-12 Gcov	6
图 12-13 Create chat	7
图 12-14 柱状图15	8
图 12-15 饼状图15	8
图 13-1 设置编译选项16	0
图 13-2 设置 Linux Tools Path16	1
图 13-3 设置 Linux Tools Path16	1
图 13-4 运行工程并查看 Profiling 信息16	2
图 13-5 Select All	3
图 13-6 Parse HexDump Data 16	3
图 13-7 解析 Profiling 信息并生成对应文件164	4
图 13-8 查看生成的 gmon.out 文件 164	4
图 13-9 打开 gmon.out 文件16	5
图 13-10 Gprof 16	5
图 13-11 查看 Profiling 信息 16	6
图 13-12 Create chat	7
图 13-13 柱状图16	8
图 13-14 饼状图164	8

表目录

表 1-1 术语、缩略语	. 2
表 2-1 MCU 支持	. 4
表 4-1 GMD 软件主要文件描述	. 6
表 5-1 File 菜单项	. 13
表 5-2 项目类型	. 16
表 5-3 软件工具链	. 16
表 5-4 Edit 菜单项	. 17
表 5-5 Source 菜单项	. 19
表 5-6 Refactor 菜单项	. 19
表 5-7 Navigate 菜单项	. 20
表 5-8 Search 菜单项	. 22
表 5-9 Project 菜单项	. 22
表 5-10 Run 菜单项	. 23
表 5-11 调试方式	. 25
表 5-12 Gowin 菜单项	. 26
表 5-13 Window 菜单项	. 27
表 5-14 Help 菜单项	. 29
表 5-15 工具栏选项	. 30
表 8-1 Target Processor	. 57
表 8-2 Optimization	. 58
表 8-3 Debugging	. 61
表 8-4 GNU ARM Cross Assembler > Preprocessor	. 62
表 8-5 GNU ARM Cross C Compiler > Includes	. 67
表 8-6 GNU ARM Cross C Compiler > Optimization	. 68

表 8-7 GNU ARM Cross C Linker > General	. 71
表 8-8 GNU ARM Cross Create Flash Image	. 74
表 8-9 GNU ARM Cross Print Size	. 75
表 8-10 Target Processor	. 76
表 8-11 Optimization	. 78
表 8-12 Debugging	. 80
表 8-13 GNU RISC-V Cross Assembler > Miscellaneous	. 84
表 8-14 GNU RISC-V Cross C Compiler > Includes	. 86
表 8-15 GNU RISC-V Cross C Compiler > Optimization	. 87
表 8-16 GNU RISC-V Cross C Compiler > Miscellaneous	. 89
表 8-17 GNU RISC-V Cross C Linker > General	. 90
表 8-18 GNU RISC-V Cross C Linker > Miscellaneous	. 92
表 8-19 GNU RISC-V Cross Create Flash Image	. 93
表 8-20 GNU RISC-V Cross Create Listing	. 94
表 8-21 GNU RISC-V Cross Print Size	. 95
表 9-1 下载设置	. 101
表 9-2 下载设置	. 102
表 9-3 下载设置	. 103
表 9-4 下载设置	. 104
表 9-5 下载设置	. 105
表 10-1 调试模式	. 106
表 10-2 Debugger 视图	. 110
表 10-3 OpenOCD 配置文件	. 110
表 10-4 Startup 视图	. 111
表 10-5 Debugger 视图	. 113
表 10-6 Arm MCU 设备型号	. 114
表 10-7 Startup 视图	. 115
表 10-8 Debugger 视图	. 117
表 10-9 OpenOCD 配置文件	. 118
表 10-10 Startup 视图	. 118

表 10-11 调试功能选项	120
表 11-1 已支持的外设功能	136
表 11-2 Debugger 视图	139
表 11-3 Arm MCU QEMU	140
表 11-4 Startup 视图	140
表 11-5 Debugger 视图	143
表 11-6 RISC-V MCU QEMU	144
表 11-7 Startup 视图	144
表 12-1 Gcov 功能	156
表 13-1 Gprof 功能	166

1 关于本手册

1.1 手册内容

本手册主要描述高云 GMD 软件的使用方法,包含平台支持、下载与安装、用户界面、创建工程、导入旧版本 GMD 创建的工程、编译工程、下载 工程、调试运行工程、QEMU 仿真调试、Code Coverage 功能、Profiling 功能等,旨在帮助用户快速熟悉 GMD 软件的使用流程,提高开发效率。本 手册中的软件界面截图和支持的产品列表等信息均参考 GMD 2025.01 版 本,因软件版本升级,部分信息可能会略有差异,具体以用户软件版本的信 息为准。

1.2 相关文档

通过登录高云半导体网站 <u>www.gowinsemi.com</u>可下载、查看以下相关 文档:

- IPUG928, Gowin_EMPU(GW1NS-4C) IDE 软件参考手册
- MUG1186, Gowin EMPU(GW5AS-25)快速开发用户手册
- IPUG536, Gowin EMPU M1 IDE 软件参考手册
- IPUG910, Gowin PicoRV32 IDE 软件参考手册
- IPUG919, Gowin_EMPU_M3 IDE 软件参考手册

1.3 术语、缩略语

本手册中的相关术语、缩略语及相关释义如表 1-1 所示。

表 1-1 术语、缩

术语、缩略语	全称	含义
GMD	GOWIN MCU Designer	高云微控制器软件开发环 境
ARM	Advanced RISC Machine	高级精简指令集计算机
RISC-V	Reduced Instruction Set Computer Five	第五代精简指令集计算机
IDE	Integrated Development Environment	集成开发环境
MCU	Microcontroller Unit	微控制器单元
FPGA	Field Programmable Gate Array	现场可编程门阵列
SoC	System on Chip	片上系统
GNU	GNU's Not Unix	自由软件操作系统
GCC	GNU Compiler Collection	GNU 编译器套件
GDB	GNU Debug	GNU 调试器
OpenOCD	Open On-Chip Debugger	开源片上调试器
QEMU	Quick Emulator	开源模拟器
CDT	C/C++ Development Tooling	C/C++开发工具
EPL	Eclipse Public License	Eclipse 公共许可证
ABI	Application Binary Interface	应用二进制接口
UART	Universal Asynchronous Receiver/Transmitter	异步收发传输器
GPIO	General Purpose Input/Output	通用输入输出

1.4 技术支持与反馈

高云半导体提供全方位技术支持,在使用过程中如有任何疑问,可直接 与公司联系:

网址: <u>www.gowinsemi.com.cn</u>

E-mail: support@gowinsemi.com

Tel: +86 755 8262 0391



GMD 软件是高云依据自有 FPGA + MCU SoC 架构的产品特性,基于 开源 GCC 编译工具链和开源 Eclipse 框架,自主研发的新一代 MCU 软件 开发环境,支持通用的 C/C++嵌入式软件开发,帮助开发人员快速实现 MCU 软件开发过程中的代码编译、链接、产生可执行文件以及下载、调试 等。

GMD 软件集成了 OpenOCD、QEMU 等调试与仿真工具,便于开发人员调试运行工程,快速定位与分析软件编程问题。

GMD 软件集成了 Gcov 和 Gprof 软件工具,支持 Code Coverage 和 Profiling 功能。Gcov 软件工具是一个测试 C/C++代码覆盖率的工具,与 GCC 互相配合,共同实现对 C/C++程序的语句、功能函数和分支覆盖测 试。Gprof 软件工具是一个性能分析工具,可以帮助开发人员理解 C/C++程 序的运行情况,通过 Gprof 可以获取到程序中各个函数的调用信息、调用次 数、执行时间等,对优化程序、提升程序运行效率具有重要作用。

GMD 软件支持高云 Arm 和 RISC-V 架构的 MCU 产品,包括 Gowin_EMPU(GW1NS-4C)、Gowin_EMPU_M1、Gowin_EMPU(GW5AS-25)、Gowin EMPU M3 和 Gowin PicoRV32 等。

GMD 软件支持图形化界面,可以快速编辑代码、查看编译结果、在线 调试与仿真,集成了高云下载软件 Programmer,可以快速启动下载功能, 实现用户设计的全流程。

2.1 参考文档

Eclipse 平台采用开放式源代码模式运作,提供 EPL 公共许可证(免费、开源)以及全球发布权利。Eclipse 本身只是一个框架平台,除了 Eclipse 平台的运行时内核之外,其他所有功能都位于不同的插件中。开发 人员既可通过 Eclipse 的不同插件来扩展平台功能,也可自主开发所需功能 的插件。一个插件也可以插入另一个插件,从而实现最大程度的集成。

由于 Eclipse 已经在社区被大量使用,一些常见的使用方法已在 Eclipse IDE 里,如果没有和具体硬件绑定,一般情况下可以参考 Eclipse 官方文档或其他教程,这里推荐几个常用的学习网站:

- <u>https://help.eclipse.org/latest/index.jsp</u>
- <u>https://eclipse-embed-cdt.github.io/</u>
- <u>https://mcuoneclipse.com/</u>

2.2 MCU 支持

GMD 软件已支持的高云 MCU、内核架构与所在 FPGA 器件如表 2-1 所示。

表 2-1 MCU 支持

MCU	内核架构	FPGA 器件
Gowin_EMPU(GW1NS- 4C)	Arm Cortex-M3	GW1NS/R/ER-4C
Gowin_EMPU_M1	Arm Cortex-M1	GW1N/R 系列
		GW2A/N/R/NR 系列
		GW3A/R 系列
		GW5A/NT/NRT/R/RT/S/ST/T 系列
Gowin_EMPU(GW5AS- 25)	Arm Cortex-M4	GW5AS-25
Gowin_EMPU_M3	Arm Cortex-M3	GW2A/N/R/NR 系列
Gowin_PicoRV32	RISC-V	GW1N/R 系列
		GW2A/N/R/NR 系列
		GW3A/R 系列
		GW5A/NT/NRT/R/RT/S/ST/T 系列



3.1 平台支持

- Windows 10/11 (64-bit)
- Ubuntu 20.04/22.04/24.04 LTS

3.2 运行环境

- 建议 4 GB 运行内存
- 建议6GB磁盘空间



4.1 软件下载

GMD 软件包括 Windows 和 Linux 两个不同系统的压缩包:

- GMD_20yy.xx_win.zip
- GMD_20yy.xx_linux.tar.gz

注!

- 软件下载前,需要先注册公司网站并登录;
- 软件安装包名称 "20yy.xx" 中的 "y" 表示发布年份, "x" 表示当年的发布版次。

用户可以在高云公司网站上,根据本地系统环境,下载对应 Windows 或 Linux 的 GMD 软件压缩包。

目前已在 Windows 10/11 64-bit、Ubuntu 20.04 LTS 上验证测试,推荐使用以上版本的操作系统。

4.2 IDE 安装

完成 GMD 软件压缩包下载后,在本地解压后即可直接使用,无需另行 安装 IDE 和设置环境变量。注意解压路径中不可包含中文。

GMD 软件无需申请 License,可直接打开使用。

例如 Windows 版本 GMD 软件 GMD_20yy.xx_win.zip, 解压后包含若 干文件夹和文件,如表 4-1 所示。

GMD 软件文件	描述
configuration	GMD 软件运行过程中产生的本地临时配置文件
dropins	用户手动添加和安装插件文件
features	软件功能相关的模块

表 4-1 GMD 软件主要文件描述

GMD 软件文件	描述
plugins	所有已安装的插件文件
tool	软件工具,例如 Programmer、GoBridgeDriver
toolchain	软件工具链,例如 GCC、OpenOCD、QEMU
GMD.exe	GMD 软件可执行文件,双击运行 GMD 软件
GMD.ini	GMD 软件初始化文件
ver.20yy.xx.txt	版本控制说明文件

4.3 驱动安装

GMD 软件支持使用 OpenOCD 调试 Arm 和 RISC-V MCU,以及使用 Segger J-Link 调试 Arm MCU。OpenOCD 调试时支持 GWU2X、 GWUSB、Olimex、J-Link 等高云自研或第三方常见的调试器,在 Windows 系统下须安装 WinUSB 软件驱动。Segger J-Link 调试时,须安装 Segger J-Link 软件驱动。

在 Windows 系统下,GMD 软件支持使用 GoBridgeDriver 驱动软件安装相关 WinUSB 软件驱动,打开 GMD 软件,选择菜单栏 "Gowin > GoBridgeDriver",打开 GoBridgeDriver 驱动软件(须 PC 管理员权限),如图 4-1 和图 4-2 所示。

🐳 workspace_cm3_h - GMD				_	
File Edit Source Refactor Navi	gate Search Project Run	Gowin Window Help			
📑 🕶 🔚 🐚 🛛 🗞 🖛 🔜 🗠) 🗠 i 📮 i 🏪 i 🔌 🕹 i 👫	↓購 Programmer	- O - 🚱 - 💁 -	🕭 🗁 🔗 👻 🛛	9 🗉 n
월 ▼ 祠 ▼ ♡ ▷ ♥ 🗘 ▼ ▷ ▼		😟 GoBridgeDriver		٩ 🛙	🖞 📠 🎋
Project Explorer 🗙 🗖 🗖		♦♦ SDK and Document →		≡ 0 × »₃	
E 🕏 🏹 🝃 🕴					5º 8 🗖
📁 cm3_demo			т	here is no active	editor
👕 cm3_freertos			t	hat provides an c	utline.
> 😂 cm3_gcov_demo					
📋 cm3_gprof_demo					
📋 cm3_rtthread_nano					
💼 cm3_ucos_iii					
	🖹 Problems 🧔 Tasks 📃	Console 🗙 🔲 Properties			
		×	우 수 🔄 🖬 🚮 😑	🔒 🖃 🛃	• 📩 •
	CDT Build Console [cm3_gco	/_demo]			
					^
					~
	<				>
🗀 cm3_gcov_demo					

图 4-1 选择 Gowin > GoBridgeDriver

图 4-2 GoBridgeDriver 驱动	软件
-------------------------	----

🔀 GoBridgeDriver V1.0	– 🗆 X
<u>D</u> evice <u>O</u> ptions <u>H</u> elp	
Device	✓ Edit
Driver	WinUSB (v6.1.7600.16385)
USB ID	
WCID	Install WCID Driver
0 devices found.	

上述几种调试器的软件驱动安装方法如下文所述。

4.3.1 GWU2X

高云 GWU2X 下载/调试器,内置高云自研 USB to JTAG 芯片,请参照 以下步骤安装 GWU2X 软件驱动。

- 1. 请将 GWU2X 设备插入 PC 主机。
- 2. 选择菜单栏 "Options > List All Devices",列举出所有已插入 PC 主机 的 USB 设备,如图 4-3 所示。

图 4-3 选择 Options > List All Devices

👷 GoB	Bridg	eDrive	er V1.0	_		\times
<u>D</u> evice	<u>O</u> p	tions	<u>H</u> elp			
	~	List /	All Devices			
Device	~	Igno	re Hubs or Composite Parents			
	~	Crea	te a Catalog File			
Driver	~	Sign	Catalog & Install Autogenerate	d Certif	icate	
USB ID		Adva	anced Mode			
WCID	_	Log	Verbosity			>
4 devices found.						

3. 选择设备 "GWU2X",选择驱动 "WinUSB",单击 "Install Driver" 安 装软件驱动,如图 4-4 所示,稍等片刻即可完成软件驱动的安装。

图 4-4 GWU2X 软件驱动安装

👷 GoB	ridgeDriver V1.0	– 🗆 X
<u>D</u> evice	<u>Options</u> <u>H</u> elp	
Device	GWU2X	✓ □Edit
Driver	WinUSB (v6. 1. 7600. 16385)	WinUSB (v6.1.7600.16385)
USB ID	33AA 0120	Reinstall Driver
4 devices	found.	T

注!

- 如果当前设备未安装过任何软件驱动,该选项显示为 "Install Driver";
- 如果当前设备已安装过当前软件驱动,该选项显示为 "Reinstall Driver";
- 如果当前设备已安装过其他软件驱动,该选项显示为"Replace Driver"。

4.3.2 GWUSB V3.3

高云 GWUSB V3.3 下载/调试器,内置 FTDI 232H 双通道芯片,请参照以下步骤安装 GWUSB V3.3 软件驱动。

- 1. 请将 GWUSB V3.3 设备插入 PC 主机。
- 选择菜单栏 "Options > List All Devices",列举出所有已插入 PC 主机的 USB 设备。
- 分别选择设备 "Dual RS232-HS (Interface 0)"和 "Dual RS232-HS (Interface 1)",选择驱动 "WinUSB",单击 "Install Driver"安装软件 驱动,如图 4-5 所示,稍等片刻即可完成软件驱动的安装。

图 4-5 GWUSB V3.3 软件驱动安装

🕅 GoBridgeDriver V1.0 – 🗆 🗙	
<u>D</u> evice <u>Options</u> <u>H</u> elp	
Device Dual RS232-HS (Interface 0)	
Driver WinUSB (v6. 1. 7600. 16385)	
USB ID 0403 6010 00 WCID X	
5 devices found.	

4.3.3 GWUSB V4.1

高云 GWUSB V4.1 下载/调试器,内置 FTDI 232H 单通道芯片,请参照以下步骤安装 GWUSB V4.1 软件驱动。

- 1. 请将 GWUSB V4.1 设备插入 PC 主机。
- 选择菜单栏 "Options > List All Devices",列举出所有已插入 PC 主机的 USB 设备。
- 3. 选择设备 "Single RS232-HS",选择驱动 "WinUSB",单击 "Install Driver" 安装软件驱动如图 4-6 所示,稍等片刻即可完成软件驱动的安装。

🕎 GoBri	idgeDriver V1.0 —		×
<u>D</u> evice (<u>O</u> ptions <u>H</u> elp		
Device	Single RS232-HS	\sim	Edit
Driver	libusbK (v3.1.0.0) WinUSB (v6.1.7600	. 16385	5) 🔺
USB ID WCID	0403 6014 Replace Driver		-
4 devices	found.		

4.3.4 Olimex

ARM-USB-TINY-H 版本 Olimex 调试器,内置 FTDI FT2232H 芯片, 请参照以下步骤安装 Olimex 软件驱动。

- 1. 请将 Olimex 设备插入 PC 主机。
- 2. 选择菜单栏 "Options > List All Devices",列举出所有已插入 PC 主机 的 USB 设备。
- 分别选择设备 "Olimex OpenOCD JTAG ARM-USB-TINY-H (Interface 0)"和 "Olimex OpenOCD JTAG ARM-USB-TINY-H (Interface 1)",选择驱动"WinUSB",单击"Install Driver"安装软件驱动如图 4-7 所示,稍等片刻即可完成软件驱动的安装。

图 4-6 GWUSB V4.1 软件驱动安装

图 4-7 Olimex 软件驱动安装

👷 GoBridgeDriver V1.0	– 🗆 X
<u>D</u> evice <u>O</u> ptions <u>H</u> elp	
Device Olimex OpenOCD JTAG ARM-USB-TIN	IY-H (Interface 0) 🗸 🗌 Edit
Driver libusbK (v3.1.0.0)	WinUSB (v6.1.7600.16385)
USB ID 15BA 002A 00	
WCID X	Replace Driver
5 devices found.	

4.3.5 J-Link WinUSB

J-Link 调试器,请参照以下步骤为 J-Link 安装 WinUSB 类型软件驱动。

- 1. 请将 J-Link 设备插入 PC 主机。
- 2. 选择菜单栏 "Options > List All Devices",列举出所有已插入 PC 主机 的 USB 设备。
- 3. 选择设备 "J-Link",选择驱动 "WinUSB",单击 "Install Driver" 安装 软件驱动如图 4-8 所示,稍等片刻即可完成软件驱动的安装。

图 4-8 J-Link WinUSB 软件驱动安装

🞇 GoBridgeDriver V1.0	- 🗆 X
<u>D</u> evice <u>O</u> ptions <u>H</u> elp	
Device J-Link	∽ □Edit
Driver jlink (v2.70.8.0)	WinUSB (v6.1.7600.16385)
USB ID 1366 0101 WCID X	Replace Driver
4 devices found.	

4.3.6 Segger J-Link

请参照 Segger 官方方法(<u>https://www.segger.com/downloads/jlink/</u>) 下载与安装 Segger J-Link 官方软件驱动,这里不再详细描述。

5_{用户界面}

双击 GMD 软件可执行文件 "GMD\GMD.exe" 打开 GMD 软件, GMD 软件用户界面如图 5-1 所示,主要包含标题栏(Title Bar)、菜单栏(Menu Bar)、工具栏(Tool Bar)、项目资源管理器(Project Explorer)、代码编辑 区(Code Editor)、控制台(Console View)、透视图(Perspective)等。

图 5-1 软件用户界面



软件用户界面的各个主要部分如下所述。

5.1 标题栏

主要显示当前软件工程的工作区、名称以及当前已打开的文件等信息。

5.2 菜单栏

主要提供一些常用菜单项,包含 File、Edit、Source、Refactor、 Navigate、Search、Project、Run、Gowin、Window、Help。

常用的菜单项如下所述。

5.2.1 File 菜单项

File 菜单项如表 5-1 所示。

表 5-1 File 菜单项

菜单项	子菜单项	功能描述	
New	Makefile Project with Existing Code	创建包含 makefile 的 C/C++项目	
	C/C++ Project	创建 C/C++项目	
	Project…	创建 C、C++、C/C++项 目	
	Convert to a C/C++ Project (Adds C/C++ Nature)	非 C/C++项目添加 C/C++ 特征,启用 C/C++工具链	
	Source Folder	创建源代码文件夹	
	Folder	创建文件夹	
	Source File	创建 C/C++源码文件	
	Header File	创建 C/C++头文件	
	File from Template	创建 C/C++模板文件	
	Class	创建 C++类结构	
	Task	跟踪和管理开发过程中的 待办事项或问题	
	Example	提供 XML 文件的编辑和 验证	
	Other	打开选择向导,显示所有 菜单项	
Open File	-	打开已存在的文件	
Open Projects from File System	-	由文件系统导入并打开现 有的项目	

菜单项	子菜单项	功能描述
Recent Files		最近打开过的文件
Close Editor	-	关闭当前编辑器
Close All Editors	-	关闭所有编辑器
Save	-	保存当前编辑器的内容
Save As	-	以新名称保存当前编辑器 的内容
Save All	-	保存所有编辑器的内容以 及未保存的变更
Revert	-	将当前编辑器的内容恢复 为已保存档案中的内容
Move	-	移动资源
Rename	-	重命名资源
Refresh	-	重新整理和加载项目资源
Convert Line Delimiters	Windows	Windows 系统格式
То	Unix	Unix 系统格式
Print	-	打印当前编辑器的内容
Import	-	导入项目的向导
Export	-	导出项目的向导
Properties	-	项目属性配置
Switch Workspace		切换至不同工作区并重启 工作区
Restart	-	重启 GMD 软件
Exit	-	退出 GMD 软件

C/C++ Project

"New > C/C++ Project" 选项如图 5-2 所示,用于创建 C/C++项目。

图 5-2 New C/C++ Project



常用选项 "C Managed Build" 或 "C++ Managed Build",例如 "C Managed Build" 如图 5-3 所示,其所包含的项目类型如表 5-2 所示,可选的软件工具链如表 5-3 所示。

图 5-3	C]	Pro	ject
-------	------------	-----	------

**	-	– 🗆 X
C Project Project name must be specified		Ď
Project name:	venace cm3 h	Browce
Choose file system: default ~	Tooldaine	-7
 Executable Empty Project Hello World ANSI C Project Hello World ANSI C Project Hello World RISC-V C Project Shared Library Empty Project Static Library Empty Project Makefile project Empty Project Show project types and toolchains online 	Arm Cross GCC MinGW GCC RISC-V Cross GCC	on the platform
? < <u>B</u> ack <u>N</u> e	xt > <u>F</u> inish	Cancel

表 5-2 项目类型

项目类型	描述
Executable	创建一个可执行的工程项目
Shared Library	创建一个共享库项目
Static Library	创建一个静态库项目
Makefile project	创建一个基于 Makefile 的项目

表 5-3 软件工具链

工具链	描述
Arm Cross GCC	Arm GCC 交叉编译工具链
RISC-V Cross GCC	RISC-V GCC 交叉编译工具链

Import

"New > Import..."选项如图 5-4 所示,用于导入已有的项目。

图 5-4 Import

🗱 Import		×
Select Create new projects from an archive file or directory.	Ľ	5
<u>S</u> elect an import wizard:		
 Existing Projects into Workspace File System Preferences Projects from Folder or Archive C/C++ Git Install Oomph RPM RPM Run/Debug Tasks Team 		*
? < <u>B</u> ack <u>N</u> ext > <u>F</u> inish	Cance	ł

常用选项"Existing Projects into Workspace"如图 5-5 所示。

🐳 Import						×
Import Projects Select a directory to sear	ch for ex	tisting Eclipse	projects.			
Select roo <u>t</u> directory: Select <u>a</u> rchive file: <u>P</u> rojects:	D:\GMD	_2025_worksį	oace\workspac	xe_ ∨	B <u>r</u> ows B <u>r</u> ows	e
 ✓ picorv32_demo (D ✓ picorv32_load_der ✓ picorv32_qemu_de 	:\GMD_2 no (D:\Gl emo (D:\(025_workspa MD_2025_wor 3MD_2025_wo	ce\workspace_ kspace\works orkspace\work	riscv\ pace_ space	<u>S</u> elect <u>D</u> eselec	All t All
< Options				>	R <u>e</u> fre	sh
☐ Searc <u>h</u> for nested pro	ojects orkspace I projects eady exis	upon compl st in the work	etion space			
Working sets Add project to work Working sets:	ing sets			~	Ne <u>w</u> S <u>e</u> lect	
?	<u>B</u> ack	<u>N</u> ext >	<u>F</u> inish	1	Cance	el

图 5-5 Import Projects

5.2.2 Edit 菜单项

Edit 菜单项如表 5-4 所示。

表 5-4 Edit 菜单项

菜单项	子菜单项	功能描述
Undo	-	恢复为编辑器的前一次变更
Redo	-	恢复已取消的变更
Cut	-	剪切
Сору	-	复制
Paste	-	粘贴
Delete	-	删除选定文本或元素选项
Select All	-	选择所有的编辑器内容
	Enclosing Element	使当前的代码选择扩展到包含当前选 择内容的最外层元素
Expand Selection to	Next Element	使当前的选择范围扩展到下一个逻辑 代码元素
	Previous Element	将选择范围扩展到上一个代码元素

菜单项	子菜单项	功能描述
	Restore Last Selection	将选择恢复到上一次的状态
Toggle Block Selection	-	切换块状选择模式
To multi-selection	-	同时选择多个位置或多段代码
Find/Replace	-	查找/替换
Find Word	-	查找目前所选文字
Find Next	-	查找目前所选文字的下一个搜寻结果
Find Previous	-	查找目前所选文字的上一个搜寻结果
Incremental Find Next	-	启动增量查找模式,增量查找下一个 搜寻结果
Incremental Find Previous	-	启动增量查找模式,增量查找上一个 搜寻结果
Bookmark	-	为目前所选文字或元素新增书签
Task	-	为目前所选文字或元素新增使用者定 义的作业
Smart Insert Mode	-	智能地切换光标在文本编辑区域时的 插入方式
Insert ChangeLog entry	-	在代码中快速插入一个变更日志条目
Show Tooltip Description	-	以浮动说明方式显示当前光标位置上 的值
Word Completion	-	根据正在输入的字符自动完成单词输 入
Quick Fix	-	根据编译器或 IDE 提供的错误和警告,自动生成修复建议或代码修复
Content Assist	-	智能提示和自动补全
Parameter Hints	-	在输入方法调用时,自动显示该方法 的参数列表
Set Encoding	-	设置当前文件的字符编码方式

5.2.3 Source 菜单项

Source 菜单项如表 5-5 所示。

表	5-5	Source	菜单项
---	-----	--------	-----

菜单项	功能描述
Toggle Comment	在选中的代码行前添加或删除单行注释
Add Block Comment	将选中的代码块注释,注释形式为块注释 (/* */)
Remove Block Comment	移除选中的块注释
Shift Right	选中的代码行向右缩进
Shift Left	选中的代码行向左缩进
Align Const	将选中区域内的常量(通常是静态常量)对齐
Correct Indentation	修正选中代码的缩进,使其符合项目的编码规范
Format	自动格式化整个文件或选中的代码
Copy Qualified Name	复制当前类、方法、变量或其他元素的完全限定名
Add Include	添加头文件的#include 指令
Organize Includes	对源文件中的#include 语句进行整理,移除未使用的 包含项,并按字母顺序排序
Sort Lines	对选中的代码行进行排序
Override Methods	快速为当前类重写一个或多个继承自父类或接口的方 法
Implement Method	为当前类实现接口或抽象类中的未实现方法
Generate Getters and Setters	根据类的字段自动生成相应的 getter 和 setter 方法
Surround With	用某种结构(如 try-catch、if、for 等)包裹选中的代码

5.2.4 Refactor 菜单项

Refactor 菜单项如表 5-6 所示。

表 5-6 Refactor 菜单项

菜单项	功能描述
Rename	重新命名所选的元素,并且更新元素的所有参照
Extract Local Variable	建立新变量,指定给目前所选的表达式,并将选择项替 换为新变量的参照
Extract Constant	从所选表达式中建立 static final 字段并替换字段参照, 重新写入其他出现相同表达式的位置
菜单项	功能描述
------------------	--
Extract Function	建立新函数,指定给目前所选的语句,并将选择项替换 为新函数的参照
Toggle Function	将选定的函数定义从头文件(类定义的内部或外部)移 动到实现文件,或返回
Hide Method	私有化选择的方法
Apply Script	应用已保存的重构列表
Create Script	导出己完成的重构列表,供后续使用
History	显示重构历史记录

5.2.5 Navigate 菜单项

Navigate 菜单项如表 5-7 所示。

菜单项	子菜单项	功能描述
Go Into	-	将当前视图聚焦到所选资源的根目录
Go To	Back	返回到之前显示的层级
	Forward	返回到上次"Back"操作之后的层级
	Up One Level	向上一级层级导航
	Resource	快速导航到特定资源
	Next Member	跳转到当前类或接口的下一个成员
	Previous Member	跳转到当前类或接口的上一个成员
	Go to Matching Bracket	跳转到与当前括号、花括号、方括号 或圆括号匹配的另一个括号位置
	Next Bookmark	跳转到下一个书签
Open Declaration	-	跳转到当前光标所在元素的声明位置
Open Type Hierarchy	-	查看当前类型(类或接口)的继承层 次结构
Open Call Hierarchy	-	查看当前方法的调用层次结构
Open Include Browser	-	查看与当前文件相关的包含关系
Toggle Source/Header	-	切换源文件(.c或.cpp文件)和对应 的头文件(.h文件)
Open Element	-	打开当前光标所在元素的详细信息
Open Type in Hierarchy…	-	在类型层次结构视图中打开当前类型

表 5-7 Navigate 菜单项

菜单项	子菜单项	功能描述
Open Element in Call Hierarchy…	-	将当前元素(方法、类等)显示在调 用层次结构视图中
Open Resource	-	显示所有资源,允许在工作区中选择 并打开任何资源
Open Discovered Type…	-	显示并打开已发现的类型
Open Task…	-	打开当前工作区中的任务列表
Activate Task	-	激活并高亮显示当前选中的任务
Deactivate Task	-	停用当前选中的任务,将其从活动状 态中移除
Open Setup	User	查看或修改与当前用户相关的设置
	Installation	查看当前安装的配置和设置
	Workspace	查看和修改与工作区相关的设置
	Self Products	查看与当前实例相关的产品设置
	Parent Models	查看和管理与当前工作区或项目的父 模型相关的设置
Open Setup Log	-	查看与项目或工作区相关的设置日志
Show In	Include Browser	显示 Include Browser 视图
	C/C++ Projects	显示 C/C++ Projects 视图
	Project Explorer	显示 Project Explorer 视图
	Problem Details	显示 Problem Details 视图
	Terminal	显示 Terminal 视图
	Outline	显示 Outline 视图
	System Explorer	显示 System Explorer 视图
	Properties	查看当前选中文件、项目或资源的属 性
Quick Outline	-	显示当前编辑器的结构大纲,帮助快 速浏览类、方法等结构
Next	-	在活动视图中导航到下一个项目
Previous	-	在活动视图中导航到上一个项目
Previous Edit Location	-	将光标定位到上次编辑的位置
Next Edit Location	-	将光标定位到下次编辑的位置
Go to Line…	-	跳转到指定行号的该行
Back	-	在编辑器中返回到之前查看的资源

菜单项	子菜单项	功能描述
Forward	-	在编辑器中前进到之前查看的资源

5.2.6 Search 菜单项

Search 菜单项如表 5-8 所示。

表 5-8 Search 菜单项

菜单项	子菜单项	功能描述
Search	-	搜索
File	-	在文件中搜索特定的文本或模式
C/C++	-	在 C 或 C++代码中进行搜索
Remote	-	在远程系统上进行搜索
Text	Workspace	在整个工作区中进行搜索
	Project	在当前项目中进行搜索
	File	在单个文件中进行搜索
	Working Set	在当前选择的工作集内进行搜索
Quick Search	-	快速搜索

5.2.7 Project 菜单项

Project 菜单项如表 5-9 所示。

表 5-9 Project 菜单项

菜单项	子菜单项	功能描述
Open Project	-	打开当前项目
Close Project	-	关闭当前项目
Build All	-	编译所有项目
Build	Set Active	设置 Release 或 Debug 为活动状态
Configurations	Manage	管理配置
	Build by Working Set	设置编译工作集
	Set Active by Working Set	设置工作集为活动状态
	Manage Working Sets…	管理工作集
Build Project	-	对当前选定的项目执行增量编译
Build Working Set	Select Working Set	对工作集中的所有项目执行增量编译
Clean	-	清除编译结果

菜单项	子菜单项	功能描述
Build Automatically	-	自动编译工作区中的所有项目
Build Targets	Create	创建
	Build…	编译
	Rebuild Last Target	重新编译最近目标
C/C++ Index	Rebuild	重新构建整个 C/C++项目的索引
	Freshen All Files	刷新项目中的所有文件
	Update with Modified Files	仅更新那些被修改过的文件的索引
	Re-resolve Unresolved Includes	重新解析那些未解析的头文件 (#include)引用
	Search for Unresolved Includes	搜索项目中所有未解析的头文件
	Create Parser Log File	生成解析器的日志文件
Properties	-	属性配置

5.2.8 Run 菜单项

Run 菜单项如表 5-10 所示。

表 5-10 Run 菜单项

菜单项	子菜单项	功能描述
Run Last Launched	-	运行上次启动过的程序
Debug Last	-	调试上次启动过的程序
Launched		
Profile Last	-	性能分析上次启动过的程序
Launched		
Profile History	-	使用性能分析历史记录中最后一次的
		性能分析配置
Profile As	Local C/C++	以本地 C/C++应用方式分析程序,直
	Application	接在主机环境中性能分析,使用本地
		安装的编译器和调试器
Profile	-	配置和管理性能分析程序时的设置选
Configurations		项
Run History	-	使用运行历史记录中最后一次的运行
		配置
Run As	C/C++ Container	以容器 C/C++应用方式运行程序,将
		可执行文件挂载到指定的 Docker 容

菜单项	子菜单项	功能描述
	Application	器中,在隔离的容器环境内运行
	Local C/C++ Application	以本地 C/C++应用方式运行程序,直 接在主机环境中运行,使用本地安装 的编译器和调试器
Run Configurations…	-	配置和管理运行程序时的设置选项
Debug History	-	使用调试历史记录中最后一次的调试 配置
Debug As	C/C++ Container Application	以容器 C/C++应用方式调试程序,将 可执行文件挂载到指定的 Docker 容 器中,在隔离的容器环境内调试
	Local C/C++ Application	以本地 C/C++应用方式调试程序,直 接在主机环境中调试,使用本地安装 的编译器和调试器
Debug Configuration…	-	配置和管理调试程序时的设置选项
Breakpoint Types	-	配置断点类型(C/C++ Breakpoint 或 C/C++ Dynamic Printf)
Toggle Breakpoint	-	在当前行新增或移除断点
Toggle Line Breakpoint	-	在代码的某一行设置一个行断点
Toggle Watchpoint	-	当前字段,新增或移除观察点
Toggle Method Breakpoint	-	指定方法的入口或在退出处设置断点
Skip All Breakpoints	-	切换禁用工作区中所有断点的模式
Remove All Breakpoint	-	永久删除工作区中的所有断点
External Tools	-	允许执行和配置外部工具(如构建脚 本、命令行工具等)

Debug Configurations

"Run > Debug Configurations…"选项,如图 **5-6** 所示,用于设置调 试选项和启动调试运行。

图 5-6 Debug Configurations

Create, manage, and run configurations	Ś.
Image: Startup Image	vse
Filter matched 22 of 24 items Revent AF ? Debug O	ply Close

常用的调试方式如表 5-11 所示。

调试方式	描述
GDB OpenOCD Debugging	使用 OpenOCD 调试 Arm 和 RISC-V MCU
GDB QEMU arm Debugging	使用 QEMU 仿真调试 Arm MCU
GDB QEMU riscv32	使用 QEMU 仿真调试 32 位 RISC-V MCU
Debugging	
GDB QEMU riscv64	使用 QEMU 仿真调试 64 位 RISC-V MCU
Debugging	
GEB SEGGER J-Link	使用 Segger 官方软件驱动和 J-Link 调试 Arm
Debugging	MCU

表 5-11 调试方式

Run Configurations

"Run > Run Configurations…"选项与"Debug Configurations…" 选项相同,这里不再详述。

5.2.9 Gowin 菜单项

Gowin 菜单项如表 5-12 所示,包含高云一些软件工具,例如 Programmer、GoBridgeDriver 等。

表 5-12 Gowin 菜单项

菜单项	功能描述
Programmer	启动下载软件 Programmer
GoBridgeDriver	启动驱动软件 GoBridgeDriver
SDK and Document	高云 MCU 相关的 SDK 和用户指南的官 方链接

Programmer

"Gowin > Programmer"选项,如图 5-7 所示,启动下载软件 Programmer。

图 5-7 Programmer

Sowin Programmer Version					-		×	
1								
3		Cardea	Device	Orentian	re di-	lleen C		IDCC
	Enable	Series		Operation	rs rile	User C	ode	0000000
1	\bowtie	General JTAG Device	JIAG_NOP	Bypass				000000
<								>
C	Output							ъх
R	eady							

GoBridgeDriver

"Gowin > GoBridgeDriver"选项,如图 5-8 所示,启动驱动软件 GoBridgeDriver。

图 5-8 GoBridgeDriver

👷 GoBridgeDriver V1.0	– 🗆 X
<u>D</u> evice <u>O</u> ptions <u>H</u> elp	
Device	✓ Edit
Driver	WinUSB (v6.1.7600.16385)
	Install WCID Driver
0 devices found.	

5.2.10 Window 菜单项

Window 菜单项如表 5-13 所示。

表 5-13 Window 菜单项

菜单项	子菜单项	功能描述	
New Window	-	打开一个新的工作区窗口,使用当 前透视图,方便同时查看和编辑多 个项目	
	Toggle Split Editor (Horizontal)	在当前窗口中横向打开一个新的编 辑器实例	
Editor	Toggle Split Editor (Vertical)	在当前窗口中垂直打开一个新的编 辑器实例	
	Clone	在当前窗口中复制一个编辑器实例	
	Hide Toolbar	隐藏工具栏	
	Hide Status Bar	隐藏状态栏	
	Toggle Full Screen	切换全屏模式	
Appearance	Maximize Active View or Editor	最大化活跃的视图或编辑器	
	Minimize Active View or Editor	最小化活跃的视图或编辑器	
	Build Targets	显示当前透视图中的 Build Targets 视图	
	C/C++ Projects	显示当前透视图中的 C/C++ Projects 视图	
Show View	Console	显示当前透视图中的 Console 视图	
	Documents	显示当前透视图中的 Documents 视 图	
	Include Browser	显示当前透视图中的 Include	

菜单项	子菜单项	功能描述
		Browser 视图
	Navigator	显示当前透视图中的 Navigator 视 图
	Outline	显示当前透视图中的 Outline 视图
	Problem Details	显示当前透视图中的 Problem Details 视图
	Problems	显示当前透视图中的 Problems 视图
	Project Explorer	显示当前透视图中的 Project Explorer 视图
	Properties	显示当前透视图中的 Properties 视 图
	Search	显示当前透视图中的 Search 视图
	Task List	显示透视图中的 Task List 视图
	Tasks	显示当前透视图中的 Tasks 视图
	Terminal	显示当前透视图中的 Terminal 视图
	Other	显示当前透视图中所有可用的视图
	Open Perspective	打开一个新的透视图
	Customize Perspective …	自定义当前透视图的菜单栏、工具 栏和视图的可见性
	Save Perspective As	将当前透视图的布局和设置保存为 新的透视图
Perspective	Reset Perspective	将当前透视图恢复到默认布局,撤 销所有自定义更改
	Close Perspective	关闭当前透视图,返回到上一个活 跃的透视图
	Close All Perspectives	关闭所有打开的透视图,清空工作 区布局
	Show System Menu	显示当前活动视图或编辑器的系统 菜单
	Show View Menu	显示当前活动视图的菜单
Navigation	Active Editor	激活编辑器,将焦点切换到当前活 动的编辑器
	Next Editor	切换到下一个打开的编辑器
	Previous Editor	切换到上一个打开的编辑器
	Switch to Editor	列出所有打开的编辑器,允许用户 选择切换编辑器

菜单项	子菜单项	功能描述	
	Next View	切换到下一个打开的视图	
	Previous View	切换到上一个打开的视图	
	Next Perspective	切换到下一个打开的透视图	
	Previous Perspective	切换到上一个打开的透视图	
Preferences	_	当前工作区的首选项,配置 GMD 软件的外观、行为和插件设置	

5.2.11 Help 菜单项

Help 菜单项如表 5-14 所示。

菜单项	功能描述		
Welcome	打开欢迎页面,了解 Eclipse 基本功能		
Help Contents	打开 Eclipse 帮助文档,详细介绍各种功能和使用方法		
Search	在帮助文档中搜索关键词		
Show Context Help	显示上下文帮助		
Show Active Keybindings	显示当前可用的快捷键列表		
Tips and Tricks	提供使用 Eclipse 的一些技巧和建议		
Report Bug and Enhancement…	引导用户提交问题或建议		
Cheat Sheets	提供交互式的操作指南,指导用户完成多步骤任 务		
Eclipse User Storage	用于以加密形式安全地存储敏感信息		
Performance Setup Tasks	用于优化 IDE 性能的工具		
Check for Updates	检查并安装 Eclipse 或已安装插件的更新		
Install New Software	安装新的插件或扩展功能		
Eclipse Marketplace	打开 Eclipse 插件市场,可方便浏览和安装插件		
About GMD	查看当前 GMD 软件的版本信息等		

表 5-14 Help 菜单项

About GMD

"Help > About GMD"选项如图 5-9 所示。

🐝 About GMD		_		×
	Gowin for Embedded C/C++ Developers (includes	Incubating o	ompone	nts)
	Version: 2025.01			
$\checkmark \checkmark \checkmark$	Copyright (C)2014-2025 Gowin Semiconductor Tec All rights reserved.	hnology Co.,	Ltd.	
	GMD is a new generation of MCU software develo developed by GOWIN according to its own FPGA a features. GMD supports the MCU compilation toolchain of A RISC-V architecture and also supports MCU compi downloading and debugging of ARM and RISC-V a	pment enviro and MCU Sol ARM architec ilation, linking architecture.	onment s C archite ture and g,	elf- cture
© 🔵	🥖 🧁 🜔 🚱 💉 🤃	\$		1
? Installatio	n Details		<u>C</u> lose	

图 5-9 About GMD

5.3 工具栏

主要提供一些常用功能的快速访问入口,包含创建文件或项目 (New)、保存文件(Save)、保存所有文件(Save All)、撤销(Undo)、 恢复(Redo)、管理配置(Manage configurations)、编译(Build、Build All)、终端(Open a Terminal)、下载(Programmer)、调试(Debug)、运 行(Run)、性能分析(Profile)、搜索(Search)等,如表 5-15 所示。

表 5-15 工具栏选项

选项	图标	功能描述
New	•	快速创建文件/项目
Save		保存当前文件
Save All	a	保存所有己打开的文件
Manage configurations	•	管理配置(包括 Debug 和 Release)
Build	×	编译当前工程项目
Build All	010	编译工作区所有已打开的工程项目

选项	图标	功能描述
Undo	4	撤销最近的编辑操作
Redo	\$	重载最近的编辑操作
Open a Terminal	Ę	打开一个终端,例如串口终端
Skip All Breakpoints	ø	调试过程中,所有断点将被标记为跳 过,程序执行时不会在这些断点处暂停
Restart	€	在不终止整个调试会话的情况下重新开 始当前方法的执行
Debug	苓 •	启动应用程序的调试会话
Run	0 -	启动应用程序的运行会话
Profile		启动应用程序的性能分析会话
Programmer	₩ 81	启动下载软件 Programmer
Open Element	(29	打开当前光标所在元素的详细信息
Open Task	Þ	打开当前工作区中的任务列表
Search	A •	搜索
Toggle Mark Occurrences	4	高亮显示当前文件中所选元素(如变 量、方法、类型等)的所有引用
Toggle Work Wrap	R	在编辑器中启用或禁用自动换行
Toggle Block Selection Mode		以矩阵方式选择和编辑文本
Show Whitespace Characters	Π	编辑器中显示空白字符
Next Annotation		快速导航到源代码中的下一个注释标记
Previous Annotation	₩	快速导航到源代码中的上一个注释标记
Last Edit Location	*>	快速跳转到上一个编辑的代码位置

选项	图标	功能描述
Next Edit Location	47	快速跳转到下一个编辑的代码位置
Back	< →	快速回到之前查看的资源或视图
Forward	\$ ▼	返回到上次"Back"操作之后的资源或 视图
Pin Editor		自动关闭编辑器
Open Perspective	Ê	打开一个新的透视图
C/C++	Fic	切换到 C/C++透视图
Debug	*	切换到 Debug 透视图

5.4 项目资源管理器

主要提供软件工程项目及其相关文件的管理和显示功能,如图 5-10 所示。



Project Explorer X	🖻 🔄 🏹 🗊 🕴 🗖 🗖
✓ E cm3_demo	
> 🎉 Binaries	
> 🔊 Includes	
> 👝 Debug	
Y 🔁 src	
> 👝 library	
> 🔁 project	
Cm3_freertos	
🗀 cm3_gcov_demo	
🗀 cm3_gprof_demo	
🗀 cm3_rtthread_nano	
📁 cm3_ucos_iii	

5.5 代码编辑区

提供基本的文件编辑及查看功能如图 5-11 所示。

"File > New > Source File"或 **"File > New > Header File"**选项用于 创建新的源文件(.c/cpp) 或头文件(.h),显示在编辑区。

"File > Close Editor"选项用于关闭编辑区当前显示的文件。"File >

Close All Editors"选项,关闭编辑区所有打开的文件。

图 5-11 代码编辑区

ic main.c ⊃	X	· 🗆
1⊖ /*		^
2 ***	***************************************	
3 * @	Offile main.c	
4 * @	Dauthor GowinSemicoductor	
5 * @	device Gowin_EMPU(GW1NS-4C)	
6 * @	Obrief Main function.	
7 ***	***************************************	
8 */		
9		
10 /* I	Includes*/	
11 #inc	clude "demo.h"	
12		
13		
14 /* D	Definitions*/	
15		
16 //Ap	oplication entry function	
1/0 int	main(Vold)	
10 1		
19 #17	And_instea_dem();	
20 21 #ond	and _ mascer_demo(),	
21 #enu		
23 #if	ADR2 MASTER DEMO	
24	ab2 master demo():	
25 #end	if the second seco	
26		
27 #if	DMM DEMO	
28	dmm_demo();	
29 #end	dif	
30		
31 #if	FIFO_DEMO	
32	fifo_demo();	
33 #end	dif	~

5.6 控制台

显示软件在运行过程中的各种处理信息,可手动切换各个视图查看不同 类型的输出信息,例如控制台(Console)、问题(Problems)、任务 (Tasks)、终端(Terminal)等。

5.6.1 控制台视图

控制台视图如图 5-12 所示,例如显示编译信息。

图 5-12 控制台视图

🖹 Problems 🧔 Tasks 📮 Console 🗙 🔲 Properties 🧬 Terminal	
🗶 🕹 🔂 📰 🔂 💳 🖳 🧲 🖬 🖬 🖬	r 📸 🔻
CDT Build Console [cm3_demo]	
./src/library/middlewares/dmm/malloc.o ./src/library/middlewares/delay/delay.o	~
<pre>./src/library/libraries/drivers/src/gw1ns4c_gpio.o ./src/library/libraries/drivers/src/gw1ns4c_i2c.o</pre>	
<pre>./src/library/libraries/drivers/src/gw1ns4c_misc.o ./src/library/libraries/drivers/src/gw1ns4c_rtc.o</pre>	
<pre>./src/library/libraries/drivers/src/gw1ns4c_spi.o ./src/library/libraries/drivers/src/gw1ns4c_syscon.o</pre>	
./src/library/libraries/drivers/src/gwins4c_timer.o ./src/library/libraries/drivers/src/gwins4c_uart.o	
//src/library/libraries/univers/src/gwins4c_wudg.o	
/sr/library/libraries/msis/cm3/device_support/suct cup/gmu/scar cup_gmins+c.0	
/src/library/libraries/cmsis/cm3/core_support/md/core_cm3.o	
Finished building target: cm3 demo.elf	
Invoking: GNU ARM Cross Create Flash Image	
arm-none-eabi-objcopy -O binary "cm3_demo.elf" "cm3_demo.bin"	
Finished building: cm3_demo.bin	
Toyoking: GNULARM Cross Print Size	
arm-none-eabi-sizeformat=berkeley "cm3 demo.elf"	
text data bss dec hex filename	
29988 1764 392 32144 7d90 cm3 demo.elf	
Finished building: cm3_demo.siz	
10:44:04 Build Finished. 0 errors, 0 warnings. (took 15s.231ms)	
	~

5.6.2 问题视图

问题视图如图 5-13 所示,例如显示编译过程中报出的警告和错误提示。

图 5-13 问题视图

🔐 Problems X 🖉 Tasks 📮 Console 🔲 Properties 🧬 Terminal 🛛 🍸 🖙 🖗 🗖 🗖								
3 errors, 0 warnings, 1 other								
Description	Resource	Path	Location	Туре				
✓ Ø Errors (3 items)								
😣 'mi' undeclared (first use in this function);	printf_demo.c	/cm3_demo/src/p	line 43	C/C++ Prob				
expected '=', ',', ';', 'asm' or '_attribute_'	printf_demo.c	/cm3_demo/src/p	line 43	C/C++ Prob				
Ø make: *** [src/project/printf/subdir.mk:20:	cm3_demo			C/C++ Prob				
 i Infos (1 item) 								
i each undeclared identifier is reported only	printf_demo.c	/cm3_demo/src/p	line 43	C/C++ Prob				

5.6.3 终端视图

"Navigate > Show In > Terminal"选项用于启动终端如图 5-14 所示。

🐝 workspace_cm3_h - cm3_de	mo/src/p	roject/printf	/printf_de	emo.c - GN	1D											-		×
<u>File E</u> dit <u>S</u> ource Refac <u>t</u> or	<u>N</u> avigate	Se <u>a</u> rch	<u>P</u> roject	<u>R</u> un <u>G</u> ow	in <u>W</u> indow	<u>H</u> elp												
📑 🕶 🗐 👘 😻 🕶 🔦 🕶 [Go	Into					\$F. •	• 🜔 •	· 🚷 -	9 -	- 10	9 6	- 🔗 -	- 🥖	ə 🗟	þ 🔲	П	
🐓 → 🦂 → 🏷 😅 🔶 → 🗠	Go	То				>									0	2	E	睑ゃ
Project Explorer 🗙 🗖	Оре	n Declaratic	n			F3				- 6	3	B≊ C	X]т 🍭	в	D		•
🖻 🕏 🏹 🔤	Ope	en Type Hier	archy			F4	L .			^			69	⊟ ↓ª₂	8	× ^s	● 茶	8 🗖
✓ [™] cm3_demo	Ope	en Call Hiera	rchy		Ctrl+A	t+H	L .					1.1	der	no.h				
> 🐝 Binaries	Ope	en Include Br	rowser		Ctrl+4	lt+I	L .					1	uar uar	t.h				
> Includes	Tog	gle Source/I	Header		Ctrl+	Tab	L .					- 1	std	io.h				
> 👝 Debug	🍅 Оре	n Element			Ctrl+Shi	ft+T							prii	ntf_dem	10(VOI	d) : I	nt	
G library	Ope	en Type in Hi	erarchy		Ctrl+Shi	t+H	L .											
> > project	Ope	n Element ir	n Call Hie	rarchy			L											
i cm3 freertos	Ope	en Resource.			Ctrl+Shi	ft+R												
📋 cm3_gcov_demo	Оре	n Discovere	d Type				L .											
📋 cm3_gprof_demo		n Task			Ctrl+	-F12	%x	:\r\n",	-3, -	з,								
cm3_rtthread_nano	Acti	vate Task			Ctr	+ F9	L .											
🔲 cm3_ucos_iii	O Dea	ctivate Task			Ctrl+Shift	+ F9	ssa	ee"):		~								
										>								
		en Setup				>	8	K 🕂	û		6)	- -	2 -	1	• -	
	Upe Ope	en Setup Log	1															
	Sho	w In			Alt+Shif	t+W>	٦	Termi	nal			1						^
	Qui	ck Outline			Ctr	l+O	1∎	Includ	e Brow	ser								
	😔 Nex	t Annotation			С	trl+.	Ec	C/C++	+ Projec	ts								
	Prev	vious Annota	tion		С	trl+,	6	Projec	t Explo	rer								
	Prev Prev	vious Edit Lo	cation		Ctr	l+Q	R	Proble	em Deta	ails								
	<⊅ Nex	t Edit Locati	on		Ctrl+Alt+R	ight		Outlin	e									
	Go	to Line			Ct	rl+L	2	Systen	n Explo	rer								
	C Bac	k			د. ۱+ ـ	left >	-	Prope	rties			J						
		vard				ight >	1											~
		VV	паріе		Smart insen	agine 2	4	7:42:	1187									

图 5-14 启动终端

可以设置 Local Terminal、SSH Terminal、Serial Terminal、Telnet Terminal 不同类型的终端,例如常用的串口终端如图 5-15 所示。

图 5-15 串口终端

Ŷ	🌢 Launch Ter	minal		—		×
	Choose term	inal: Serial	Terminal			~
	Settings					
	Serial port:	COM5				~
	Baud rate:	38400				~
	Data size:	8				\sim
	Parity:	None				\sim
	Stop bits:	1				\sim
	Encoding:	Default (ISC)-8859-1)			~
	_					
	0		01/		~	
	\bigcirc		OK		Cano	el

串口终端输出串口打印信息如图 5-16 所示。

图 5-16 串口终端输出

🖹 Problems 🛛 🕫 Tasks	📮 Console	Properties	🎜 Terminal 🛛	M 🖣	🕞 🚮	۵	
🖳 COM55 🗙							
Hello world!							\sim
5 = 5							
-214/48364/ = - max 1	Integer						
cnar a = a							
hex 00 = 00							
signed -3 = unsigned	4294967293	= hex ffffff	fd				
0 message(s)							
0 message(s) with %							
							14



6.1 新建工程

菜单栏 "File > New > C/C++ Project" 选项或工具栏 "New > C/C++

Project"(" □")选项如图 6-1 所示,创建 C/C++工程项目。

图 6-1 新建工程



常用项目类型 "C Managed Build" 和 "C++ Managed Build",分别为 新建 C 工程和 C++工程,请根据实际需求来选择。

例如新建 C 工程, Arm 和 RISC-V MCU 新建工程的步骤分别如下所述。

6.1.1 新建工程(Arm MCU)

1. 选择项目类型和软件工具链

创建项目名称和项目位置,选择项目类型和软件工具链,如图 6-2 所示。

图 6-2 选择项目类型和软件工具链

**	—	
C Project Create C project of selected type		Ď
Project name: test_demo ✓ Use default location Location: D:\GMD_2025_workspace\works Choose file system: default ∨	space_cm3_h\test_demc	B <u>r</u> owse
Project type:	Toolchains: Arm Cross GCC MinGW GCC RISC-V Cross GCC	
✓ Show project types and toolchains only	if they are supported on	the platform
? < <u>B</u> ack <u>N</u> ext	t > <u>F</u> inish	Cancel

例如:

- Project name: test_demo
- Location: Use default location,使用当前工作区
- Project type: Empty Project
- Toolchains: Arm Cross GCC
- 2. 选择平台和配置

选择平台和配置,包含"Debug"和"Release"如图 6-3 所示。

**	— 🗆 X								
Select Configurations Select platforms and configurations you wish to deploy on	Ď								
Project type: Executable Toolchains: Arm Cross GCC Configurations:									
☑ ऄ Debug ☑ ऄ Release	Select all Deselect all								
	Advanced settings								
Use "Advanced settings" button to edit project's properties. Additional configurations can be added after project creation. Use "Manage configurations" buttons either on toolbar or on property pages.									
? < <u>B</u> ack <u>N</u> ext > <u>F</u> in	ish Cancel								

图 6-3 选择平台和配置

例如:

Configurations: 勾选 Debug 和 Release。

3. 选择软件工具链和路径

选择具体的 Arm 交叉编译软件工具链以及所在路径如图 6-4 所示。 GMD 软件已集成好软件工具链,以及已配置好路径,一般情况下默认选择 即可。

**		_		×
GNU Arm Cross	Toolchain ain and configure path. AArch64 projects require an AArch64 toolchain!			2
Toolchain name:	xPack GNU Arm Embedded GCC (arm-none-eabi-gcc)			~
Toolchain path:	\${eclipse_home}\toolchain\gowin\gcc_arm\bin	Browse.	xP	ack
On macOS use S	hift+Cmd+'.' to show the hidden folders while browsing the file system. xpm uses a .content folder	to store	the bir	naries.
?	< <u>B</u> ack <u>N</u> ext > <u>F</u> inish		Cance	el

图 6-4 选择工具链和路径

例如:

- Toolchain name: xPack GNU Arm Embedded GCC (arm-none-eabigcc);
- Toolchain path: \${eclipse_home}\toolchain\gowin\gcc_arm\bin。

6.1.2 新建工程(RISC-V MCU)

1. 选择项目类型和软件工具链

创建项目名称和项目位置,选择项目类型和软件工具链,如图 6-5 所示。

**	- 🗆 X
C Project Create C project of selected type	
Project name: test_demo	
Use gerauit location Location: D:\GMD_2025_workspace\w Choose file system: default	rorkspace_riscv\test_demo Browse
 Project type: Executable Empty Project Hello World ANSI C Project Hello World Arm C Project Hello World RISC-V C Project Shared Library Static Library Static Library Makefile project 	t I colchains: Arm Cross GCC MinGW GCC RISC-V Cross GCC t
? < <u>B</u> ack	Next > <u>Finish</u> Cancel

图 6-5 选择项目类型和软件工具链

例如:

- Project name: test_demo
- Location: Use default location,使用当前工作区
- Project type: Empty Project
- Toolchains: RISC-V Cross GCC
- 2. 选择平台和配置

选择平台和配置,包含"Debug"和"Release"如图 6-6 所示。

**				×					
Select Configura Select platforms	tions and configurations you wish to deploy on			Ŷ					
Project type: Toolchains: Configurations:	Executable RISC-V Cross GCC								
☑ ѷ Debug ☑ ѷ Release		De	Select all eselect a	11					
		Advan	ced sett	ings					
Use "Advanced settings" button to edit project's properties. Additional configurations can be added after project creation. Use "Manage configurations" buttons either on toolbar or on property pages.									
?	< <u>B</u> ack <u>N</u> ext > <u>F</u> in	ish	Car	ncel					

图 6-6 选择平台和配置

例如:

Configurations: 勾选 Debug 和 Release。

3. 选择软件工具链和路径

选择具体的 RISC-V 交叉编译软件工具链以及所在路径如图 6-7 所示。 GMD 软件已集成好软件工具链,以及已配置好路径,一般情况下默认选择 即可。

**				×
GNU RISC-V Cros	s Toolchain			•
Select the toolch	ain and configure path			4
Toolchain name:	xPack GNU RISC-V Embedded GCC (riscv-none-elf-gcc)			~
Toolchain path:	\${eclipse_home}\toolchain\gowin\gcc_riscv\bin	Browse	. xPa	ack
On macOS use S	hift+Cmd+'.' to show the hidden folders while browsing the file system. xpm uses a .content folder	to store t	he bin	aries.
?	< <u>B</u> ack <u>N</u> ext > <u>Finish</u>		Cance	;

图 6-7 选择工具链和路径

例如:

- Toolchain name: xPack GNU RISC-V Embedded GCC (riscv-noneelf-gcc);
- Toolchain path: \${eclipse_home}\toolchain\gowin\gcc_riscv\bin。

6.2 导入已有工程

高云提供所有 MCU 产品的参考设计,用户可以直接导入使用,导入已 有工程的步骤如下所述。

- 1. "File > Import..."选项,开始导入已有工程。
- 2. 选择"Existing Projects into Workspace"如图 6-8 所示。

M Import		×
W mport		^
Select	_	-
Create new projects from an archive file or directory.	2	5
Select an import wizard:		
type filter text		
🗸 🗁 General		^
🕼 Archive File		
😭 Existing Projects into Workspace		
🛱 File System		
Preferences		
Projects from Folder or Archive		
> 🥟 C/C++		
> 📂 Git		
> 🦻 Install		
> 🔁 Oomph		
> 🗁 RPM		
> 🤛 Run/Debug		~
(?) < Back Next > Finish	Cance	

图 6-8 选择导入的方式

3. 选择要导入的已有工程,以及设置选项。

点击"Browse",选择需要导入的工程路径如图 6-9 所示。需要导入的工程成功被 IDE 识别,可以勾选"Copy projects into workspace"选项,将导入的工程复制到当前工作区。

冬	6-9	选择需要导入的工程	睈
---	-----	-----------	---

👹 Import			_	- 🗆	×
Import Projects Select a directory to sea	ch for exi	sting Eclipse p	rojects.		
 Select root directory: Select <u>a</u>rchive file: <u>P</u>rojects: 	D:\GMD	2025_workspa	ce\workspace_ ~	B <u>r</u> ow	/se
☑ picorv32_demo (D	:\GMD_20)25_workspace	workspace_riscv	<u>S</u> elec	t All ect All
 Options Search for nested products of the search of the sea	ojects orkspace I projects eady exist	upon completi t in the worksp	on ace	R <u>e</u> fr	esh
Working sets	ing sets		~	Ne <u>w</u> S <u>e</u> lect.	
?	<u>B</u> ack	<u>N</u> ext >	<u>F</u> inish	Can	cel

7 导入旧版本 GMD 创建的工程

从 GMD 2025.01 开始的软件版本中,因为软件工具链做了较大版本的 更新,如果用户在新的 GMD 软件中想要使用 GMD V1.2 及以下的旧版软件 创建的工程,或者使用旧版高云 MCU SDK,需要参照本章内容进行修改。

将旧版 GMD 中的工程导入到 GMD 2025.01 新版软件中时,或者使用 旧版 MCU SDK (旧版 SDK 指的是在 GMD 2025.01 发布之前所发布过的 SDK)所创建的工程,因为工程配置使用的是旧版 GCC,当找不到对应的 软件工具链时,会出现编译报错等问题,导致软件工程无法正常使用。

请参照以下方法,分别重新配置导入的 Arm MCU 和 RISC-V MCU 工程的软件工具链。

7.1 修改导入的 Arm MCU 工程

- 1. 参照 6.2 导入已有工程节导入已有工程,导入旧版 GMD 创建的 Arm MCU 工程。
- 编译工程时,因找不到对应的旧版软件工具链,会出现编译报错如图 7-1 所示。

₩ workspace test - cm3 demo	/src/project/led/led_demo.c - GMD — 🗌 🗙
File Edita Course Reference I	Neight Stande Deciset Due Caule Minden Hele
: 🛄 🕈 📾 🐨 🛛 🐨 🕶 💊 🕇 🔤	
	▼ 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
陷 Project Explo 🗙 📮 🗖	Led_demo.c ×
🖻 🕏 🏹 🖙 i	13 ^ I La V × 3 ● ₩ 8
✓ ∰ cm3_demo	14 /* Definitions
> 🗊 Includes	16 #if LED DEMO 🗳 gpio.h
> 👝 Debug	17 delay.h
V 🖶 src	18 /* Includes: */
> 👝 library	29 #include "gelay.h"
✓ project	21
> 🕞 ahb2_master	22 22 /* Definitions: */
> 🍋 apb2_master	24
> 👝 dmm	25 //Application entry function
> 👝 gpio_in_intr	260 int led_demo(void)
> 👝 hyper_ram	27 1 28 SystemInit(); //Initializes system clock
> 👝 i2c master	29 gpio_init(); //Initializes GPIO
> 👝 interrupt	30 delay_init(); //Initializes delay functions
> 👝 keyscan	31 32 while(1)
Y 🚌 led	33 {
> 🔂 led_demo.c	GPIO_ResetBit(GPIO0,GPIO_Pin_0); //LED1 on delay_sec(1): //1 sec
> 🗁 printf	< >>
> 📂 psram	🔋 Problems 🧖 Tasks 🔍 Console X 🔲 Properties 🛛 💥 😌 😚 🖼 🖬 🔲 💭 🖃 🔻 🖻 🕶 🗖
> 📂 rtc	CDT Build Console [cm3 demo]
> 📂 spi_flash	make all
> 📂 spi_master	Building file:/src/project/wdog/wdog_demo.c
> 📂 timer	Invoking: GNU ARM cross C Compiler arm-none-reahi-gcc _mrojecontex-m3 _mthumb -00 _fmessage-length=0 _fsigned-char _ffunction-sections _fdata-section
> 📂 uart_rx	e:/gmd/toolchain/gowin/build_tools/bin/sh: arm-none-eabi-gcc: not found
> 📂 uart_rx_intr	<pre>make: *** [src/project/wdog/subdir.mk:20: src/project/wdog/wdog_demo.o] Error 127</pre>
> 📂 user_intr	make all terminated with exit code 2. Build might be incomplete.
> 👝 wdog	18:11:36 Build Failed. 1 errors, 0 warnings. (took 534ms)
> 🖻 demo.h	
> 尾 main.c	<
	Writable Smart Insert 1:1:0

图 7-1 Arm 工程导入后编译报错

3. 修改软件工具链,重新选择可用的 Arm 工具链。

选定当前工程,右键选择 "Properties > C/C++ Build > Settings > Toolchains",修改软件工具链。

如图 7-2 所示, "Name" 下拉列表中重新选择 "xPack GNU Arm Embedded GCC (arm-none-eabi-gcc)" 软件工具链。

ype filter text	Settings	⟨- + -
ype filter text Builders C(C++ Build Build Variables Environment JSON Compilation Dat Logging Settings Tool Chain Editor C(C++ General Linux Tools Path MCU Project Natures Project References Run/Debug Settings	Settings Configuration: Debug [Active] Tool Settings Toolchains Tool Settings Toolchains Configuration: Image: Image	Manage Configurations with an age Configurations with a start and a start and a start and a start and a start a start and a start a star
Task Repository Task Tags Validation WikiText	Archiver: devkiPy o ARM EABI (arm-eabi-gcc) DEF Hex/Bin converter: Yagarto, Summon, etc. ARM EABI (arm-eabi-gcc) DEF Yagarto, Summon, etc. ARM EABI (arm-eabi-gcc) DEF Yagarto, Summon, etc. ARM EABI (arm-eabi-gcc) DEF Isting generator: Linaro AArch64 bare-metal ELF (aarch64 Linaro AArch64 big-endian bare-metal EL inaro AArch64 big-endian Linux GNU (arch64-linux Build command: Linaro AArch64 big-endian Linux GNU (arch64-linux Build command: Custom (arm-none-eabi-gcc) Remove command: rm Prefer local xpacks/.bin path Toolchain path:	RECATED none-eabi-gcc) DEPRECATED arch64-none-elf-gcc) i-elf-gcc) iLF (aarch64_be-elf-gcc) x-gnu-gcc) aarch64_be-linux-gnu-gcc)
	(to change it use the <u>global</u> or <u>workspac</u> Build tools path: \${eclipse_home}\toolchain\gowin\build_t (to change it use the <u>global</u> or <u>workspac</u> ✓ Create flash image ☐ Create extended listing	<u>re</u> preferences pages or the <u>project</u> properties page) tools\bin <u>re</u> preferences pages or the <u>project</u> properties page)
2	☑ Print size	Apply and Close Costel

图 7-2 软件工具链设置

7.2 修改导入的 RISC-V MCU 工程

- 1. 参照 6.2 导入已有工程节导入已有工程,导入旧版 GMD 创建的 RISC-V MCU 工程。
- 编译工程时,因找不到对应的旧版软件工具链,会出现编译报错如图 7-3 所示。

🐳 workspace_test - picorv32_de	emo/src/main.c - GMD			_		×
<u>File Edit Source Refactor M</u>	<u>N</u> avigate Se <u>a</u> rch <u>P</u> roject <u>R</u> un <u>G</u> owin <u>W</u> indow <u>H</u> elp					
📑 🕶 🔚 🕼 😸 🕶 🚮		+ : .	/ 🕫 🗄	e a r		
				Q	i 🗈 🖪	d 💠
Project Explo 🗙 🗖 🗖	i main.c × □		Be O	Х 🗐 Т	⊛ B '	
	4 * @author GowinSemicoductor	^	59	F 14 📎	∿ ^s ● 3	# 8
× ⇔ picorv32 demo	5 * @device Gowin_PicoRV32			config.h	~	~
> 🔊 Includes	6 * @Driet Main body. 7 ************************************			picory32.	h	
> 🕞 Debug	8 */			firmware.	h	
× = src	9			ira.h		
) Carban	10 /* Includes			simpleuar	t h	
> Co lib	12 #include "picorv32.h"			whuarth	un	
> D config h	13 #include "firmware.h"			whenie h		
> m conlig.n	14 #include "irq.h"			wbgpio.n		
> [a] custom_ops.s	15 #include "simplevart.n" 16 #include "whyart.h"			wospinasi DEV. CWG	n.n	
> ic irq.c	17 #include "wbgpio.h"	-	#	DEV_GWS	A25_15	
> h irq.h	18 #include "wbspiflash.h"		#	MEM_MA	P_ADDRI	
> c loader.c	19	-	#	MEM_MA	P_ADDR2	-
> 🚾 main.c	20 21 /* Definitions		#	MEM_MA	P_ADDR1	
> h picorv32.h	22	-	#	MEM_MA	P_ADDR2	£
> 🔝 start.S	23 //User-defined device		#	MEM_MA	P_ADDR1	
sections_debug.lds	24 #define DEV_GW5A25_15	-	#	MEM_MA	P_ADDR2	£
sections_xip.lds	25 26 //SPT-Flash memory man address	~	#	get_char()	l.	~
sections.lds	< > >		<			>
	🖹 Problems 🖉 Tasks 📃 Console 🗙 🔲 Properties 🛛 🗱 🐥 😚 🕵 🗔) , F	🛃 🗐 🔻	📑 👻	- 8
	CDT Build Console [picorv32_demo]					
	Invoking: GNU RISC-V Cross C Compiler riscv-none-embed-gcc -march=rv32imc -mabi=ilp32 -mtune=size -mcmodel=medany -msma -msave-restore -00 -fmessage-length=0 -ffunction-sections -fdata-sections -fno-bu \GMD_2025_workspace\workspace_test\picorv32_demo\src" -I"D:\GMD_2025_workspace\wo \bsp" -I"D:\GMD_2025_workspace_test\picorv32_demo\src\lib" -std=gnu1 - \dmathcal{emb}	ll-da iltin rkspa Wa,-	ta-limi -g3 -] ce_test	it=8 -mstr ["D: t\picorv32	ict-ali _demo∖s	gn î
	adnins="src/lib/tirmware.o.lst" -MMD -MP -MF"src/lib/tirmware.d" -MT"src/lib/tirm "src/lib/firmware.o" "/src/lib/firmware.c"	ware.	oc -	-0		
	e:/gmd/toolchain/gowin/build tools/bin/sh: riscv-none-embed-gcc: not found					
	<pre>make: *** [src/lib/subdir.mk:23: src/lib/firmware.o] Error 127</pre>					
	"make all" terminated with exit code 2. Build might be incomplete.					
	17:46:09 Build Failed. 1 errors, 0 warnings. (took 1s.430ms)					~

图 7-3 RISC-V 工程导入后编译报错

3. 修改软件工具链,重新选择可用的 RISC-V 工具链。

选定当前工程,右键选择 "Properties > C/C++ Build > Settings > Toolchains",修改软件工具链。

如图 7-4 所示, "Name" 下拉列表中重新选择 "xPack GNU RISC-V Embedded GCC (riscv-none-elf-gcc)" 软件工具链。

/pe filter text	Settings	← < ⇒
 Resource Builders C/C++ Build Build Variables 	Configuration: Debug [Active]	✓ Manage Configurations
Environment JSON Compilation Dat	🛞 Tool Settings 🛞 Toolchains 🔳 Devices 🟮 Container Settings 🎤 Build St	teps 🚇 Build Artifact 🎇
Logging Settings	Name: GNU MCU RISC-V GCC (riscv-none-embed-gcc) DEPRECA	TED
Tool Chain Editor	Architecture: xPack GNU RISC-V Embedded GCC (riscv-none-elf-gcc) RISC-V GCC/Newlib (riscv64-unknown-elf-gcc)	
Linux Tools Path	Prefix: RISC-V GCC/Linux (riscv64-unknown-linux-gnu-gcc) RISC-V GCC/RTEMS (riscv64-unknown-rtems-gcc)	
MCU Project Natures	C compiler: Compiler: Custom (riscy-none-embed-gc Custom (riscy-none-embed-gc) DEPRECA Custom (riscy-none-eff-acc)	cc) DEPRECATED TED
Project References Run/Debug Settings	C++ compiler: g++	
Task Repository Task Tags	Archiver: ar	
Validation WikiText	Listing generator: objdump	
	Size command: size	
	Build command: make	
	Remove command: rm	
	□ Prefer local xpacks/.bin path	
	Toolchain path: (to change it use the global or workspace preferences page	ges or the project properties page)
	Build tools path: [\$(eclipse_home)\toolchain\gowin\build_tools\bin	ges er me <u>projeer</u> properties page)
	(to change it use the <u>global</u> or <u>workspace</u> preferences page	ges or the <u>project</u> properties page)
	⊡ Create flash image	
	✓ Create extended listing ✓ Print size	

图 7-4 RISC-V 工具链设置

4. 修改编译选项,填写额外的扩展指令选项"_zicsr"。

选定当前工程,右键选择 "Properties > C/C++ Build > Settings > Tool Settings > Target Processor", "Other extensions" 选项填写 "_zicsr",如 图 7-5 所示。

₩ Properties for picorv32_der	no		- 🗆 X
	Settings		<□ ▼ □> ▼ 8
 > Resource Builders ✓ C/C++ Build Build Variables Environment JSON Compilation Dat Logging 	Configuration: Debug [Active]	Container Settings	 ✓ Manage Configurations Muild Steps Build Artifact "2 RV32I (-march=rv32i*) ✓
Settings Tool Chain Editor > C/C++ General Linux Tools Path > MCU	 Øptimization Warnings Debugging SOLU RISC-V Cross Assembler Preprocessor 	Multiply extensio Atomic extensio Floating point Compressed ext	n (RVM) n (RVA) None ✓ tension (RVC)
Project Natures Project References Run/Debug Settings > Task Repository	隧 Includes 隆 Warnings 隆 Miscellaneous	Other extensions Integer ABI	_zicsr ILP32 (-mabi=ilp32*) ~
Task Tags > Validation WikiText	 ♥ GNU RISC-V Cross C Compiler 	Floating point ABI Tuning Code model	None ~ Size (-mtune=size) ~ Medium Any (-mcmodel=medany) ~
	 ➢ Warnings ➢ Miscellaneous ✓ ভ GNU RISC-V Cross C Linker ➢ General 	Small data limit Align	8 Strict (-mstrict-align)
	 ➢ Clibraries ➢ Miscellaneous ✓ ※ GNU RISC-V Cross Create Flash Image 	Small prologue/	epilogue (-msave-restore) erations to call library functions (-mmemcpy)
	 № Seneral ♥ SGNU RISC-V Cross Create Listing ֎ General ♥ SGNU RISC-V Cross Print Size @ General 		
< >			Restore <u>D</u> efaults <u>Apply</u>
?			Apply and Close Cancel

图 7-5 编译选项设置



GMD 软件集成多种软件工具链,支持 Arm MCU 和 RISC-V MCU 工程的编译。

8.1 编译工具

GMD 软件集成 Gowin GNU 软件工具链,是高云在 GNU 基础版本上 针对高云的 MCU 产品特性进行了再次开发。GNU 软件工具链是由 GNU 项 目提供的一套完整的软件开发工具链,包含编译器、调试器、链接器、库文 件等一系列用于软件开发和构建的必需工具。GNU 软件工具链以其开源、 跨平台、高度可定制和强大的功能特性,成为了全球开发者社区广泛使用的 开发工具集。

GMD 软件在创建工程时,如果开发人员创建 Arm MCU 工程,可以选择软件工具链 "xPack GNU Arm Embedded GCC (arm-none-eabi-gcc)",如图 8-1 所示;如果开发人员创建 RISC-V MCU 工程,可以选择软件工具链 "xPack GNU RISC-V Embedded GCC (riscv-none-elf-gcc)",如图 8-2 所示。

ype filter text	Settings	
 Resource Builders C/C++ Build Build Variables Environment 	Configuration: Debug [Active]	Manage Configurations Ontainer Settioner & Build Steps Build Artifact *-
JSON Compilation Dat Logging Settings Tool Chain Editor > C/C++ General Linux Tools Path > MCU Project Natures Project References Run/Debug Settings > Task Repository Task Tags > Validation WikiText	Name: xPack GNU Arm Embedded Architecture: ARM (AArch32) ~ Prefix: arm-none-eabi- Suffix:	I GCC (arm-none-eabi-gcc)
	□ Prefer local xpacks/.bin path Toolchain path: \$(eclipse_home)\toolchain\((to change it use the global Build tools path: \$(eclipse_home)\toolchain\((to change it use the global © Create flash image □ Create extended listing ♥ Print size	gowin\gcc_arm\bin or <u>workspace</u> preferences pages or the <u>project</u> properties page) gowin\build_tools\bin or <u>workspace</u> preferences pages or the <u>project</u> properties page)

图 8-1 Arm 软件工具链

	Settings		$\langle \neg \bullet \circ \rangle$
Resource Builders C/C++ Build Build Variables	Configuration: Deb	ug [Active]	✓ Manage Configurations.
Environment JSON Compilation Dat	🛞 Tool Settings 🛞	Toolchains 📕 Devices 🔋 Container S	Settings 🎤 Build Steps 🙅 Build Artifact 🎽
Logging	Name:	xPack GNU RISC-V Embedded GCC (ri	iscv-none-elf-gcc) ~
Tool Chain Editor	Architecture:	RISC-V ~	
C/C++ General	Prefix:	riscv-none-elf-	
MCU	Suffix:		
Project Natures	C compiler:	gcc	
Project References	C++ compiler:	g++	
Task Repository	Archiver:	ar	
Task Tags	Hex/Bin converter:	obicopy	
Validation WikiText	Listing generator:	objdump	
The Car	Size command:	size	
	Build command:	make	
	Remove command:	rm	
	Prefer local xpac	ks/.bin.path	
	Toolchain path:	{eclipse home}\toolchain\gowin\gcc	riscv\bin
		(to change it use the <u>global</u> or <u>worksp</u>	ace preferences pages or the project properties page
	Build tools path:	\${eclipse_home}\toolchain\gowin\build	d_tools\bin
		(to change it use the <u>global</u> or <u>worksp</u>	<u>ace</u> preferences pages or the <u>project</u> properties page
	✓ Create flash image	ie .	
	Create extended	listing	
>	Print size		

图 8-2 RISC-V 软件工具链

8.2 设置工程

在创建工程后,编译工程前,需要根据用户应用需求来设置工程的各个 编译、链接选项。

选定当前工程,右键选择 "Properties > C/C++ Build > Settings > Tool Settings",设置工程。例如,Arm MCU 工程设置如图 8-3 所示,RISC-V MCU 工程设置如图 8-4 所示。

Settings > Resource Builders < C/C++ Build Build Variables Environment JSON Compilation Dat Logging Settings Tool Chain Editor > C/C++ General Linux Tools Path > MCU Project Natures Project References Run/Debug Settings > Task Repository Task Tags > Validation WikiText WikiText Settings Soft Optimization Bincludes Preprocessor Bincludes Project References Run/Debug Settings > Validation WikiText WikiText Bincludes Preprocessor Warnings Warnings Warnings Warnings Warnings Warnings Warnings Warnings Miscellaneous Warnings Warnings Miscellaneous Warnings Warnings Warnings Warnings Sonu ARM Cross C Linker Sonu ARM Cross Create Flash Image Wiscellaneous <t< th=""><th></th></t<>	
Resource Builders C/C++ Build Build Variables Environment JSON Compilation Dat Logging Tool Settings Tool Chain Editor C/C++ General Linux Tools Path Coffiguration MCU Settings Project Natures Preprocessor Run/Debug Settings GNU ARM Cross Assembler Validation Settings Validation Soft Compilation WikiText Soft NU ARM Cross C Compiler Settings Instruction ect WikiText Soft NU ARM Cross C Linker Soft NU ARM Cross C Linker Feature crc Soft NU ARM Cross C Linker Soft NU ARM Cross C Linker Soft NU ARM Cross C Linker Soft NU ARM Cross C Linker Soft NU ARM Cross C Linker Feature crc Soft NU ARM Cross C Linker Feature sind Soft NU ARM Cros	✓ Manage Configurations gs ≫ Build Steps ♥ Build Artifact ♥ 2 pu) cortex-m3 ✓ 4 Toolchain default ✓ 7 Thumb (-mthumb) ✓ 6 ork (-mthumb-interwork) Toolchain default ✓ 7 Toolchain default ✓ 7 Toolchain default ✓ 7
Builders Configuration: Debug [Active] C/C++ Build Build Variables Environment JSON Compilation Dat Logging Tool Settings Toolchains Devices Container Settings Build Tool Chain Editor Optimization Arm family (-mcpu) corts C/C++ General Debugging Architecture (-march) Tool Linux Tools Path Warnings Instruction set Thum MCU Project Natures Preprocessor Instruction set Thum Project References Warnings Float ABI Tool WikiText GNU ARM Cross C Compiler Float ABI Tool WikiText GNU ARM Cross C Compiler TrustZone (-mcmse) Marnings WikiText GNU ARM Cross C Linker Feature crc Tool WikiText Wiscellaneous Feature crc Tool Wiscellaneous Sig GNU ARM Cross Create Flash Image General Feature sind Enable Wiscellaneous Sig GNU ARM Cross Create Flash Image Gode model Small	✓ Manage Configurations gs ➢ Build Steps Build Artifact »2 pu) cortex-m3 ~ arch) Toolchain default ~ Thumb (-mthumb) ~ ~ ork (-mthumb-interwork) Toolchain default ~ Toolchain default ~ ~
 C/C++ Build Build Variables Environment JSON Compilation Dat Logging Settings Tool Chain Editor C/C++ General Linux Tools Path MCU Project Natures Project References Run/Debug Settings Task Repository Task Rags Validation WikiText Settings GNU ARM Cross C Compiler Preprocessor Instruction set Tool Unaligned access Tool WikiText GNU ARM Cross C Compiler Settings Miscellaneous Settings Miscellaneous Settings Miscellaneous Settings Miscellaneous Settings Miscellaneous Settings Miscellaneous Settings Settings Settings Settings Settings Settings Settings Settings Miscellaneous Settings Settin	gs → Build Steps → Build Artifact → 2 pu) cortex-m3 → arch) Toolchain default Thumb (-mthumb) ork (-mthumb-interwork) Toolchain default → Toolchain default → Toolchain default
Build Variables Environment JSON Compilation Dat Logging Settings Tool Chain Editor C/C++ General Linux Tools Path MCU Project Natures Project Natures Run/Debug Settings Task Repository Task Repository Task Tags Validation WikiText Ø Optimization Ø NU ARM Cross C Compiler Preprocessor Ø Niscellaneous Y 🕸 GNU ARM Cross C Compiler Project Keferences Run/Debug Settings Task Repository Task Repository Task Tags Validation WikiText Ø Optimization Ø SINU ARM Cross C Compiler Prestores C Ø SINU ARM Cross C Linker Ø GNU ARM Cross Create Flash Image Ø Miscellaneous Y 🕸 GNU ARM Cross Create Flash Image Ø Miscellaneous Y 🕸 GNU ARM Cross Create Flash Image Ø GNU ARM Cross Create Flash Image <	gs Duild Steps Duild Artifact Du cortex-m3 Contex-m3 Con
Environment JSON Compilation Dat Image: Tool Settings image: Tool Chains image: Tool Chains image: Tool Chain Editor Image: Tool C	as Puild Steps Puild Artifact P2 build Cortex-m3 Toolchain default Thumb (-mthumb) ork (-mthumb-interwork) Toolchain default Toolchain default Toolchain default
JSON Compilation Dat Image: Control of the second seco	ou) cortex-m3 v arch) Toolchain default v Thumb (-mthumb) v ork (-mthumb-interwork) Toolchain default v
Settings Image: Processor Arm family (-mcpu) cortex Tool Chain Editor Image: Processor Architecture (-march) Tool C/C++ General Image: Processor Instruction set Thum Linux Tools Path Image: Processor Image: Processor Image: Processor MCU Image: Processor Image: Processor Image: Processor Image: Processor Project Natures Image: Processor Image: Processor Image: Processor Image: Processor Run/Debug Settings Image: Processor Image: Processor Image: Processor Image: Processor Task Repository Image: Processor Image: Processor Image: Processor Image: Processor Validation Image: Processor Image: Processor Image: Processor Image: Processor Image: Processor WikiText Image: Processor Image: Processor Image: Processor Image: Processor Image: Processor WikiText Image: Processor Image: Processor Image: Processor Image: Processor Image: Processor WikiText Image: Processor Image: Processor Image: Processor Image: Processor <td< td=""><td>ou) cortex-m3 arch) Toolchain default Thumb (-mthumb) ~ ork (-mthumb-interwork) Toolchain default Toolchain default</td></td<>	ou) cortex-m3 arch) Toolchain default Thumb (-mthumb) ~ ork (-mthumb-interwork) Toolchain default Toolchain default
Tool Chain Editor Image: Second Sec	arch) Toolchain default Thumb (-mthumb) ork (-mthumb-interwork) Toolchain default Toolchain default
C/C++ General Warnings Instruction set Thurn Linux Tools Path > S GNU ARM Cross Assembler Instruction set Thurn MCU Project Natures Preprocessor Endianness Tool Project Natures > Warnings Float ABI Tool Warnings > Warnings Float ABI Tool Task Repository > S GNU ARM Cross C Compiler FPU Type Tool Task Tags > S GNU ARM Cross C Compiler Instruction set Tool Validation > S GNU ARM Cross C Compiler Instructore (-mcmse) WikiText > Optimization AArch64 family General > Miscellaneous > S GNU ARM Cross C Linker Feature crc Tool > Miscellaneous > S GNU ARM Cross Create Flash Image Feature simd Endianeous > S GNU ARM Cross Create Flash Image Feature simd Endianeous S Code model Smather	Thumb (-mthumb) ~ ork (-mthumb-interwork) Toolchain default ~
Linux Tools Path Instruction set Thurn MCU S GNU ARM Cross Assembler Thumb interwork (-mthur Preprocessor Includes Miscellaneous Foat ABI Tool Y S GNU ARM Cross Create Flash Image Miscellaneous Y S GNU ARM Cross Create Flash Image Code model Small Small Code model Small Small Small Code model Small	Thumb (-mthumb)
MCU Image: Second and the second an	ork (-mthumb-interwork) Toolchain default Toolchain default
Project Natures Preprocessor Endianness Tool Project References Warnings Float AB1 Tool Run/Debug Settings Wiscellaneous FPU Type Tool Task Repository V S GNU ARM Cross C Compiler PU Type Tool Validation Preprocessor Unaligned access Tool WikiText Optimization TrustZone (-mcmse) General Wiscellaneous Feature crc Tool Wiscellaneous Feature crc Tool Wiscellaneous Feature crypto Tool Wiscellaneous Feature fp Tool Wiscellaneous Feature simd Endianeous Wiscellaneous Segeneral Code model Wiscellaneous Feature simd Endianeous Wiscellaneous Feature simd E	Toolchain default
Project References	Toolchain default
Run/Debug Settings Task Repository Task Tags Validation WikiText	Loolchain detault
Task Repository S GNU ARM Cross C Compiler FPU Type Tool Task Tags S GNU ARM Cross C Compiler Unaligned access Tool Validation Includes Includes Includes WikiText Optimization Arch64 family Gene S GNU ARM Cross C Linker S GNU ARM Cross C Linker Feature crc Tool S GNU ARM Cross C Linker S GNU ARM Cross C Linker Feature fp Tool S GNU ARM Cross C Corate Flash Image Feature simd Enable S GNU ARM Cross Create Flash Image Code model Small	
Task Tags Preprocessor Unaligned access Tool Validation Includes TrustZone (-mcmse) WikiText Optimization AArch64 family Gene Warnings AArch64 family Gene Gene Wiscellaneous Feature crc Tool Wiscellaneous Feature fp Tool Wiscellaneous Feature fp Tool Wiscellaneous Feature simd Enable Wiscellaneous Code model Small	Toolchain default
Validation Includes Imagine diccess Formattion WikiText Imagine diccess Imagine diccess Formattion Imagine diccess	Toolchain default
WikiText Optimization ITrustZone (-mcmse) Warnings AArch64 family Gene Wiscellaneous Feature crc Tool Seneral Feature crypto Tool Wiscellaneous Feature fp Tool Miscellaneous Feature fp Tool Miscellaneous Feature simd Enable General Feature simd Enable Miscellaneous Code model Small	
Warnings AArch54 family Gene Miscellaneous Feature crc Tool Solution Feature crypto Tool General Feature crypto Tool Miscellaneous Feature crypto Tool Miscellaneous Feature fp Tool Miscellaneous Feature simd Enable GRU ARM Cross Create Flash Image General Code model Son Son Son Son	icmse)
Image: Miscellaneous Feature crc Tool Image: Miscellaneous Feature crypto Tool Image: Miscellaneous Feature fp Tool Image: Miscellaneous Feature fp Tool Image: Miscellaneous Feature simd Enable Image: Miscellaneous Code model Small	Generic (-mcpu=generic)
✓ Sign GNU ARM Cross C Linker Feature crypto Tool	Toolchain default
Sequence Feature crypto Tool Ibraries Feature fp Tool Ibraries Feature fp Tool Ibraries Feature simd Enable Ibraries Code model Small	
Libraries Feature fp Tool Miscellaneous Solution Feature simd Enable General Code model Small	Toolchain default
Miscellaneous Miscellaneous Sonu ARM Cross Create Flash Image General Code model Small	Toolchain default
	- 11 14 1 h
Code model Sma	Enabled (+simd)
A MA CALLARA Course Daint Cine	Small (-mcmodel=small)
Strict align (-mstrict-align	nstrict-align)
(2) General	-
Other target flags	s

图 8-3 Arm MCU 工程设置
Resource Builders C/C++ Build Build variables Environment JSON Compilation Dat Logging Settings Tool Chain Editor C/C++ General Linux Tools Path MCU Project Natures Project References Run/Debug Settings Task Repository Task Tags WikiText WikiText GNU RISC-V Cross C Compiler Bound Loads Mixeellaneous WikiText
Miscellaneous Miscellaneous Solu RISC-V Cross Create Flash Image Beneral Solu RISC-V Cross Create Listing Solu RISC-V Cross Create Listing Beneral Solu RISC-V Cross Create Listing Beneral Solu RISC-V Cross Create Listing Solu RISC-V Cross Create Listing Solu RISC-V Cross Create Listing Beneral Solu RISC-V Cross Create Listing Beneral Solu RISC-V Cross Create Listing Solu RISC-V Cross Create Listing

图 8-4 RISC-V MCU 工程设置

Arm MCU 和 RISC-V MCU 工程设置如下所述。

8.2.1 Arm MCU 工程设置

以 Gowin_EMPU(GW1NS-4C) 为例进行说明。

Target Processor

设置 Arm 目标处理器如图 8-5 所示。

🖲 Tool Settings 🛞 Toolchains 🔳 Devices 🚺	Container Settings	🎤 Build Steps 🙅 Build Artifact 🔌	
🖄 Target Processor	Arm family (-mcpu)	cortex-m3	~
🖄 Optimization	Architecture (-march) Toolchain default	~
👺 Warnings	Architecture (-march		· ·
🖄 Debugging	Instruction set	Thumb (-mthumb)	\sim
✓ Song GNU ARM Cross Assembler Song GNU Song GNU	Thumb interwork	(-mthumb-interwork)	
2 Preprocessor	Endianness	Toolchain default	~
Includes	Endiamicss		
Warnings	Float ABI	Toolchain default	
Miscellaneous	FPU Type	Toolchain default	
Preprocessor	Unaligned access	Toolchain default	~
Includes			-
🖉 Optimization	IrustZone (-mcms	se)	
🖉 Warnings	AArch64 family	Generic (-mcpu=generic)	
🖄 Miscellaneous	Feature crc	Toolchain default	
🗸 🛞 GNU ARM Cross C Linker	_		
🖄 General	Feature crypto	Toolchain default	
🖉 Libraries	Feature fp	Toolchain default	
Miscellaneous Sources Create Flack Image	Feature simd	Enabled (+simd)	
General			
Scheren	Code model	Small (-mcmodel=small)	
General	Strict align (-mstrict-align)		
	Other target flags		
		L	

图 8-5 Target Processor

选项设置如表 8-1 所示。

表 8-1 Target Processor

选项	设置	说明
Arm family (-mcpu)	cortex-m3	Arm 内核型号,已支持 Cortex- M1/M3/M4
Architecture (- march)	Toolchain default	Arm 架构
Instruction set	Thumb (-mthumb)	Arm 指令集
Thumb interwork (- mthumb)	-	Thumb 互操作,在同一个可执行文件 中混合使用 32-bit 和 16-bit Thumb 指 令集
Endianness	Toolchain default	字节序,例如 Little endian 或 Big endian
Float ABI	-	浮点 ABI 支持
FPU Type	-	硬件浮点支持
Unaligned access	Toolchain default	非对齐访问

选项	设置	说明
TrustZone (- mcmse)	-	编译器会生成支持 TrustZone 安全特性的代码
Other target flags	-	额外的目标处理器编译选项

Optimization

设置编译优化等级,例如-O0、-O1、-O2、-O3、-Os、-Ofast、-Og 或-Oz,用于平衡编译速度、代码性能与体积,如图 8-6 所示。

图 8-6 Optimization

选项设置如表 8-2 所示。

表 8-2 Optimization

选项	设置
Optimization Level	None (-O0)
Message length (-fmessage-length = 0)	\checkmark
'char' is signed (-fsigned-char)	\checkmark

选项	设置
Function sections (-ffunction-sections)	\checkmark
Data sections (-fdata-sections)	\checkmark
No common unitialized (-fno-common)	-
Do not inline functions (-fno-inline- functions)	-
Assume freestanding environment (- ffreestanding)	-
Disable builtin (-fno-builtin)	-
Single precision constants (-fsingle- precision-constant)	-
Position independent code (-fPIC)	-
Link-time optimizer (-flto)	-
Disable loop invariant move (-fno-move- loop-invariants)	-
Other optimization flags	-

Warnings

设置警告信息开关,可帮助开发人员提早发现潜在问题,并在需要时将 警告视为错误,如图 8-7 所示。

图 8-7 Warnings

Debugging

设置调试等级,例如-g、-g1、-g3等生成不同详细级别的调试信息,如 图 8-8 所示。

🛞 Tool Settings 🛞 Toolchains 🔳 Devices 🏮	Container Settings 🎤 E	Build Steps 🚇 Build Artifact 🔭	
Target Processor Optimization	Debug level	Default (-g)	~
Warnings	Debug format	Toolchain default	\sim
Debugging	Generate prof inform	nation (-p)	
✓ 險 GNU ARM Cross Assembler	Generate gprof infor	mation (-pg)	
Preprocessor	Other debugging flags		
🖉 Includes	outer debugging hugs		
🖄 Warnings			
🖄 Miscellaneous			
✓ S GNU ARM Cross C Compiler			
Preprocessor			
🖄 Includes			
🖄 Optimization			
🖄 Warnings			
🖄 Miscellaneous			
🗸 🛞 GNU ARM Cross C Linker			
🖄 General			
🖄 Libraries			
🖄 Miscellaneous			
V 🛞 GNU ARM Cross Create Flash Image			
🖄 General			
V 🛞 GNU ARM Cross Print Size			
🖄 General			

图 8-8 Debugging

选项设置如表 8-3 所示。

表 8-3 Debugging

选项	设置
Debug level	Default (-g)
Debug format	Toolchain default
Generate prof information (-p)	-
Generate gprof information (-pg)	-
Other debugging flags	-

GNU ARM Cross Assembler

设置 Arm MCU 汇编编译选项,包含 Preprocessor、Includes、Warnings、Miscellaneous。

1. Preprocessor

设置宏定义,用于添加项目相关的预编译宏如图 8-9 所示。

🛞 Tool Settings 🛞 Toolchains 🔳 Devices 📋	Container Settings	🎤 Build Steps	🚇 Build Artifact	»2
 Target Processor Optimization Warnings Debugging S GNU ARM Cross Assembler Preprocessor Includes Warnings Miscellaneous S GNU ARM Cross C Compiler Preprocessor Includes Optimization Warnings Miscellaneous S GNU ARM Cross C Linker General 	Use preprocesso Do not search sy Preprocess only Defined symbols	or (stem directorie (-E) (-D) BSS	s (-nostdinc)	الج
 Contain Clibraries Miscellaneous GNU ARM Cross Create Flash Image General GNU ARM Cross Print Size General 	Undefined symbo	Is (-U)		

图 8-9 GNU ARM Cross Assembler > Preprocessor

选项设置如表 8-4 所示。

表 8-4 GNU ARM Cross Assembler > Preprocessor

选项	设置
Use preprocessor	\checkmark
Do not search system directions (- nostdinc)	-
Preprocess only (-E)	-
Defined symbols (-D)	STARTUP_CLEAR_BSS
Undefined symbols (-U)	-

2. Includes

指定头文件搜索路径,可填写绝对路径、相对路径或 Eclipse 变量(例 如\${workspace_loc}),如图 8-10 所示。

🛞 Tool Settings 🛞 Toolchains 🔳 Devices 🔋	Container Settings	🎤 Build Steps	🚇 Build Artifa	act "2
 Target Processor Optimization Warnings Debugging S GNU ARM Cross Assembler Preprocessor Includes Warnings Miscellaneous S GNU ARM Cross C Compiler 	Include paths (-I)			· 산 · 산 ·
 W GNU ANM Cross C Compiler Preprocessor Includes Optimization Warnings Miscellaneous W GNU ARM Cross C Linker General Libraries Miscellaneous S GNU ARM Cross Create Flash Image 	Include system p	aths (-isystem)] 상 샷
@ General ♥ 酸 GNU ARM Cross Print Size 않 General	Include files (-incl	lude)		<u> </u>

图 8-10 GNU ARM Cross Assembler > Includes

3. Warnings

按需填写额外的警告选项如图 8-11 所示。

🛞 Tool Settings 🛞 Toolchains 🔳 Devices 🔋 Container Settings 🎤 Build Steps 🙅 Build Artifact 🐾	
 Tool Settings is Toolchains Devices Container Settings is Build Artifact "2 Target Processor Optimization Warnings Debugging So RNU ARM Cross Assembler Preprocessor Includes Warnings Miscellaneous So GNU ARM Cross Cinker General Sonu ARM Cross Print Size General 	

图 8-11 GNU ARM Cross Assembler > Warnings

4. Miscellaneous

按需填写额外的编译器选项如图 8-12 所示。

🛞 Tool Settings 🛞 Toolchains 🔳 Devices 🔋	Container Settings	🎤 Build Steps	Build Artifact "2	
🖄 Target Processor	Assembler flags		🛃 🔊 🗟 취 🗄	
🖄 Optimization				
🖄 Warnings				
🖄 Debugging				
🗸 🛞 GNU ARM Cross Assembler				
Preprocessor				
🖄 Includes				
🖄 Warnings				
🖄 Miscellaneous				
🗸 🛞 GNU ARM Cross C Compiler				
Preprocessor				
Includes				
Optimization				
🖄 Warnings				
A Miscellaneous				
V 🛞 GNU ARM Cross C Linker				
🖄 General				
🖄 Libraries				
A Miscellaneous				
✓ ℜ GNU ARM Cross Create Flash Image				
A General				
V 🛞 GNU ARM Cross Print Size				
A General				
	Generate assem	bler listing (-Wa	,-adhlns="\$@.lst")	
	Save temporary	files (save-tem	ps Use with caution!)	
	Verbose (-v)			
	Other assembler fla	ags		

图 8-12 GNU ARM Cross Assembler > Miscellaneous

GNU ARM Cross C Compiler

设置 Arm MCU C 编译选项,包含 Preprocessor、Includes、 Optimization、Warnings、Miscellaneous。

1. Preprocessor

设置宏定义,用于添加项目相关预编译宏如图 8-13 所示。



图 8-13 GNU ARM Cross C Compiler > Preprocessor

2. Includes

指定头文件搜索路径,可填写绝对路径、相对路径或 Eclipse 变量(例 如\${workspace_loc}),如图 8-14 所示。

🛞 Tool Settings 🛞 Toolchains 📕 Devices 📋	Container Settings 🎤 Build Steps	Puild Artifact »2
 Target Processor Optimization Warnings Debugging S GNU ARM Cross Assembler Preprocessor Includes Warnings Miscellaneous S GNU ARM Cross C Compiler Preprocessor 	Include paths (-1) *{workspace_loc:/{ProjName}/src */{workspace_loc:/{ProjName}/src */{workspace_loc:/{ProjName}/src */{workspace_loc:/{ProjName}/src */{workspace_loc:/{ProjName}/src */{workspace_loc:/{ProjName}/src */{workspace_loc:/{ProjName}/src */{workspace_loc:/{ProjName}/src */{workspace_loc:/{ProjName}/src */{workspace_loc:/{ProjName}/src */{workspace_loc:/{ProjName}/src */{workspace_loc:/{ProjName}/src */{workspace_loc:/{ProjName}/src */{workspace_loc:/{ProjName}/src */{workspace_loc:/{ProjName}/src */{workspace_loc:/{ProjName}/src */{workspace_loc:/{ProjName}/src */{workspace_loc:/{ProjName}/src */{workspace_loc:/{ProjName}/	(library/libraries/cmsis/cm3/dev /library/libraries/drivers/inc)* /library/middlewares/delay)* /library/middlewares/delay)* /library/middlewares/pyper_ran /library/middlewares/pyper_ran /library/middlewares/psi_flash)* /library/middlewares/spi_flash)*
 Preprocessor Includes Optimization Warnings Wascellaneous S GNU ARM Cross C Linker General Libraries Miscellaneous S GNU ARM Cross Create Flash Image 	Include system paths (-isystem)	<u>•</u> 또 접 아 알
 ➢ General ✓ ℜ GNU ARM Cross Print Size ➢ General 	Include files (-include)	<table-cell> 🔊 🖗 ਨੂੰ। ਨੂ।</table-cell>

图 8-14 GNU ARM Cross C Compiler > Includes

选项设置如表 8-5 所示。

表 8-5 GNU ARM Cross C Compiler > Includes

选项	设置
Include paths (-I)	"\${workspace_loc:/\${ProjName}/src/library/libraries/cmsis/ cm3/core_support/gmd}"
	"\${workspace_loc:/\${ProjName}/src/library/libraries/cmsis/ cm3/device_support}"
	"\${workspace_loc:/\${ProjName}/src/library/libraries/drivers /inc}"
	"\${workspace_loc:/\${ProjName}/src/library/middlewares/d elay}"
	"\${workspace_loc:/\${ProjName}/src/library/middlewares/d mm}"
	"\${workspace_loc:/\${ProjName}/src/library/middlewares/g pio}"
	"\${workspace_loc:/\${ProjName}/src/library/middlewares/hy per_ram}"
	"\${workspace_loc:/\${ProjName}/src/library/middlewares/ps ram}"

选项	设置
	"\${workspace_loc:/\${ProjName}/src/library/middlewares/sp i_flash}"
	"\${workspace_loc:/\${ProjName}/src/library/middlewares/u art}"
	"\${workspace_loc:/\${ProjName}/src/project}"
Include system paths (-isystem)	-
Include files (-include)	-

3. Optimization

设置 C 标准及扩展,用于开启对应语言特性,以及按需额外的优化选项,如图 8-15 所示。

图 8-15 GNU ARM Cross C Compiler > Optimization

🛞 Tool Settings 🛞 Toolchains 📓 Devices 🧯 Container Settings 🎤 Build Steps 🏆 Build Artifact 🗟 Binary Parsers 🦏

🖄 Target Processor	Language standard	GNU ISO C11 (-std=gnu11)	~
🖄 Optimization			_
🖄 Warnings	Other optimization flags		
🖄 Debugging			
✓			
Preprocessor			
🖄 Includes			
🖄 Warnings			
🖄 Miscellaneous			
✓ Son GNU ARM Cross C Compiler Son Compiler			
Preprocessor			
🖄 Includes			
🖄 Optimization			
🖄 Warnings			
🖄 Miscellaneous			
✓ Solver Service Serv			
🖄 General			
🖄 Libraries			
🖄 Miscellaneous			
🗸 🛞 GNU ARM Cross Create Flash Image			
🖄 General			
✓ S GNU ARM Cross Print Size			
🖄 General			
	<		>

选项设置如表 8-6 所示。

表 8-6 GNU ARM Cross C Compiler > Optimization

选项	设置
Language standard	GNU ISO C11 (-std=gnu11)

选项	设置
Other optimization flags	-

4. Warnings

按需填写额外的警告选项如图 8-16 所示。

图 8-16 GNU ARM Cross C Compiler > Warnings

🛞 To	ool	Settings	🛞 Toolchains	Devices	٥	Container	Settings	🎤 Build Steps	🚇 Build Artifact	»2
	<u>8</u> 3	Target P	rocessor			│ □ Warn if	a olobal	function has no	prototype (-Wmi	ssina-ı
	 &@}	Optimiza	tion			Warn if	a functio	n has no aro tvr	pe (-Wstrict-proto	types)
		Warning	5			- Warn if	wrong c	ast (-Whad-fun	rtion-cast)	()))))))
	<u>8</u>	Debuaai	na						cuon-cast)	
~	89	GNU AR	M Cross Asseml	bler		Other war	ning flags	s		
	Ĩ	🖄 Prepro	ocessor							
	í	🖄 Includ	es							
	6	🖄 Warni	ngs							
	6	🖄 Misce	llaneous							
~	89	GNU AR	M Cross C Com	piler						
	6	🖄 Prepro	ocessor							
	ĺ	🧏 Includ	es							
	ĺ	🖄 Optim	ization							
	Ć	🖄 Warni	ngs							
	_ (🤌 Misce	llaneous							
×	89	GNU AR	M Cross C Linke	er						
	6	🥦 Gener	al							
	6	🖄 Misso	llanaous							
) 883		M Cross Create	Elach Image						
	w (Sener 🖏	al	nash image						
~	89	GNU AR	u. M Cross Print Si	ze						
	Ű,	🖄 Gener	al							
		-								
									_	>

5. Miscellaneous

按需填写额外的编译器选项如图 8-17 所示。

图 8-17 GNU ARM Cross C Compiler > Miscellaneous

GNU ARM Cross C Linker

设置 Arm MCU C 链接选项,包含 General、Libraries、 Miscellaneous。

1. General

如图 8-18 所示, Script files (-T)用于指定链接脚本文件。

🛞 Tool Settings 🛞 Toolchains 🔳 Devices 🏮	Container Settings 🎤 Build St	eps 舋 Build Artifact 🗟 Binary Parsers 🎇
🖄 Target Processor	Script files (-T)	🗐 🗐 😼 중 문)
🖄 Optimization	"\$fworkspace_loss/\$(DroiNam)	a) / esc // ibsaries / escie / escie / escie / escie
🖄 Warnings	\${workspace_loc./s{ProjName	ey/src/library/libraries/chisis/chis/device_sup
🖄 Debugging		
✓ S GNU ARM Cross Assembler		
Preprocessor		
🖄 Includes		
🖉 Warnings		
🖄 Miscellaneous		
✓ Son GNU ARM Cross C Compiler		
Preprocessor		
Includes		
Optimization		
🖄 Warnings		
Miscellaneous		
✓ Son GNU ARM Cross C Linker		
🖄 General		
🖄 Libraries		
Miscellaneous		
✓		
🖄 General		
✓ SGNU ARM Cross Print Size		
🖄 General		
		· · · · · · · · · · · · · · · · · · ·
	Do not use standard start fi	les (-nostartfiles)
	Do not use default libraries	(-nodefaultlibs)
	No startup or default libs (-	nostdlib)
	Remove unused sections (->	(linkergc-sections)
	Print removed sections (-Xli	nkerprint-gc-sections)
	Omit all symbol information	(-s)

图 8-18 GNU ARM Cross C Linker > General

选项设置如表 8-7 所示。

表 8-7 GNU ARM Cross C Linker > General

选项	设置
Script files (-T)	"\${workspace_loc:/\${ProjName}/src/ library/libraries/cmsis/cm3/device_s upport/startup/gmd/linker/gw1ns4c_ flash.ld}"
Do not use standard start files (-nostartfiles)	-
Do not use default libraries (-nodefaultlibs)	-
No startup or default libs (-nostdlib)	-
Remove unused sections (-Xlinkergc- sections)	\checkmark
Print removed sections (-Xlinkerprint-gc- sections)	-
Omit all symbol information (-s)	-

2. Libraries

设置链接库编译选项如图 8-19 所示。在 Libraries (-I)文本框中按需填

写要链接的库。Library search path (-L)用于设置静态库(.a)和共享库(.so)的搜索路径。

图 8-19 GNU ARM Cross C Linker > Libraries

🛞 Tool Settings 🛞 Toolchains 🔳 Devices 🏮 Container Settings 🎤 Build Steps 🙅 Build Artifact 类 Target Processor Libraries (-l) 🛃 🔊 🗟 신 문] 🖄 Optimization 🖄 Warnings 🖄 Debugging ✓
 Song GNU ARM Cross Assembler
 Song GNU
 Song G Preprocessor 🖄 Includes 🖄 Warnings 🖄 Miscellaneous 🖄 Preprocessor 🖄 Includes 🖄 Optimization 🖄 Warnings Miscellaneous Group libraries (-Wl,--start-group, ... ,--end-group) 🖄 General Library search path (-L) 🛃 🔊 🗟 신 신 🖄 Libraries Aiscellaneous 🖄 General 🖄 General

3. Miscellaneous

按需设置或填写额外的链接选项如图 8-20 所示。

🛞 Tool Settings 🛞 Toolchains 🔳 Devices 🏮	Container Settings 🎤 Build Steps 🙅 Build Artifact 🎇
 Target Processor Optimization Warnings Debugging S GNU ARM Cross Assembler Preprocessor Includes Warnings Miscellaneous S GNU ARM Cross C Compiler 	Linker flags (-Xlinker [option]) <table-cell> 🗟 ਨੇ 🖓</table-cell>
 Preprocessor Includes Optimization Warnings Miscellaneous S GNU ARM Cross C Linker General Libraries Miscellaneous S GNU ARM Cross Create Flash Image 	Other objects 🕢 🗟 ਨੇ। 🖓
ጅ General ♥ 평 GNU ARM Cross Print Size 遂 General	Generate map [*\${BuildArtifactFileBaseName}.map" Cross reference (-Xlinkercref) Print link map (-Xlinkerprint-map) Use newlib-nano (specs=nano.specs) Use float with nano printf (-u _printf_float) Use float with nano scanf (-u _scanf_float) Do not use syscalls (specs=nosys.specs) Verbose (-v) Other linker flags

图 8-20 GNU ARM Cross C Linker > Miscellaneous

GNU ARM Cross Create Flash Image

设置 Arm MCU 可执行文件的输出格式,例如 Raw binary、Intel HEX 等,如图 8-21 所示。

	0		
🛞 Tool Settings 🛞 Toolchains 📕 Devices 🚺	Container Settings 🎤	Build Steps 🙅 Build Artifact	»2
Processor	Output file format (-O)	Raw binary	~
2 Optimization	Section: -j .text		
Warnings	Section: -j .data		
	Other sections (-i)	🔊 🔊	
Signu ARM Cross Assembler		1	E U XI
Preprocessor			
warnings			
Miscellaneous			
Ono Activi Cross C Compiler			
Warnings			
Solution Clinker			
General			
M Libraries			
Miscellaneous			
✓ ℜ GNU ARM Cross Create Flash Image			
A General			
V 🕅 GNU ARM Cross Print Size			
General			
	Other flags		
L		L	

图 8-21 GNU ARM Cross Create Flash Image

选项设置如表 8-8 所示。

表 8-8 GNU ARM Cross Create Flash Image

选项	设置
Output file format (-O)	Raw binary
Section: -j .text	-
Section: -j .data	-
Other sections (-j)	-
Other flags	-

GNU ARM Cross Print Size

用于输出 Arm MCU 可执行文件中各段(text、data、bss)以及总大小的统计,包含 Berkeley、SysV 格式,不同格式只会改变输出格式,不会影响实际段大小的计算,如图 8-22 所示。

图 8-22 GNU ARM Cross Print Size

选项设置如表 8-9 所示。

表 8-9 GNU ARM Cross Print Size

选项	设置
Size format	Berkeley
Show totals	\checkmark
Other flags	-

8.2.2 RISC-V MCU 工程设置

以 Gowin_PicoRV32 为例进行说明。

Target Processor

设置 RISC-V 目标处理器如图 8-23 所示。

🛞 Tool Settings 🛞 Toolchains 📕 Devices 📋 C	Container Settings	🎘 Build Steps	ዋ Build Artifact	». 2
 Tool Settings Toolchains Devices Contraction Target Processor Optimization Warnings Debugging GNU RISC-V Cross Assembler Preprocessor Includes Warnings Miscellaneous GNU RISC-V Cross C Compiler Preprocessor 	Container Settings	Build Steps RV32I (-marcl on (RVM) in (RVA) None tension (RVC) _zicsr ILP32 (-mabi= None	Build Artifact h=rv32i*)	»2
 Preprocessor Includes Optimization Warnings Miscellaneous S GNU RISC-V Cross C Linker General Libraries Miscellaneous S GNU RISC-V Cross Create Flash Image 	Tuning Code model Small data limit Align Small prologue, Force string ope Other target flags	Size (-mtune= Medium Any 8 Strict (-mstric /epilogue (-msa erations to call	:size) (-mcmodel=meda t-align) ave-restore) library functions (any) ~ (-mmemcpy)
 ➢ General ◇ ➢ GNU RISC-V Cross Create Listing ➢ General ◇ ※ GNU RISC-V Cross Print Size ➢ General 				

图 8-23 Target Processor

选项设置如表 8-10 所示。

表 8-10 Target Processor

选项	设置	说明
Architecture	RV32I (-march = rv32i)	RISC-V 架构
Multiply extension (RVM)	\checkmark	乘除法指令
Atomic extension (RVA)	-	原子指令
Floating point	None	浮点指令
Compressed extension (RVC)	\checkmark	压缩指令
Other extensions	_zicsr	额外的扩展指令
Integer ABI	ILP32 (-mabi = ilp32*)	整数运算的 ABI
Floating point ABI	None	浮点运算的 ABI
Tuning	Size (-mtune = size)	为特定的处理器微架构优 化代码生成

选项	设置	说明
Code model	Medium Any (- mcmodel = medany)	代码模型
Small data limit	8	小数据区的大小限制
Align	Strict (-mstrict-align)	对齐方式
Small prologue/epilogue (- msave-restore)	\checkmark	编译器可以选择使用外部 函数来保存和恢复寄存器
Other target flags	-	额外的目标处理器编译选 项

Optimization

设置编译优化等级,例如-O0、-O1、-O2、-O3、-Os、-Ofast、-Og 或-Oz,用于平衡编译速度、代码性能与体积,如图 8-24 所示。

图 8-24 Optimization

🛞 Tool Settings 🛞 Toolchains 🔳 Devices 📋 Container Settings 🎤 Build Steps 🙅 Build Artifact 🎽

🖄 Target Processor	Optimization Level	None (-00)	\sim
 Optimization Optimization Warnings Debugging S GNU RISC-V Cross Assembler Preprocessor Includes Warnings Miscellaneous S GNU RISC-V Cross C Compiler Preprocessor Includes Optimization Warnings Miscellaneous S GNU RISC-V Cross C Linker General Libraries GNU RISC-V Cross C create Flash Image General S GNU RISC-V Cross Create Flash Image General S GNU RISC-V Cross Print Size General 	Message length (-fmee 'char' is signed (-fsigne Function sections (-ffur Data sections (-fdata-s No common unitialized Do not inline functions Assume freestanding e Disable builtin (-fno-bu Single precision consta Position independent of Link-time optimizer (-fl Disable loop invariant Other optimization flags	India ('00) issage-length=0) ed-char) inction-sections) ections) d (-fno-common) (-fno-inline-functions) invironment (-ffreestanding) iiltin) iiltin) iintin (-fsingle-precision-constant) i:code (-fPIC) to) move (-fno-move-loop-invariants)	

选项设置如表 8-11 所示。

选项	设置
Optimization Level	None (-O0)
Message length (-fmessage-length = 0)	\checkmark
'char' is signed (-fsigned-char)	-
Function sections (-ffunction-sections)	\checkmark
Data sections (-fdata-sections)	\checkmark
No common unitialized (-fno-common)	-
Do not inline functions (-fno-inline-functions)	-
Assume freestanding environment (- ffreestanding)	-
Disable builtin (-fno-builtin)	\checkmark
Single precision constants (-fsingle-precision- constant)	-
Position independent code (-fPIC)	-
Link-time optimizer (-flto)	-
Disable loop invariant move (-fno-move-loop- invariants)	-
Other optimization flags	-

表 8-11 Optimization

Warnings

设置警告信息开关,可帮助用户提早发现潜在问题,并在需要时将警告 视为错误,如图 8-25 所示。

图 8-25 Warnings

Debugging

设置调试等级,例如-g、-g1、-g3等生成不同详细级别的调试信息,如 图 8-26 所示。

🛞 Tool Settings 🛞 Toolchains 🔳 Devices 🔋 C	Container Settings 🎤 Bu	ild Steps 🚇 Build Artifact 🎇
Target Processor	Debug level	Default (-g) \lor
2 Optimization	Debug format	Toolchain default ~
Warnings Warnings	Generate prof inform	nation (-n)
Solution States Accomplete	Generate profilition	mation (p)
Breprocessor	Generate gpror mor	mation (-pg)
	Other debugging flags	
Warnings		
Miscellaneous		
✓		
Includes		
Optimization		
🖉 Warnings		
🖄 Miscellaneous		
✓		
🖄 General		
🖄 Libraries		
🖄 Miscellaneous		
♥ Solve GNU RISC-V Cross Create Flash Image		
🖄 General		
V 🛞 GNU RISC-V Cross Create Listing		
🖉 General		
SOLUTION NOT Size		
🖉 General		
]	

图 8-26 Debugging

选项设置如表 8-12 所示。

表 8-12 Debugging

选项	设置
Debug level	Default (-g)
Debug format	Toolchain default
Generate prof information (-p)	-
Generate gprof information (-pg)	-
Other debugging flags	-

GNU RISC-V Cross Assembler

设置 RISC-V MCU 汇编编译选项,包含 Preprocessor、Includes、Warnings、Miscellaneous。

1. Preprocessor

设置宏定义,用于添加项目相关的预编译宏如图 8-27 所示。





2. Includes

指定头文件搜索路径,可填写绝对路径、相对路径或 Eclipse 变量(例 如\${workspace_loc}),如图 8-28 所示。



图 8-28 GNU RISC-V Cross Assembler > Includes

3. Warnings

按需填写额外的警告选项如图 8-29 所示。

图 8-29 GNU RISC-V Cross Assembler > Warnings

4. Miscellaneous

按需填写额外的编译器选项如图 8-30 所示。

🛞 Tool Settings 🛞 Toolchains 🔳 Devices 📋 Co	ntainer Settings 🎤 Build Steps 🎇
🖄 Target Processor	Assembler flags 🗿 🗐 🗐
Optimization	
🖉 Warnings	
Debugging	
✓ Sign GNU RISC-V Cross Assembler	
Preprocessor	
Includes	
🖉 Warnings	
🖄 Miscellaneous	
✓ Solve Service Compiler Solve Service Compiler Solve Service	
Preprocessor	
🖄 Includes	
🖄 Optimization	
🖄 Warnings	
🖄 Miscellaneous	
✓ S GNU RISC-V Cross C Linker	
🖄 General	
🖄 Libraries	
🖄 Miscellaneous	
✓ S GNU RISC-V Cross Create Flash Image	
🖄 General	
✓	
🖄 General	
✓ Son GNU RISC-V Cross Print Size	
🖄 General	
	Generate assembler listing (-Wa,-adhlns="\$@
	Save temporary files (save-temps Use with c
	Verbose (-v)
	Other assembler flags
	< >

图 8-30 GNU RISC-V Cross Assembler > Miscellaneous

选项设置如表 8-13 所示。

表 8-13 GNU RISC-V Cross Assembler > Miscellaneous

选项	设置
Assembler flags	-
Generate assembler listing (-Wa,-adhIns="\$@.lst")	\checkmark
Save temporary files (save-temps Use with caution!)	-
Verbose (-v)	-
Other assembler flags	-

GNU RISC-V Cross C Compiler

设置 RISC-V MCU C 编译选项,包含 Preprocessor、Includes、 Optimization、Warnings、Miscellaneous。

1. Preprocessor

设置宏定义,用于添加项目相关预编译宏,如图 8-31 所示。

图 8-31 GNU RISC-V Cross C Compiler > Preprocessor

🛞 Tool Settings 🛞 Toolchains 🏢 Devices 🧯 Container Settings 🎤 Build Steps 🚇 Build Artifact 🤌

🖉 Target Processor	Do not search system directories ((-nostdinc)
🖉 Optimization	Preprocess only (-E)	
🖉 Warnings	Defined symbols (-D)	🗿 🔊 🕲 🏷 Ar
🖄 Debugging		🖬 🖦 🗠 U - K I
🗸 🛞 GNU RISC-V Cross Assembler		
🖄 Preprocessor		
🖄 Includes		
🖄 Warnings		
🖄 Miscellaneous		
🗸 🛞 GNU RISC-V Cross C Compiler		
🖄 Preprocessor		
🖄 Includes		
🖄 Optimization		
🖄 Warnings		
🖄 Miscellaneous		
🗸 🛞 GNU RISC-V Cross C Linker		
🖄 General	Undefined symbols (11)	
🖄 Libraries	Undermed symbols (-0)	🛃 📰 쩐 산 전[
🖄 Miscellaneous		
🗸 🛞 GNU RISC-V Cross Create Flash Image		
🖄 General		
✓		
🖄 General		
V 🛞 GNU RISC-V Cross Print Size		
🖄 General		
][

2. Includes

指定头文件搜索路径,可填写绝对路径、相对路径或 Eclipse 变量(例 如\${workspace_loc}),如图 8-32 所示。



图 8-32 GNU RISC-V Cross C Compiler > Includes

选项设置如表 8-14 所示。

表 8-14 GNU RISC-V Cross C Compiler > Includes

选项	设置
Include paths (-I)	"\${workspace_loc:/\${ProjName}/src}"
	"\${workspace_loc:/\${ProjName}/src/bsp}"
	"\${workspace_loc:/\${ProjName}/src/lib}"
Include system paths (- isystem)	-
Include files (-include)	-

3. Optimization

设置 C 标准及扩展,用于开启对应语言特性,以及按需额外的优化选项,如图 8-33 所示。

🛞 Tool Settings 🛞 Toolchains 🔳 Devices 📋 C	ontainer Settings 🎤 Build	d Steps 🙅 Build Artifact 🔭
Processor Processor	Language standard	GNU ISO C11 (-std=gnu11)
Optimization		_
🖄 Warnings	Other optimization flags	
Debugging		
✓		
Preprocessor		
Includes		
🖉 Warnings		
Miscellaneous		
V 🕅 GNU RISC-V Cross C Compiler		
A Includes		
Optimization		
2 Warnings		
Miscellaneous		
V 🕅 GNU RISC-V Cross C Linker		
🖉 General		
Libraries		
Miscellaneous		
V 🕅 GNU RISC-V Cross Create Flash Image		
🖉 General		
V 🕅 GNU RISC-V Cross Create Listing		
🖉 General		
V 🕅 GNU RISC-V Cross Print Size		
🖄 General		
	<	>

图 8-33 GNU RISC-V Cross C Compiler > Optimization

选项设置如表 8-15 所示。

表 8-15 GNU RISC-V Cross C Compiler > Optimization

选项	设置
Language standard	GNU ISO C11 (-std=gnu11)
Other optimization flags	-

4. Warnings

按需填写额外的警告选项如图 8-34 所示。

	Container Sattings A Build Stars A Build Artifact »
Tool settings Toolchains Devices	
🖄 Target Processor	Warn if a global function has no prototype (-Wmissing-
🖄 Optimization	Warn if a function has no arg type (-Wstrict-prototypes)
🖄 Warnings	Warn if wrong cast (-Wbad-function-cast)
🖄 Debugging	Other warning flags
🗸 🛞 GNU RISC-V Cross Assembler	
🖄 Preprocessor	
🖄 Includes	
🖉 Warnings	
🖄 Miscellaneous	
✓ Solve GNU RISC-V Cross C Compiler Solve GNU S	
Preprocessor	
Includes	
Optimization	
Warnings	
Miscellaneous	
GNU RISC-V Cross C Linker	
Ceneral	
Miscellaneous Monocola Crosse Crosse Elash Image	
General General	
GNUL RISC-V Cross Create Listing	
General	
Solieran	
General	
	< >

图 8-34 GNU RISC-V Cross C Compiler > Warnings

5. Miscellaneous

按需填写额外的编译器选项如图 8-35 所示。

 Tool Settings Dolchains Devices Container Settings Puild Steps Puild Artifact *2 Target Processor Optimization Warnings Debugging GRU RISC-V Cross Assembler Preprocessor Includes Warnings Miscellaneous Gontrainer Settings Miscellaneous GRU RISC-V Cross C Linker General Libraries Miscellaneous GRU RISC-V Cross Create Listing General General GRU RISC-V Cross Print Size General GRU RISC-V Cross Print Size General Gener
 ➢ Target Processor ➢ Optimization ➢ Warnings ➢ Debugging ➢ BOUR RISC-V Cross Assembler ➢ Preprocessor ➢ Includes ➢ Warnings ➢ Miscellaneous ✓ ➢ GNU RISC-V Cross C Compiler ➢ Preprocessor ※ Includes ※ Optimization ※ Warnings ※ Miscellaneous ✓ ➢ GNU RISC-V Cross C Linker ※ General ※ GNU RISC-V Cross Create Flash Image ※ GNU RISC-V Cross Print Size ※ GNU RISC-V Cross Print Size ※ General ※ GNU RISC-V Cross Print Size ※ General

图 8-35 GNU RISC-V Cross C Compiler > Miscellaneous

选项设置如表 8-16 所示。

表 8-16 GNU RISC-V Cross C Compiler > Miscellaneous

选项	设置
Generate assembler listing (-Wa,- adhIns="\$@.lst")	\checkmark
Save temporary files (save-temps Use with caution)	-
Verbose (-v)	-
Other compiler flags	-

GNU RISC-V Cross C Linker

设置 RISC-V MCU C 链接选项,包含 General、Libraries、 Miscellaneous。

1. General

如图 8-36 所示, Script files (-T)用于指定链接脚本文件。

🛞 Tool Settings 🛞 Toolchains 🔳 Devices 🏮 C	Container Settings 🎤 Build Steps 🙅 Build Artifact 🎇
🖉 Target Processor	Script files (-T) 🕢 🛃 🗟 취 위
🖄 Optimization	"\$/workspace_log/\$/ProiName)/crs/sections_lds)"
🖄 Warnings	workspace_oc./strojtvame//src/sections.ids/
🖄 Debugging	
✓	
Preprocessor	
🖄 Includes	
🖄 Warnings	
🖄 Miscellaneous	
✓	
Preprocessor	
🖄 Includes	
🖄 Optimization	
🖄 Warnings	
🖄 Miscellaneous	
✓	
🖄 General	
🖄 Libraries	
🖄 Miscellaneous	
✓	
🖄 General	
✓	
🖄 General	
✓	
🖄 General	Do not use standard start files (-nostartfiles)
	Do not use default libraries (-nodefaultlibs)
	No startup or default libs (-nostdlib)
	Remove unused sections (-Xlinkergc-sections)
	Print removed sections (-Xlinkerprint-gc-sections)
	Omit all symbol information (-s)

图 8-36 GNU RISC-V Cross C Linker > General

选项设置如表 8-17 所示。

表 8-17 GNU RISC-V Cross C Linker > General

选项	设置
Script files (-T)	"\${workspace_loc:/\${ProjName}/src/ sections.lds}"
Do not use standard start files (-nostartfiles)	\checkmark
Do not use default libraries (-nodefaultlibs)	-
No startup or default libs (-nostdlib)	-
Remove unused sections (-Xlinkergc- sections)	\checkmark
Print removed sections (-Xlinkerprint-gc- sections)	-
Omit all symbol information (-s)	-

2. Libraries

设置链接库编译选项,如图 8-37 所示。Libraries (-I),按需填写要链接的库。Library search path (-L),设置静态库(.a)和共享库(.so)的搜索路径。

图 8-37 GNU RISC-V Cross C Linker > Libraries

Tool Settings Toolchains Devices	ontainer settings 🎤 build steps 👱	Build Artifact 2
Target Processor Optimization	Libraries (-l)	행 방 정 산 산
Warnings		
🖉 Debugging		
✓		
Preprocessor		
🖄 Includes		
🖄 Warnings		
🖄 Miscellaneous		
✓		
Preprocessor		
🖄 Includes		
🖄 Optimization		
🖄 Warnings		
Miscellaneous	Group libraries (W/L, start group	and aroun)
V 🛞 GNU RISC-V Cross C Linker	Group libraries (-wi,start-group	, ,end-group)
General	Library search path (-L)	• 6] # 1] 월 상[유]
(23) Libraries		
Miscellaneous		
Miscellaneous SGNU RISC-V Cross Create Flash Image		
Miscellaneous Miscellaneous Signu RISC-V Cross Create Flash Image Beneral ONU RISC V Create Create Listing		
Boundary Miscellaneous Miscellaneous Sonu RISC-V Cross Create Flash Image Boundary General Sonu RISC-V Cross Create Listing		
 Boundary Construction Bou		
 Wiscellaneous Source Source Source Flash Image Gource Source Source		
 Wiscellaneous Source Source Source Flash Image Gource Source Source		
 Wiscellaneous Source Source Flash Image Goural 		
 Wiscellaneous Source Source Flash Image GNU RISC-V Cross Create Flash Image GNU RISC-V Cross Create Listing General GNU RISC-V Cross Print Size General 		
 Wiscellaneous S GNU RISC-V Cross Create Flash Image General GNU RISC-V Cross Create Listing General GNU RISC-V Cross Print Size General 		

3. Miscellaneous

按需设置或填写额外的链接选项如图 8-38 所示。
🛞 Tool Settings 🚷 Toolchains 🔳 Devi	ces 📋	Container Settings	🎤 Build Steps	🚇 Build Artifact	»2	
To anoth Data and an			the states		17 L O .	
Ortiniation		Linker flags (-X	linker [option])	6 10 20	한말	
Optimization						
Debugging						
GNU RISC-V Cross Assembler						
Preprocessor						
includes						
Warnings						
Miscellaneous						
Signu RISC-V Cross C Compiler						
Preprocessor		Other objects		🛃 🔊 🗟	신 문티	
2 Includes						
Optimization						
Warnings						
Miscellaneous						
✓ GNU RISC-V Cross C Linker						
Beneral Beneral						
Libraries						
GNU RISC-V Cross Create Hash	mage					
Beneral		Generate map	"\${BuildArtifa	ctFileBaseName}.r	nap"	
GINU RISC-V Cross Create Listing		Cross reference	e (-Xlinkercre	f)		
General Size		Print link map	(-Xlinkerprint	-map)		
General		Use newlib-na	no (specs=nar	no.specs)		
General		Use float with	nano printf (-u	printf float)		
		Use float with nano scanf (-u_scanf float)				
		Verbose (-v)				
		Other linker flags	-mcmodel=m	edany		
		unker nage		couny		

图 8-38 GNU RISC-V Cross C Linker > Miscellaneous

选项设置如表 8-18 所示。

表 8-18 GNU RISC-V Cross C Linker > Miscellaneous

选项	设置
Linker flags (-Xlinker [option])	-
Other objects	-
Cross reference (-Xlinkercref)	-
Print link map (-Xlinkerprint-map)	-
Use newlib-nano (specs=nano.specs)	按需选择
Use float with nano printf (-u _printf_float)	按需选择
Use float with nano scanf (-u _scanf_float)	按需选择
Do not use syscalls (specs = nosys.specs)	-
Verbose (-v)	-
Other linker flags	-mcmodel=medany

GNU RISC-V Cross Create Flash Image

设置 RISC-V MCU 可执行文件的输出格式,例如 Raw binary、Intel HEX 等,如图 8-39 所示。

图 8-39 GNU RISC-V Cross Create Flash Image

🛞 Tool Settings 🛞 Toolchains 📕 Devices 📋 C	Container Settings	🎤 Build Steps	🚇 Build Artifact	»2
 Tool Settings Toolchains Devices C Target Processor Optimization Warnings Debugging S GNU RISC-V Cross Assembler Preprocessor Includes Warnings Miscellaneous S GNU RISC-V Cross C Compiler Preprocessor Includes Optimization Warnings Miscellaneous S GNU RISC-V Cross C Linker General Libraries Miscellaneous S GNU RISC-V Cross Create Flash Image General S GNU RISC-V Cross Create Listing General S GNU RISC-V Cross Print Size General S GNU RISC-V Cross Print Size General 	Container Settings Output file forma Section: -j .text Section: -j .dat Other sections .irqsec .ldsec .btsec	Build Steps It (-O) Raw bin t t a (-j)	Puild Artifact	
	Other flags			

选项设置如表 8-19 所示。

表 8-19 GNU RISC-V Cross Create Flash Image

选项	设置
Output file format (-O)	Raw binary
Section: -j .text	\checkmark
Section: -j .data	\checkmark
Other sections (-j)	.irqsec
	.ldsec
	.btsec
Other flags	-

GNU RISC-V Cross Create Listing

用于生成包含汇编代码、符号信息和调试信息的列表文件,通常为.lst 文件,这些列表文件对于调试、性能分析和代码审查非常有用,如图 8-40 所示。

图 8-40 GNU RISC-V Cross Create Listing

🛞 Tool Settings 🛞 Toolchains 📕 Devices 🔋 Contair	er Settings 🎤 Build Steps 🚇 Build Artifact 🎇
 Tool Settings Toolchains Devices Contain Target Processor Optimization Warnings Debugging S GNU RISC-V Cross Assembler Preprocessor Miscellaneous S GNU RISC-V Cross C Compiler Preprocessor Includes Optimization Warnings Miscellaneous S GNU RISC-V Cross C Linker General Libraries Miscellaneous S GNU RISC-V Cross Create Flash Image General S GNU RISC-V Cross Print Size General S GNU RISC-V Cross Print Size General 	splay source (source -S) splay all headers (all-headers -x) mangle names (demangle -C) splay debugging info (debugging -g) sassemble (disassemble -d) splay file headers (file-headers -f) splay relocation info (reloc -r) splay symbols (syms -t) de lines (wide -w) flags

选项设置如表 8-20 所示。

表 8-20 GNU RISC-V Cross Create Listing

选项	设置
Display source (source -S)	\checkmark
Display all headers (all-headers -x)	\checkmark
Demangle names (-demangle -C)	\checkmark
Display debugging info (debugging -g)	-
Disassemble (disassemble -d)	-
Display file headers (file-headers -f)	-
Display line numbers (line-numbers -l)	\checkmark
Display relocation info (reloc -r)	-

选项	设置
Display symbols (syms -t)	-
Wide lines (wide -w)	\checkmark
Other flags	-

GNU RISC-V Cross Print Size

用于输出 RISC-V MCU 可执行文件中各段(text、data、bss)以及总 大小的统计,包含 Berkeley、SysV 格式,不同格式只会改变输出格式,不 会影响实际段大小的计算,如图 8-41 所示。

图 8-41 GNU RISC-V Cross Print Size

🛞 Tool Settings 🛞 Toolchains 🔳 Devices 📋 🤇	Container Setti	ings 🎤 I	Build Steps	🙅 Build Artifact	»2
 Tool Settings Toolchains Devices C Coptimization Coptimization Coptimization Coptimization Coptimization Coptimization Coptimization Coptimization Preprocessor Includes Warnings Miscellaneous Soft RISC-V Cross C Compiler Preprocessor Includes Optimization Warnings Miscellaneous Soft RISC-V Cross C Compiler Miscellaneous Soft RISC-V Cross C Linker General Miscellaneous Soft RISC-V Cross C Create Flash Image General Soft RISC-V Cross C Create Listing General Sonu RISC-V Cross Print Size 	Size format	ings 🎤 f	Build Steps	Build Artifact	»2 ~
 @ General @ GNU RISC-V Cross Create Listing @ General @ GNU RISC-V Cross Print Size @ General 					

选项设置如表 8-21 所示。

表 8-21 GNU RISC-V Cross Print Size

选项	设置
Size format	Berkeley
Нех	-

选项	设置
Show totals	\checkmark
Other flags	-

8.3 编译工程

8.3.1 编译方法

Arm MCU 和 RISC-V MCU 编译方法有以下几种,如图 8-42 所示:

- 选定当前工程,右键选择 "Build Project"
- 选择工具栏 "Build" (**●**)
- 选择工具栏 "Build All"(量)
- 选择菜单栏 "Project > Build All"
- 选择菜单栏 "Project > Build Project"

图 8-42 Build

🐝 workspace_test - picorv32	_demo/src/main.c - GMD				_		
<u>File Edit Source Refactor</u>	<u>N</u> avigate Se <u>a</u> rch <u>P</u> roje	t <u>R</u> un <u>G</u> owin <u>W</u> indow	<u>H</u> elp				
📑 🕶 🖬 🕼 😸 🕶 🍊 🕶	🗟 🖓 😒 i 📮 i 🎭 🛛 (Open Project	• 🔅 • O	- 🤮 - 💁 -	🙋 🗁 🛷 🔻		
📝 🛊 🖻 🗉 🖬 🖢 🛉	. ▼ \$> c* \$> ₹ (Close Project			Q	i 🖻 🚾 🔅	۶
🔓 Project Ex 🗙 🗖 🗖	🖻 main.c 🗡 📑 B	Build All Ctrl	I+B		₽0 × °2	- 8	3
🖻 🕏 🍸 👕 🕴	58	Build Configurations	>	^	💱 🖻 🖓 😿	💉 🔍 💥	
📋 cm3_demo	60 #define AHB	Build Project			8		
> 📂 picorv32_demo	61 #define PIC	Build Working Set	Build Project	AHBREGDE	🖬 config	.h	^
	63⊖ //Open WB e: (Clean	Build Hoject		picorv	32.h	
	64 //Register I	Build Automatically			🔛 firmwa	are.h	
	66 {	Build Targets	>		urq.h	warth	
	67IO un 68IO un	C/C++ Index	>		u wbuar	t.h	
	69IO un: 70 } WBREGDEMO	Properties			🖬 wbgpi	o.h	
	71				wbspit	W5A60 GWC	
	72 //Address Definit	ions open up a		~	# DEV_G	WJA00_GW2	~
	<			>	<	>	
	🖹 Problems 🖉 Tasks 토	Console × 🔲 Properties	🗙 🕂 🗘 🔁		5 🖻 🚽	📑 🗖 🗖	3
	CDT Build Console [picorv32	_demo]					_
	Invoking: GNU RISC-V Cr	oss Print Size				'	^
	riscv-none-elf-sizef	ormat=berkeley "picorv32	_demo.elf"				
	7552 100 112	7764 1e54 picorv3	2_demo.elf				
	Finished building: pico	rv32_demo.siz					
	13:56:15 Build Finished	. 0 errors, 0 warnings.	(took 8s.718ms)				
< >>							¥
👺 picorv32_demo			1				

8.3.2 编译工程

Arm MCU 工程编译示例如图 8-43 所示。

🗱 workspace_test - cm3_demo/src/project/printf/printf_demo.c - GMD × _ <u>F</u>ile <u>E</u>dit <u>S</u>ource Refac<u>t</u>or <u>N</u>avigate Se<u>a</u>rch <u>P</u>roject <u>R</u>un <u>G</u>owin <u>W</u>indow <u>H</u>elp 🖻 + 📓 🐘 🗞 + 🐘 🗇 🖓 + 📄 👘 🔌 🕹 👭 🎯 + 🞯 + 🕼 + 🕑 + 🛠 + 🕖 + 🥵 + 🥭 📁 🖉 + 📝 🖉 🗐 🔳 월 ▼ 祠 ▼ ∜ ↔ ♦ ▼ ↔ ▼ 📑 Q 🗄 😰 🛱 🏘 🗝 🗖 🗄 О 🗙 🗐 Т 💿 В 🖓 🗖 E 🕏 🍸 🖙 🕴 을 🕒 🖧 😿 💉 🛛 🗮 🖇 13 14 /* Definitions -----🖬 demo.h 15 > 🖑 Binaries 💵 uart.h > 🗊 Includes 💵 stdio.h > 👝 Debug printf_demo(void) : int ✓ → src > 👝 library 21 22 23 /* Definitions: */ 24 25 //Application entry > 👝 project icorv32_demo //Application entry function 26⊖ int printf_demo(void) 27 28 29 30 { SystemInit(); uart_init(UART0, 38400, //Initializes system clock //Initializes UART0 //<u>Baudrate</u> 31 1, 1, //Tx 32 //<u>Rx</u> < > 🗶 | 🕂 😚 🔄 📰 👪 🔚 🖳 🛃 🖃 🕶 🗂 🖛 🗖 🖹 Problems 🧟 Tasks 📮 Console 🗙 🔲 Properties CDT Build Console [cm3_demo] Invoking. Uno Ann cross create riasin image arm-none-eabi-objcopy -0 binary "cm3_demo.elf" "cm3_demo.bin" Finished building: cm3_demo.bin Invoking: GNU ARM Cross Print Size arm-none-eabi-size --format=berkeley --totals "cm3_demo.elf" text data bss dec hex filename 29988 1764 392 32144 7d90 cm3_demo.elf 29988 1764 392 32144 7d90 (TOTALS) Finished building: cm3_demo.siz 14:21:46 Build Finished. 0 errors, 0 warnings. (took 6s.935ms) 📂 cm3_demo

图 8-43 Arm MCU 工程编译示例

RISC-V MCU 工程编译示例如图 8-44 所示。

🐝 workspace_test - picorv32_d	emo/src/main.c - GMD	- 🗆 ×	:
<u>File Edit Source Refactor</u>	<u>N</u> avigate Se <u>a</u> rch <u>P</u> roject <u>R</u> un <u>G</u> owin <u>W</u> indow <u>H</u> elp		
📬 🔻 🔚 🐚 🦥 🕶 🚳 🖛 🖥) 🖉 🗠 🖳 🗶 👫 🎯 🔻 🗳 👻 🐨 🐨 🐨 🐨 🖉 🕶 🐎 🔹 🔘 🖛 🚱 🖛 🚱 🖛 🚱 🖉 🖉 🔗	R 🗐 🖬	
🖢 🕶 😽 🕶 🖘 🗇 📼 🔿		९ 🗄 🖬 🕯	ģ.
🍋 Project Explo 🗙 🗖 🗖	🖻 main.c 🗙 🗖 🗖 📴 O	Х 🗐 Т 💿 В 🗂 Е	3
□ 🕏 🏹 👕 🕴	58 ^ 69	🖻 🎝 😿 🗙 🖉 🗮	000
🗀 cm3_demo	60 #define AHBREGDEMO BASE OPEN AHB BASE	l config.h	^
✓ ﷺ picorv32_demo	61 #define PICO_AHBREGDEMO ((AHBREGDEMO_RegDef *) AHBREGDEMO_BASE)	picorv32.h	
> 👯 Binaries		I firmware.h	
> 🗊 Includes	64 //Register Definitions	/ irq.h	
> 👝 Debug	650 typedef struct	simpleuart.h	
Y 👝 src	66 { 67 IO unsigned int PEG9: //0v00	l wbuart.h	
> 👝 bsp	68 IO unsigned int REG1; //0x04	wbgpio.h	
> 👝 lib	69IO unsigned int REG2; //0x08	wbspiflash.h	
> 🖻 config.h	70 } WBREGDEMO_RegDet;	DEV_GW5A60_GW2A	
> 🔝 custom_ops.S	72 //Address Definitions	MEM_MAP_ADDR1	
> 💽 irq.c	73 #define WBREGDEMO0_BASE OPEN_WB_BASE	MEM_MAP_ADDR2	
> h irq.h	74 #define PICO_WBREGDEMO0 ((WBREGDEMO_RegDet *) WBREGDEMO0_BASE)	MEM_MAP_ADDR1	
> 💽 loader.c	76	MEM_MAP_ADDR2	
> 💽 main.c	77 uint8_t irq00_flag = 0; //Timer interrupt	MEM_MAP_ADDR1	
> h picorv32.h	< > #	MEM_MAP_ADDR2	۷
> S start.S	🖹 Problems 🧟 Tasks 📮 Console 🗙 🔲 Properties 🛛 🗶 🕂 🚱 🔚 👪 📻 🖳 🧲	🛛 🛃 🖳 🕶 📩 🖛 🗖	3
sections_debug.ids	CDT Build Console [picorv32_demo]		
sections_xip.ids	riscv-none-elf-objdumpsourceall-headersdemangleline-numberswide "picorv32 der	mo.elf" >	^
= sections.ids	"picorv32_demo.lst"		
	Finished building: picorv32_demo.lst		
	Invoking: GNU RISC-V Cross Print Size		
	riscv-none-elf-sizeformat=berkeley "picorv32_demo.elf"		
	7552 100 112 7764 1e54 picorv32 demo.elf		
	Finished building: picorv32_demo.siz		
	09:39:38 Build Finished. 0 errors, 0 warnings. (took 16s.218ms)		
	Weitable Const Incert 1, 1, 0		~
	whitable Smart insert 1 ; 1 ; 0		

图 8-44 RISC-V MCU 工程编译示例



9.1 下载工具

GMD 软件集成了高云下载软件 Programmer,用来下载 Arm MCU 和 RISC-V MCU 的可执行文件,如图 9-1 所示。

👾 Gowin Programmer Version \times File Edit Tools About and and the setting and the se Enable Series Device Operation MCU/FW/(HEX) File 1 🗹 GW1NSR GW1NSR-4C Firmware Erase,Program D:/GMD_2025_workspace/workspace_test/... < Output ð × Info Cable found: Gowin USB Cable(WINUSB)/0/403/null (USB location:403) (SN: null) Info Cost 0.02 second(s) Target Cable: Gowin USB Cable(WINUSB)/0/403/null@10MHz Info Target Device: GW1NSR-4C(0x0100981B) Info Operation "Firmware Erase,Program" for device#1... Info Info Status Code is: 0x00019020 Finished. Info Info Cost 4.53 second(s) Ready

图 9-1 下载工具

9.2 下载方法

Arm MCU 和 RISC-V MCU 下载方法有以下几种,如图 9-2 所示:

- 选择工具栏 "Programmer" (₩)
- 选择菜单栏 "Gowin > Programmer"

图 9-2 Programmer

🗱 workspace_test - picorv32_demo/src/main.c - GMD	- 0	×
<u>F</u> ile <u>E</u> dit <u>S</u> ource Refactor <u>N</u> avigate Se <u>a</u> rch <u>P</u> roject <u>Run</u> <u>G</u> owin <u>W</u> indow <u>H</u> elp		
🗂 ▼ 🔚 🐚 🛞 ▼ 🗞 ▼ 📾 🖉 🏷 🖳 🏪 🍬 🕹 👯 👯 Programmer 🔰 ▼ 🔘 ▼ 🥵 ▼ 🚱 ♥ 🗁	A 🔹	
🏼 🎝 😜 📴 🌐 🖷 🖓 🔻 🎋 マジロ ござ 🗘 🖛 ロシーマ 📑 📩 🧱 GoBridgeDriver	۹ 🗄	1 🔂 🎋
Project Ex X 🗖 🗖 🕼 main.c X	»2	- 0
Image: Second	2 X X config.h bicorv32.h irmware.h rq.h impleuart.h vbuart.h vbgpio.h vbspiflash.ł DEV_GW5A(• ૠ
		>
😰 Problems 🖉 Tasks 🖳 Console X 🔲 Properties 🕷 🦊 😯 🚳 🔝 🖓 层 🖉 💽	⊒ ▼ 🖸 י	
CDT Build Console [picorv32_demo] Throwing. Gum Alsc-v Cross Create Listing riscv-none-elf-objdumpsourceall-headersdemangleline-numberswide "picorv "picorv32_demo.lst" Finished building: picorv32_demo.lst Invoking: GNU RISC-V Cross Print Size riscv-none-elf-sizeformat=berkeley "picorv32_demo.elf" text data bss dec hex filename 7552 100 112 7764 1e54 picorv32_demo.elf Finished building: picorv32_demo.siz	/32_demo.e	elf" > ^
🖻 nicon/32 demo		~
picorosz_demo		

9.3 下载工程

Arm MCU 和 RISC-V MCU 工程的 Programmer 下载设置与所用 FPGA 器件直接相关,以下给出几个典型的下载设置。

1. GW1NS/R-4C

例如 Gowin_EMPU(GW1NS-4C),如图 9-3 和表 9-1 所示。

图 9-3 下载设置	
🙀 Device configuration	
- Device Operation	

Anne Wala	ucut ut. J.
Access Mode:	mLU Mode ~
)peration:	Firmware Erase, Program 🗸 🗸
Rivera France P	
rirmware Erase, r	ogram
Yogropping Optic	
Programming Optic	15
Programming Optic	us 300 test/gowin empu/cm3 demo/impl/pnr/cm3 demo.fs
Programming Optic	ns 300_test/gowin_empu/om3_demo/impl/pnr/om3_demo.fs]
Programming Optic Pile name: sktop Retained User	ns 300_test/gowin_empu/cm3_demo/impl/pnr/cm3_demo.fs] 'lash Data
Programming Optic File name: sktop/ Retained User	ns 300_test/gowin_empu/cm3_demo/impl/pnr/cm3_demo.fs 'lash Data
Programming Optic Pile name: Sktop, Retained User User Flash Ini	ns 300_test/gowin_empu/cm3_demo/impl/pnr/cm3_demo.fs Tash Data Halization
Programming Optic Pile name: sktop Retained User User Flash Ini	ns 3700_test/gowin_empu/cm3_demo/impl/pnr/cm3_demo.fs] Tash Data Tialization
Programming Optic Programming Optic Programming Skipp Retained User User Flash Ini W/MCU/Binary Ing	ns 370D_test/gowin_empu/cm3_demo/impl/pnr/cm3_demo.fs lash Data dialization at Options
Programming Optic Vile name: Sktop, Retained User User Flash Ini WW/MCU/Binary Inp	ns JMD_test/gowin_empu/om3_demo/impl/pnr/om3_demo.fs Plash Data Halization Int Options

表 9-1 下载设置

选项	设置
Access Mode	MCU Mode
Operation	Firmware Erase, Program
	Firmware Erase, Program, Verify
File name	.fs 文件
Firmware/Binary File	.bin 文件

?

 \times

2. GW1NSER-4C

例如 Secure FPGA Gowin_EMPU(GW1NS-4C),如图 9-4 和表 9-2 所示。

图 9-4 下载设置

Device configura	ition (>
Device Uperation		
Access Mode:	SecureFPGA Mode	~
Operation:	Firmware Erase, Program Securely	~
Firmware Erase, fi Key Authenticati	rogram, verity with Security(Une lime frogrammable on Code)	ror
Programming Optio	ns	
Programming Optio Sile name: sktop/	ns GMD_test/gowin_empu/cm3_demo/impl/pnr/cm3_demo.fs]
Programming Optio File name: sktop/ Retained User	ns GMD_test/gowin_empu/cm3_demo/impl/pnr/cm3_demo.fs Flash Data]
Programming Optio File name: sktop/ Retained User User Flash Ini	ns GMD_test/gowin_empu/om3_demo/impl/pnr/om3_demo.fs Flash Data tialization]
Programming Optio File name: sktop/ Retained User User Flash Ini FW/MCU/Binary Inp Firmware/Binary F	ns GMD_test/gowin_empu/cm3_demo/impl/pnr/cm3_demo.fs Flash Data tialization ut Options ile: e/workspace_test/cm3_demo/Debug/cm3_demo.bin]

表 9-2 下载设置

选项	设置
Access Mode	SecureFPGA Mode
Operation	Firmware Erase, Program Securely
File name	.fs 文件
Firmware/Binary File	.bin 文件

3. GW1N/R-9 (C)

例如 Gowin_EMPU_M1 和 Gowin_PicoRV32, 如图 9-5 和表 9-3 所示。

🛊 Device configura	tion	? X
-Device Operation -		
Access Mode:	MCV Mode L	~
Operation:	Firmware Erase, Program	~
Firmware Erase, Pr	rogram	
Programming Optio	ns	
Programming Optio File name: s/liuk	ns ai/Desktop/GMD_test/cm1_demo/impl/pnr/cm1_demo.	fs
Programming Optio File name: s/liuk	ns ai/Desktop/GMD_test/cm1_demo/impl/pnr/cm1_demo. Flash Data	fs
Frogramming Optio File name: s/liuk Retained User : User Flash Ini	ns ai/Desktop/GMD_test/cm1_demo/impl/pnr/cm1_demo. Flash Data tialization	fs
Programming Optio File name: s/liuk Retained User : User Flash Ini FW/MCU/Binary Inp	ns ai/Desktop/GMD_test/cm1_demo/impl/pnr/cm1_demo. Flash Data tialization ut Options	fs
Programming Optio File name: s/liuk Retained User : User Flash Ini FW/MCU/Binary Inp Firmware/Binary F	ns ai/Desktop/GMD_test/cm1_demo/impl/pnr/cm1_demo. Flash Data tialization ut Options ile: e/workspace_test/cm1_demo/Debug/cm1_demo.b	fs

图 9-5 下载设置

表 9-3 下载设置

选项	设置
Access Mode	MCU Mode L
Operation	Firmware Erase, Program
	Firmware Erase, Program, Verify
File name	.fs 文件
Firmware/Binary File	.bin 文件

4. GW2A/N/R/NR

例如 Gowin_EMPU_M1 和 Gowin_PicoRV32, 如图 9-6 和表 9-4 所示。

图 9-6 下载设置

🙀 Device configuration	?	×	
Device Operation			
Access Mode:	External Flash Mode	\sim	
Operation:	exFlash C Bin Erase, Program thru GAO-Bridge	\sim	
exFlash C Bin Erase, Pr	ogram thru GAO-Bridge	^	
Note! When programming exFlash using GAO Bridge in DUAL BOOT mode, please ensure that the "Start Address" is set to 0x0000000; otherwise, the FPGA will fail to load data from exflash.			
-External Flash Options	External Flash Options		
Device:	Generic Flash	\sim	
Start Address:	0x400000		
-FW/MCU/Binary Input Opt	ions		
Firmware/Binary File:	e/workspace_test/cm1_demo/Debug/cm1_demo.bin		
	Save Cano	el	

表 9-4 下载设置

选项	设置
Access Mode	External Flash Mode
Operation	exFlash C Bin Erase, Program thru GAO-Bridge
	exFlash C Bin Erase, Program, Verify thru GAO- Bridge
Device	Generic Flash
Start Address	GW2A/N/NR/R-18 (C/X): 0x400000 GW2A/N-55 (C): 0x400000
Firmware/Binary File	.bin 文件

5. GW5A/NT/NRT/R/RT/S/ST/T

例如 Gowin_EMPU_M1 和 Gowin_PicoRV32, 如图 9-7 和表 9-5 所示。

图 9-7 下载设置

🙀 Device configuration		?	×	
Device Operation				
Access Mode:	External Flash Mode 5A		\sim	
Operation:	exFlash C Bin Erase, Program 5A		\sim	
exFlash C Bin Erase, Pro	ogram 5A			
Device:	Generic Flash		\sim	
Start Address:	0x600000			
FW/MCU/Binary Input Opt Firmware/Binary File: e	ions :/workspace_test/cm1_demo/Debug/cm1_dem Save	o. bin Can	 cel	

表 9-5 下载设置

选项	设置
Access Mode	External Flash Mode 5A
Operation	exFlash C Bin Erase, Program 5A
	exFlash C Bin Erase, Program, Verify 5A
Device	Generic Flash
Start Address	GW5ANRT/NT/RT/T-15 (A/B): 0x100000
	GW5A/R/S-25 (A/B): 0x100000
	GW5A/T-60B: 0x400000
	GW5AT-75 (B/C): 0x600000
	GW5A/S/ST/T-138 (B/C): 0x600000
Firmware/Binary File	.bin 文件

10调试运行工程

10.1 调试工具

GMD 软件集成了 Gowin OpenOCD 软件调试工具,是高云在 OpenOCD 基础版本上针对高云的 MCU 产品特性进行的再次开发。 OpenOCD 是一个开源的片上调试器,旨在提供针对嵌入式设备的调试、系 统编程和边界扫描功能。OpenOCD 的功能需要调试仿真器来辅助完成,调 试仿真器是一个提供调试目标电信号的小型硬件单元。

GMD 软件还支持使用 J-Link 调试,请前往 Segger 官方网站,根据自己的操作系统下载与安装 Segger J-Link 驱动软件。

GMD 软件也集成了 Gowin QEMU 软件工具,是高云在 QEMU 基础版本上针对高云的 MCU 产品特性进行的再次开发。QEMU 是一个托管的虚拟机,提供多种硬件和外设模型,可以直接在线模拟仿真功能,不需要一个真实的物理平台。

10.2 注意事项

高云所有的 MCU 产品都不支持 GMD 软件在调试过程中自动下载,请 在每次调试运行前,务必先使用下载工具 Programmer 下载想要调试运行的 可执行文件,然后在启动调试,确保每次调试运行的是当前工程。

10.3 调试模式

GMD 软件支持多种高云 MCU 产品、多种调试模式、多种调试仿真器,如表 10-1 所示。

表 10-1 调试模式

MCU 产品	调试模式	调试仿真器
Gowin_EMPU(GW1NS-	GDB OpenOCD	GWU2X
4C)	Debugging	GWUSB (dual FTDI)
		GWUSB (single FTDI)

MCU 产品	调试模式	调试仿真器	
		J-Link (WinUSB driver)	
	GDB SEGGER J-Link Debugging	J-Link (Segger J-Link driver)	
	GDB QEMU arm Debugging	-	
Gowin_EMPU_M1	GDB OpenOCD	GWU2X	
	Debugging	GWUSB (dual FTDI)	
		GWUSB (single FTDI)	
		J-Link (WinUSB driver)	
	GDB SEGGER J-Link Debugging	J-Link (Segger J-Link driver)	
Gowin_EMPU(GW5AS- 25)	GDB SEGGER J-Link Debugging	J-Link (Segger J-Link driver)	
Gowin_EMPU_M3	GDB SEGGER J-Link Debugging	J-Link (Segger J-Link driver)	
Gowin_PicoRV32	GDB OpenOCD	GWU2X	
	Debugging	GWUSB (dual FTDI)	
		GWUSB (single FTDI)	
		J-Link (WinUSB driver)	
		Olimex	
	GDB QEMU riscv32 Debugging	-	

10.4 调试方法

Arm MCU 和 RISC-V MCU 调试方法有以下几种,如图 10-1 所示:

- 选定当前工程,右键选择 "Debug As > Debug Configurations…"
- 选择工具栏 "Debug > Debug Configurations…" (🏘 🔹)
- 选择菜单栏 "Run > Debug Configurations…"

₩ workspace_cm3_h - GMD -	×
<u>F</u> ile <u>E</u> dit <u>S</u> ource Refactor <u>N</u> avigate Se <u>a</u> rch <u>P</u> roject <u>Run</u> <u>G</u> owin <u>W</u> indow <u>H</u> elp	
🖸 🕈 🐷 🐚 🗞 🔻 🦓 🔻 🎆 ! 🖓 ! 🖓 ! 🖳 ! 🏪 ! 🍬 🔄 🗛 Run Last Launched 🛛 🛛 Ctrl+F11 🖿 💁 🖛 🥔 🐼 🖓 💎 ! 🍠 🐼 💷 🏢 🏢	
½	126 🕸
🎦 Project Explorer X " 🗖 🗋 main.c 🙆 de 💡 Profile Last Launched 🗖 🗖 😤 O X 🗐 T 💿 B 🕁 D 🖓	
Image: Second gene of the second gene o	88
Organic Method Breaknaint	
Problems # Tas `s Skip All Breakpoints Ctrl+Alt+B	
Remove All Breakpoints	9 👻
CDT Build Console [cr	
Invoking: GNU ARM Cross Print Size arm-none-eabi-sizeformat-berkeley "cm3_demo.elf" text data bss dec hex filename 29988 1764 392 32144 7d90 cm3_demo.elf Finished building: cm3_demo.siz 16:48:38 Build Finished. 0 errors, 0 warnings. (took 9s.246ms)	~
🔓 cm3_demo	

图 10-1 Debug

10.5 调试设置

"Debug Configurations"中,创建一种模式的调试视图,设置调试选项,如图 10-2 所示。

图 10-2 创建调试视图

🐳 Debug Configurations	- D X
Create, manage, and run configurations	- The second sec
Create, manage, and run configurations	 Configure launch settings from this dialog: Press the 'New Configuration' button to create a configuration of the selected type. Press the 'New Prototype' button to create a launch configuration prototype of the selected type. Press the 'Export' button to export the selected configurations. Press the 'Duplicate' button to copy the selected configuration. Press the 'Delete' button to remove the selected configuration. Press the 'Ielete' button to configure filtering options. Select launch configuration(s) and then select 'Link Prototype' menu item to link a prototype. Select launch configuration(s) and then select 'Lunk Prototype' menu item to unlink a prototype. Select launch configuration(s) and then select 'Re Values' menu item to reset with prototype values. Edit or view an existing configuration by selecting it. Configure launch perspective settings from the 'Perspectives' preference page.
C cm3 demo det G GDB QEMU grua G GDB QEMU riscv32 Debugging G GDB QEMU riscv32 Debugging G GDB SEGGER J-Link Debugging G Launch Group Filter matched 22 of 24 items	
	Debug Close

SUG549-2.0

包含 Arm MCU 和 RISC-V MCU 的调试设置,分别如下所述。

10.5.1 Arm MCU 调试设置

例如 Gowin_EMPU(GW1NS-4C)支持 GDB OpenOCD Debugging、 GDB SEGGER J-Link Debugging、GDB QEMU arm Debugging 调试模 式,如表 10-1 所示。

GDB OpenOCD Debugging

1. Main 视图

Main 视图中的调试设置如图 10-3 所示。

图 10-3 Main 视图

📄 Main 🕸 Debugger 🕨 Startup 🧤 Source 🔲 Co	ommon 🔀 SVD Path	
Project:		
cm3_demo	B	owse
C/C++ Application:		
Debug\cm3_demo.elf		
	<u>V</u> ariables Searc <u>h</u> Project B <u>i</u>	owse
Build (if required) before launching		
Build Configuration: Select Automatically		~
◯ Enable auto build	◯ Disable auto build	
Use workspace settings	Configure Workspace Settings	

2. Debugger 视图

Debugger 视图中的调试设置如图 10-4 和表 10-2 所示。

🗎 Main 🕸 Debugg	er 🕨 Startup 🦻 Source 🔲 Common 🔀 SVD Path		
OpenOCD Setup			
Start OpenOCD	locally		
Executable path:	{openocd_path}/\${openocd_executable}	Browse	Variables
Actual executable:	E:\GMD\\toolchain\gowin\openocd\bin/openocd.exe		
	(to change it use the <u>global</u> or <u>workspace</u> preferences pages or the <u>pr</u>	<u>oject</u> prope	rties page)
GDB port:	3333		
Telnet port:	4444		
Tcl port:	6666		
Config options:	-f board/gowin_u2x_m3.cfg		~
			~
✓ Allocate console	e for OpenOCDAllocate console for the telne	t connectior	1
GDB Client Setup			
✓ Start GDB sessi	on		
Executable name:	\${cross_prefix}gdb\${cross_suffix}	Browse	Variables
Actual executable:	arm-none-eabi-gdb		
Other options:			
Commands:	set mem inaccessible-by-default off		~
			× .
Remote Target			
Remote Target Host name or IP a	ddress: localhost		
Remote Target Host name or IP a Port number:	ddress: localhost 3333		

图 10-4 Debugger 视图

表 10-2 Debugger 视图

选项	设置
Start OpenOCD locally	\checkmark
GDB port	3333
Config options	-f board/x.cfg
	Arm MCU 产品的 x.cfg 对应关系如表 10-3 所示。
Start GDB session	\checkmark
Commands	set mem inaccessible-by-default off

表 10-3 OpenOCD 配置文件

Arm MCU 产品	调试仿真器	配置文件
Gowin_EMPU(GW1NS-	GWU2X	gowin_u2x_m3.cfg
4C)	GWUSB (dual FTDI)	gowin_ftdi_dual_m3.cfg
	GWUSB (single FTDI)	gowin_ftdi_single_m3.cfg

Arm MCU 产品	调试仿真器	配置文件	
	J-Link (WinUSB driver)	gowin_jlink_m3.cfg	
Gowin_EMPU_M1	GWU2X	gowin_u2x_m1.cfg	
	GWUSB (dual FTDI)	gowin_ftdi_dual_m1.cfg	
	GWUSB (single FTDI)	gowin_ftdi_single_m1.cfg	
	J-Link (WinUSB driver)	gowin_jlink_m1.cfg	

3. Startup 视图

Startup 视图中的调试设置如图 10-5 和表 10-4 所示。

图 10-5 Startup 视图

Main 🕸 Debugger 🕨 Startup 🦆 Source 🔲 Common 🛃 SVD Path		
Initialization Commands		· · · · · · · · · · · · · · · · · · ·
☑ Initial Reset. Type: init		
		^ ~
Enable Arm semihosting		
Load Symbols and Executable		
✓ Load symbols		
● Use project binary: cm3_demo.elf		
○ Use file:	Workspace	File System
Symbols offset (hex):		
Load executable		
 Use project binary: cm3_demo.elf 		
◯ Use file:	Workspace	File System
Executable offset (hex):		
Runtime Options		
Debug in RAM		
Run/Restart Commands		
Pre-run/Restart reset Type: halt (always executed at Restart)		
		^
		~
Set program counter at (hex):		
☑ Set breakpoint at: main		
└ Continue		,

表 10-4 Startup 视图

选项	设置
Initial Reset.	\checkmark
Туре	init
Enable Arm semihosting	\checkmark

选项	设置
Load symbols	\checkmark
Load executable	×
Debug in RAM	×
Pre-run/Restart reset	\checkmark
Туре	halt
Set program counter at (hex)	-
Set breakpoint at	main
Continue	\checkmark

GDB SEGGER J-Link Debugging

1. Main 视图

Main 视图中的调试设置如图 10-6 所示。

图 10-6 Main 视图

📄 Main 🏇 Debugger 🕨 Startup 🦻 Source 🔲 Common 🛃 SV	/D Path		
Project:			
cm3_demo	<u>B</u> rowse		
C/C++ Application:			
Debug\cm3_demo.elf			
	<u>V</u> ariables Searc <u>h</u> Project B <u>r</u> owse		
Build (if required) before launching			
Build Configuration: Select Automatically	~		
○ Enable auto build ○ Disabl	O Disable auto build		
Use workspace settings <u>Configure</u>	settings <u>Configure Workspace Settings</u>		

2. Debugger 视图

Debugger 视图中的调试设置如图 10-7 和表 10-5 所示。

J-Link GDB Server Setup	-		
✓ Start the J-Link GDB server locally □Connect to running target			
Executable path: \$(jlink_path)/\$(jlink_gdbserver) Browse Variable	es		
Actual executable: C:/Program Files (x86)/SEGGER/JLink/JLinkGDBServerCL.exe			
(to change it use the global or workspace preferences pages or the project properties	age)		
Device name: Cortex-M3 Supported device r	ames		
Endianness: Little Big 			
Connection: OUSB OIP (USB serial or IP name/address)			
Interface: O SWD			
Initial speed: O Auto O Adaptive I Fixed 1000 kHz			
GDB port: 2331			
SWO port: 2332 Verify downloads 🗹 Initialize registers on	start		
Telnet port: 2333 🔽 Local host only 🗌 Silent			
Log file: Brow	se		
Other options: -singlerun -strict -timeout 0 -nogui			
Allocate console for the GDB server			
GDB Client Setup			
Executable name: \${cross_prefix}gdb\${cross_suffix} Browse Variab	es		
Actual executable: arm-none-eabi-gdb			
Other options:	_		
Commands: set mem inaccessible-by-default off	^		
	~		
Remote Target			
Host name or IP address: localhost			
Port number: 2331			
Force thread list update on suspend			
<	۱ <u>ـد.</u> ۲		

图 10-7 Debugger 视图

选项	设置
Start the J-Link GDB server locally	\checkmark
Connect to running target	×
Device name	例如 Cortex-M3
	Arm MCU 产品对应的设备型号如表 10-
	6 所示。
Endianness	Little
Connection	USB
Interface	JTAG
	SWD

表 10-5 Debugger 视图

选项	设置
Initial speed	Fixed
GDB port	2331
Verify downloads	\checkmark
Initialize registers on start	\checkmark
Local host only	\checkmark
Silent	-
Other options	-singlerun -strict -timeout 0 -nogui
Allocate console for semihosting and SWO	\checkmark
Commands	set mem inaccessible-by-default off

表 10-6 Arm MCU 设备型号

Arm MCU 产品	设备型号
Gowin_EMPU(GW1NS-4C)	Cortex-M3
Gowin_EMPU_M1	Cortex-M1
Gowin_EMPU(GW5AS-25)	Cortex-M4
Gowin_EMPU_M3	Cortex-M3

3. Startup 视图

Startup 视图中的调试设置如图 10-8 所示和表 10-7 所示。

冬	10-8	Startup	视图
---	------	---------	----

🖹 Main 🏇 Debugger 🕨 Startup 🧤 Source 🔲 Common 🕂 SVD Path				
Initialization Commands				
✓ Initial Reset and Halt Type: Low speed: 1000 kHz				
JTAG/SWD Speed: Auto Adaptive Fixed kHz				
✓ Enable flash breakpoints				
☐ Enable semihosting Console routed to: ☑ Telnet □ GDB client				
Enable SWO CPU freq: 0 Hz. SWO freq: 0 Hz. Port mask: 0x1				
×				
Load Symbols and Executable				
✓ Load symbols				
Use project binary: cm3_demo.elf				
O Use file: Workspace File System				
Symbols offset (hex):				
Load executable				
Use project binary: cm3_demo.elf				
OUse file: Workspace File System				
Executable offset (hex):				
Runtime Options				
RAM application (reload after each reset/restart)				
Run/Restart Commands				
✓ Pre-run/Restart reset Type: (always executed at Restart)				
Set program counter at (hex):				
✓ Set breakpoint at: main				
✓ Continue				
Restore defaults				

表 10-7 Startup 视图

选项	设置
Initial Reset and Halt	\checkmark
JTAG/SWD Speed	Auto
Enable flash breakpoints	\checkmark
Enable semihosting Console routed to	Telnet
Load symbols	\checkmark
Load executable	X
RAM application (reload after each reset/restart)	×

选项	设置
Pre-run/Restart reset	\checkmark
Set program counter at (hex)	-
Set breakpoint at	main
Continue	\checkmark

GDB QEMU arm Debugging

Arm MCU 产品的 QEMU 仿真,请参照第 11 章 QEMU 仿真调试。

10.5.2 RISC-V MCU 调试设置

例如 Gowin_PicoRV32 支持 GDB OpenOCD Debugging、GDB QEMU riscv32 Debugging 调试模式,如表 10-1 所示。

GDB OpenOCD Debugging

1. Main 视图

Main 视图中的调试设置如图 10-9 所示。

图 10-9 Main 视图

📄 Main 🕸 Debugger 🕨 Star	tup 🦻 Source 🔲 Common 🔀 SVD Path		
Project:			
picorv32_demo		<u>B</u> rowse	
C/C++ Application:			
Debug\picorv32_demo.elf			
	<u>V</u> ariables Searc <u>h</u> Project	B <u>r</u> owse	
-Build (if required) before laund	hing		
Build Configuration: Select Au	Build Configuration: Select Automatically		
◯ Enable auto build	ble auto build O Disable auto build		
Use workspace settings	Use workspace settings <u>Configure Workspace Settings</u>		

2. Debugger 视图

Debugger 视图中的调试设置如图 10-10 和表 10-8 所示。

OpenOCD Setup					
Start OpenOCD	locally				
Executable path:	{openocd_path}/\${openocd_executable}	Browse	Variables		
Actual executable:	E:\GMD\\toolchain\gowin\openocd\bin/openocd.exe				
	(to change it use the <u>global</u> or <u>workspace</u> preferences pages or the	<u>project</u> pro	perties page)		
GDB port:	3333				
Telnet port:	4444				
Tcl port:	6666				
Config options:	-f board/gowin_u2x_pico.cfg		^		
			¥		
Allocate console	a for OpenOCD Allocate console for the teln	et connectio	on		
GDB Client Setup					
Start GDB sessio		Durauna	Manialalaa		
Executable name:	\${cross_prefix}gab\${cross_suffix}	Browse	variables	1	
Actual executable:	: riscv-none-elf-gdb				
Other options:					
Commands:	set mem inaccessible-by-default off		<u>^</u>		
	set remotetimeout 250				
			×		
Remote Target					
Host name or IP a	ddress: localhost				
Port number:	3333				
]Force thread list u	pdate on suspend				

图 10-10 Debugger 视图

表 10-8 Debugger 视图

选项	设置
Start OpenOCD locally	\checkmark
GDB port	3333
Config options	-f board/x.cfg
	RISC-V MCU 产品的 x.cfg 对应关系如表
	10-9 所示。
Start GDB session	\checkmark
Commands	set mem inaccessible-by-default off
	set arch riscv:rv32
	set remotetimeout 250

RISC-V MCU 产品 调试仿真器 配置文件 Gowin_PicoRV32 GWU2X gowin_u2x_pico.cfg GWUSB (dual FTDI) gowin_ftdi_dual_pico.cfg GWUSB (single FTDI) gowin_ftdi_single_pico.cfg J-Link (WinUSB driver) gowin_jlink_pico.cfg Olimex gowin_olimex_pico.cfg

表 10-9 OpenOCD 配置文件

3. Startup 视图

Startup 视图中的调试设置如图 10-11 和表 10-10 所示。

图 10-11 Startup 视图

📄 Main 🕸 Debugger 🕨 Startup 🦃 Source 🔲 Common 🚼 SVD Path	
Initialization Commands	^
☑ Initial Reset. Type: init	
^	
×	
Enable Arm semihosting	
Load Symbols and Executable	
✓ Load symbols	
Use project binary: picorv32_demo.elf	
Ouse file: Workspace File System	
Symbols offset (hex):	
☑ Load executable	
Use project binary: picorv32_demo.elf	
O Use file: Workspace File System	
Executable offset (hex):	
Runtime Options	
☑ Debug in RAM	
Run/Restart Commands	
Pre-run/Restart reset Type: halt (always executed at Restart)	
^	
×	
Set program counter at (hex):	
☑ Set breakpoint at: main	
Continue	~

表 10-10 Startup 视图

选项	设置
Initial Reset.	\checkmark

选项	设置
Туре	init
Enable Arm semihosting	×
Load symbols	\checkmark
Load executable	×
Debug in RAM	\checkmark
Pre-run/Restart reset	\checkmark
Туре	halt
Set program counter at (hex)	-
Set breakpoint at	main
Continue	\checkmark

GDB QEMU riscv32 Debugging

RISC-V MCU 产品的 QEMU 仿真,请参照第 11 章 QEMU 仿真调试。

10.6 运行工程

例如在运行 Gowin_EMPU(GW1NS-4C 工程时,可以搭配使用 Segger J-Link 调试仿真器。

10.6.1 启动调试视图

完成调试设置后,单击"Debug",启动调试视图,运行工程,如图 10-12 所示。

🗱 workspace_test - cm3_demo/src/project/printf/printf_dem	o.c - GMD						-	
<u>File Edit Source Refactor Navigate Search Project</u>	<u>R</u> un <u>G</u> owin <u>W</u> indow <u>H</u> elp							
i 🗂 🕶 🔚 🕼 i 🖓 😒 i 🖳 i 🗙 🕨 💷 🛤 🕺 🧟	o .e i> 🗟 😒 🍪 🐫 🔅 🗸 🛛) • 💁 • 🙋 🗁 🤞	² • 🛃 ≎ i 🖗 • ł	yi • 🏷 🕫 🔶	• 🗢 • 🖻	5	Q	8 🖬 💀
🎋 Debug 🗙 🍋 Project Explorer 🛛 🙀 it+ 🖇 🗖 🗖	ⓑ demo.h	i main.c		- 0	(x)= Var ×	⁰₀ Bre 🤄	f Ex 🔀 Pe	er 🖻 🗖
✓	33 0, //]	[x interrupt		^			🖾 🍂 E	1 📑 🖻 🕴
✓ ⑦ cm3_demo.elf	34 0, //	<pre>1x interrupt Ix overflow interru</pre>	pt		Name	Туре	Value	
✓	36 0); //	x overflow interru	pt		> ptr	char *	0x7020 <	
printf_demo() at printf_demo.c:50 0x12c	37 chan *ntn = "Hollo work	41.5			(x)= i	int	5	
main() at main.c:60 0x19c	39 int i = 5;	. ,			(x)= bs	unsigned i	32	
JLinkGDBServerCL.exe	40 unsigned int bs = sizeo	f(int)*8;			(x)• mi	int	-2147483	
arm-none-eabi-gdb	41 int mi;							
Semihosting and SWV	43 mi = (1 << (bs-1)) + 1;							
	44							
	45 printf(%s\r\n , ptr); 46 printf("%d = 5\r\n", i)							
	<pre>47 printf("%d = - max inter</pre>	, ger\r\n", mi);						
	48 printf("char %c = 'a'\r'	\n", 'a'); "ff\;						
	<pre>>50 printf("hex %02x = 00\r"</pre>	(n", 0);						
	51 printf("signed %d = uns	igned %u = hex %x∖r	\n", -3, -3, -3);					
	52 print+("%d %s(s)", 0, "	message");						~
	54 printf("%d %s(s) with %	K\r\n" , 0, "m essage	");					
	55							
	57 }							
	58			~				
	< #and1+			>	<			>
	📮 Console 🚟 Registers 🖹 Problem	s 🜔 Executables 🖉	Terminal 🗙 🛃 Deb	ugger Console	Memory	🗉 🗤 🛄 🛙	🔒 🔠 🖪 (e - e
	COM58 X							
	Hello world!							~
	5 = 5							
	char a = 'a'							
	hex ff = ff							
< >								~
		Writable	Smart Insert	50:1:1257				10

图 10-12 启动调试视图

10.6.2 调试功能选项

调试功能选项如表 10-11 所示。

选项	功能说明
Resume (继续运行暂停的调试线程
Suspend (临时暂停调试线程
Terminate (停止调试线程
Step Into (🏊)	单步运行调试线程的当前语句,进入子 函数调用
Step Over (😨)	单步运行调试线程的当前语句,跳过子 函数调用
Step Return (🚅)	单步运行调试线程的当前语句,直到返 回其上一级调用者
Restart (停止本次调试线程,重启新的调试线程
Instruction Stepping Mode (i>)	进入反汇编视图,单步运行汇编指令

表 10-11 调试功能选项

10.7 调试视图

启动调试线程后,GMD 软件自动调用相应的调试视图。为调用指定的 调试视图,可以选择菜单栏 "Window > Show View > Other…",如图 10-13 所示。"Show View > Debug"下拉列表中,选择想要显示的调试视 图,如图 10-14 所示。

图 10-13 Window > Show View > Other...

File fidit Source Refactor Navigate Search Project Bun Gowin Window Help Image: Source Refactor Navigate Search Project Bun Gowin Navigation Image: Source Refactor Navigate Search Project Explorer Image: Source Refactor Navigate Search Project Explorer Image: Source Refactor Navigate Search Project Explorer Image: Source Refactor Navigate Search Project Explorer Image: Source Refactor Navigate Search Project Explorer Image: Source Refactor Navigate Search Project Explorer Image: Source Refactor Navigate Search Project Explorer Image: Source Refactor Navigate Search Project Explorer Image: Source Refactor Navigate Search Project Bun Gowin Mindow Image: Source Search Project Explorer Image: Source Refactor Navigate Search Project Bun Gowin Mindow Image: Source Search Project Search Pro	🐝 workspace_test - cm3_demo/src/project/p	rintf/printf_demo.c - GMD				_	o x
Image: Second	<u>F</u> ile <u>E</u> dit <u>S</u> ource Refac <u>t</u> or <u>N</u> avigate Se	<u>a</u> rch <u>P</u> roject <u>R</u> un <u>G</u> owin	<u>Window</u> <u>H</u> elp				
Editor ** Debug × Project Explorer ** Debug × Project Explorer ** Com3_demo. Debug jink (GDB SEGGER) ** Com3_demo. Bebug iink (GDB SEGGER) ** Com3_demo.eff ** Printf_demo() at printf_demo() ** Drintf_demo() at printf_demo() ** SystemInit(); ** Drintf_demo() ** SystemInit(); ** Drintf_demo() ** Drintf_demo() ** Drintf_demo() ** SystemInit(); ** Drintf_demo() ** Drintf_demo() </td <td></td> <td>■ N 7. 70. 10 i> i</td> <td>R New Window</td> <td>19</td> <td>🖌 🕶 🗁 🛷 🔹 🗋</td> <td>s 🖇</td> <td></td>		■ N 7. 70. 10 i> i	R New Window	19	🖌 🕶 🗁 🛷 🔹 🗋	s 🖇	
* Debug X Project Explorer Project Explorer<	월 - 得 - ♡ -♡ - ○ - 🛃		Editor			٩	2 🖬 🔯
20 #include <stdio.h> Show View ● Breakpoints Alt+Shift+Q, B 21 alue 21 22 /* Definitions: */ Navigation ● Debug Alt+Shift+Q, C alue 22 /* Definitions: */ Perspective ● Debug Console Alt+Shift+Q, C alue 23 /* Definitions: */ Preferences ● Debug Debug 22 22 24 //Application entry Preferences ● Debug Support 22 24 <t< td=""><td>💠 Debug 🗙 陷 Project Explorer 🛛 🗖</td><td>h demo.h 🖻 printf_d</td><td>e Appearance ></td><td></td><td>□ (x)= V</td><td>🗙 🎭 в 🕵 е 🔀</td><td>P</td></t<></stdio.h>	💠 Debug 🗙 陷 Project Explorer 🛛 🗖	h demo.h 🖻 printf_d	e Appearance >		□ (x)= V	🗙 🎭 в 🕵 е 🔀	P
 	🖻 🐜 i> 🕴	20 #include <stdio.h< td=""><td>Show View</td><td>•</td><td>Breakpoints</td><td>Alt+Shift+Q, B</td><td>1 🛃 🖻</td></stdio.h<>	Show View	•	Breakpoints	Alt+Shift+Q, B	1 🛃 🖻
<pre>v @ cm3_demo.elf v @ Thread #1 57005 (Suspended : Ste</pre>	✓ 💽 cm3_demo Debug jlink [GDB SEGGER J-	21 22	Perspective >		Console	Alt+Shift+Q, C	alue
Printead #1 \$7005 (Guspended : Str ²⁴ //Application entry preferences main() at main.c600 0x19c main() at main.c600 0x19c seminosting and SWV ²⁵ //SystemInit(); //Initializes us uart_init(UART0, //Ix interrupt d, //Ix interrupt d, //Ix overflow int d, //Ix overflow int decar *ptr = "Hello world!"; unsigned int bs = sizeof(int)*8; Debugs Console × IIII Registers Problems Alt+Shift+Q, Z Project Explorer Read 4 bytes @ address 0x0000002C (Data = 0xF94 Reading 8 bytes @ address 0x0000002C (Data = 0xF94 Reading 8 bytes @ address 0x00000728 Reading 8 bytes @ address 0x00000728 Reading 8 bytes @ address 0x0000728 Reading 8 bytes @ address 0x000072	∨ 🎲 cm3_demo.elf	23 /* Definitions: *,	Navigation >	*	Debug		x7020 <
<pre>printf_demo() at printf_demo.ci main() at main.c:60 0x19c J LinkGDBServerCLexe main() at main.c:60 0x19c J LinkGDBServerCLexe main() at main.c:60 0x19c J LinkGDBServerCLexe main() at main.c:60 0x19c main() at main.centers main() at main.centers</pre>	✓ P Thread #1 57005 (Suspended : Ste	24 25 //Application ent	N		Debugger Console		
■ main() at main.cb0 0x19c JLinkGDBServerCLexe JLinkGDBServerCLexe JunkGDBServerCLexe Junk	printf_demo() at printf_demo.c:/	26⊖ int printf_demo(v	Preferences	ø	Debug Sources		2
JunkobserverLexe Semihosting and SWV Semifor Semifore SWV Semihosting and SWV </td <td>main() at main.c:60 0x19c</td> <td>27 {</td> <td>//Initializes su</td> <td></td> <td>Disassembly</td> <td></td> <td>2147483</td>	main() at main.c:60 0x19c	27 {	//Initializes su		Disassembly		2147483
30 38400, //Budcate 31 1, //X 32 1, //X 33 0, //X 34 0, //X 0; //X 36 char *ptr = "Hello world!"; 39 unsigned int bs = sizeof(int)*8; 0 int i = 5; 0 unsigned int bs = sizeof(int)*8; 0 Console × IMR Registers Problems © Execute 0 maddress 0x0000001c (Data = 0xF9) Reading 8 bytes @ address 0x000007028 Reading 8 bytes @ address 0x000007028 0 Variables 0 Variables 0 Variables	JLinkGDBServerCL.exe	29 uart_init(UAR	0, //Initializes UA	v 🕗	Error Log	Alt+Shift+Q, L	
1, //k 22 1, //k 33 0, //IX interrupt 34 0, //IX interrupt 35 0, //IX overflow int 36 char *ptr = "Hello world!"; Modules 37 Modules Problems 38 char *ptr = "Hello world!"; Problems 39 int i = 5; Problems 40 unsigned int bs = sizeof(int)*8; Problems 5 0 Project Explorer 6 Console × WR Registers Problems 6 Console × WR Registers Problems 7 March Signals Templates 7 Read 4 bytes @ address 0x0000026 Pate = 0x466 Terminal 8 Bytes @ address 0x00007028 Procentrol V 8 Bytes @ address 0x00007028 Procentrol V 7 Terminal Procentrol V V	Semibosting and SWV	30 3844	0, //Baudrate	O	Executables		L
33 0, //Ix interrupt 34 0, //Ix interrupt 35 0, //Ix overflow int 36 0); //Rx overflow int 37 Modules 38 char *ptr = "Hello world!"; 39 int i = 5; 40 unsigned int bs = sizeof(int)*8; 40 Problems 40 Project Explorer 40 Project Explorer 41 Signals 42 Project Explorer 43 Bytes @ address 0x0000006C (Data = 0x46) 44 bytes @ address 0x00000728 45 Signals 46 Trace Control 47 Project Explorer 48 Bytes @ address 0x0000728 49 Project Explorer 49 Project Explorer 49 Project Explorer 40 Project Explorer 40 Bytes @ address 0x0000728 40 Bytes @ address 0x0000728 41 Bytes @ address 0x0000728 42 Project Explorer 43 Bytes @ ad		32 1 ,	//Rx	ଙ୍କୁ	Expressions		
34 0, //Xs interrupt 35 0, //Xs overflow int 36 0); //Xs overflow int 37 0); //Rs overflow int 38 char *ptr = "Hello world!"; 39 int i = 5; 40 unsigned int bs = sizeof(int)*8; 40 unsigned int bs = sizeof(int)*8; 40 Problems 41+Shift+Q, X 42 Project Explorer 43 Registers 44 bytes @ address 0x00000066 (Data = 0xF9; 44 bytes @ address 0x00007020 45 Signals 46 Tareminal 47 address 0x00007020 48 bytes @ address 0x00007020 49 Bytes @ address 0x00007020 40 Bytes @ address 0x00007020 41 Signals 42 Dytes @ address 0x00007020 43 Bytes @ address 0x00007020 44 Bytes @ address 0x00007020 44 Bytes @ address 0x00007028 44 Bytes @ address 0x00007028 44 Bytes @ address 0x00007028		33 0,	//Tx interrupt	0	Memory		
36 0); //®: overflow int Modules 37 char *ptr = "Hello world!"; Dutline Alt+Shift+Q, O 39 int i = 5; Peripherals Problems Alt+Shift+Q, X 40 unsigned int bs = sizeof(int)*8; Problems Alt+Shift+Q, X © Console × IIII Registers Problems Execute Project Explorer Image: Console × IIII Registers Problems Execute Project Explorer Image: Console × IIII Registers Problems Execute Project Explorer Image: Console × IIII Registers Problems Execute Project Explorer Image: Console × IIII Registers Project Explorer Image: Console × Image: Console × Image: Console × Read 4 bytes @ address 0x0000006C (Data = 0xF9) Figure © Signals Image: Console × Image: Console × Reading & bytes @ address 0x00007020 Image: Console × Read + bytes @ address 0x00007020 Image: Console × Imag		34 0, 35 0,	//RX interrupt //Tx overflow in	1	Memory Browser		
3/2 char *ptr = "Hello world!"; B: Outline Alt+Shift+Q, O 39 int i = 5; unsigned int bs = sizeof(int)*8; Peripherals Problems Alt+Shift+Q, X 40 visit Registers Problems Execute Project Explorer V V V 10 Console X Hill Registers Problems Execute Project Explorer V <		36 0);	//Rx overflow in	nt 🛋	Modules		^
39 int i = 5; unsigned int bs = sizeof(int)*8; Image: Project Splore Image: Project Splore Project Explore Image: Project Splore Image: Project Splore Image: Project Splore		37 38 char *ptr = "	ello world!":	믈	Outline	Alt+Shift+Q, O	
40 unsigned int bs = sizeof(int)*8; Problems Alt+Shift+Q, X Console X Win Registers Problems Execute Cm3_demo Debug jilnk [GDB SEGGER J-Link Debugging] Registers Signals Read 4 bytes @ address 0x000000C6 (Data = 0x+5) Signals Templates Reading 8 bytes @ address 0x00007028 @ Terminal Reading 8 bytes @ address 0x00007028 @ Trace Control Reading 8 bytes @ address 0x00007028 @ Trace Control Reading 8 bytes @ address 0x00007028 Winables Alt+Shift+Q, V		39 int i = 5;	,	累	Peripherals		
Image: Console X in Registers Image: Problems Image: Console X in Registers Image: Console X in		40 unsigned int	os = sizeof(int)*8;		Problems	Alt+Shift+Q, X	×
Console × IIII Registers Problems Executa Project Explorer Registers m3_demo Debug jilnk [GDB SEGGER J-Link Debugging] Read 4 bytes @ address 0x0000006C (Data = 0x454 Reading 8 bytes @ address 0x000007020 Reading 8 bytes @ address 0x00007020 Cother		<		-	Progress		>
cm3_demo Debug jlink [GDB SEGGER J-Link Debugging] Registers Signals Read 4 bytes @ address 0x00000026 (Data = 0xF94 Templates Templates Reading 8 bytes @ address 0x000007020 Terminal Terminal Reading 8 bytes @ address 0x00007020 Trace Control Trace Control Reading 8 bytes @ address 0x00007028 Wariables Alt+Shift+Q,V		Console × 🔡 Registe	rs 🖹 Problems 🚺 Execut	a 🔁	Project Explorer		y - 🗆
cm3_demo Debug jilnk [GDB SEGGER J-Link Debugging] is signals Read 4 bytes @ address 0x0000019C (Data = 0xf94 Templates Templates Reading 8 bytes @ address 0x00007020 Terminal Trace Control Reading 8 bytes @ address 0x00007020 Trace Control Variables Alt+Shift+Q, V Reading 8 bytes @ address 0x00007028 Image: Control Variables Alt+Shift+Q, V				1010	Registers		P - 📬 -
Read 4 bytes @ address 0x00000016 (Data = 0xf94 Templates Reading 8 bytes @ address 0x00000102 (Data = 0x461 Templates Reading 8 bytes @ address 0x00007020 Terminal Reading 8 bytes @ address 0x00007020 Trace Control Reading 8 bytes @ address 0x00007020 Variables Alt+Shift+Q, V Variables		cm3_demo Debug jlink [GD	B SEGGER J-Link Debugging] <u>-</u>	Signals		
Reading 8 bytes @ address 0x00007020 P Terminal Reading 8 bytes @ address 0x00007020 Trace Control Reading 8 bytes @ address 0x00007020 Image: Control Contrect Control Control Control Control Control Control Control Contre		Read 4 bytes @ address Read 4 bytes @ address	0x000000006 (Data = 0xF9 0x0000019C (Data = 0x46	1 🖬	Templates		<u>^</u>
Reading 8 bytes @ address 0x00007028 Reading 8 bytes @ address 0x00007020 Reading 8 bytes @ address 0x00007028 (x)= Variables Alt+Shift+Q, V		Reading 8 bytes @ addre	ss 0x00007020	æ	Terminal		
Reading 8 bytes @ address 0x00007028 (x)= Variables Alt+Shift+Q, V		Reading 8 bytes @ addre Reading 8 bytes @ addre	ss 0x00007028	8	Trace Control		
		Reading 8 bytes @ addre	ss 0x00007028	(x)=	Variables	Alt+Shift+Q, V	
					Other	Alt+Shift+O_O	×
	< >>	<		-		rationality of	

type filter text	^
type filter text	^
 ✓ Debug ● Breakpoints ☆ Debug ☑ Debugger Console @ Debug Sources 	^
 Breakpoints Debug Debugger Console Debug Sources 	
☆ Debug ₽ Debugger Console ഈ Debug Sources	
🛃 Debugger Console 🎯 Debug Sources	
🕮 Debug Sources	
🔤 Disassembly	
Executables	
ଙ୍କୁ Expressions	
Launch Configurations	
Memory	
Memory Browser	
Modules	
GS Resources	J
Doriphorals	·
Open Cancel	

图 10-14 Debug View

一些常用的调试视图如下所述。

10.7.1 内存视图

内存(Memory)视图以内存监视器列表的形式检查和修改程序的内存,如图 10-15 所示。每个内存监视器都关联一段指定基地址或表达式的内存,每个内存监视器的数据都可以以多种预定义的格式显示。

图 10-15 内存视图

📮 Console り Register	s 🖹 Problems	Executable	s 🖉 Terminal	强 Debugge	r Console 📋	Memory $ imes$			
					1010 1010	🖹 🛃 📑	U 🖓	• •	000
Monitors 🛛 🖶 💥	🕷 printf_demo	: 0x80 <hex></hex>	🗙 🐈 New R	enderings					
printf_demo	Address	0 - 3	4 - 7	8 - B	C - F				^
	00000080	01040000	03040000	05040000	07040000				
	00000090	09040000	0B040000	0D040000	0F040000				
	000000A0	11040000	13040000	15040000	17040000				
	000000B0	19040000	1B040000	1D040000	1F040000				
	00000C0	80B588B0	04AF00F0	ABF90023	03930023				
	000000D0	02930023	01930023	00930123	01224FF4				
	000000E0	16414FF0	402000F0	A5F81E4B	FB600523				
	000000F0	BB602023	7B607B68	013B0122	02FA03F3				
	00000100	01333B60	F9681848	00F0A8FA	B9681748				
	00000110	00F0A4FA	39681648	00F0A0FA	61211548				
	00000120	00F09CFA	FF211448	00F098FA	00211348				
	00000130	00F094FA	6FF00203	6FF00202	6FF00201				
	00000140	0F4800F0	8BFA0F4A	00210F48	00F086FA				
	00000150	0E4800F0	E9FA0B4A	00210D48	00F07EFA				
	00000160	FEE700BF	20700000	30700000	38700000				
	00000170	44700000	5C700000	6C700000	7C700000				
	00000180	8C700000	B0700000	B8700000	C4700000				~

建立内存监视器

 暂停程序运行后,在调试视图中选择一个想要监视的线程或堆栈结构, 如图 10-16 所示。

图 10-16 选择要监视的线程或堆栈



内存视图中,选择 "Monitors > Add Memory Monitor" (♣),如图
 10-17 所示。"Monitor Memory"中,输入一个想要查看的内存地址或

表达式,如图 **10-18** 所示。

图 10-17 Add Memory Monitor



图 10-18 Monitor Memory

🗱 Monitor Memory	×
Enter address or expres monitor:	sion to
printf_demo	~
🕐 ок	Cancel

3. 内存视图中,默认显示一个传统格式的内存渲染器,如图 10-19 所示。

📮 Console 💧	Reg	jisters 🚦	🛚 Problems	Executable	s 🖉 Terminal	🛃 Debugge	r Console 📋	Memory $ imes$			
							1010 1010	° 🛃 ⇒ti	U 🔁	•	000
Monitors	-	X %	printf_demo	: 0x80 <hex></hex>	🗙 🐈 New R	enderings					
oprintf_c	demo		Address	0 - 3	4 - 7	8 - B	C - F				^
			00000080	01040000	03040000	05040000	07040000				
			00000090	09040000	0B040000	0D040000	0F040000				
			000000A0	11040000	13040000	15040000	17040000				
			000000B0	19040000	1B040000	1D040000	1F040000				
			000000C0	80B588B0	04AF00F0	ABF90023	03930023				
			000000D0	02930023	01930023	00930123	01224FF4				
			000000E0	16414FF0	402000F0	A5F81E4B	FB600523				
			000000F0	BB602023	7B607B68	013B0122	02FA03F3				
			00000100	01333B60	F9681848	00F0A8FA	B9681748				
			00000110	00F0A4FA	39681648	00F0A0FA	61211548				
			00000120	00F09CFA	FF211448	00F098FA	00211348				
			00000130	00F094FA	6FF00203	6FF00202	6FF00201				
			00000140	0F4800F0	8BFA0F4A	00210F48	00F086FA				
			00000150	0E4800F0	E9FA0B4A	00210D48	00F07EFA				
			00000160	FEE700BF	20700000	30700000	38700000				
			00000170	44700000	5C700000	6C700000	7C700000				
			00000180	8C700000	B0700000	B8700000	C4700000				~

图 10-19 传统格式的内存渲染器

也可以选择 "New Renderings… > Floating Point > Add Rendering(s)",如图 10-20 所示,创建一个浮点格式的内存渲染器,如图 10-21 所示。

📮 Ca	onsole	1919 Reg	gisters [Problems	C Executat	oles	🖉 Termin	al 🖳	Debugger	Console	🚺 Memo	ory ×				
									_	1010	010 📑 🛃	⇒tä	Щ <	3 4	•	000
Monit	ors	-	X %	printf_demo	<hex></hex>	÷ 1	New Rende	rings								
	printf	demo		Memory Me	onitor: printf	den	no : 0x80		_							
				Select rende	ering(s) to cr	eate										
				Floating Po	int								Add	Rende	ring	(s)
				Traditional								!			-	
				Hex												
				Signed Inte	ger											
				Unsigned I	nteger											
				Hex Intege	r											

图 10-20 创建浮点格式的内存渲染器

图 10-21 浮点格式的内存渲染器

🗐 Console り Registers 🔝 Problem	ns 🜔 Executables 🖉	Terminal 🛛 🗟 Debu	gger Console 📋 N	lemory ×		
			1010 1010	🛃 🍕 🗒 🕏	- -	000
1onitors 🛛 💠 💥 🙀 printf_den	no <hex> printf_d</hex>	emo : 0x80 <floatin< th=""><th>g Point> 🗙 🐈 Ne</th><th>w Renderings</th><th></th><th></th></floatin<>	g Point> 🗙 🐈 Ne	w Renderings		
printf_demo 0x000000	00 1.086320E-19	Infinity	Infinity	Infinity		^
0x000000	10 Infinity	Infinity	Infinity	0.00000E0		
0x000000	20 0.00000E0	0.00000E0	0.00000E0	Infinity		
0×000000	30 Infinity	0.00000E0	Infinity	Infinity		
0×000000	40 Infinity	Infinity	Infinity	Infinity		
0x000000	50 Infinity	Infinity	Infinity	Infinity		
0x000000	60 Infinity	Infinity	0.00000E0	Infinity		
0x000000	70 Infinity	Infinity	Infinity	Infinity		
0×000000	80 Infinity	Infinity	Infinity	Infinity		
0x000000	90 Infinity	Infinity	Infinity	Infinity		
0x000000	A0 Infinity	Infinity	Infinity	Infinity		
0×000000	B0 Infinity	Infinity	Infinity	Infinity		
0×000000	C0 -9.946888E-10	-1.593026E29	6.991763E-18	6.970025E-18		
0×000000	D0 6.970024E-18	6.970023E-18	7.024232E-18	-6.564301E31		
0×000000	E0 -2.565683E29	-1.586123E29	1.041834E7	7.230481E-18		
0x000000	F0 8.694101E-18	4.748375E24	1.751400E-18	-1.045626E31		
0x000001	00 5.395651E19	1.560679E5	-4.385868E35	1.550429E5		
0×000001	10 -4.282022E35	1.540169E5	-4.178176E35	1.527095E5		
0×000001	20 -4.074330E35	1.516880E5	-3.970485E35	1.506600E5		
0-000001	> > > > > > > > > > > > > > > > > > > >	> 0470575 >7	0 6100036 30	Tafiaito		×

4. 右键内存渲染器的任意位置,弹出的下拉菜单中选择"Endian",可以 修改数据显示的字节顺序或可寻址大小,如图 10-22 所示。

🖳 Console り 👭 Registers	s 🖹 Problems 🌔	🕽 Executables 🖉 Terminal 🛛 🗟 Deb	ugger	Console 🚺 M	lemory ×	- 0
				1010 1010 📑	🛃 🍕 🗒 🔁	- 8
Monitors 🛛 🖕 🗶 🎗	🐞 printf_demo <	Hex> printf_demo <floating poi<="" td=""><td>nt> ×</td><td>🗧 🐈 New Rend</td><td>derings</td><td></td></floating>	nt> ×	🗧 🐈 New Rend	derings	
orintf_demo	0×00000000	1.086320E-19 Infinity	I	nfinity	Infinity	1
	0x00000c es	Add Watchpoint (C/C++)		ıfinity	0.00000E0	
	0×000000	Find and Replace Ctr	I+F	.000000E0	Infinity	
	0x000000			ifinity	Infinity	
	0x000000 🕂	Add Rendering		ıfinity	Infinity	
	0x000000 💥	Remove Rendering		ıfinity	Infinity	
	0×000000	Endian	~	Pia	Infinity	
	0x000000	chuid an in T		Big	nfinity	
	0x000000	Floating Point Type	~	Little	nfinity	
	0x000000	Displayed Precision	>	ofinity	Infinity	
	0×000000	Columns	>	ifinity	Infinity	
	0×000000	Update Mode	>	ıfinity	Infinity	
	0x000000			.991763E-18	6.970025E-18	
	0x000000	Copy To Clipboard		024232E-18	-6.564301E31	
	0×000000	Copy Address		041834E7	7.230481E-18	
	0x000000	Reset To Base Address		751400E-18	-1.045626E31	
	0×000001	Refresh		385868E35	1.550429E5	
	0x000001	-4.2020222233 1.34010323		178176E35	1.527095E5	
	0x00000120	-4.074330E35 1.516880E5	-3	.970485E35	1.506600E5	
	0.000000130	> OCCONESE > ONTOFTE 37	0	c100000 00	Tafinito	

也可以修改每个单元格中的内存值,但是如果输入了不恰当的值,程序 运行就会宕机。

5. 程序运行过程中,会以红色标记所存储内容的偏差,如图 10-23 所示。

冬	10-23	存储内容显示
---	-------	--------

📮 Console	iiii Reg	gisters 🖁	🛛 Problems	Executable	s 🖉 Termina	🛛 🖳 Debugge	er Console	🚺 Memory 🗙	-	
							1019 101	• 🖹 🛃 斗	🛄 🔁 🖁 🔻	000
Monitors	-	X 🔆	printf_demo	: 0x80 <hex></hex>	× printf_den	no <floating p<="" td=""><td>oint> 📢</td><td>🕨 New Renderin</td><td>gs</td><td></td></floating>	oint> 📢	🕨 New Renderin	gs	
print	f_demo		Address	0 - 3	4 - 7	8 - B	C - F			^
			00000000	A 00400020	A 89030000	A D1030000	A D303000	00		
			00000010	A D5030000	A D7030000	A D9030000	A 0000000	0		
			00000020	▲ 00000000	▲ 00000000	▲ 00000000	A DB0300	00		
			00000030	A DD030000	▲ 00000000	A DF030000	A E103000	0		
			00000040	▲ E3030000	▲ E5030000	A E7030000	A E903000	0		
			00000050	▲ EB030000	▲ ED030000	▲ EF030000	A F103000	0		
			00000060	▲ F3030000	▲ F5030000	▲ 00000000	A F703000	0		
			00000070	▲ F9030000	▲ FB030000	A FD030000	A FF03000	0		
			00000080	▲ 01040000	▲ 03040000	▲ 05040000	A 0704000	0		
			00000090	▲ 09040000	▲ 0B040000	▲ 0D040000	A 0F04000	0	0x80	
			0A000000	▲ 11040000	▲ 13040000	▲ 15040000	A 1704000	0	0,00	
			000000B0	▲ 19040000	▲ 1B040000	▲ 1D040000	▲ 1F04000	0		
			000000C0	▲ 80B588B0	▲ 04AF00F0	ABF90023	A 0393002	23		
			000000D0	▲ 02930023	01930023	▲ 00930123	▲ 01224FF	4		
			000000E0	▲ 16414FF0	▲ 402000F0	A5F81E4B	A FB60052	23		
			000000F0	▲ BB602023	▲ 7B607B68	▲ 013B0122	▲ 02FA03F	3		
			00000100	▲ 01333B60	▲ F9681848	▲ 00F0A8FA	A B968174	48		~

设置渲染效果

内存渲染器默认以黑色文本显示,可以参照以下步骤以不同颜色区分渲 染的数据和地址。 1. 内存视图中,选择"View Menu"(⁸),弹出的下拉菜单中可以选择

"Traditional Rendering Preferences..."或 "Floating Point Rendering Preferences...",分别设置传统格式或浮点格式的内存渲染器的渲染效果,如图 10-24 所示。

图 10-24 设置渲染效果

📃 Console り	Reg	isters [🛛 Problems	Executable	es 🖉 Termina	l 🖳 Deb	ougger Console 🔋 Memory 🗙	
							1019 1010 📑 🛃 🎿 🗒 🕰	- 8
Monitors	÷	× %	printf_demo	:0x80 <hex></hex>	× printf_den	no <floa< td=""><td>Layout</td><td>></td></floa<>	Layout	>
oprintf_de	emo		Address	0 - 3	4 - 7	8 - B	Find/Replace	
			00000000	∆ 00400020	A 89030000	A D103	Find Next	
			00000010	A D5030000	A D7030000	▲ D903	Traditional Rendering Preferences	
			00000020	▲ 00000000	A 00000000	▲ 0000	Floating Point Rendering Preferences	
			00000030	A DD030000	▲ 00000000	A DF03	Droforopcoc	1
			00000040	▲ E3030000	A E5030000	🛦 E703	Floating Point Rendering Preferences	
			00000050	▲ EB030000	A ED030000	▲ EF03	a contractings Preferences	
			00000060	▲ F3030000	▲ F5030000	▲ 00000	0000 🔥 F7030000	
			00000070	▲ F9030000	A FB030000	A FD030	0000 🔥 FF030000	
			00000080	▲ 01040000	A 03040000	▲ 05040	0000 🔥 07040000	
			00000090	▲ 09040000	▲ 0B040000	▲ 0D040	0000 🔥 0F040000	
			000000A0	▲ 11040000	A 13040000	<mark>▲</mark> 15040	0000 🔥 17040000	
			000000B0	▲ 19040000	A 18040000	▲ 1D040	0000 🔥 1F040000	
			000000C0	▲ 80B588B0	A 04AF00F0	ABF90	023 🔥 03930023	
			000000D0	A 02930023	▲ 01930023	A 00930	0123 🔥 01224FF4	
			000000E0	▲ 16414FF0	▲ 402000F0	A5F81	E4B 🔥 FB600523	
			000000F0	A BB602023	▲ 7B607B68	▲ 013B0	0122 🔥 02FA03F3	
			00000100	▲ 01333B60	▲ F9681848	▲ 00F0A	A8FA 🔥 B9681748	~

 例如 "Preferences > C/C++ > Debug > Traditional Memory Rendering", 勾掉 "Use Global Text Color",可以自定义 Text Color, 如图 10-25 所示。
🗱 Preferences (Filtered)	— C	x c
type filter text ×	Traditional Memory Rendering 🗘 👻	⊲> - 8
 ✓ C/C++ ✓ Debug Traditional Memory Rendering 	Traditional Memory Rendering Use Global Text Color Iext Color: Brighten Alternate Cells	· · · ·
	Background Color:	
	History <u>T</u> rail Levels Show cross reference informationE.g. Variables, Registers Restore <u>D</u> efaults	Apply
? 🎽 🖉 🖲	Apply and Close Ca	ncel

图 10-25 设置内存渲染效果

视图工具栏功能

内存视图工具栏包含如下功能:

● Import ([™]): 引入一组指定的内存值,如图 10-26 所示。

图 10-26 Import Memory

🔯 Import Memory		\times					
Format: Plain Text V							
Restore to address: 0x0							
File name:			Browse	e			
Scroll to restore address							
?	ОК		Cancel				

• Export (¹⁰⁰): 把内存渲染器显示的内存值引出到一个文件中,如图 10-27 所示。

图 10-27 Export Memory

🐝 Export Memory		_		×			
Format: Plain Text V							
Start address: 0x0 End address: 0x0	Length: C)					
File name:	Browse]					
	OK		Canad				
	OK		Cancel				

GMD 软件支持三种引入/引出内存数据的格式: Plain Text、RAW Binary、SRecord。Plain Text 和 RAW Binary 一般更常用。Plain Text 包含 十六进制,一个字节保存两个十六进制字符。RAW Binary 包含二进制值。

10.7.2 内存浏览器视图

内存浏览器(Memory Brower)视图是内存视图的一种备选替代方式, 用于检查和修改进程内存。

建立内存监视器

1. 程序运行暂停后,在内存浏览器视图的文本框中输入想要查看的内存地 址或表达式,选择 "Go",建立一个内存监视器,如图 10-28 所示。

图 10-28 内存浏览器视图

ms 🖉 Terminal	🚺 Memory Browser 🛛	🙀 Debugger Console	
			1010 1010 📑 🖻 🖇
			✓ Go New Tab
	ems	ms Prominal Memory Browser X	ems Prerminal Memory Browser X R Debugger Console

2. 内存浏览器视图中,默认显示一个传统格式的内存渲染器,如图 10-29 所示。

Console	Registers 🖹	Problems 🖉	Terminal	🚺 Memor	ry Browser	×	Debugger Console	- 8
								1019 1010 📑 🛃 🖇
printf_demo								✓ Go New Tab
0x80 - printf_der	mo <tradition< th=""><th>al> X</th><th></th><th></th><th></th><th></th><th></th><th></th></tradition<>	al> X						
0x00000080 00	0000401 00000	403 00000405	00000407	00000409	0000040B	000004	40D	^
0x0000009C 00	000040F 00000	411 0000041	00000415	00000417	00000419	000004	41B	
0x000000B8 00	000041D 00000	41F B088B580	F000AF04	2300F9AB	23009303	230093	302μ.°.	.ð«ù.###
0x000000D4 23	3009301 23019	300 F44F2201	F04F4116	F0002040	4B1EF8A5	230560	0FB##."Oô./	40ð @ .ð ¥¢.K û`.#
0x000000F0 23	32060BB 687B6	07B 22013B01	L F303FA02	603B3301	481868F9	FAA8F@	000 »`#{`{h.;.".u	i.ó.3;`ùh.H.ð¨ú
0x0000010C 48	81768B9 FAA4F	000 48166839	FAA0F000	48152161	FA9CF000	481421	1FF ¹h.H.ð¤ú9h.H.ð	δ úa!.Η.δ.úÿ!.Η
0x00000128 F/	A98F000 48132	100 FA94F000	0302F06F	0202F06F	0102F06F	F00048	80F .ð.ú.!.H.ð.úoð	ნ იð იშ H.ð
0x00000144 4/	A0FFA8B 480F2	100 FA86F000	F000480E	4AØBFAE9	480D2100	FA7EF	000 .ú.J.!.H.ð.ú.⊦	1.ðéú.J.!.H .ð∼ ú
0x00000160 B	F00E7FE 00007	7020 0000703	00007038	00007044	0000705C	000070	06C þç.; p0p8	Dp ∖p lp. .
0x0000017C 00	000707C 00007	708C 000070B	00007088	000070C4	000070C8	AF00B5	580 pp°pp	ÄpÈpμ.⁻
0x00000198 FF	F92F7FF 46182	300 B580BD80	AF00B086	60B960F8	2300607A	E00C61	17B ÿ÷.ÿ.#.F.½.μ.°	°.⁻¢`¹`z`.#{a.à
0x000001B4 69	97B68BA F9934	413 46193000	2040F04F	F8CAF000	3301697B	687B61	17B ºh{i.D.ù.0.F0∂	0@ .ðÊ¢{i.3{a{h
0x000001D0 42	29A697A 687BE	3EE 37184618	BD8046BD	AF00B480	46BDBF00	4770B0	C80 zi.B îÓ{h .F.7 ½A	.½¿½F.¼pG
0x000001EC A	F00B480 46BDB	F00 4770BC80	AF00B480	46BDBF00	4770BC80	AF00B4	480	ХЕ.%рб
0x00000208 46	6BDBF00 47708	C80 AF00B480	46BDBF00	4770BC80	AF00B480	46BDBF	F00 .:%F.%pG	%F.%pG %F
0x00000224 47	770BC80 AF00E	480 46BDBF00	4770BC80	B088B580	60F8AF00	607A60	0B9 .¼pG.´.゙.¿½F.)	φG .μ.° .⁻ø` ¹`z`
0x00000240 68	87B603B 763BB	2DB B2DB6838	6ABB767B	76BBB2DB	B2DB6AFB	6B3B76	6FB ;`{hÛ²;v;hÛ²{\	/≫jÛ²≫vûjÛ²ûv;k ∨

图 10-29 传统格式的内存渲染器

还可以建立一个浮点格式的内存渲染器。选择"View Menu"([§]),

弹出的下拉菜单中选择 "Default Rendering > Floating Point > New Tab", 如图 10-30 和图 10-31 所示,建立一个浮点格式的内存渲染器。

📮 Console	🔐 Registers 🛚	ر Problems	🖉 Terminal	🔋 Memo	ry Browser	· 🗙 🖳 De	ebugger C	Console 5	
								1012 1010 📑	* 8
printf_demo	D				Flo	ating Point		Default Rendering	>
0x80 - printf_	demo <traditiona< th=""><th>l> X</th><th></th><th></th><th>• Tra</th><th>aditional</th><th></th><th>Find/Replace</th><th></th></traditiona<>	l> X			• Tra	aditional		Find/Replace	
0x0000080	00000401 00000	403 000004	05 00000407	00000409	0000040B	0000040D		Find Next	
0x0000009C	0000040F 00000	411 000004	13 00000415	00000417	00000419	0000041B		Clear Expressions	
0x000000B8	0000041D 00000	41F B088B5	80 F000AF04	2300F9AB	23009303	23009302			_
0x000000D4	23009301 23019	300 F44F22	01 F04F4116	F0002040	4B1EF8A5	230560FB	#	.# ."Oô .AOð @ .ð ¥ø.Kû`.:	#
0x000000F0	232060BB 687B6	07B 22013B	01 F303FA02	603B3301	481868F9	FAA8F000	» #{`	{h.;.".ú.ó.3;`ùh.H.ð¨ı	ú
0x0000010C	481768B9 FAA4F	000 481668	39 FAA0F000	48152161	FA9CF000	481421FF	¹h.H.ð	¤ú9h.H.ð úa!.H.ð.úÿ!.H	н
0x00000128	FA98F000 48132	100 FA94F0	00 0302F06F	0202F06F	0102F06F	F000480F	.ð.ú.!	.H.ð.úoðoðH.d	ð
0x00000144	4A0FFA8B 480F2	100 FA86F0	00 F000480E	4A0BFAE9	480D2100	FA7EF000	.ú.J.!	.H.ð.ú.H.ðéú.J.!.H.ð~	ú
0x00000160	BF00E7FE 00007	020 000070	30 00007038	00007044	0000705C	0000706C	þç.¿ p	0p8pDp\p1p.	
0x0000017C	0000707C 00007	08C 000070	BØ 000070B8	000070C4	000070C8	AF00B580	p .p	°ppÄpÈpμ.	-
0x00000198	FF92F7FF 46182	300 B580BD	80 AF00B086	60B960F8	2300607A	E00C617B	ÿ÷.ÿ.#	.F.½.μ.°.¯φ`¹`z`.#{a.	à
0x000001B4	697B68BA F9934	413 461930	00 2040F04F	F8CAF000	3301697B	687B617B	⁰h {i .D	.ù.0.FOð@ .ðÊø{i.3{a{l	h
0x000001D0	429A697A 687BD	3EE 371846	18 BD8046BD	AF00B480	46BDBF00	4770BC80	zi.BîÓ	{h.F.7%F.%.´.`.¿%F.%p	G
0x000001EC	AF00B480 46BDB	F00 4770BC	80 AF00B480	46BDBF00	4770BC80	AF00B480		%F.%pG.´.`.¿%F.%pG.´.`	-
0x00000208	46BDBF00 4770B	C80 AF00B4	80 46BDBF00	4770BC80	AF00B480	46BDBF00	.¿%F.%	pG.1	F
0x00000224	4770BC80 AF00B	480 46BDBF	00 4770BC80	B088B580	60F8AF00	607A60B9	.%pG.í	. ⁻ .¿½F.¼pG.μ.°. ⁻ φ`¹`z	•
0x00000240	687B603B 763BB	2DB B2DB68	3B 6ABB767B	76BBB2DB	B2DB6AFB	6B3B76FB	;`{hÛ²	;v;hÛ²{v»jÛ²»vûjÛ²ûv;	k v

图 10-30 建立浮占格式的内左渲染器

📮 Console	👫 Registers Registers	blems 🖉 Terminal	Memory Browse	er 🔀 🖳 Debugger	r Console	• 🗆
					1019 LOID IO1	°00
printf_dem	o				✓ Go New Ta	ab
0x80 - printf	demo <traditional></traditional>	0x80 - printf_der	no <floating point=""></floating>	X		
0x00000080	Infinity	Infinity	Infinity	Infinity	Infinity	^
0x00000094	Infinity	Infinity	Infinity	Infinity	Infinity	
0x000000A8	Infinity	Infinity	Infinity	Infinity	Infinity	
0x000000BC	Infinity	-9.946888E-10	-1.593026E29	6.991763E-18	6.970025E-18	
0x000000D0	6.970024E-18	6.970023E-18	7.024232E-18	-6.564301E31	-2.565683E29	
0x000000E4	-1.586123E29	1.041834E7	7.230481E-18	8.694101E-18	4.748375E24	
0x00000F8	1.751400E-18	-1.045626E31	5.395651E19	1.560679E5	-4.385868E35	
0x0000010C	1.550429E5	-4.282022E35	1.540169E5	-4.178176E35	1.527095E5	
0x00000120	-4.074330E35	1.516880E5	-3.970485E35	1.506600E5	-3.866639E35	
0x00000134	3.847957E-37	9.619893E-38	Infinity	-1.588048E29	2.358947E6	
0x00000148	1.465640E5	-3.503178E35	-1.588048E29	2.293434E6	1.445160E5	
0x0000015C	-3.309278E35	-5.035399E-1	Infinity	Infinity	Infinity	
0x00000170	Infinity	Infinity	Infinity	Infinity	Infinity	
0x00000184	Infinity	Infinity	Infinity	Infinity	-1.170601E-10	
0x00000198	Infinity	9.736750E3	-9.591895E-7	-1.170425E-10	1.068636E20	
0x000001AC	6.959324E-18	-4.046201E19	1.899593E25	-9.558104E34	9.804000E3	~
<						>

图 10-31 浮点格式的内存渲染器

3. 右键内存渲染器的任意位置,弹出的下拉菜单中选择"Endian",可以 修改数据显示的字节顺序或可寻址大小,如图 10-32 所示。

图 10-32 设置字节顺序

📮 Console	nin Regi	isters 🖹 Problems 🖉 Terminal	🚺 Mem	nory Browser $ imes$	🛃 Del	bugger Console 🗖	
						101g 1010 📑 🖻	8
printf_demo v Go New Tab							
0x80 - printf_demo <traditional> X 0x80 - printf_demo <floating point=""></floating></traditional>							
0x0000080	0 60	Add Watchpoint (C/C++)		9 0000040B 000	00040D		^
0x0000009C	0	Find and Bankar	CH L	.7 00000419 00	00041B		
0x000000B8	0	Find and Replace	Ctri+F	B 23009303 230	009302	µ.°.⁻.ð«ù.###	
0x000000D4	2	Panes	>	0 4B1EF8A5 230	0560FB	##."Oô.AOð@ .ð¥ø.Kû`.#	
0x000000F0	2	Endian	>	Big	900	»` #{`{h.;.".ú.ó.3;`ùh.H.ðïú	
0x0000010C	4	Text	>	• Little	LEE	¹h.H.ð¤ú9h.H.ð úa!.H.ð.úÿ!.H	
0x00000128	F.	Coll Size	(- Little	30F	.ð.ú.!.H.ð.úoðoðoðH.ð	
0x00000144	4.	Cell Size		9 480D2100 FA	7EF000	.ú.J.!.H.ð.ú.H.ðéú.J.!.H.ð~ú	
0x00000160	В	Radix	>	4 0000705C 000	00706C	þç.¿ p0p8pDp∖p1p	
0x0000017C	0	Columns	>	4 000070C8 AF	00B580	pp°p,pÄpÈpμ.⁻	
0x00000198	F	Update Mode	>	8 2300607A E0	0C617B	ÿ÷.ÿ.#.F.½.µ.°.⁻¢`¹`z`.#{a.à	
0x000001B4	6			0 3301697B 687	7B617B	≌h{i.D.ù.0.FOð@ .ðÊø{i.3{a{h	
0x000001D0	4	Сору	>	0 46BDBF00 47	70BC80	zi.BîÓ{h.F.7½F.½.´.`.¿½F.¼pG	
0x000001EC	Α	Reset To Base Address		0 4770BC80 AF	00B480		
0x00000208	4	Go to Address		0 AF00B480 46	BDBF00	.¿%F.%pG.´.¯.¿%F.%pG.´.¯.¿%F	
0x00000224	4	Refresh		0 60F8AF00 60	7A60B9	.¼pG.´.¯.¿½F.¼pG.μ.°.¯φ`¹`z`	
0x00000240	68788	038 70385205 BZUB6838 0A55707	5 76555ZL	B B2DB6AFB 6B	3B76FB	;`{hÛ²;v;hÛ²{v»jÛ²»vûjÛ²ûv;k	~

4. "View Menu"([§])弹出的下拉菜单中选择"Find/Replace…",可以 查找和替换选定的内存监视的指定数据,如图 10-33 所示。

🐳 Find / Replace Me	emory				
Find: Replace With: Direction	Range]
Forward Backward	Start address: End address:	0x00000000 0xFFFFFFF		~	
Format ASCII String Hexadecimal Octal Binary Decimal Byte Sequence	Options Wrap Search	h itive			
?			Find Replace/Find Replace All	Find All Replace Close	

图 10-33 Find/Replace

10.7.3 寄存器视图

寄存器(Registers)视图显示寄存器位域级别的 CPU 信息,可以直接 在单元格中修改寄存器或位域的值,如图 10-34 所示。

图 10-34 寄存器视图

📮 Console 🚻 Registers 🗙 🖹 Problems	; 🖉 Terminal 🔋 Memory Browser 🛛 🙀 De	bugger Console 🔋 Memory
		🏝 🎫 🕒 📑 😫
Name	Value	Description ^
✓ ₩ General Registers		General Purpose and FPU Register Group
1919 rO	10	
1919 r1	0	
10101 r2	574619648	
1910 r3	301989888	
1010 0101 r4	952682120	
1000 r5	838334808	
10101 r6	3143087189	
1010 r7	536887264	
1919 r8	2374924313	
1000 r9	809339810	
1919 r10	3705169638	
1001 r11	1429610927	
10101 r12	21	
10101 sp	0x20003fd0	
1010 lr	353	v
<		>

10.7.4 变量视图

变量(Variables)视图显示了当前作用域中所有变量的名称和值,如 图 10-35 所示。当程序暂停运行时,可以查看和监视变量值的变化。

(x)= Variables	🗙 💁 Break	cpoints 🛭 Expressions 🚡 Peripherals	🖾 🎫 🕞 📬 😤 📮 🗖
Name	Туре	Value	
> 🔶 ptr	char *	0x7020 <_fini+4> "Hello world!"	
(×)= i	int	5	
(×)= bs	unsigned int	32	
(×)= mi	int	-2147483647	
<			>

图 10-35 变量视图

10.7.5 表达式视图

表达式(Expressions)视图允许添加和监视自定义表达式的值,如图 10-36 所示,这对于计算复杂表达式的值或在调试过程中测试代码片段非常 有用。

图 10-36 表达式视图

(x)= Variables 💁 Breakpoints 🖉	🙀 Expressions 🗙 🔀 Peripherals	🗄 🎿 🕞 🕂 🗙 💥 📬 🖻 🖇 🗖 🗖
Expression	Туре	Value
printf_demo	int (void)	{int (void)} 0xc0 <printf_demo></printf_demo>
uart_init	void (UART_TypeDef *, uint32_t, uint3	{void (UART_TypeDef *, uint32_t, uint3
> 🔹 ptr	char *	0x7020 <_fini+4> "Hello world!"
(×)= mi	int	-2147483647
(×)= i	int	5
(×)= bs	unsigned int	32
🐈 Add new expression		
	·	

10.7.6 断点视图

断点(Breakpoints)视图列出了软件工程中所有已设置的断点,用户可以在这里管理断点,包括启用、禁用、添加、删除断点,以及为断点设置条件,如图 10-37 所示。

图 10-37 断点视图

(x)= Variables	● Breakpoints ×	🕵 Expressions	x	*	æ	÷	`Ø	Ŧ	E	\$₽}	00	
🗹 💣 (fu	nction: main] [type: T	emporary]										
🗹 🔎 prin	ntf_demo.c [line: 29]											
🗹 🔎 prin	ntf_demo.c [line: 38]											
🗹 🔎 prin	ntf_demo.c [line: 40]											
🗹 🔎 prin	ntf_demo.c [line: 45]											
🗹 🔎 prin	ntf_demo.c [line: 46]											
🗹 🔎 prin	ntf_demo.c [line: 47]											
🗹 🔎 prin	ntf_demo.c [line: 48]											
🗹 🔎 prin	ntf_demo.c [line: 49]											
🖂 🔎 prin	ntf_demo.c [line: 50]											
N	P 1 6 4	at a la star										

No details to display for the current selection.

10.7.7 反汇编视图

反汇编(Disassembly)视图显示了高级语言所对应的汇编代码,用于 在调试过程中查找指令问题,如图 10-38 示。

图 10-38 反汇编视图

0	(x)= Variables	Breakpoints	🛠 Expressions	🛄 Disassem	bly $ imes$						
				ł	Enter lo	cation here	~ 8	🟠 😫 🛯	3 🖂	<u>ح</u>	000
	0000015c:	bl 0x6	5c <printf></printf>								^
	56	<pre>while(1);</pre>									
۵	00000160:	b.n 0x1	60 <printf_demo< td=""><td>+160></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></printf_demo<>	+160>							
	00000162:	nop									
	00000164:	strb r0,	[r4, #0]								
	00000166:	movs r0,	r0								
	00000168:	strb r0,	[r6, #0]								
	0000016a:	movs r0,	r0								
	0000016c:	strb r0,	[r7, #0]							- 1	
	0000016e:	movs r0,	r0								
	00000170:	strb r4,	[r0, #1]								
	00000172:	movs r0,	r0								
	00000174:	strb r4,	[r3, #1]								
	00000176:	movs r0,	r0								
	00000178:	strb r4,	[r5, #1]								
	0000017a:	movs r0,	r0								
	0000017c:	strb r4,	[r7, #1]								
	0000017e:	movs r0,	r0								
	00000180:	strb r4,	[r1, #2]								_
	00000182:	movs r0,	r0								
	00000184:	strb r0,	[r6, #2]								
	00000186:	movs r0,	r0								~
		<								>	

10.7.8 控制台视图

控制台(Console)视图显示进程输出过程,如图 10-39 所示,例如调试过程输出。

图 10-39 控制台视图

📮 Console 🗙 🕮 Registers 🔝 Problems 🚺 Executables 🖉 Terminal	🛃 Debugger Console				
	🔳 🗙 💥 🗟 🚮 🖬 🖉 🖅 🛃	• 📬 •			
cm3_demo Debug jlink [GDB SEGGER J-Link Debugging]					
Reading register (CONTROL = 0x 0)		^			
Reading 720 bytes @ address 0x00000000					
Reading 688 bytes @ address 0x000002D0					
Starting target CPU					
Debugger requested to halt target					
Target halted (PC = 0x00000160)					
Reading all registers					
Read 4 bytes @ address 0x00000160 (Data = 0xBF00E7FE)					
Read 4 bytes @ address 0x00000160 (Data = 0xBF00E7FE)					
Reading 720 bytes @ address 0x00000000					
Reading register (MSP = 0x20003FD0)					
Reading register (PSP = 0xD673C7BC)					
Reading 64 bytes @ address 0x20003FC0					
Read 4 bytes @ address 0x0000019C (Data = 0x46182300)					
Reading 8 bytes @ address 0x0000/020					
Reading 8 bytes @ address 0x0000/028					
Reading register (PRIMASK = 0X 0)					
Reading register (DASEPRI = 0X 0)					
Reading register (CONTROL - 0x 0)					
Reading 8 bytes @ address 0x00007020					
Peading 8 bytes @ address 0x00007020					
Reading 8 bytes @ address 0x00007020					
Reading 8 bytes @ address 0x00007020					
		~			
<		>			

11 QEMU 仿真调试

11.1 QEMU 工具

GMD 软件集成了 Gowin QEMU 软件工具,是高云在 QEMU 基础版本 上针对高云的 MCU 产品特性进行的再次开发。QEMU 是一个托管的虚拟 机,提供多种硬件和外设模型,可以直接在线模拟仿真功能,不需要一个真 实的物理平台。

GMD 2025.01 版本的 QEMU 工具已支持 Gowin_EMPU(GW1NS-4C) 和 Gowin_PicoRV32, 各自分别已支持的外设功能如表 11-1 所示。

MCU 产品	已支持外设	功能说明
Gowin_EMPU(GW1NS-4C)	UART	输入和输出、中断处理
	GPIO	输入和输出
	SysTick	计数、中断处理
	Timer	计数、中断处理
Gowin_PicoRV32	UART	输入和输出
	GPIO	输入和输出

表 11-1 已支持的外设功能

11.2 QEMU 使用方法

Arm MCU 和 RISC-V MCU 进入 QEMU 的方法有以下几种:

- 选定当前工程,右键选择 "Debug As > Debug Configurations…"
- 选择工具栏 "Debug > Debug Configurations…"(**)
- 选择菜单栏 "Run > Debug Configurations…"

Arm MCU 和 RISC-V MCU 分别使用不同的 QEMU 工具,设置方法如下所述。

11.2.1 Arm MCU QEMU 设置

例如 Gowin_EMPU(GW1NS-4C),选择 GDB QEMU arm Debugging,如图 11-1 所示。

图 11-1 GDB QEMU arm Debugging

🗱 Debug Configurations					×
Create, manage, and run configurations				Ŕ	5.
Image: Construct of the second sec	Name: cm3_demo debug qemu Main Debugger Startup Project: cm3_demo C/C++ Application: Debug\cm3_demo.elf Build (if required) before launching Build Configuration: Select Automati C Enable auto build Use workspace settings	Source Common S SVD Path	ild ace Settings	Browse	
< >> Filter matched 22 of 24 items			Re <u>v</u> ert	Apply	
?			<u>D</u> ebug	Close	•

1. Main 视图

Main 视图中的 QEMU 设置如图 11-2 所示。

图 11-2 Main 视图

📄 Main 🏇 Debugger	🕨 🕨 Startup 🧤 Source 🔲 Co	mmon 🚡 SVD Path		
<u>P</u> roject:				
cm3_demo				<u>B</u> rowse
C/C++ Application:				
Debug\cm3_demo.elf				
		<u>V</u> ariables	Searc <u>h</u> Project	B <u>r</u> owse
Build (if required) bef	fore launching			
Build Configuration:	Select Automatically			~
◯ Enable auto build		⊖ Disable auto bui	d	
● Use workspace set	tings	Configure Workspa	ce Settings	

2. Debugger 视图

Debugger 视图中的 QEMU 设置如图 11-3 和表 11-2 所示。

🗎 Main 🏇 Debugger 🕨 Startup 🤤 Source 🔲 Common 🛃 SVD Path						
QEMU Setup		^				
Start QEMU locally						
Executable path: \${qemu_arm_path}/\${qemu_arm_executable} Browse Variable	les					
□ Prefer the xpacks/.bin path						
Actual executable: E:\GMD\\toolchain\gowin\qemu_arm\bin/qemu-system-arm.exe						
(to change it use the <u>global</u> or <u>workspace</u> preferences pages or the <u>project</u> properties p	page)					
QEMU machine name: gowin_cm3						
QEMU CPU name: cortex-m3						
GDB port: 1234						
More options: -serial stdio -nodefaults -S						
	~					
☐ Enable Arm semihosting ☐ Do not open graphic windows	Enable Arm semihosting					
Command line: cm3_demo						
Allocate console for QEMU						
Delay (seconds): 0						
GDB Client Setup						
Everytable name: Stoross prefix)adhStoross suffix) Browse Variab	les					
Actual executable: arm-none-eabi-gdb						
Other options:						
Commands: set mem inaccessible-by-default off	\sim					
	~					
Remote Target						
Host name or IP address: localhost						
Port number: 1234						
Force thread list update on suspend		U				
<	>	Ť				

图 11-3 Debugger 视图

表 11-2 Debugger 视图

选项	设置
Start QEMU locally	\checkmark
QEMU machine name	如表 11-3 所示
QEMU CPU name	如表 11-3 所示
GDB port	1234
More options	-serial stdio -nodefaults -S
Enable Arm semihosting	\checkmark
Do not open graphic windows	\checkmark
Allocate console for QEMU	\checkmark
Delay (second)	0
Commands	set mem inaccessible-by-default off

表 11-3 Arm MCU QEMU

Arm MCU	Machine name	CPU name
Gowin_EMPU(GW1NS-4C)	gowin_cm3	cortex-m3

3. Startup 视图

Startup 视图中的 QEMU 设置如图 11-4 和表 11-4 所示。

图 11-4 Startup 视图

📄 Main 🕸 Debugger 🕨 Startup 🦃 Source 🔲 Common 🛃 SVD Path		
Initialization Commands		^
Initial Reset		
		^
		× .
Load Symbols and Executable		
✓ Load symbols		
• Use project binary: cm3_demo.elf		
⊖ Use file:	Workspace	File System
Symbols offset (hex):		
✓ Load executable		
 Use project binary: cm3_demo.elf 		
⊖Use file:	Workspace	File System
Executable offset (nex).		
Runtime Options		
Debug in RAM		
Run/Restart Commands		
Pre-run/Restart reset (always executed at Restart)		
		<u>^</u>
		×
Set program counter at (hex):		
☑ Set breakpoint at: main		
☑ Continue		

表 11-4 Startup 视图

选项	设置
Initial Reset.	×
Load symbols	\checkmark
Load executable	\checkmark
Debug in RAM	×
Pre-run/Restart reset	\checkmark
Set program counter at (hex)	-
Set breakpoint at	main

选项	设置
Continue	\checkmark

11.2.2 RISC-V MCU QEMU

例如 Gowin_PicoRV32,选择 GDB QEMU riscv32 Debugging,如图 11-5 所示。

图 11-5 GDB QEMU riscv32 Debugging

🗱 Debug Configurations			_		×
Create, manage, and run configurations				K	j.
Image: Second	Name: picorv32_qemu_demo debug qem Main Debugger Startup S Project: picorv32_qemu_demo C/C++ Application: Debug\picorv32_qemu_demo.elf Build (if required) before launching Build Configuration: Select Automatica O Enable auto build © Use workspace settings	nu Source Common P SN Variables S ally O Disable auto b Configure Works	VD Path	Browse	
0			<u>D</u> ebug	Clos	e

1. Main 视图

Main 视图中的 QEMU 设置如图 11-6 所示。

图 11-6 Main 视图

📄 Main 🕸 Debugger 🕨 Startup 🦆 Source	🔲 Common 🔀 SVD Path	
Project:		
picorv32_qemu_demo		<u>B</u> rowse
C/C++ Application:		
Debug\picorv32_qemu_demo.elf		
	<u>V</u> ariables Searc <u>h</u> Project	B <u>r</u> owse
Build (if required) before launching		
Build Configuration: Select Automatically		~
◯ Enable auto build	⊖ Disable auto build	
 Use workspace settings 	Configure Workspace Settings	

2. Debugger 视图

Debugger 视图中的 **QEMU** 设置如图 **11-7** 和表 **11-5** 所示。

📄 Main 🕸 Debugger 🛛	▶ Startup 🦆 Source 🔲 Common 🔓 SVD Path	
QEMU Setup		^
Start QEMU locally		
Executable path:	{qemu_riscv32_path}/\${qemu_riscv32_executable}	Browse
Prefer the xpacks/.b	in path	
Actual executable:	E:\GMD\\toolchain\gowin\qemu_riscv\bin/qemu-s	ystem-riscv32.exe
	(to change it use the <u>global</u> or <u>workspace</u> prefere	nces pages or the <u>project</u> prope
QEMU machine name:	gowin_pico	
QEMU CPU name:	gowin-picorv32	
GDB port:	1234	
More options:	-serial stdio -nodefaults -S	
Enable Arm semihos	sting 🖸 Do not oper	n graphic windows
Command line:	picorv32_qemu_demo	
Allocate console for	QEMU	
Delay (seconds):	0	
GDB Client Setup		
Executable name: \${c	ross_prefix}gdb\${cross_suffix}	Browse
Actual executable: riscv-none-elf-gdb		
Other options:		
Commands:inaccessible_by_default off		
sec	men maccessible-by-default on	
Remote Target		
Host name or IP addre	ess: localhost	
Port number:	1234	
Port number:	1204	
Force thread list upda	te on suspend	v
<		>

图 11-7 Debugger 视图

表 11-5 Debugger 视图

选项	设置
Start QEMU locally	\checkmark
QEMU machine name	如表 11-6 所示
QEMU CPU name	如表 11-6 所示
GDB port	1234
More options	-serial stdio -nodefaults -S
Enable Arm semihosting	\checkmark
Do not open graphic windows	\checkmark
Allocate console for QEMU	\checkmark
Delay (second)	0

选项	设置
Commands	set mem inaccessible-by-default off

表 11-6 RISC-V MCU QEMU

RISC-V MCU	Machine name	CPU name
Gowin_PicoRV32	gowin_pico	gowin-picorv32

3. Startup 视图

Startup 视图中的 QEMU 设置如图 11-8 和表 11-7 所示。

图 11-8 Startup 视图

Main 🎋 Debugger 🕨 Startup 🦃 Source 🔲 Common 🔀 SVD Path
Initialization Commands
Initial Reset
Load Symbols and Executable
∠ Load symbols
● Use project binary: picorv32_qemu_demo.elf
O Use file: Workspace File System
Symbols offset (hex):
Load executable
Use project binary: picorv32_qemu_demo.elf
O Use file: Workspace File System
Executable offset (hex):
Runtime Options
🗹 Debug in RAM
Run/Restart Commands
✓ Pre-run/Restart reset (always executed at Restart)
Set program counter at (hex):
Set breakpoint at: main
└ Continue

表 11-7 Startup 视图

选项	设置
Initial Reset.	×
Load symbols	\checkmark

选项	设置
Load executable	\checkmark
Debug in RAM	\checkmark
Pre-run/Restart reset	\checkmark
Set program counter at (hex)	-
Set breakpoint at	main
Continue	\checkmark

11.3 仿真调试工程

例如 Gowin_EMPU(GW1NS-4C)。

完成 QEMU 设置后,单击"Debug",启动调试视图,运行工程,如图 11-9 所示。

图 11-9 启动 QEMU 仿真调试



12Code Coverage 功能

12.1 关于 Code Coverage 功能

GMD 软件支持 Code Coverage 功能, Code Coverage 功能是借助于 GCC 编译器提供的 Gcov 工具来查看指定源码文件的代码覆盖率,可以帮助开发人员确定他们的测试用例是否重复,是否覆盖了被测代码的所有分支 和路径。

Gcov 是一个测试 C/C++代码覆盖率的工具,与 GCC 互相配合,共同 实现对 C/C++文件的语句覆盖、功能函数覆盖和分支覆盖测试。

GMD 软件中,通过带特定-coverage 编译选项编译指定源码文件,一 是可以在实际开发板上运行,并配合 semihost 功能则可以收集需要的 coverage 文件(gcda/gcno 文件),或者二是可以直接在 QEMU 上模拟运 行。在 Gcov 工具的配合下,以图形化的方式展示。

- .gcno 文件是使用 GCC 编译器的-ftest-coverage 选项编译源代码时生成的,包含了重构基本块图和为块分配源代码行号的信息。
- .gcda 文件是在使用 GCC 编译器的-fprofile-arcs 选项编译的目标文件运行时生成的。每个使用该选项编译的目标文件都会生成一个单独的.gcda 文件,包含了弧转移计数、值分布计数以及一些摘要信息。

一般情况下直接使用-coverage 选项就可以指示编译器产生上述文件, 注意.gcda 文件是在运行时产生的。

使用 Code Coverage 功能时,建议使用-O0 优化选项编译,这样覆盖的信息才会尽可能的准确。

12.2 使用 Code Coverage 功能

例如在 QEMU 平台上模拟运行 Gowin_EMPU(GW1NS-4C)。

12.2.1 创建工程

创建一个用于 Code Coverage 功能的软件测试工程,相比于常规使用

的软件工程,该工程中需要有几个特殊的代码段:

- 添加额外的 gcov.c 文件
- 添加并调用 gcov_init 函数,初始化 Gcov 功能
- 调用 gcov_collect 函数,收集相关 Profiling 信息
- 配合 Code Coverage 功能,修改 startup.S 汇编文件、flash.ld 链接文件

具体实现请参照相关参考设计,例如 Gowin_EMPU(GW1NS-4C)的相关参考设计: cm3_gcov_demo

12.2.2 设置相关选项

设置编译选项

工程创建后,对想要进行代码分析的文件或文件夹设置一个额外的编译选项-coverage,然后编译工程。

- 可以指定整个工程的所有源代码文件;
- 或者指定某个/某些源代码文件。
- 1. 指定所有源代码

选定当前工程,右键选择 "Properties > C/C++ Build > Settings > Tool Settings > GNU Arm Cross C Compiler > Miscellaneous > Other compiler flags",添加 "-coverage",如图 12-1 所示。



图 12-1 指定所有源代码

2. 指定某个/某些源代码

选定某个文件,右键选择 "Properties > C/C++ Build > Settings > GNU Arm Cross C Compiler > Other compiler flags",添加 "-coverage", 如图 12-2 所示。

🛞 Tool Settings 🧻 Container Settings 🎤 B	uild Steps
✓ Signature GNU Arm Cross C Compiler	nerate assembler listing (-Wa,-adhIns="\$@.lst") ve temporary files (save-temps Use with caution!) rbose (-v) • compiler flags -coverage

图 12-2 指定某个源代码

设置 Linux Tools Path

在使用 Code Coverage 功能的过程中,需要调用 Parse HexDump Data 功能,因不同用户环境的差异性,建议先在"Linux Tools Path"中指定编译工具链。

Windows 系统下,手动指定工具链路径及前缀信息,例如 "\${eclipse_home}\toolchain\gowin\gcc_arm\bin"和"arm-none-eabi-", 如图 12-3 所示。

Linux 系统下,使用系统默认环境路径即可,即"Use the System Environment PATH",如图 12-4 所示。

选定当前工程,右键选择 "Properties > Linux Tools Path",指定编译工具链及其路径。

Properties for cm3_gcov_d	emo				Х
type filter text	Linux To	ols Path	4	• =	₩ 00
 Resource Builders C/C++ Build 	◯ Use the ● Preper	e System Environment PATH nd string to PATH			
> C/C++ General	Options:	Custom			~
MCU		{eclipse_home}\toolchain\gowin\gc	c_arm∖bin		
Project Natures	Prefix:	arm-none-eabi-			
Project References Run/Debug Settings > Task Repository Task Tags > Validation WikiText		Rectore	Defaulte	400	
?		Apply and Cl	ose	Cancel	

图 12-3 设置 Linux Tools Path

图 12-4 设置 Linux Tools Path

	Properties for cm3_gcov_demo		8
type filter text	Linux Tools Path	• => •	000
 Resource Builders C/C++ Build C/C++ General Linux Tools Path MCU Project Natures Project References Run/Debug Settings Task Repository Task Tags Validation WikiText 	Use the System Environment PATH Prepend string to PATH Options: Custom Prefix: Restore Defaults	Αρρίγ	
?	Cancel	ply and Clo	se

12.2.3 编译工程

在编译通过的工程的 Debug 文件夹中,可以看到已经生成了几个.gcno 文件,如图 12-5 所示。

图 12-5 编译工程



12.2.4 运行工程及使用 Code Coverage 功能

工程编译完成后,可以运行或调试工程,可以选择 QEMU 模拟运行, 也可以调试实际的开发板。例如使用 QEMU 模拟运行,在 GMD 软件的控 制台中可以看到 Profiling 信息输出。如果是在开发板上调试,则是在串口 输出中可以找到 Profiling 信息输出。

参照如下步骤运行工程并使用 Code Coverage 功能:

 "Debug As > Debug Configurations… > GDB QEMU arm Debugging",在 QEMU 中启动调试运行,运行工程并在控制台中查看 Profiling 信息,如图 12-6 所示。



图 12-6 运行工程并查看 Profiling 信息

2. 输出的 Profiling 信息需要解析后 GMD 软件才可以正确读取,在控制台 内任意位置右键,在弹出的下拉菜单中选择 "Select All",选中所有的 输出信息,如图 12-7 所示。

E Console X 1919 Regis	tare	🔍 Problems 🕥 Evecutabl	as 🖪 Dabuggar	Console	Memori			-	-
	sters		es 📲 Debugger	Console			_		-
cm3_gcov_demo debug q 0c0000000b4fbc513b1dab	emi 958	ı [GDB QEMU arm Debuggin 988330a40000a101	9]		(% B.		₽ •		•
080000000000000000000000000000000000000	000	300000010c000000							
7010416053aeb7f1b8833	0a4	0000a10108000000							
000000000000000000000000000000000000000	001	0c00000068431c65							
206ecec80T531a510000a	101	9000000020000000							
2_0000000200000000000000000000000000000	000. 000	20000000000000000							
00000000000000000000000000000000000000	000	-101000000000000000							
010f000000000000200d0	000	00000000e1010000							
00000006c040000000	- b	C-4	Cul. V						
e0010000000000000010f	8	Cut	Ctri+X						
00000000e10100000000		Сору	Ctrl+C						
000000010c000000a060	Ē	Paste	Ctrl+V						
0000a101100000007304		Salact All	Ctel 1 A						
000000000000000000000000000000000000000		Select All	Ctri+A						
d>0507180000000000000000000000000000000000		Find/Replace							
d30c426ee8e95ddb0000		E IN .	or law						
000000000000000000000000000000000000000		Find Next	Ctrl+K						
b88330a40000a1010800		Find Previous	Ctrl+Shift+K						
00000010c00000de99		Open Link							
0000a101100000000000		Class.							
00000000000000000000000000000000000000	Ξĸ	Clear							
b88330a40000a1010800	86	Remove All Terminated							
000000010c000000b7bc		T 1 1 101 1 101							
000031011000000000000		Terminate/Disconnect All							
e8e95ddb0000001010800	۲	GitHub	>						
00000010c000005bd7	6	Parse HexDump Data							
0000a101080000000000	100	·							
		Scroll Lock							
CREATE: D:\GMD_2025_	R	Word Wrap		lg\src/p	roject/pri	.ntf/gcov.gcda			i.
Dump coverage data f		Preferences							U
<	-			_				>	Ŧ
•									

3. 然后在控制台中任意位置再次右键,选择"Parse HexDump Data",如图 12-8 所示。

图 12-7 Select All

	0				
Console X Mili Registers M Proble	ms 🕖 Executables	R Debugger Console	Memory		- 0
		🔳 💥	🔆 🔆 🗟 🗟 🖓	🖶 🗶 🛃 🖬	3 🕶 📑 👻
cm3_gcov_demo debug qemu [GDB QEM	U arm Debugging]				
0c000000b4fbc513b1dab958b88330a400	00a101				~
08000000000000000000000000000000000000	000000				
7010416053aeb7f1b88330a40000a10108	000000				
000000000000000000000000000000068	431c65				
2b6ecec80†531a510000a101900000002c	000000				
0000000200000000000000020000000	000000				
200000000000000000000000000000000000000	000000				
000000000000000000000000000000000000000	000000				
00000006c040000000000006e04000000	000000				
e001000000000000010f00000000000	Cut	Ctrl+X			
00000000e1010000000000002b0000	Conv	Ct-LLC			
000000010c000000a0601914756e4e	сору	Cuite			
0000a101100000073040000000000	Paste	Ctrl+V			
00000000000000000000000000000000000000	Select All	Ctrl+A			
91857180000001011000000000000000000000000					
d30c426ee8e95ddb00000000000000000000000000000000000	Find/Replace				
00000000000000010c000008b2a74	Find Next	Ctrl+K			
b88330a40000a10108000000000000	Find Desidence	Carl Chiffe / K			
00000010c00000de991f7beaf5ec	Find Previous	Ctri+Shirt+K			
0000a10110000000000000000000000	Open Link				
00000000000000010c000000fb22f6	Clear				
b88330a40000a10108000000160000					
000000010c00000b7bc7c709df3b5	Remove All Termin	ated			
000000000000000000000000000000000000000	Terminate/Disconn	ect All			
e8e95ddb0000a101080000000000000	GitHub		>		
000000010c000005bd73f2b05906b					
0000a1010800000000000000000000	Parse HexDump Da	ita			
	Scroll Lock	Parse HexDump Data	11 1 151		
CREATE: D: \GMD_2025_Workspace \	Mand Mass		ect/printf/gc	ov.gcda	
Dump coverage data finish	word wrap				
	Preferences				~
<					>

图 12-8 Parse HexDump Data

4. 此时 GMD 软件会对输出的文件进行分析,并将结果分别保存在对应的 文件中,如图 12-9 所示。

图 12-9 解析 Profiling 信息并生成对应文件

Parse and Generate next/ump rinish	×
The generated files are as follows : Dr/GMD_2025_workspace/workspace_cm3_h/cm3_gcov_demo/Debug/src/library/libraries/drivers/src/gw1ns4c_i2c.gcda Dr/GMD_2025_workspace/workspace_cm3_h/cm3_gcov_demo/Debug/src/library/libraries/drivers/src/gw1ns4c_iart.gcda Dr/GMD_2025_workspace/workspace_cm3_h/cm3_gcov_demo/Debug/src/library/libraries/drivers/src/gw1ns4c_uart.gcda Dr/GMD_2025_workspace/workspace_cm3_h/cm3_gcov_demo/Debug/src/library/libraries/drivers/src/gw1ns4c_uart.gcda Dr/GMD_2025_workspace/workspace_cm3_h/cm3_gcov_demo/Debug/src/library/libraries/drivers/src/gw1ns4c_uart.gcda Dr/GMD_2025_workspace/workspace_cm3_h/cm3_gcov_demo/Debug/src/library/libraries/drivers/src/gw1ns4c_misc.gcda Dr/GMD_2025_workspace/workspace_cm3_h/cm3_gcov_demo/Debug/src/library/libraries/drivers/src/gw1ns4c_rtc.gcda Dr/GMD_2025_workspace/workspace_cm3_h/cm3_gcov_demo/Debug/src/library/libraries/drivers/src/gw1ns4c_rtc.gcda Dr/GMD_2025_workspace/workspace_cm3_h/cm3_gcov_demo/Debug/src/library/libraries/drivers/src/gw1ns4c_rtc.gcda Dr/GMD_2025_workspace/workspace_cm3_h/cm3_gcov_demo/Debug/src/library/libraries/drivers/src/gw1ns4c_rtc.gcda Dr/GMD_2025_workspace/workspace_cm3_h/cm3_gcov_demo/Debug/src/library/libraries/drivers/src/gw1ns4c_imer.gcda Dr/GMD_2025_workspace/workspace_cm3_h/cm3_gcov_demo/Debug/src/library/libraries/drivers/src/gw1ns4c_timer.gcda Dr/GMD_2025_workspace/workspace_cm3_h/cm3_gcov_demo/Debug/src/library/libraries/drivers/src/gw1ns4c_timer.gcda Dr/GMD_2025_workspace/workspace_cm3_h/cm3_gcov_demo/Debug/src/library/libraries/drivers/src/gw1ns4c_timer.gcda Dr/GMD_2025_workspace/workspace_cm3_h/cm3_gcov_demo/Debug/src/library/libraries/drivers/src/gw1ns4c_timer.gcda Dr/GMD_2025_workspace/workspace_cm3_h/cm3_gcov_demo/Debug/src/library/libraries/drivers/src/gw1ns4c_grda_ Dr/GMD_2025_workspace/workspace_cm3_h/cm3_gcov_demo/Debug/src/library/libraries/drivers/src/gw1ns4c_grda_ Dr/GMD_2025_workspace/workspace_cm3_h/cm3_gcov_demo/Debug/src/library/libraries/drivers/src/gw1ns4c_grda_ Dr/GMD_2025_workspace/w	
OK	~

5. 再次查看工程的 Debug 文件夹,可以看到产生了对应的.gcda 文件,如 图 12-10 所示。

图 12-10 查看生成的对应文件



6. 双击某个想要查看 Code Coverage 的.gcda 文件, 打开 Gcov 工具, 如 图 12-11 所示。

图 12-11 打开.gcda 文件

₩ Gcov - Open coverage results	×
Binary File Please enter here the binary file which produced the coverage data.	
Coverage result	
 Show coverage details for gcov.c only. Show coverage for the whole selected binary file 	
OK Cancel	

7. 然后可以看到对应用程序的分析结果,在结果中显示了某个文件或某个 方法在程序运行过程中是否执行到,以及代码执行覆盖比等数据。双击 Gcov中的某一行,GMD软件会自动打开对应的文件,并对文件中的代 码着色,绿色表示在程序运行过程中有被执行到,红色表示在程序运行 过程中没有被执行到,如图 12-12 所示。

开发人员可以参考 Gcov 的结果,并对代码做出相应的优化。

图 12-12 Gcov

🗱 workspace_cm3_h - cm3_gcov_demo/src/pr	oject/printf/main.c - GMD)						_		×	<
<u>F</u> ile <u>E</u> dit <u>S</u> ource Refac <u>t</u> or <u>N</u> avigate Se <u>a</u> r	rch <u>P</u> roject <u>R</u> un <u>G</u> owi	in <u>W</u> indow <u>H</u>	<u>H</u> elp								
📑 👻 🔚 🐚 🛛 👻 🔨 🗸 🔛 🔜	🔌 🕹 👭 👩 🕶 🚳	- 🖻 - 🞯	- * - 0 - (} - 9 - 10	📁 🛷 🕶 📝 😜 🗟	T II	図・目・セ	- c> 🔶 ·	- <>	- -	1
								Q	: 😭	6	*
	D. tar						V ET O			-	
	ic printi_demo.c	g main.c X	,					5 📺 D	- \1		8
🖻 🧏 V 🗟 8	19				<u>^</u>		87 E I	z 🔍 🖉	₩	. 8	
Cm3_demo	200 Volu gcov_init (volu)					printf_demo.h				
Cm3_treentos	22 void (**p)(void);					stadet.n				
Pinarios	23 extern uint 24 uint32 t be	32_tinit_a	array_start,i t)& init array	nit_array_end; start:	/* linker deti		untsz_t : unsig	gned long i 2 +	n		
> (3) Includes	25 uint32_t en	d = (uint32_1	t)&init_array_	end;			acov init(void) :	void			
x 🕞 Debug	26						main(void) : int	Volu			
× G src	27 While(beg <e< td=""><td>na)</td><td></td><td></td><td></td><td>•</td><td>mani(void) : me</td><td></td><td></td><td></td><td></td></e<>	na)				•	mani(void) : me				
	29 p = (vo	id(**)(void))beg; /* get fun	ction pointer	*/						
× Ch project	30 (*p)();	/* call cons	structor */	/							
× 🕞 printf	32 1 Deg	sizeor(p), /	next pointer	/	¥						
> 🔜 gcov.o - [arm/le]	<				>						
> 🔜 main.o - [arm/le]	🖹 Problems 🧔 Tasks	📮 Console [🔲 Properties 🛯 🖉 T	Terminal 🐻 gcor	v ×	Ē	👏 🖾 💷 🚔	20	> -		
> 🗟 printf_demo.o - [arm/le]											
i gcov.d	program runs = 0										
🕞 gcov.gcda	program file : D:\GMD_ timestamp + 2025 (4/20)	2025_workspac	e\workspace_cm3_	h\cm3_gcov_dem	o\Debug\cm3_gcov_de	emo.elf					
🕞 gcov.gcno	O type filter text	10:07								_	
📄 main.d	/ type linter text	T = 111	1 I		0 01						
😰 main.gcda	Name	Total Lines	Instrumented	Executed Lines	Coverage %						
😰 main.gcno	✓ Summary	6,567	1,252	159	12.7%						
📄 printf_demo.d	> core_cm3.c	112	82	0	0.0%					- 1	
😰 printf_demo.gcda	> core_cm3.h	1,646	19	0	0.0%					- 1	
😰 printf_demo.gcno	> delay.c	75	20	0	0.0%					- 11	
🗋 subdir.mk	> gcov.c	200	100	82	52.50%					- 1	
> 🕸 cm3_gcov_demo.elf - [arm/le]	> gpio.c	206	10	0	0.0%					- 1	
cm3_gcov_demo.bin	> gw1ns4c_gpio.c	202	100	0	0.0%					- 1	
cm3_gcov_demo.map	> gw1ns4c_ize.c	172	35	0	0.0%					- 1	
là makefile	> gw1ns4c_miscie	82	37	0	0.0%					- 1	
bjects.mk	> gw1ns4c_relic	341	78	0	0.0%					- 1	
i sources.mk	> gw1ns4c_syscon.c	135	18	0	0.0%					- 11	
> 🤁 src	> gw1ns4c timer.c	237	52	0	0.0%						
📄 readme.bxt	> gw1ns4c uart.c	365	82	25	30.49%					- 1	
cms_gpror_demo	> gw1ns4c wdog.c	318	68	0	0.0%						
cms_rttnread_nano	> hyper ram.c	204	69	0	0.0%						
	> main.c	43	12	11	91.67%						
	> malloc.c	193	70	0	0.0%					~	
						1					

Gcov 工具相关功能如表 12-1 所示。

表 12-1 Gcov 功能

选项	功能说明
Show/Hide columns (指定要显示的列
Export to CSV (引出分析数据到.csv 文件
Sorting (分析数据的排列次序
Sort coverage per file (按文件对分析数据排序
Sort coverage per function (按功能对分析数据排序

选项	功能说明
Create chat ()	使用指定的分析数据,创建柱状图或饼 状图

8. 可以创建柱状图或饼状图。在 Gcov 中,选定要创建图形的文件,选择

"Create chat..."(**冬**),如图 12-13 所示。

图 12-13 Create chat...

🐳 Create chart from selection	×
🚱 Chart builder	
Select your chart type	
●Bar graph Vertical bars	
⊖ Pie chart	
Select the columns(s) to show	
☑ Instrumented Lines	
Executed Lines	
Coverage %	
Select all Deselect all	
OK Cancel	

创建的柱状图如图 12-14 所示, 创建的饼状图如图 12-15 所示。



图 12-14 柱状图

图 12-15 饼状图



13Profiling功能

13.1 关于 Profiling 功能

GMD 软件支持 Profiling 功能, Profiling 功能是借助 GCC 编译器的 Binutils 中的 Gprof 工具,来查看指定文件中函数的运行时间和调用次数, 以及调用关系。Gprof 可以用来确定程序的瓶颈,以便进行性能优化。 Gprof 通过在程序运行时收集数据来工作,然后生成一个报告,该报告显示 每个函数在程序中占用 CPU 时间的百分比以及函数之间的调用关系。

Gprof 是一个性能分析工具,可以帮助开发人员理解 C/C++程序的运行 情况,通过 Gprof 可以获取到程序中各个函数的调用信息、调用次数、执行 时间等,对优化程序、提升程序运行效率具有重要作用。

GMD 软件中,通过带特定-pg 编译选项编译指定源码文件,一是可以 在实际开发板上运行,并配合 semihost 功能则可以收集需要的 gmon.out 文件,或者二是可以直接在 QEMU 上模拟运行。在 Gprof 工具的配合下, 以图形化的方式展示。

产生的这个 gmon.out 文件需要配合编译器并且实际上板或 QEMU 模 拟运行,并且运行环境支持文件的读取,才可以进行有效的 Profiling 功能。

13.2 使用 Profiling 功能

例如在 QEMU 平台上模拟运行 Gowin_EMPU(GW1NS-4C)。

13.2.1 创建工程

创建一个用于 Profiling 功能的软件测试工程,相比于常规使用的软件 工程,该工程中需要有几个特殊的代码段:

- 添加额外的 gmon.c、gmon.h、profile.c、profile.h 文件
- 调用 gmon 相关函数
- 配合 Profiling 功能,修改 startup.S 汇编文件、flash.ld 链接文件

具体实现请参照相关参考设计,例如 Gowin_EMPU(GW1NS-4C)的相 关参考设计: cm3_gprof_demo

13.2.2 设置相关选项

设置编译选项

工程创建后,对想要进行代码分析的文件或文件夹设置一个额外的编译 选项-pg,然后编译工程。

选定某个文件,右键选择 "Properties > C/C++ Build > Settings > GNU Arm Cross C Compiler > Other compiler flags",添加 "-pg",如图 13-1 所示。

图 13-1 设置编译选项

Solution Generate assembler listing (-Wa,-adhlns="\$@.lst") Preprocessor Save temporary files (save-temps Use with caution!) Optimization Verbose (-v) Warnings Other compiler flags -pg	💱 Tool Settings 📋 Container Settings 🎤	Build Steps
	 Solution Solution<	Generate assembler listing (-Wa,-adhIns="\$@.lst") Save temporary files (save-temps Use with caution!) Verbose (-v) Other compiler flags -pg

设置 Linux Tools Path

在使用 Profiling 功能的过程中,需要调用 Parse HexDump Data 功能,因不同用户环境的差异性,所以建议先在"Linux Tools Path"中指定编译工具链。

Windows 系统下,手动指定工具链路径及前缀信息,例如 "\${eclipse_home}\toolchain\gowin\gcc_arm\bin"和"arm-none-eabi-",如图 13-2 所示。

Linux 系统下,使用系统默认环境即可,即"Use the System Environment PATH",如图 13-3 所示。

选定当前工程,右键选择 "Properties > Linux Tools Path",指定编译工具链及其路径。

Properties for cm3_gprof_	demo		_		×				
type filter text	Linux To	ols Path	$\langle \neg$	• 🔿	▼ 00				
> Resource Builders > C/C++ Build	○ Use the System Environment PATH ● Prepend string to PATH								
> C/C++ General	Options:	Custom			~				
> MCU		\${eclipse_home}\toolchain\gowin\gcc_ar	m∖bin						
Project Natures	Prefix:	arm-none-eabi-							
Project References Run/Debug Settings > Task Repository Task Tags > Validation WikiText		Restore <u>D</u> efa	ults	Appl	У				
?		Apply and Close]	Cancel					

图 13-2 设置 Linux Tools Path

图 13-3 设置 Linux Tools Path

	Properties for cm3_gprof_demo		8
type filter text	Linux Tools Path	⇔ ▼ ⇔ ▼	00
 Resource Builders C/C++ Build C/C++ General Linux Tools Path MCU Project Natures Project References Run/Debug Settings Task Repository Task Tags Validation WikiText 	Use the System Environment PATH Prepend string to PATH Options: Custom Prefix:	Apply	
?	Cancel	Apply and Clo	se

13.2.3 运行工程及使用 Profiling 功能

工程编译完成后,可以运行或调试工程,可以选择 QEMU 模拟运行, 也可以调试实际的开发板。例如使用 QEMU 模拟运行,在 GMD 软件的控 制台中可以看到 Profiling 信息输出。如果是在开发板上调试,则是在串口 输出中可以找到 Profiling 信息输出。

参照如下步骤运行工程并使用 Profiling 功能:

 "Debug As > Debug Configurations… > GDB QEMU arm Debugging",在 QEMU 中启动调试运行,运行工程并在控制台中查看 Profiling 信息,如图 13-4 所示。





2. 输出的 Profiling 信息需要解析后 GMD 软件才可以正确读取,在控制台内任意位置右键,在弹出的下拉菜单中选择 "Select All",选中所有的输出信息,如图 13-5 所示。

🗐 Console	×	1888 Registers	🖹 Problems	Executables	🖳 Debugger Console	🚺 Memory		- 6	3
						🗕 🗙 🦎 🗟 🚮 🗗 🚝	1	- 🔿	Ŧ
m3_gprof_de	em	o debug qemu	I [GDB QEMU a	rm Debugging]					
0000000000	000	000000000000000000000000000000000000000	000000000000000000000000000000000000000	000					^
000000000°°° 0000000000000	ł	Cut		Ctrl+X	1				
00000000		Conv		Ctrl+C					
00000000	-	Deste		Cultiv					
00000000	E	Paste		Ctri+v					
0000000		Select All		Ctrl+A					
0000000		Find/Replace.							
00000000		Find Next	Select All	Ctrl+K					
000000000		Find Dravieur		Chel (Shift) K					
000000000				CULTINICTIC					
00000000	_	Open Link							
0000000	k	Clear							
00000000	2	Remove All T	erminated						
0000000	2	Terminate/Di	connect All						
00000000		Citude							
000000000000000000000000000000000000000		GILHUD							
00000000	2	Parse HexDur	mp Data						
00000000		Scroll Lock							
00000000	20	Word Wrap							
8090000b					-				
80c0000f		Preferences							
PEATE: gmg		out			_				
ALATL: BIIO		ouc							
ump profil	lin	g data finis	hed						
									×
5								>	

3. 然后,在控制台中任意位置再次右键,选择"Parse HexDump Data", 如图 13-6 所示。

图 13-6 Parse HexDump Data

■ Console × 1000 Registers [🕺 Problems 🜔 Executables	🙀 Debugger Console	Memory	- 8
cm3_gprof_demo debug qemu	[GDB QEMU arm Debugging]		■ X ¾ 🗟 🚮 🗗 🧲 🦉	! ≓ ⊑ - ☆ -
	0000000000000 000000000000 00000000000			^
000000000000000000000000000000000000000	Cut	Ctrl+X		
000000000000000000000000000000000000000	Сору	Ctrl+C		
000000000000000000000000000000000000000	Paste	Ctrl+V		
000000000000000000000000000000000000000	Select All	Ctrl+A		
000000000000000000000000000000000000000	Find/Replace			
000000000000000000000000000000000000000	Find Next	Ctrl+K		
000000000000000000000000000000000000000	Find Previous	Ctrl+Shift+K		
000000000000000000000000000000000000000	Open Link			
000000000000000000000000	Clear			
000000000000000000000000000000000000000	Remove All Terminated			
00000000000000000000	Terminate/Disconnect All			
000000000000000000000000000000000000000	GitHub			
0000000000000000000000	Parse HevDump Data			
d408000058080000010000				
080c0000fc080000010000	Scroll Lock Parse HexDum	p Data		
	Word Wrap			
CREATE: gmon.ouc	Preferences			
Dump profiling data fin is m	20			
 <				>
4. 此时 GMD 软件会对输出的文件进行分析,并将结果分别保存在对应的 文件中,如图 13-7 所示。

图 13-7 解析 Profiling 信息并生成对应文件

Parse and Generate HexDump Finish	×
The generated files are as follows : D:\GMD_2025_workspace\workspace_cm3_h\cm3_gprof_demo\gmon.out	^
	~
ОК	

5. 查看工程的 Debug 文件夹,可以看到产生了对应的 gmon.out 文件,如 图 13-8 所示。

图 13-8 查看生成的 gmon.out 文件



6. 双击 gmon.out 文件, 打开 Gcov 工具, 如图 13-9 所示。

图 13-9 打开 gmon.out 文件

🗱 Gmon File Viewer: binary file		;	×
Binary File Please enter here the binary file which D:\GMD_2025_workspace\workspa	h produced the <u>W</u> orkspace	profile data. <u>F</u> ile System	
	ОК	Cancel	

7. Gprof 工具启动,就可以看到对应用程序的分析结果,显示了文件、方法的调用关系等,如图 13-10 所示。

图 13-10 Gprof

🖹 Problems 🧔 Tasks 📮 Console 🔲	Properties (🖉 Terminal	😼 gprof 🛛			
				🕀 🖻 🍕	ی• 😒 🐼 📙 🕹 🖒 🕻	🔊 🌑
gmon file: D:\GMD_2025_workspace\workspace_cm3_h\cm3_gprof_demo\gmon.out program file: D:/GMD_2025_workspace/workspace_cm3_h/cm3_gprof_demo/Debug/cm3_gprof_demo.elf timestamp: 2025/4/30 11:18 4 bytes per bucket, each sample counts as 1.000ms						
P type filter text						
Name (location)	Samples	Calls	Tin	ne/Call	% Time	
✓ Summary	25				100.0%	
> ??	8				32.0%	
> core_cm3.h	0				0.0%	
> gw1ns4c_uart.c	9				36.0%	
✓ main.c	7				28.0%	
∽ main	7	1	7.0	00ms	28.0%	
> main (main.c:33)	2				8.0%	
> main (main.c:41)	1				4.0%	
> main (main.c:42)	1				4.0%	
> main (main.c:45)	1				4.0%	
> main (main.c:47)	1				4.0%	
> main (main.c:49)	1				4.0%	
> retarget.c	1				4.0%	
> startup_gw1ns4c.S	0				0.0%	

8. 双击 Gprof 中的某一行, GMD 软件就会自动打开对应的源文件并定位 到对应的行, 如图 13-11 所示。

₩ workspace_cm3_h - cm3_gprof_demo/src/project/printf/main.c - GMD				×				
<u>File E</u> dit <u>S</u> ource Refactor <u>N</u> avigate Se <u>a</u> rch <u>P</u> roject <u>R</u> un <u>G</u> owin <u>W</u> indow <u>H</u> elp								
📑 🕶 🔚 🕼 🥸 🕶 🗞 🕶 🔜 🥔	🗠 🕒 🍬 🕹 😓 開 🔂 🕶 🛸	• 🖸 • 🎯 • 🐐	• 🜔 • 🚱 • 🤇	🎍 🔹 🥭 🥖	🔗 🕶 🍠 🔊 🚱 🔲	Π.		
∲ ▼ ∲ ▼ ∜⊃ ⊂∛ (⊃ ▼ ⇔ ▼ ⊡	1					۹.	😰 🖪	d 🎋
Project Explorer 🗙 🗖 🗖	🖻 main.c 🗡			- 8	ЕО X 🗒 Т 🛞 В	🔓 D	- 5	1 8
E 🔄 🏹 🖆 🕴	<pre>41 printf("%d = 5\r\n",</pre>	i);		^	😜 📄 📮 🕅	ء 💉 و) 茶 (3 🗖
📋 cm3_demo	42 printf("%d = - max ir	<pre>nteger\r\n", mi); '\r\n" 'a');</pre>			uart.h			
cm3_freertos	4) $print(char & a + f(r, n^*), a);$ 44 $printf("hex & x = f(r, n^*), a);$							
📁 cm3_gcov_demo	45 printf("hex %02x = 00	0\r\n", 0);	~		💶 gmon.h			
∨ 📂 cm3_gprof_demo	46 printf("%d %s(s)", 0, $printf("%d %s(s)", 0)$	"message"):	3x\r\n", -3, -	3, -3)	main(void) : int			
> 🖑 Binaries	47 printf("\r\n");	/						
> 🗊 Includes	49 printf("%d %s(s) with	n %%\r\n", 0, "mes	sage");					
🗸 📂 Debug	51 mcleanup(2);							
> 👝 src	52							
> 🔅 cm3_gprof_demo.elf - [arm	53 while(1); 54 }							
cm3_gprof_demo.bin	55			~				
cm3_gprof_demo.map	<			>				
lò makefile	🕄 Problems 🧖 Tasks 🗉 Consol	e 🔲 Properties 🧢	Terminal 🗔 on	rof X				1
objects.mk			gp		98		-	
jo sources.mk	amon file: D:\GMD 2025 workspace	a)workspace.cm3.h)	rm3 aprof demo			<u>va</u> "4e		×
> 🦲 src	program file: D:/GMD 2025 worksp	ace/workspace cm3	h/cm3 aprof der	mo/Debug/cm3	gprof demo.elf			
gmon.out	timestamp: 2025/4/30 11:18							
= readme.txt	4 bytes per bucket, each sample co	ounts as 1.000ms						
	Ptype filter text							_
	Name (location)	Samples	Calls	Time/Call	% Time			^
	✓ Summary	25			100.0%			
	> ??	8			32.0%			
	> core_cm3.h	0			0.0%			
	> gw1ns4c_uart.c	9			36.0%			
	✓ main.c	7			28.0%			
	✓ main	7	1	7.000ms	28.0%			
	> main (main.c:33)	2			8.0%			
	> main (main.c:41)	1			4.0%			
	> main (main.c:42)	1			4.0%			
	> main (main.c:45) 1 4.0%							
	> main (main.c:47) 1 4.0%							
> main (i] D:\GMD_2025_workspace\workspace\cm3_h\cm3_gprof_demo\Debug//src/project/printf/main.c:47								
Writable Smart Insert 47:1:1308								

图 13-11 查看 Profiling 信息

Gprof 工具相关功能如表 13-1 所示。

表 13-1 Gprof 功能

选项	功能说明
Show/Hide columns (🅙)	指定要显示的列
Export to CSV (引出分析数据到.csv 文件
Sorting (分析数据的排列次序
Sort samples per file (按文件对分析数据排序
Sort samples per function (按功能对分析数据排序
Sort samples per line (🔁)	按行对分析数据排序

选项	功能说明
Display function call graph(查看函数之间的调用关系图
Switch sample/time(切换采样次数/时间百分比视图
Create chat ()	使用指定的分析数据,创建柱状图或饼 状图

9. 可以创建柱状图或饼状图。在 Gprof 中,选定要创建图形的文件,选择

"Create chat..." (**S**),如图 13-12 所示。

图 13-12 Create chat...

🐳 Create chart from selection	×
 ♦ Chart builder Select your chart type ● Bar graph Vertical bars ○ Pie chart 	
Select the columns(s) to show ✓ Samples ✓ Calls ✓ Time/Call ✓ % Time	
Select all Deselect all	
	OK Cancel

例如,创建的柱状图如图 13-13 所示,创建的饼状图如图 13-14 所示。



图 13-13 柱状图

图 13-14 饼状图



