GMD (GOWIN MCU Designer)软件

用户指南

SUG549-2.0, 2025-06-13

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

图目录

表目录

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 www.gowinsemi.com 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、缩略语

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：www.gowinsemi.com.cn

E-mail：support@gowinsemi.com

Tel：+86 755 8262 0391

2 概述

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：

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

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 平台支持

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 6 GB 磁盘空间

4 下载与安装

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。

表 4-1 GMD 软件主要文件描述

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

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。

图 4-1 选择 Gowin > GoBridgeDriver

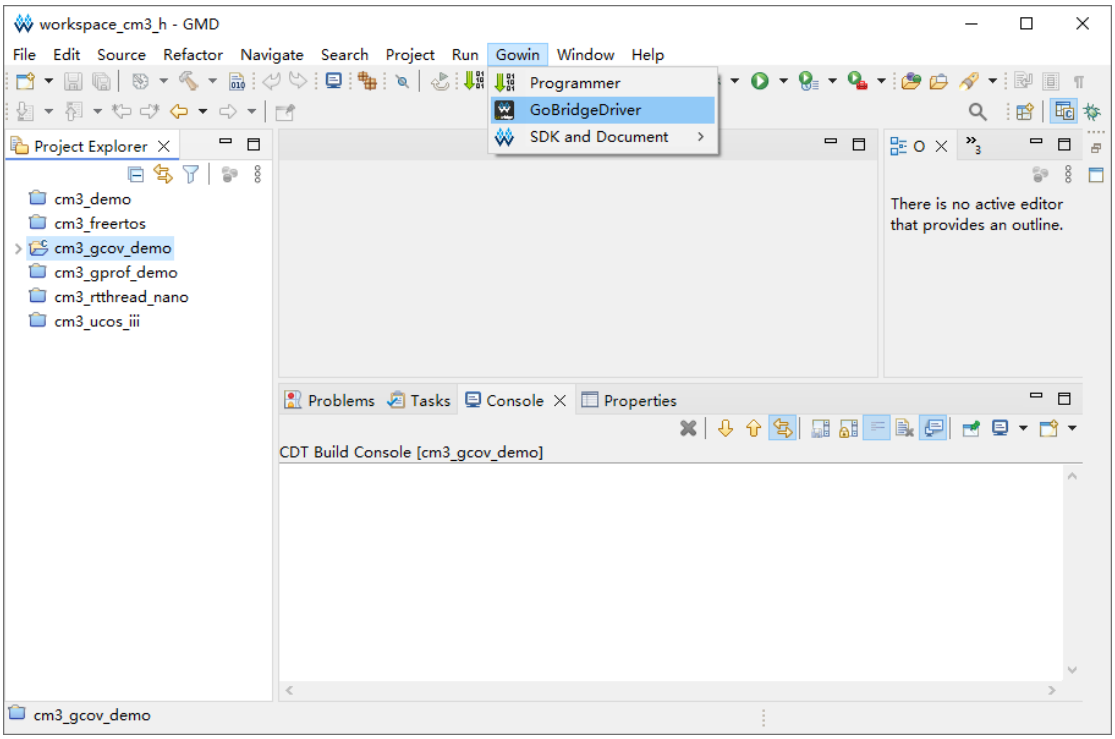
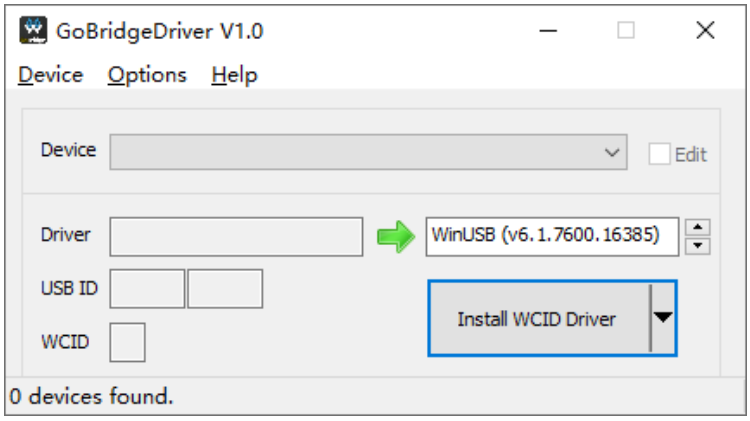


图 4-2 GoBridgeDriver 驱动软件



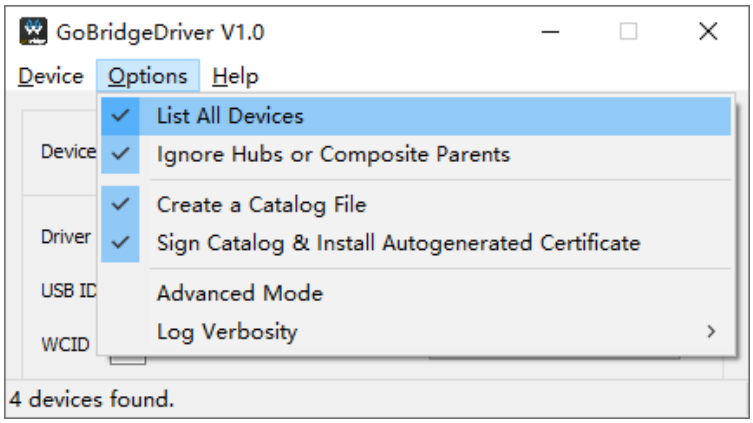
上述几种调试器的软件驱动安装方法如下文所述。

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。

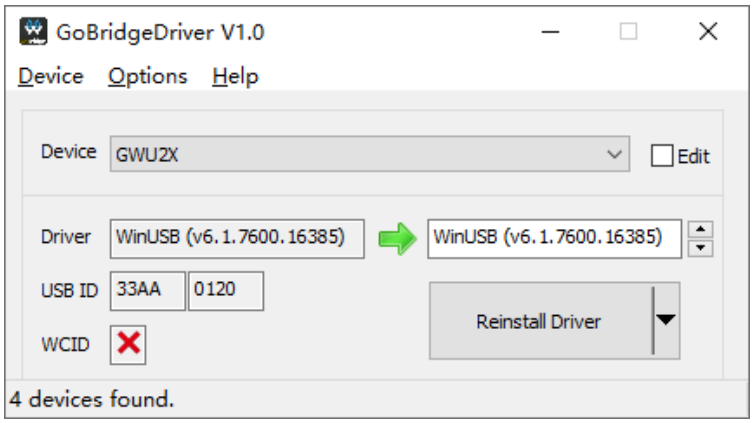
1. 请将 GWU2X 设备插入 PC 主机。
2. 选择菜单栏“Options > List All Devices”，列举出所有已插入 PC 主机的 USB 设备，如图 4-3 所示。

图 4-3 选择 Options > List All Devices



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



注！

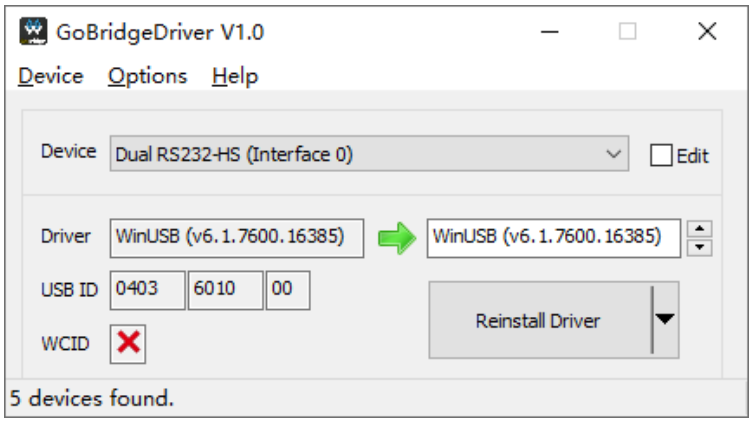
- 如果当前设备未安装过任何软件驱动，该选项显示为“Install Driver”；
- 如果当前设备已安装过当前软件驱动，该选项显示为“Reinstall Driver”；
- 如果当前设备已安装过其他软件驱动，该选项显示为“Replace Driver”。

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。
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. 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

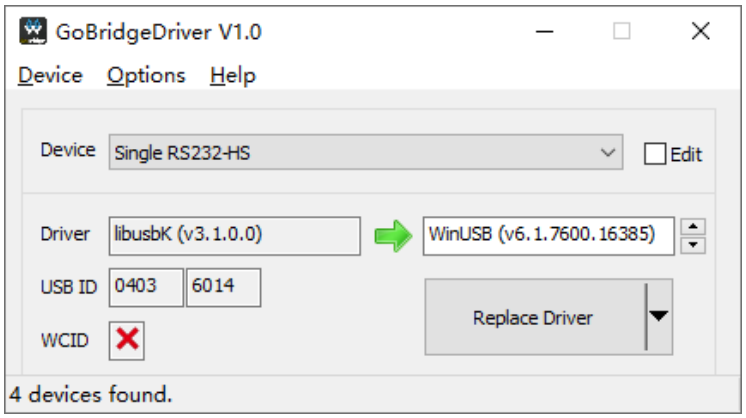


4.3.3 GWUSB V4.1

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。
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“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。

图 4-6 GWUSB V4.1 软件驱动安装

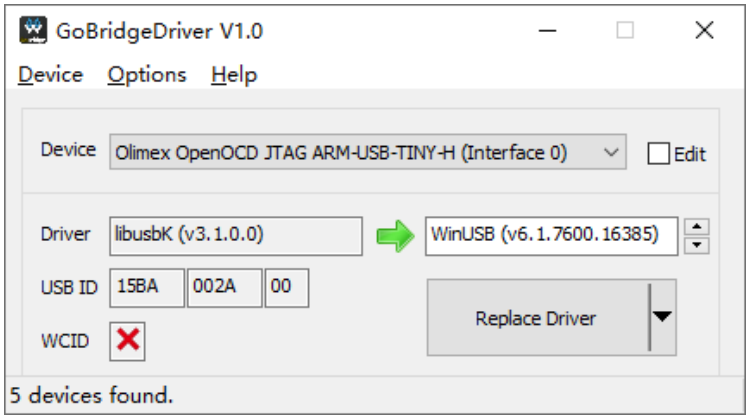


4.3.4 Olimex

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。
3. 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 4-7 Olimex 软件驱动安装

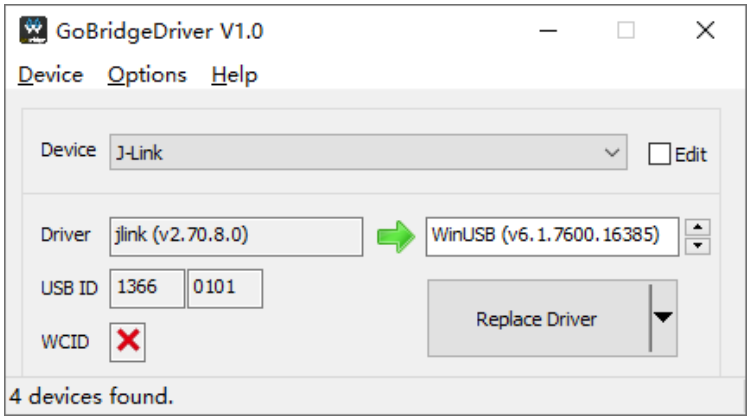


4.3.5 J-Link WinUSB

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



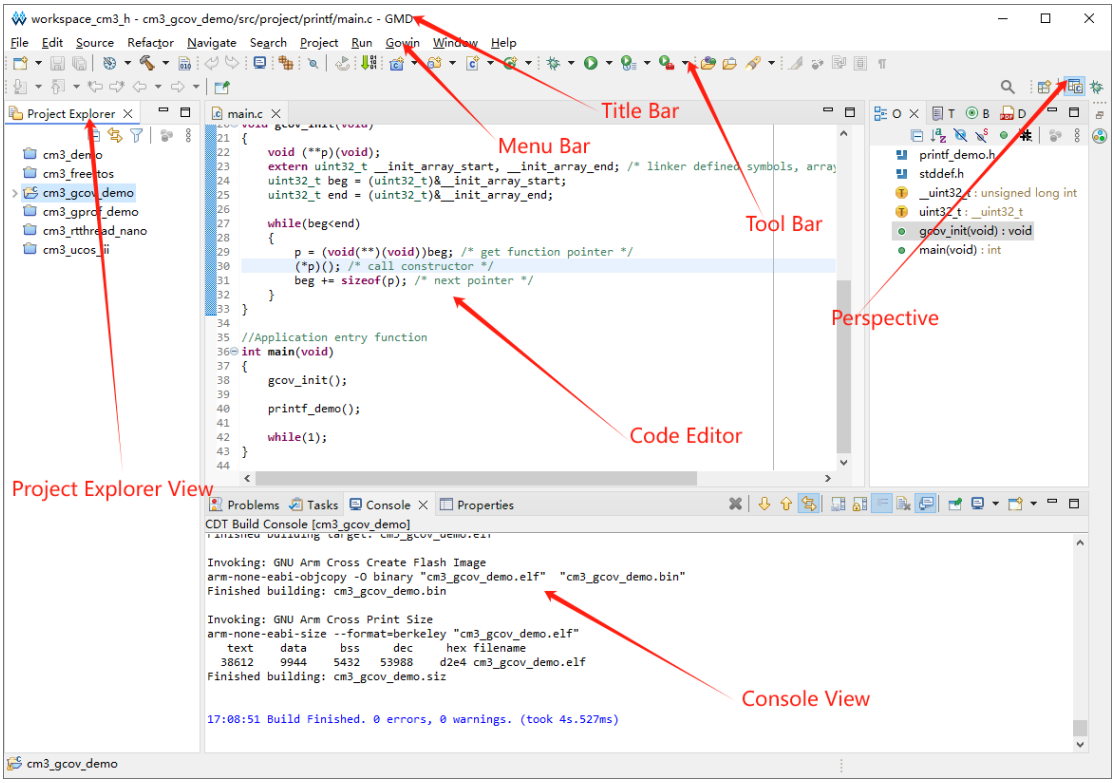
4.3.6 Segger J-Link

请参照 Segger 官方方法（https://www.segger.com/downloads/jlink/）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

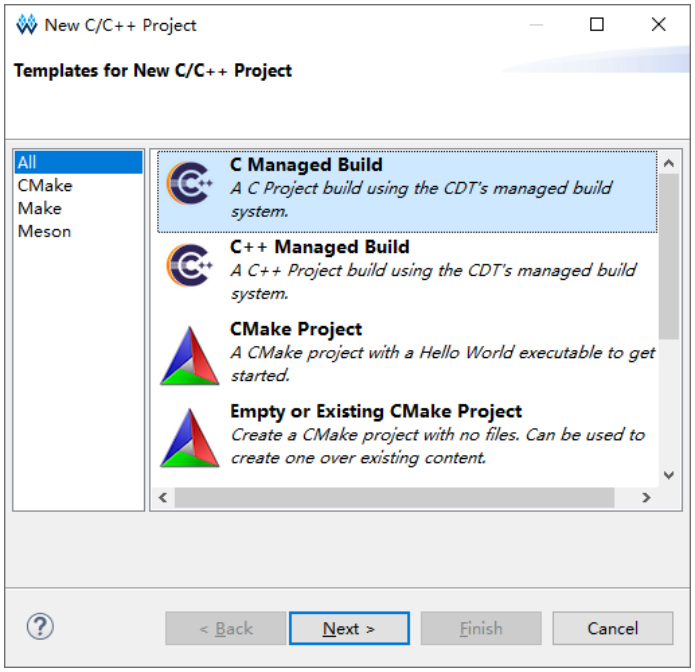
菜单项	子菜单项	功能描述
New	Makefile Project with Existing Code	创建包含 makefile 的 C/C++项目
	C/C++ Project	创建 C/C++项目
	Project…	创建 C、C++、C/C++项目
	Convert to a C/C++ Project (Adds C/C++ Nature)	非 C/C++项目添加 C/C++特征，启用 C/C++工具链
	Source Folder	创建源代码文件夹
	Folder	创建文件夹
	Source File	创建 C/C++源码文件
	Header File	创建 C/C++头文件
	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 To	Windows	Windows 系统格式
	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 Project

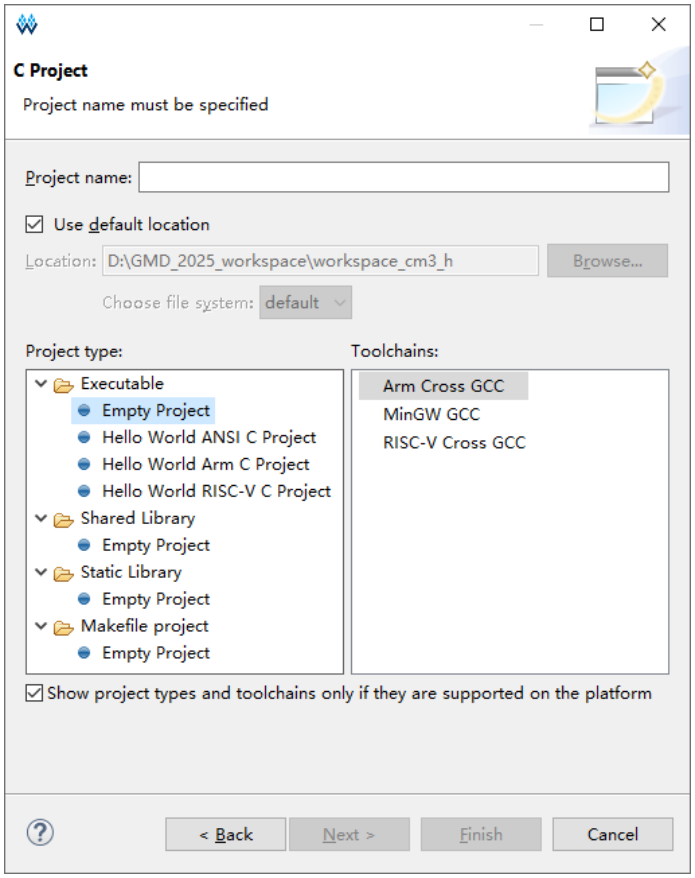


表 5-2 项目类型

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

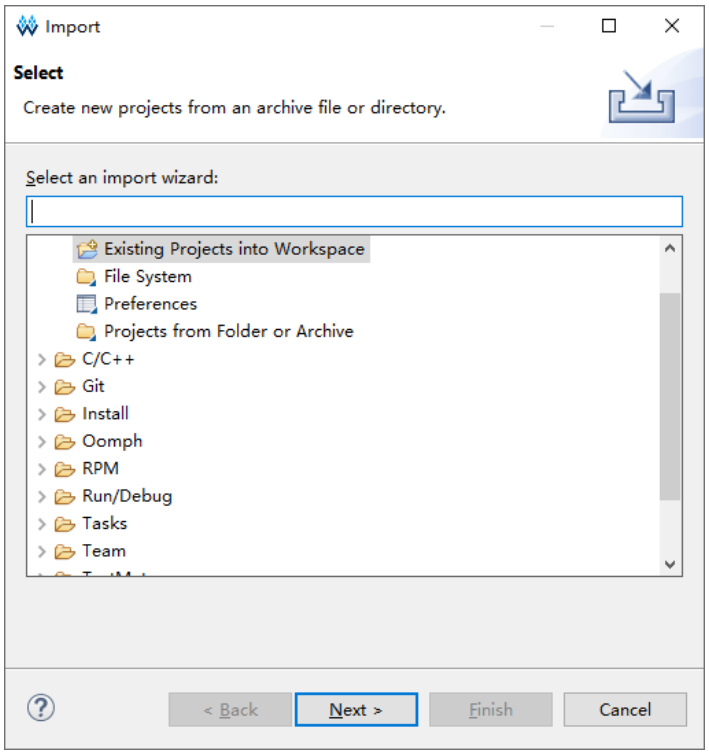
表 5-3 软件工具链

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

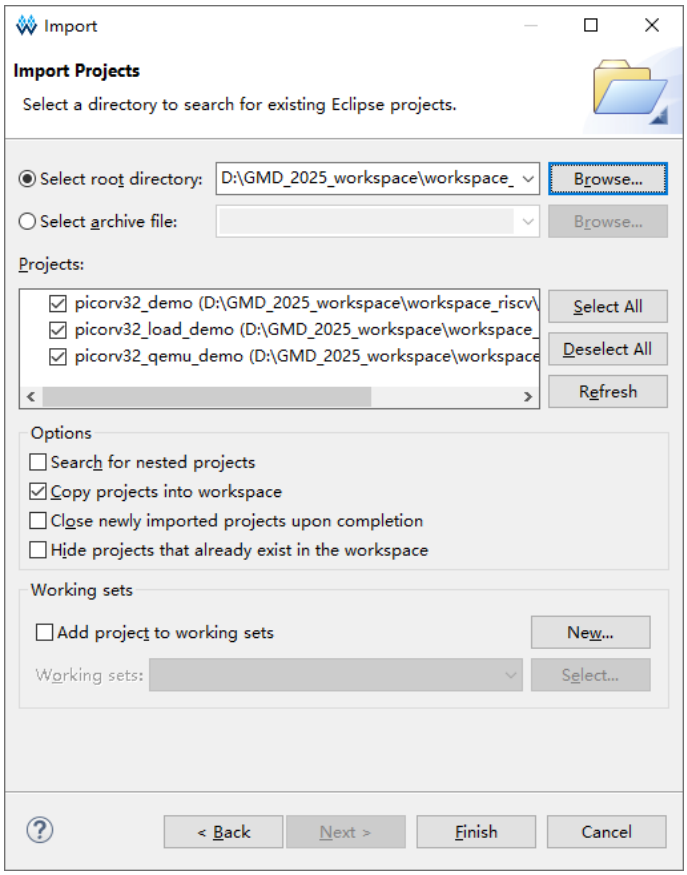
“New > Import…”选项如图 5-4 所示，用于导入已有的项目。

图 5-4 Import



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

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
Copy	-	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
Expand Selection to	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
	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…	导出已完成的重构列表，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。

表 5-7 Navigate 菜单项

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

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 Configurations	Set Active	设置 Release 或 Debug 为活动状态
	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

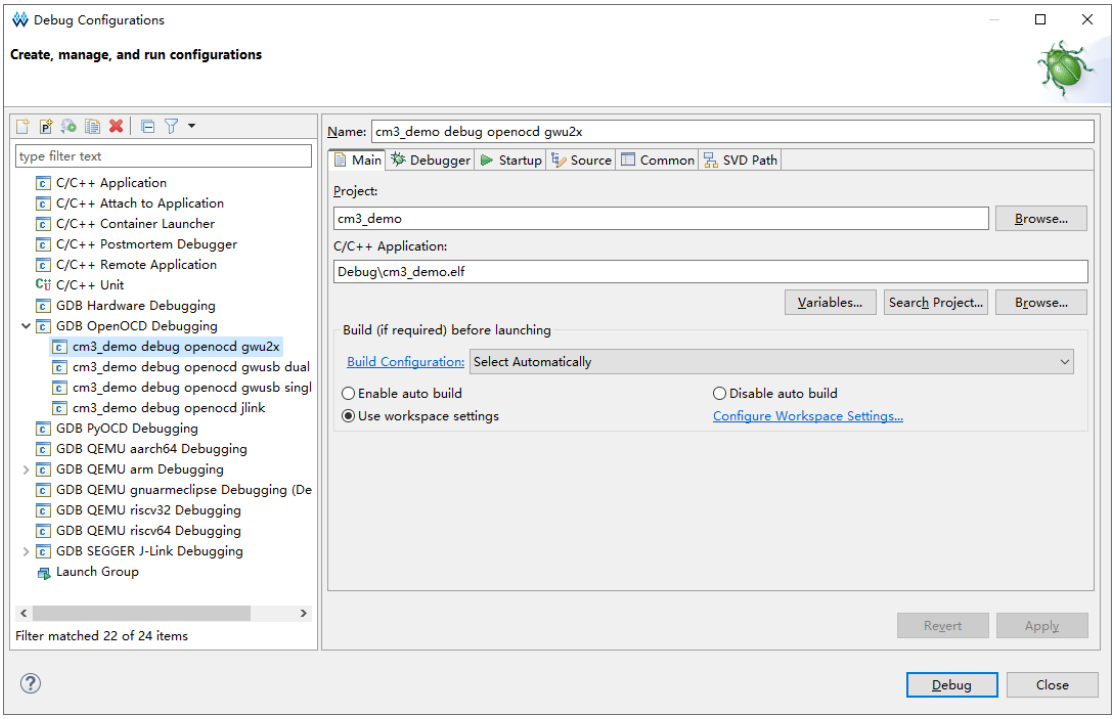
菜单项	子菜单项	功能描述
Run Last Launched	-	运行上次启动过的程序
Debug Last Launched	-	调试上次启动过的程序
Profile Last Launched	-	性能分析上次启动过的程序
Profile History	-	使用性能分析历史记录中最后一次的性能分析配置
Profile As	Local C/C++ Application	以本地 C/C++应用方式分析程序，直接在主机环境中性能分析，使用本地安装的编译器和调试器
Profile Configurations…	-	配置和管理性能分析程序时的设置选项
Run History	-	使用运行历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



常用的调试方式如表 5-11 所示。

表 5-11 调试方式

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 Debugging	使用 QEMU 仿真调试 32 位 RISC-V MCU
GDB QEMU riscv64 Debugging	使用 QEMU 仿真调试 64 位 RISC-V MCU
GEB SEGGER J-Link Debugging	使用 Segger 官方软件驱动和 J-Link 调试 Arm MCU

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。

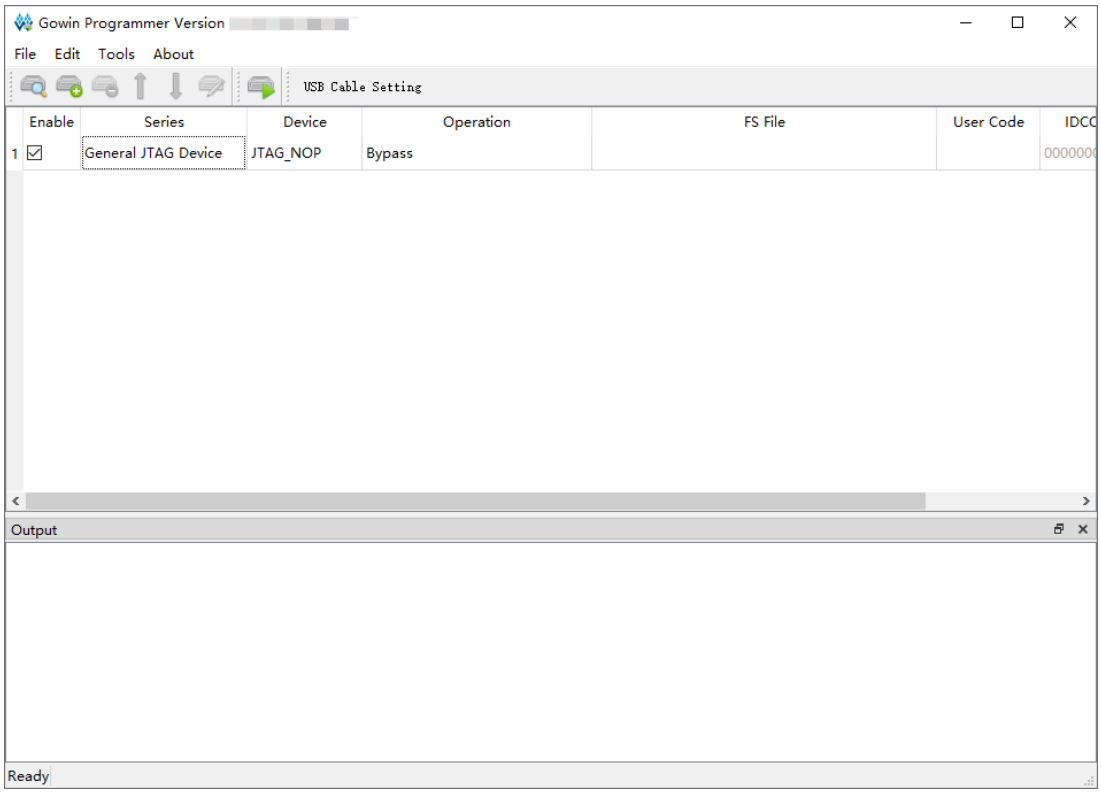
表 5-12 Gowin 菜单项

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。

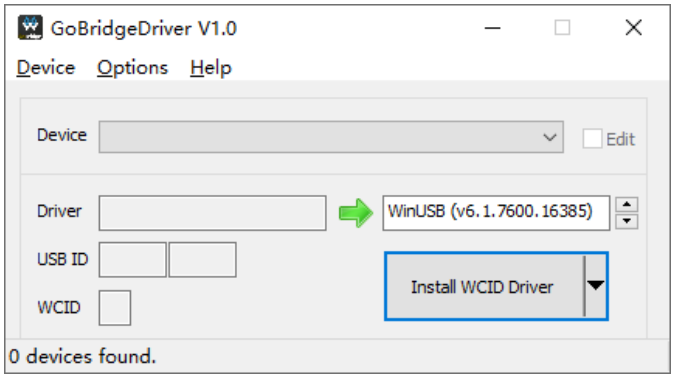
图 5-7 Programmer



GoBridgeDriver

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

图 5-8 GoBridgeDriver



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
Editor	Toggle Split Editor (Horizontal)	在当前窗口中横向打开一个新的编辑器实例
	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
Appearance	Hide Toolbar	隐藏工具栏
	Hide Status Bar	隐藏状态栏
	Toggle Full Screen	切换全屏模式
	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
Show View	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
	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
Perspective	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
	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
Navigation	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
	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。

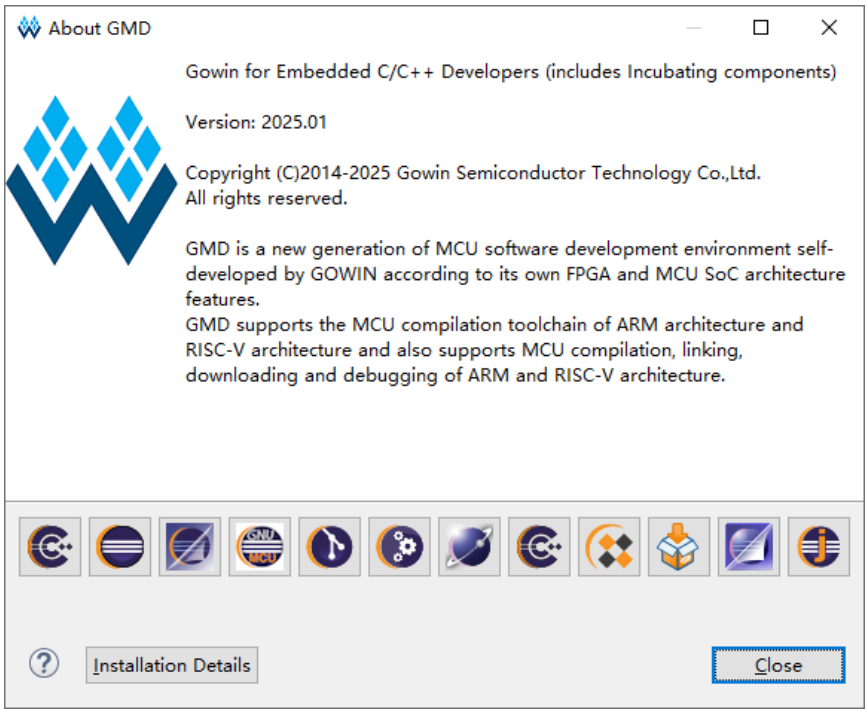
表 5-14 Help 菜单项

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

About GMD

“Help > About GMD”选项如图 5-9 所示。

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		保存所有已打开的文件
Manage configurations		管理配置（包括 Debug 和 Release）
Build		编译当前工程项目
Build All		编译工作区所有已打开的工程项目

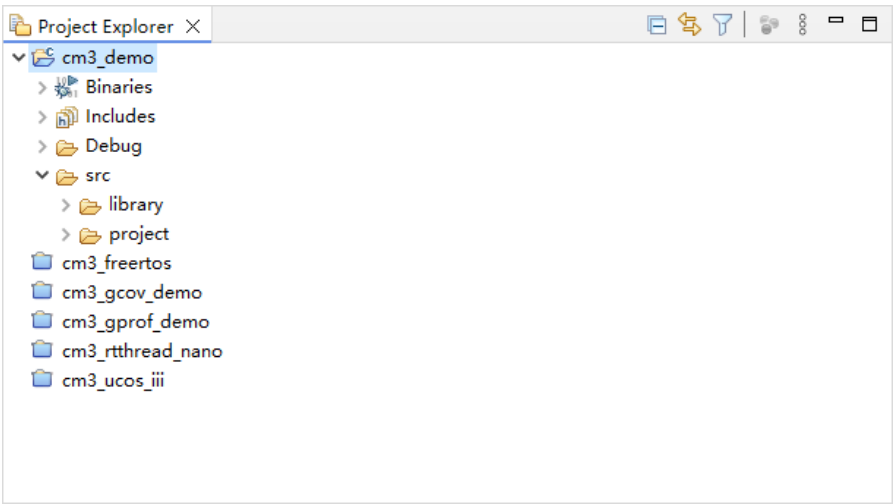
选项	图标	功能描述
Undo		撤销最近的编辑操作
Redo		重载最近的编辑操作
Open a Terminal		打开一个终端，例如串口终端
Skip All Breakpoints		调试过程中，所有断点将被标记为跳过，程序执行时不会在这些断点处暂停
Restart		在不终止整个调试会话的情况下重新开始当前方法的执行
Debug		启动应用程序的调试会话
Run		启动应用程序的运行会话
Profile		启动应用程序的性能分析会话
Programmer		启动下载软件 Programmer
Open Element		打开当前光标所在元素的详细信息
Open Task		打开当前工作区中的任务列表
Search		搜索
Toggle Mark Occurrences		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		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		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		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		打开一个新的透视图
C/C++		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。

图 5-10 项目资源管理器



5.5 代码编辑区

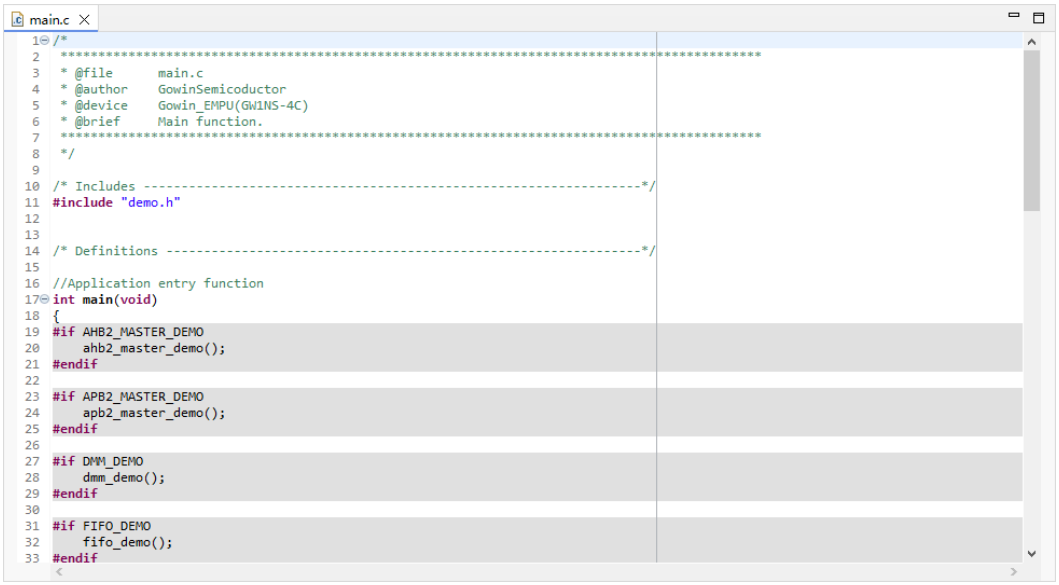
提供基本的文件编辑及查看功能如图 5-11 所示。

“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



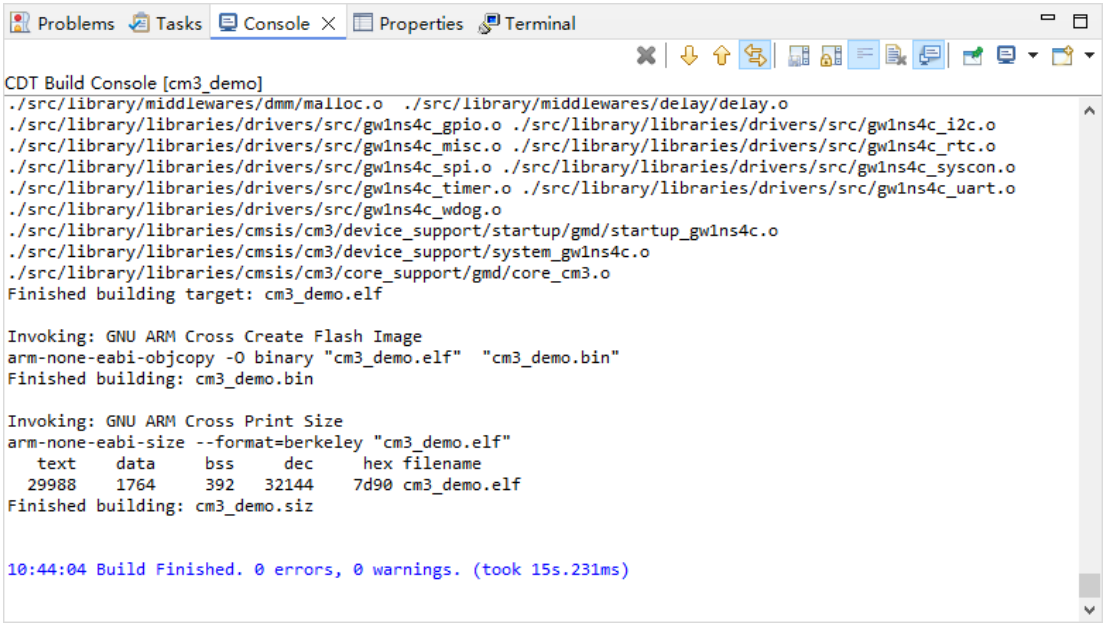
5.6 控制台

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。

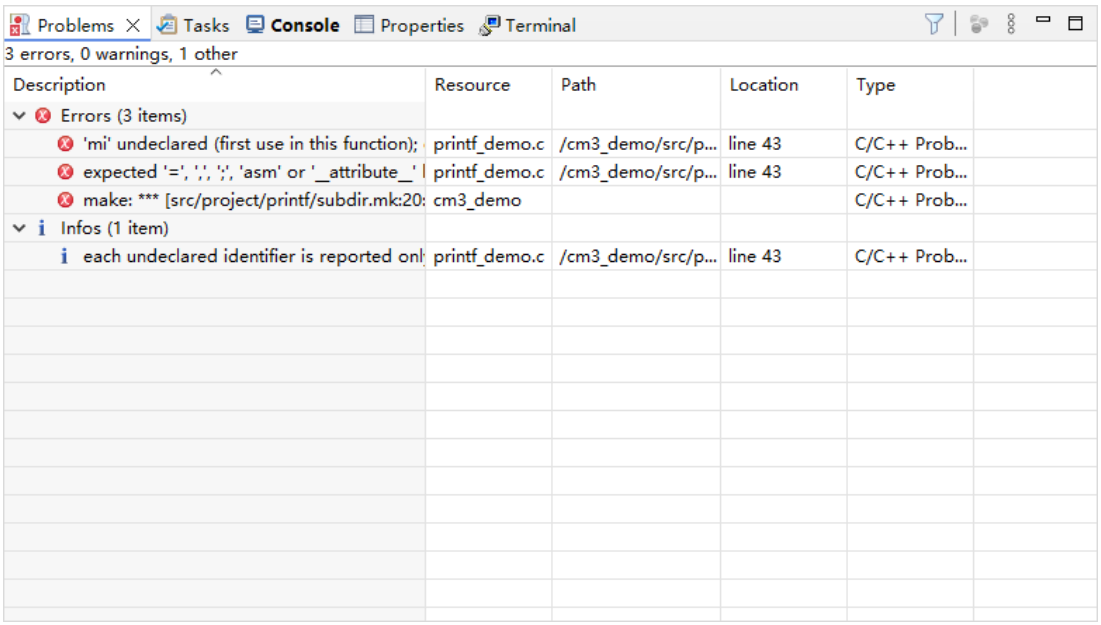
图 5-12 控制台视图



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。

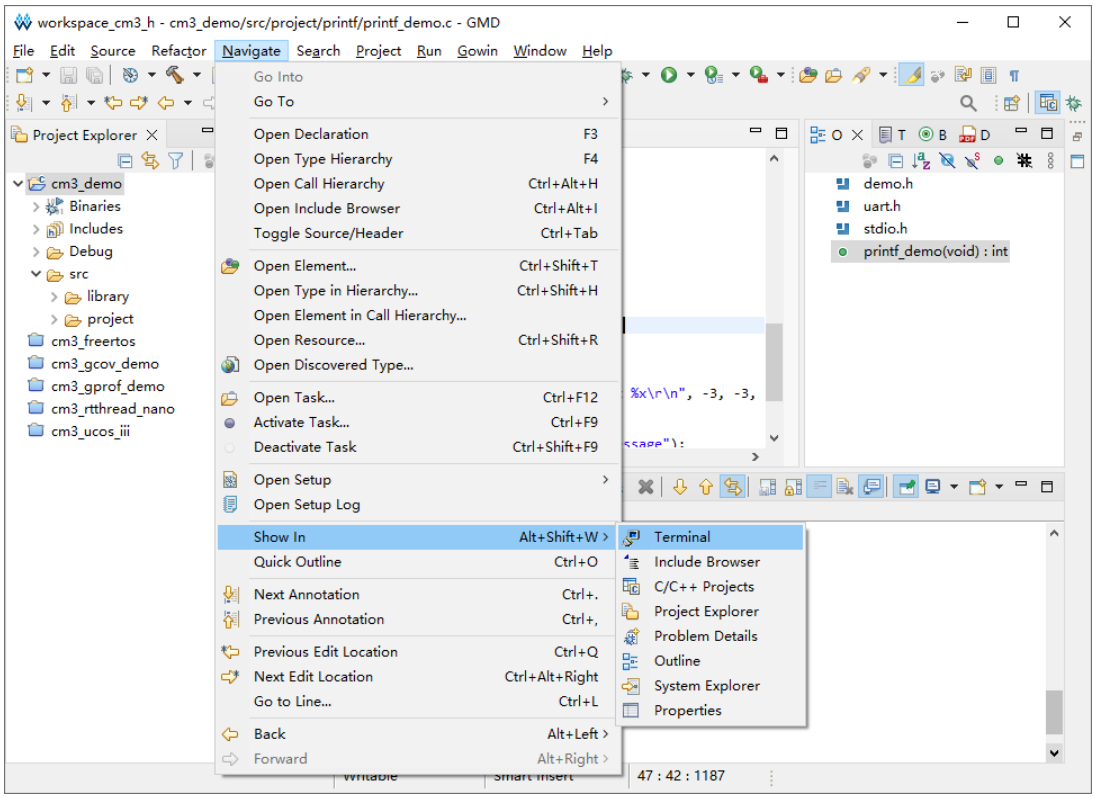
图 5-13 问题视图



5.6.3 终端视图

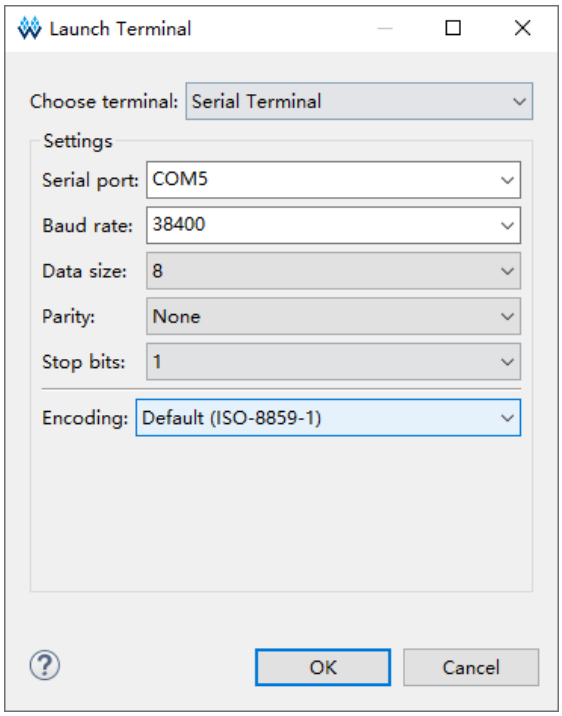
“Navigate > Show In > Terminal”选项用于启动终端如图 5-14 所示。

图 5-14 启动终端



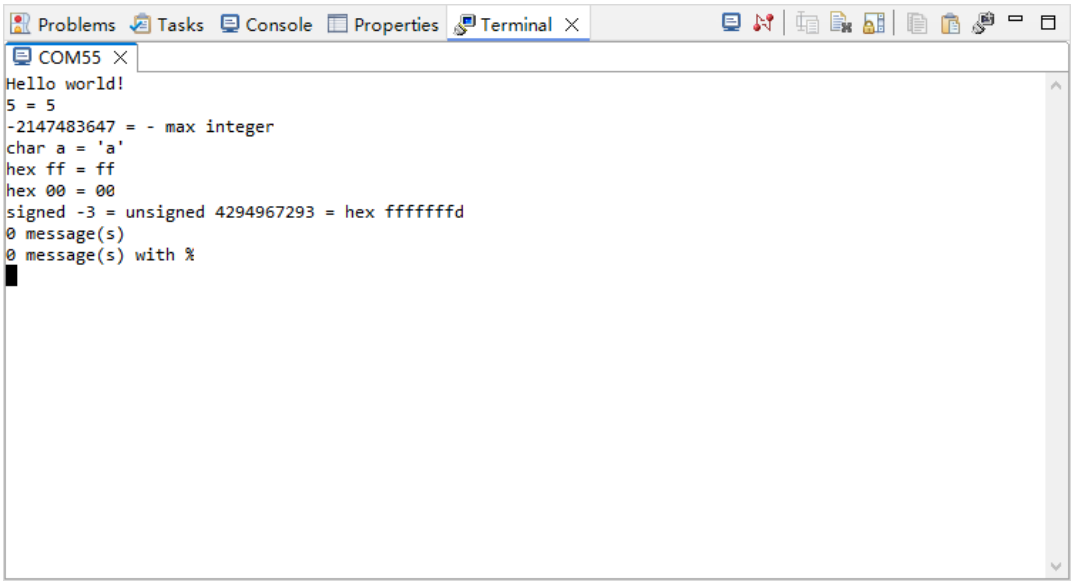
可以设置 Local Terminal、SSH Terminal、Serial Terminal、Telnet Terminal 不同类型的终端，例如常用的串口终端如图 5-15 所示。

图 5-15 串口终端



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

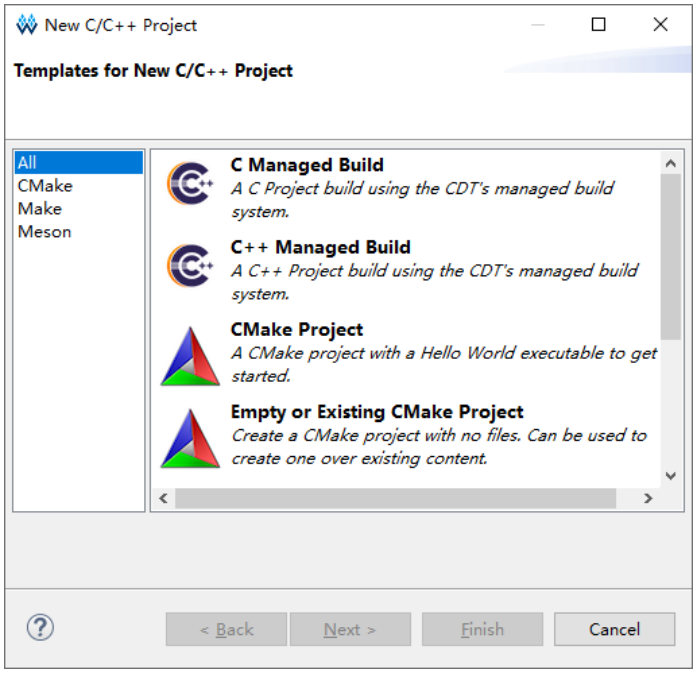


6 创建工程

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。

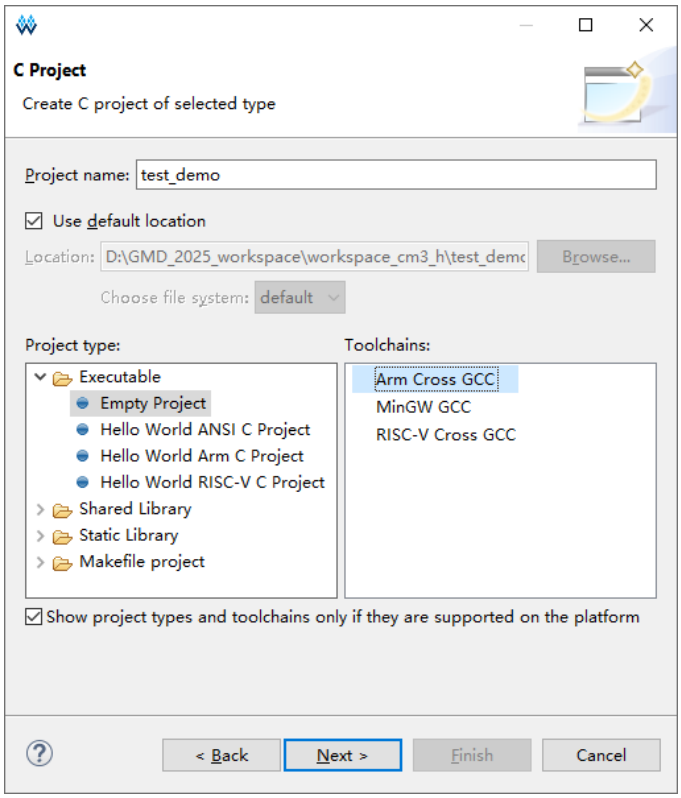
例如新建 C 工程，Arm 和 RISC-V MCU 新建工程的步骤分别如下所述。

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



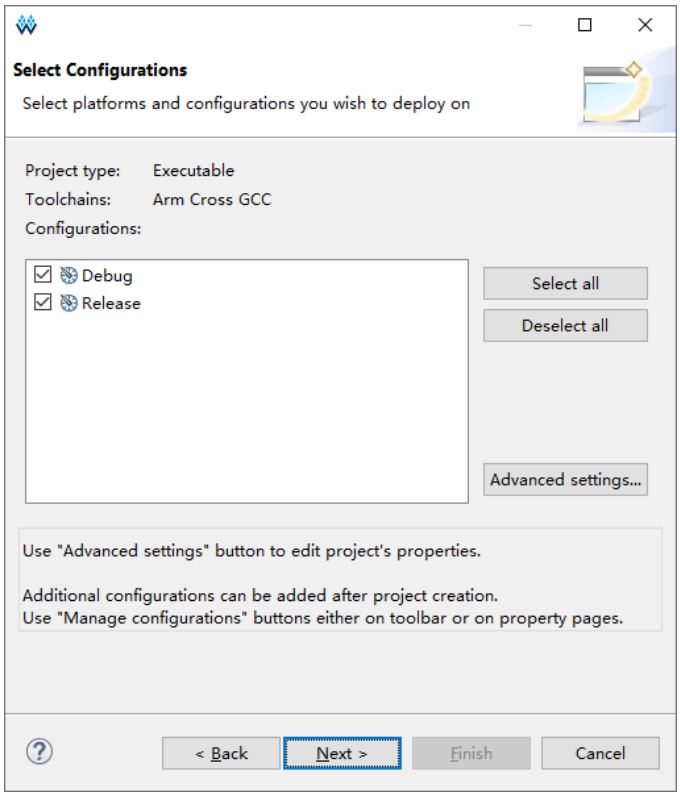
例如：

- 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。

图 6-3 选择平台和配置



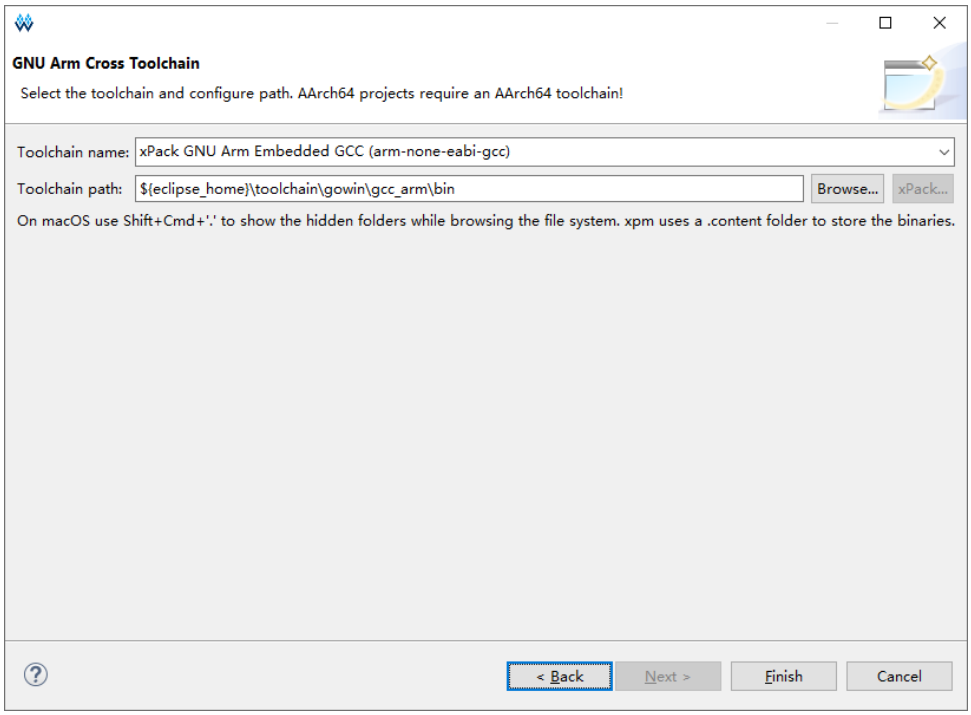
例如：

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。

图 6-4 选择工具链和路径



例如：

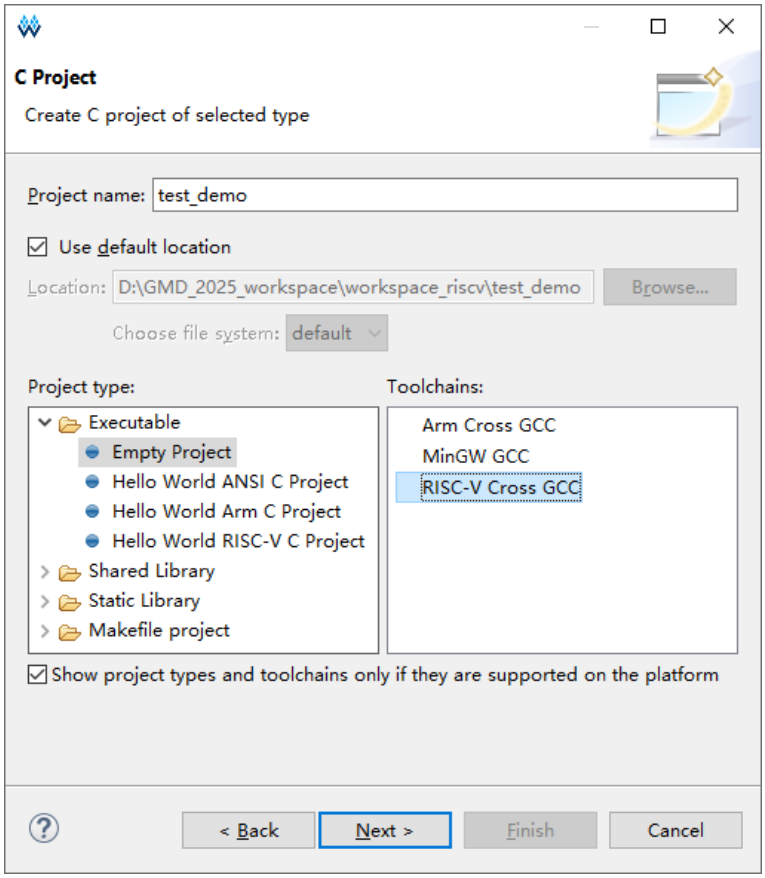
- Toolchain name：xPack GNU Arm Embedded GCC (arm-none-eabi-gcc)；
- Toolchain path：${eclipse_home}\toolchain\gowin\gcc_arm\bin。

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。

图 6-5 选择项目类型和软件工具链



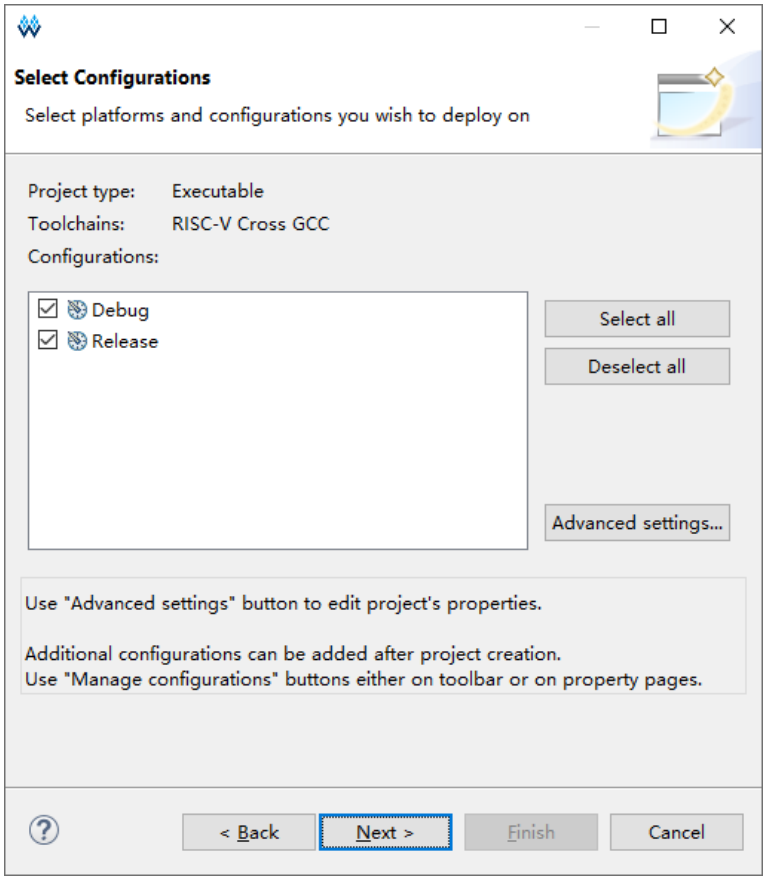
例如：

- 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。

图 6-6 选择平台和配置



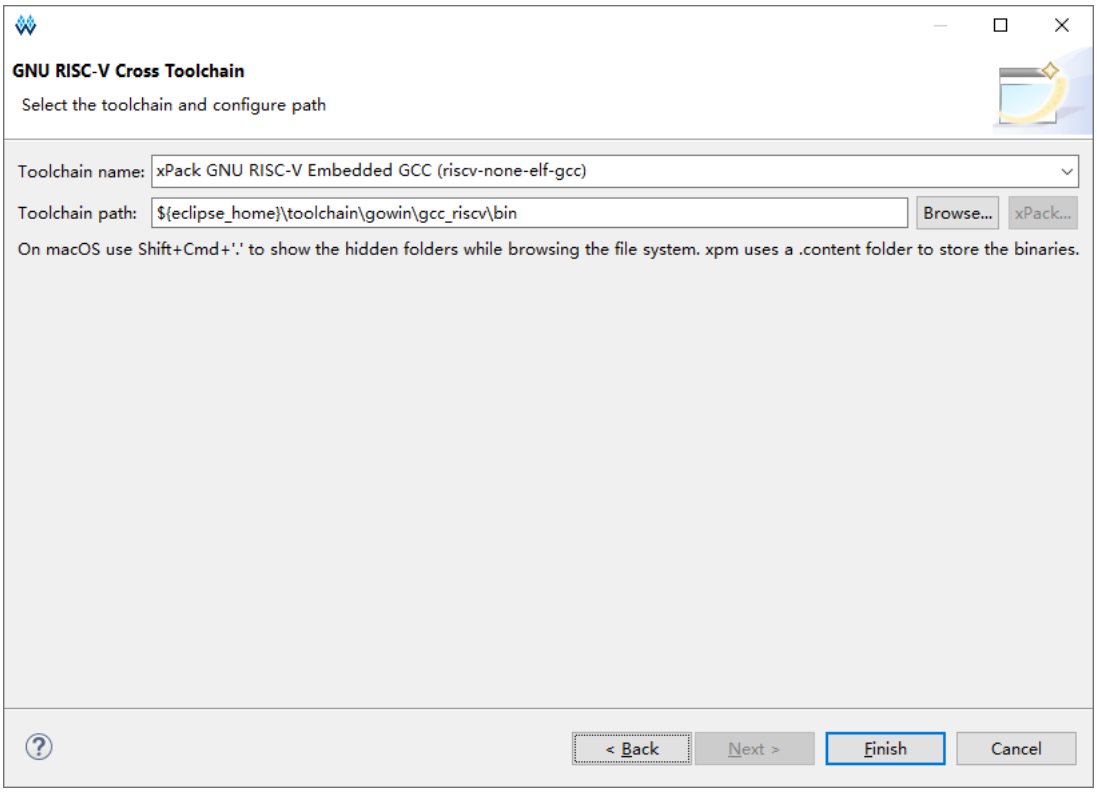
例如：

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。

图 6-7 选择工具链和路径



例如：

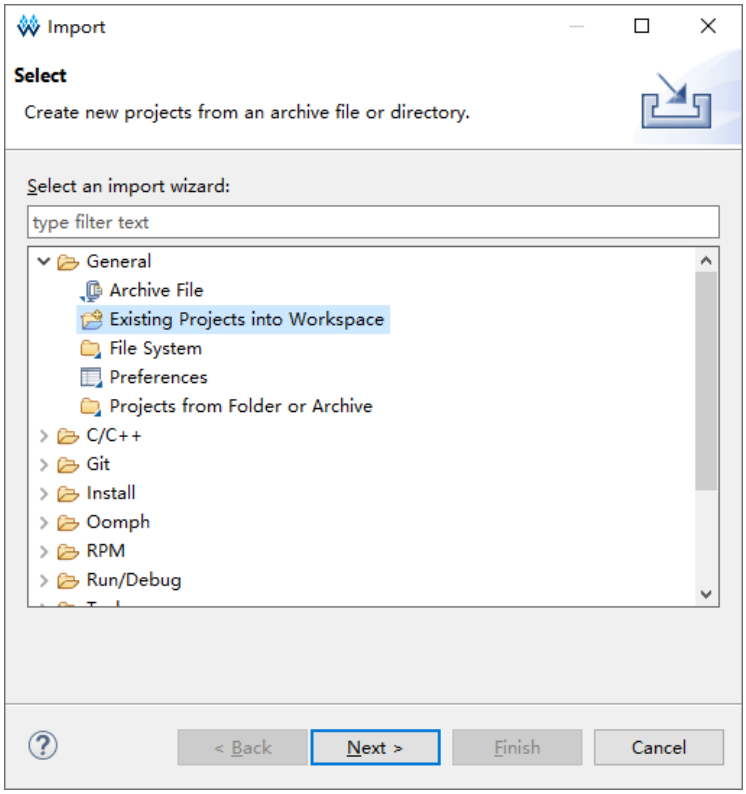
- Toolchain name：xPack GNU RISC-V Embedded GCC (riscv-none-elf-gcc)；
- Toolchain path：${eclipse_home}\toolchain\gowin\gcc_riscv\bin。

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。

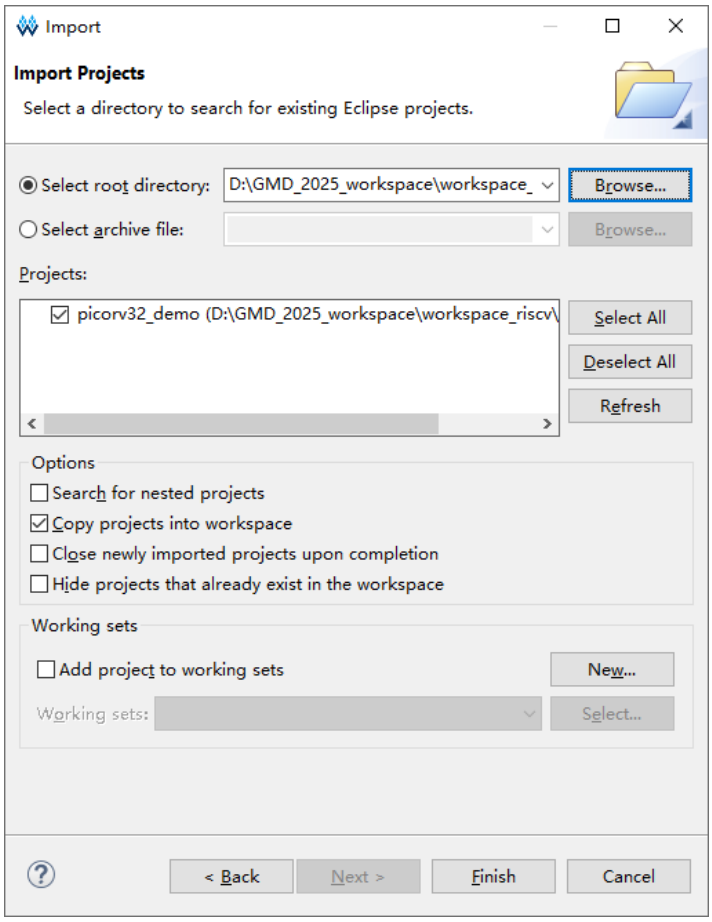
图 6-8 选择导入的方式



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。

图 6-9 选择需要导入的工程



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。

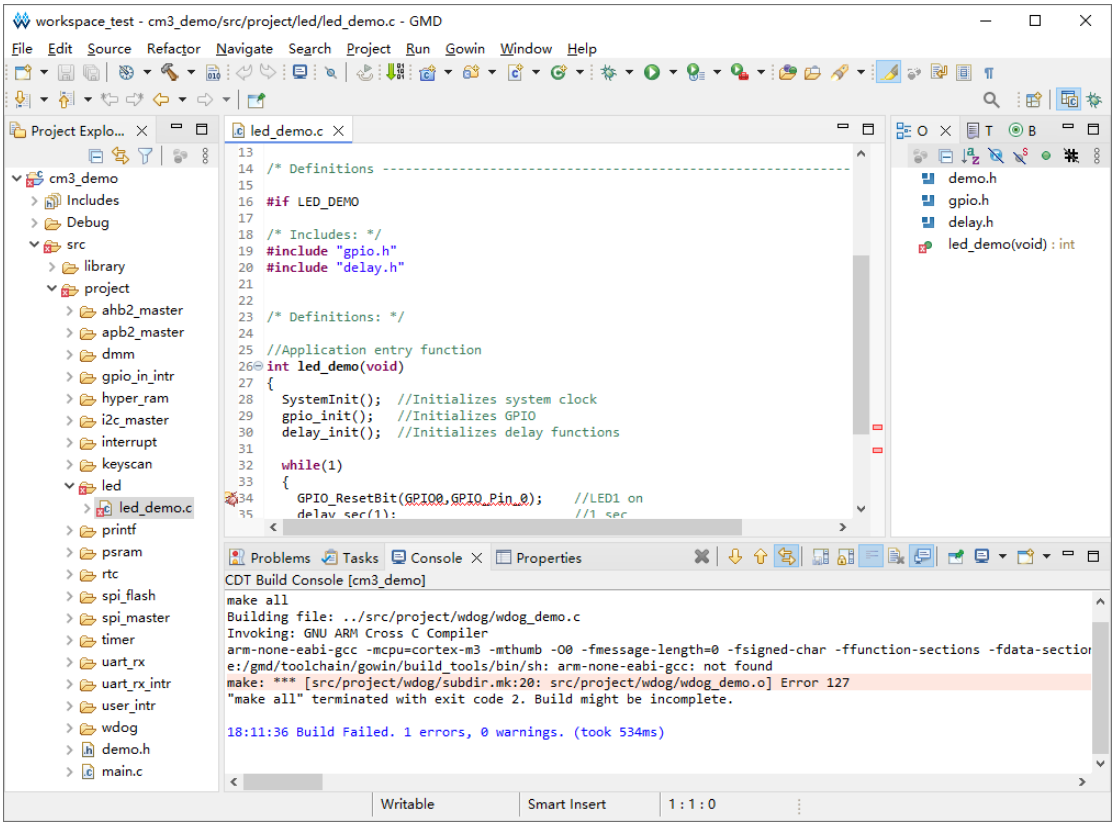
将旧版 GMD 中的工程导入到 GMD 2025.01 新版软件中时，或者使用旧版 MCU SDK（旧版 SDK 指的是在 GMD 2025.01 发布之前所发布过的 SDK）所创建的工程，因为工程配置使用的是旧版 GCC，当找不到对应的软件工具链时，会出现编译报错等问题，导致软件工程无法正常使用。

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。
2. 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。

图 7-1 Arm 工程导入后编译报错

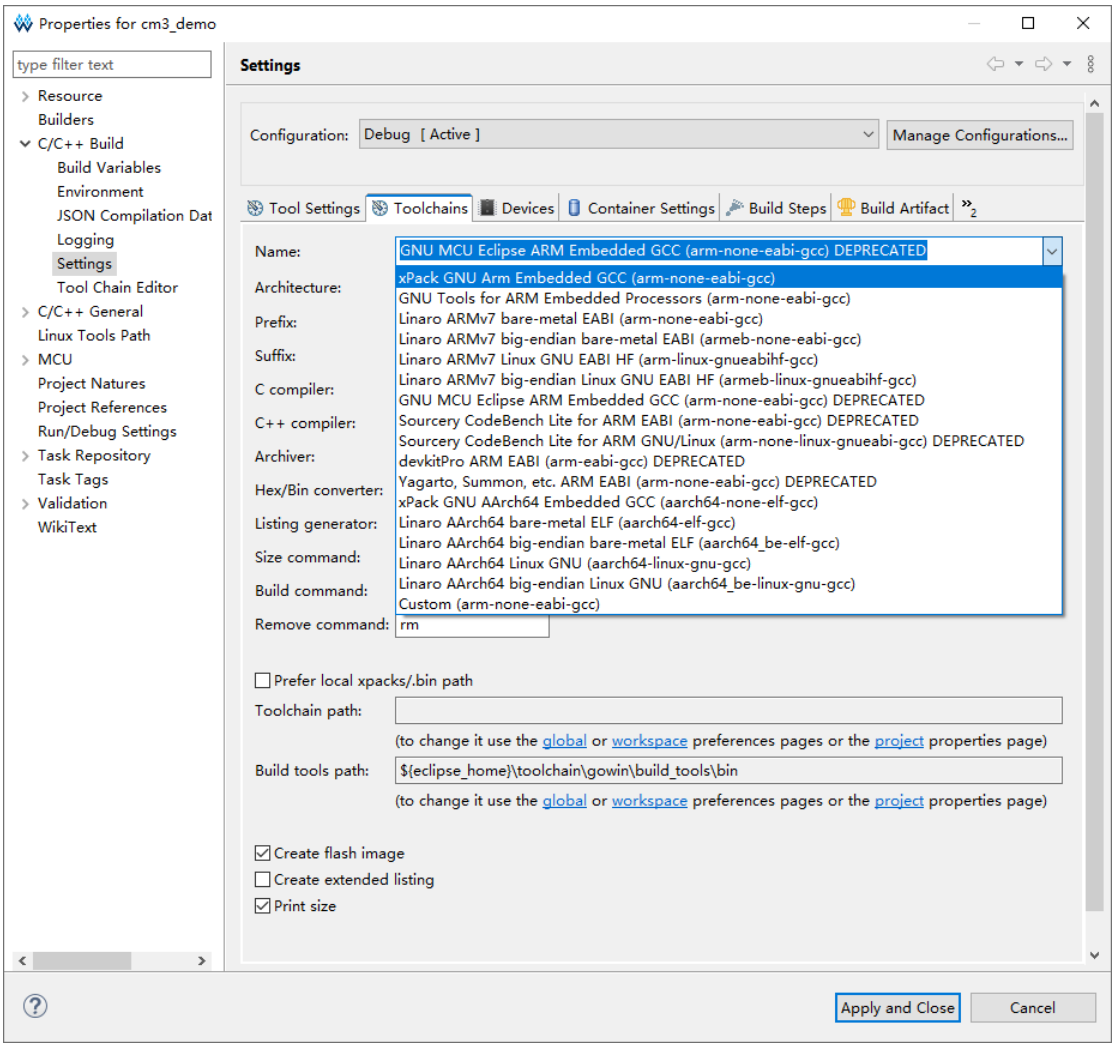


3. 修改软件工具链，重新选择可用的 Arm 工具链。

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。

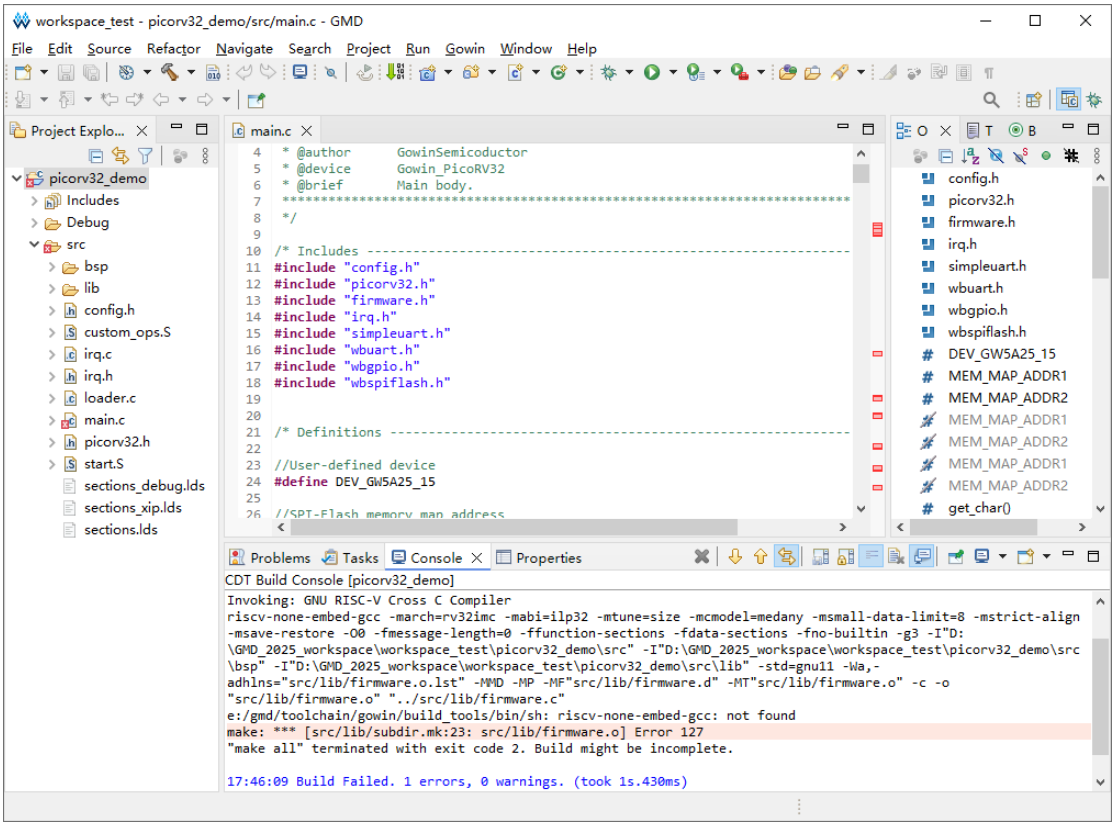
图 7-2 软件工具链设置



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。
2. 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。

图 7-3 RISC-V 工程导入后编译报错

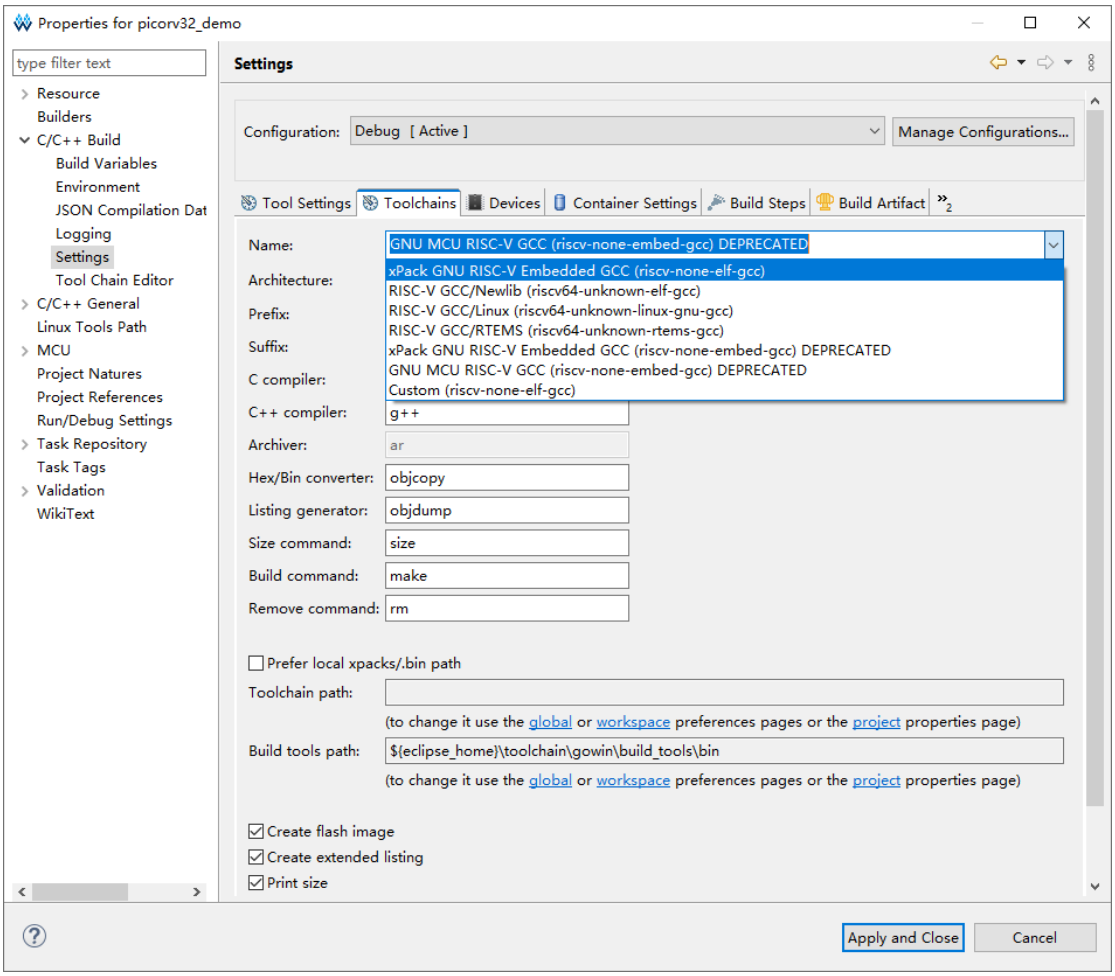


3. 修改软件工具链，重新选择可用的 RISC-V 工具链。

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。

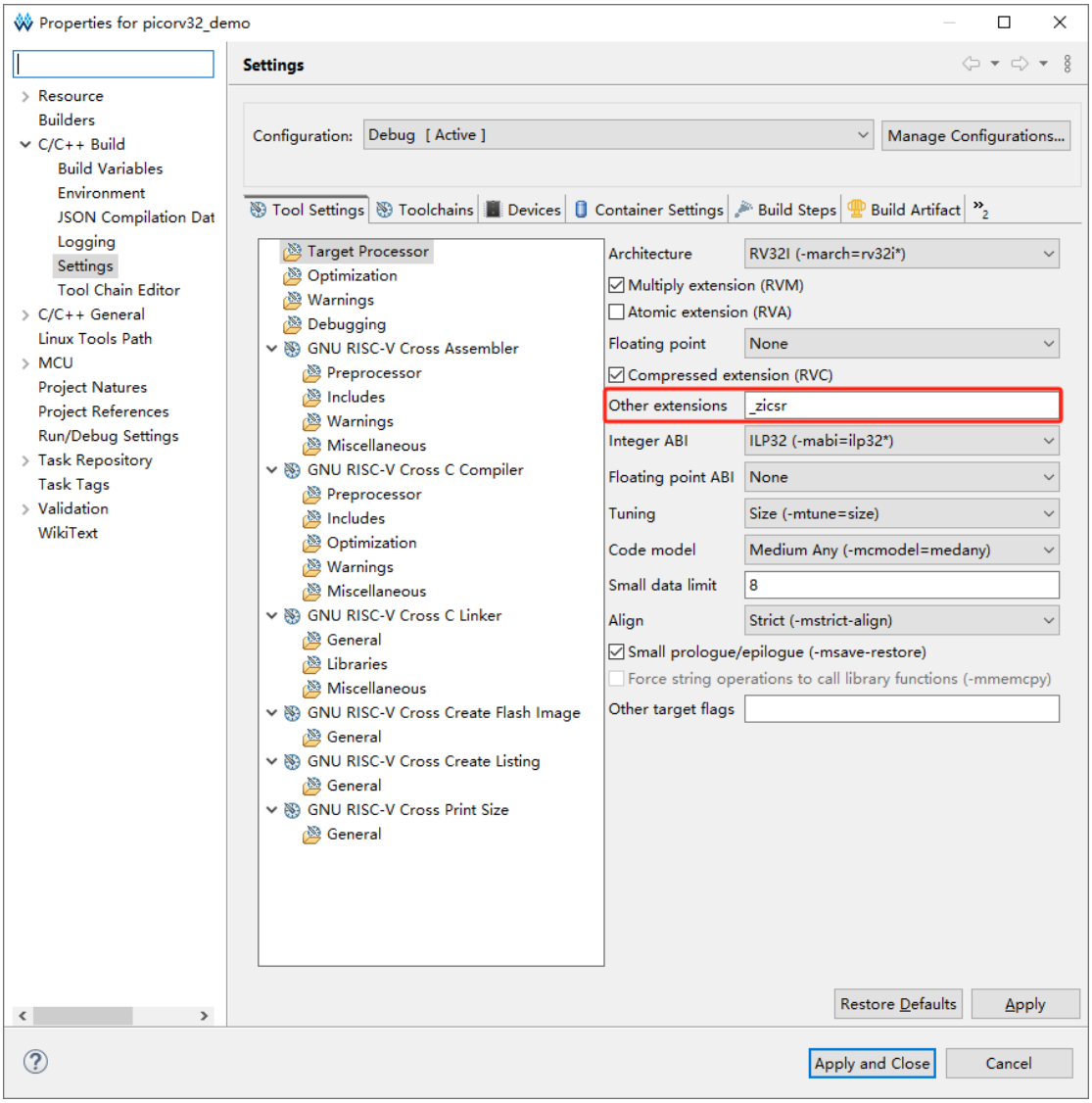
图 7-4 RISC-V 工具链设置



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。

图 7-5 编译选项设置


Properties for picorv32_demo
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
Settings
Configuration: Debug [Active]
Manage Configurations...
Tool Settings
Toolchains
Devices
Container Settings
Build Steps
Build Artifact
Target Processor
Optimization
Warnings
Debugging
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
Libraries
Miscellaneous
GNU RISC-V Cross Create Flash Image
General
GNU RISC-V Cross Create Listing
General
GNU RISC-V Cross Print Size
General
Architecture
RV32I (-march=rv32i*)
Multiply extension (RVM)
Atomic extension (RVA)
Floating point
None
Compressed extension (RVC)
Other extensions
_zicsr
Integer ABI
ILP32 (-mabi=ilp32*)
Floating point ABI
None
Tuning
Size (-mtune=size)
Code model
Medium Any (-mcmodel=medany)
Small data limit
8
Align
Strict (-mstrict-align)
Small prologue/epilogue (-msave-restore)
Force string operations to call library functions (-mmemcpy)
Other target flags
Restore Defaults
Apply
Apply and Close
Cancel

8 编译工程

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。

图 8-1 Arm 软件工具链

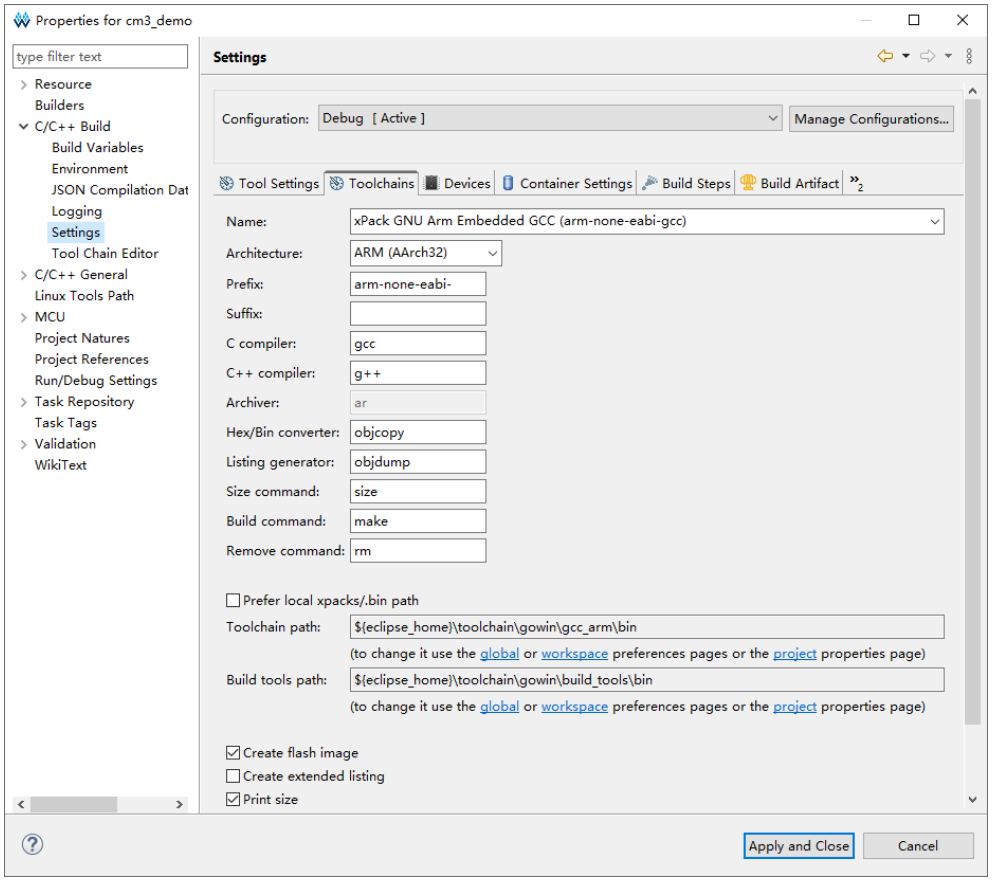
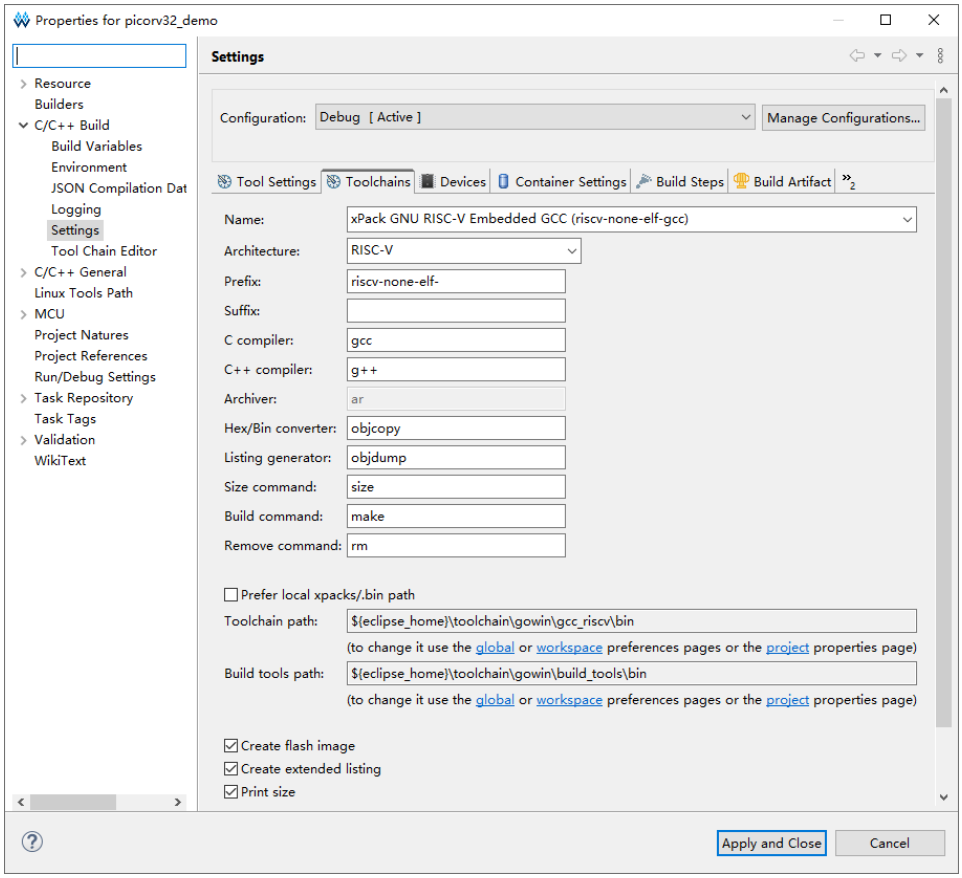


图 8-2 RISC-V 软件工具链



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。

图 8-3 Arm MCU 工程设置

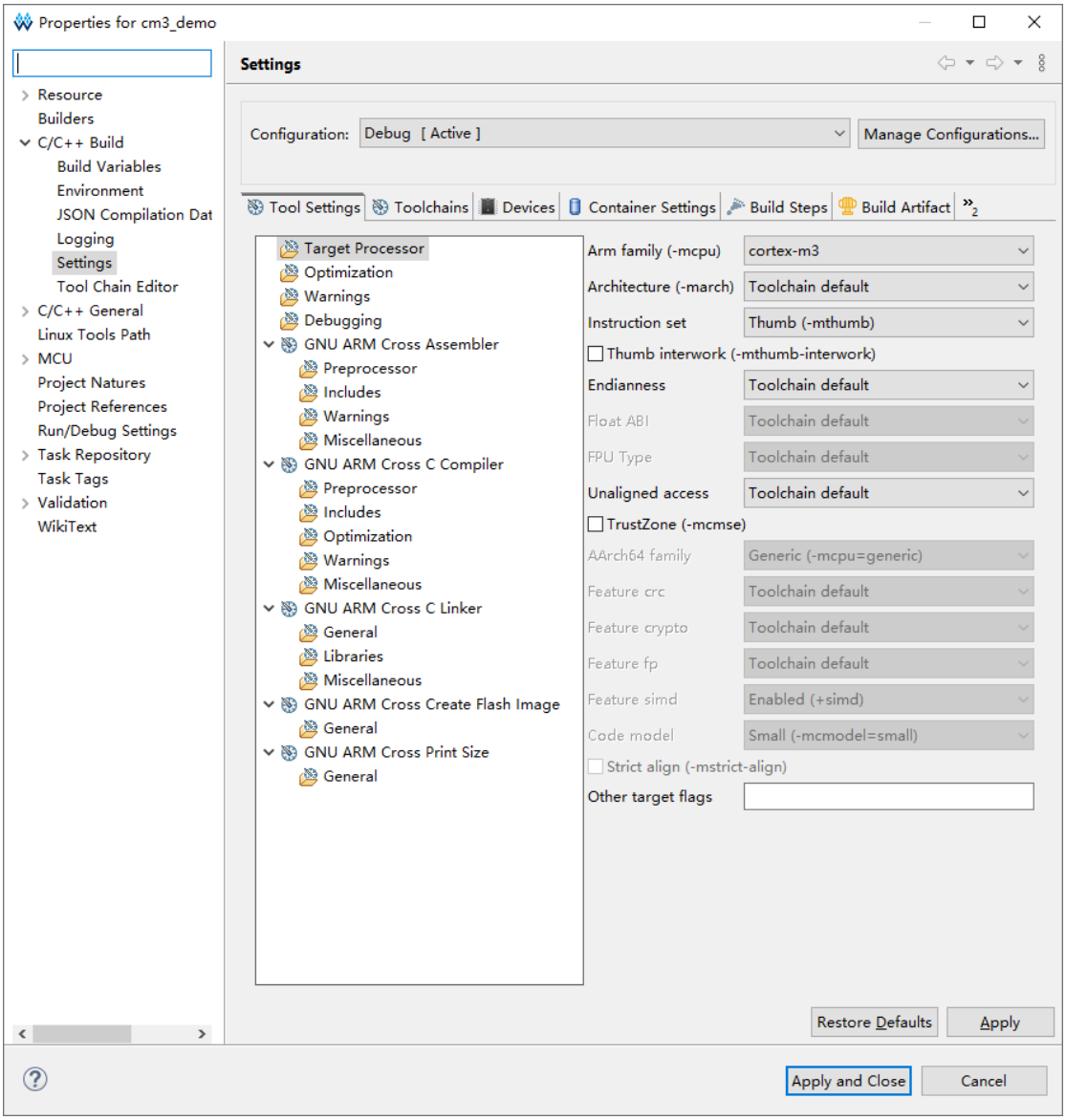
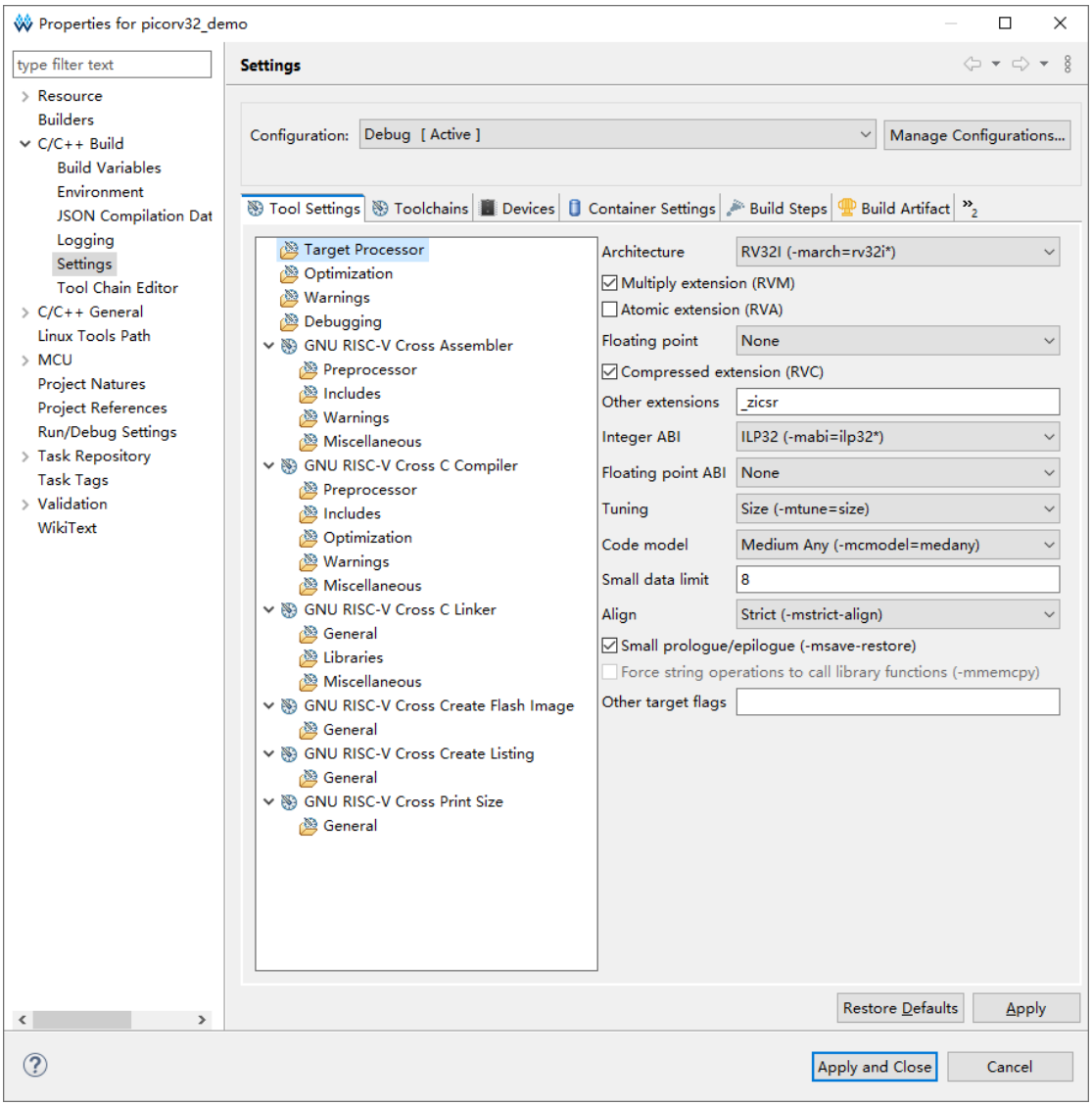


图 8-4 RISC-V MCU 工程设置



Arm MCU 和 RISC-V MCU 工程设置如下所述。

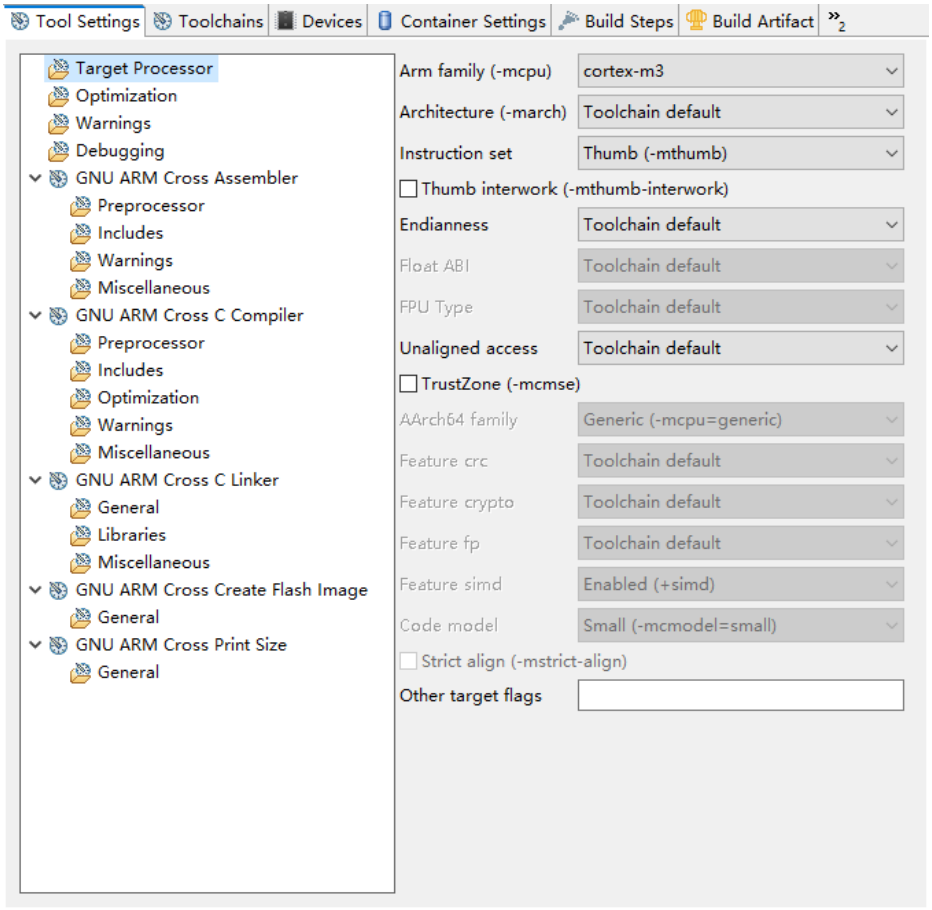
8.2.1 Arm MCU 工程设置

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。

图 8-5 Target Processor



选项设置如表 8-1 所示。

表 8-1 Target Processor

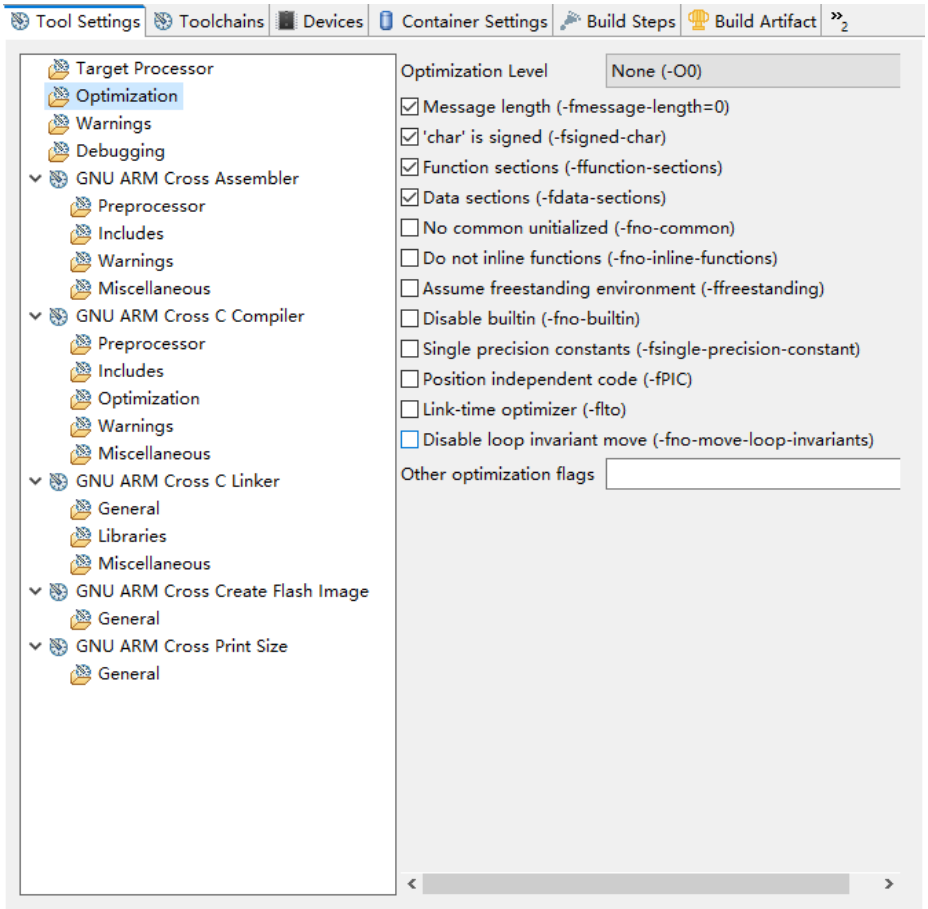
选项	设置	说明
Arm family (-mcpu)	cortex-m3	Arm 内核型号，已支持 Cortex-M1/M3/M4
Architecture (-march)	Toolchain default	Arm 架构
Instruction set	Thumb (-mthumb)	Arm 指令集
Thumb interwork (-mthumb)	-	Thumb 互操作，在同一个可执行文件中混合使用 32-bit 和 16-bit Thumb 指令集
Endianness	Toolchain default	字节序，例如 Little endian 或 Big endian
Float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

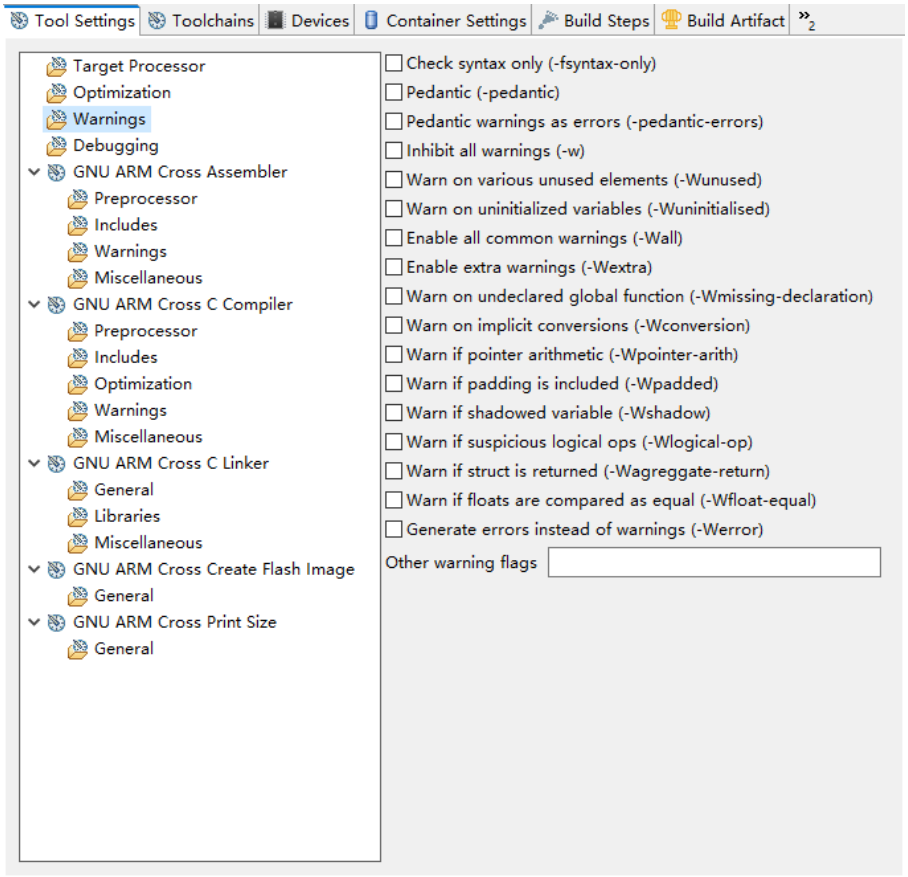
选项	设置
Optimization Level	None (-O0)
Message length (-fmessage-length = 0)	√
‘char’ is signed (-fsigned-char)	√

选项	设置
Function sections (-ffunction-sections)	√
Data sections (-fdata-sections)	√
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。

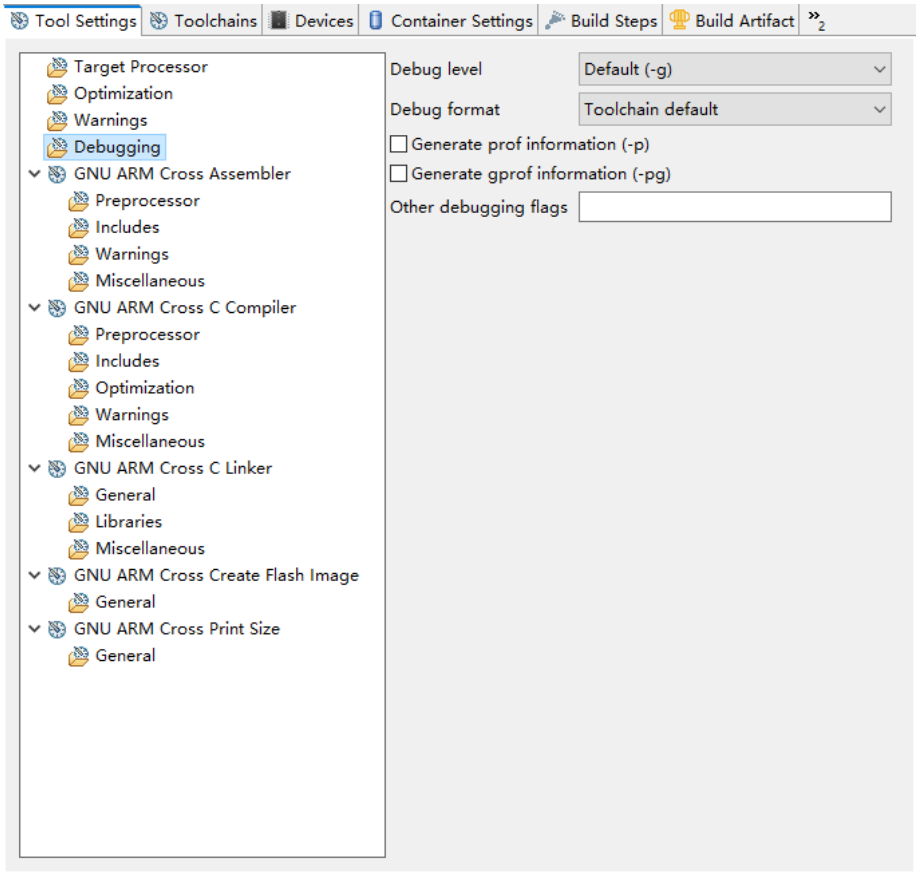
图 8-7 Warnings



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。

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	-

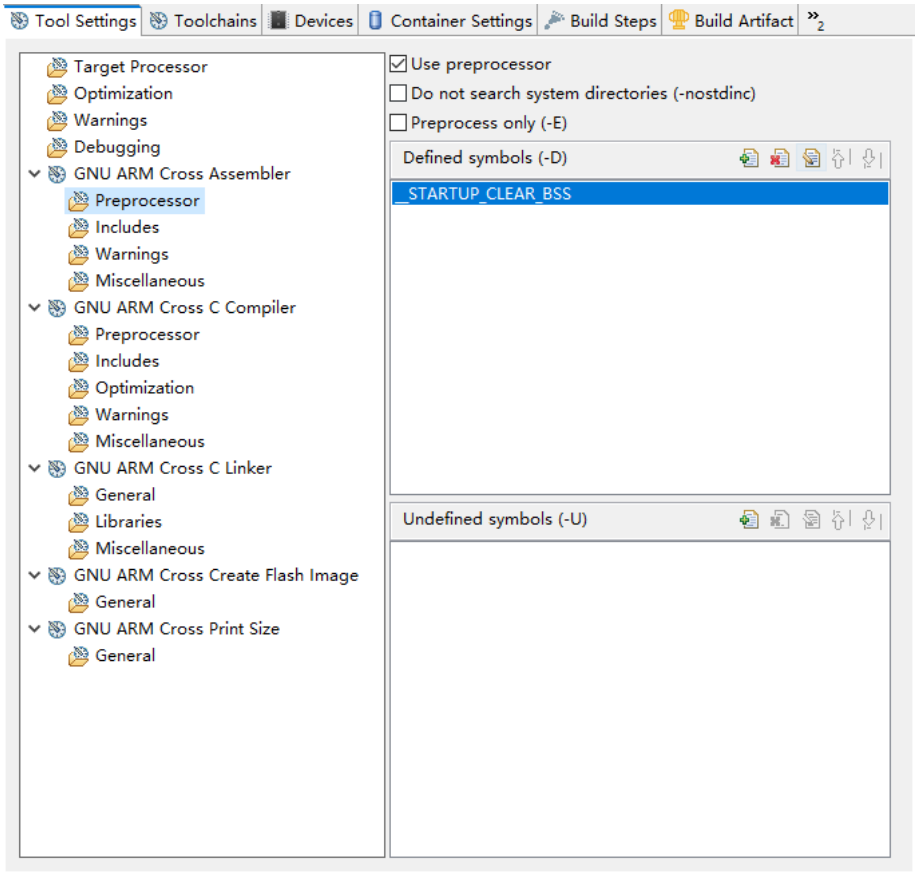
GNU ARM Cross Assembler

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。

图 8-9 GNU ARM Cross Assembler > Preprocessor



选项设置如表 8-4 所示。

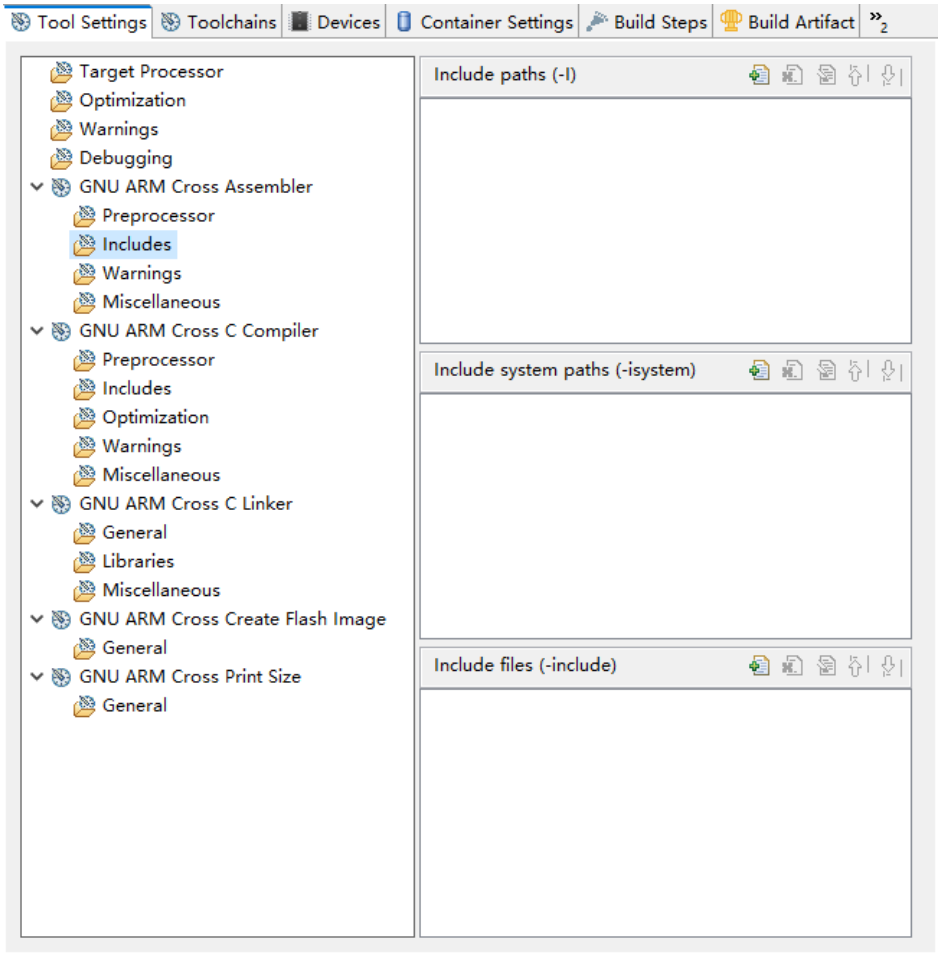
表 8-4 GNU ARM Cross Assembler > Preprocessor

选项	设置
Use preprocessor	√
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。

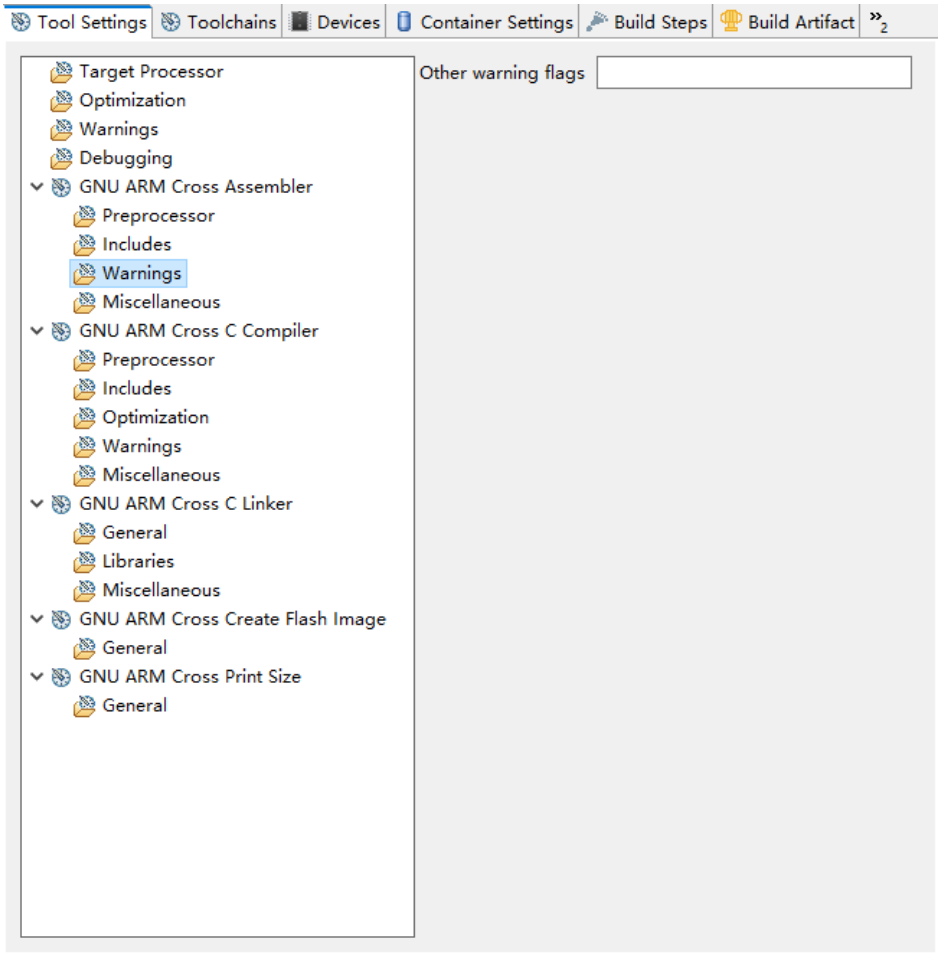
图 8-10 GNU ARM Cross Assembler > Includes



3. Warnings

按需填写额外的警告选项如图 8-11 所示。

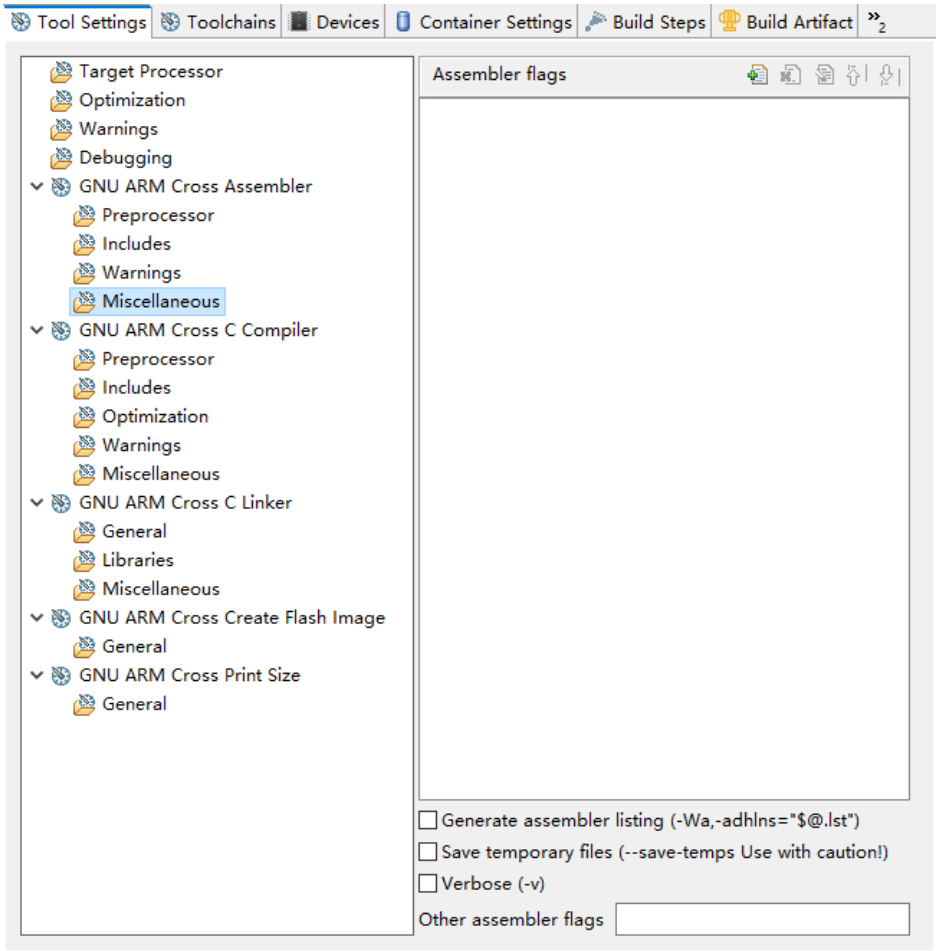
图 8-11 GNU ARM Cross Assembler > Warnings



4. Miscellaneous

按需填写额外的编译器选项如图 8-12 所示。

图 8-12 GNU ARM Cross Assembler > Miscellaneous



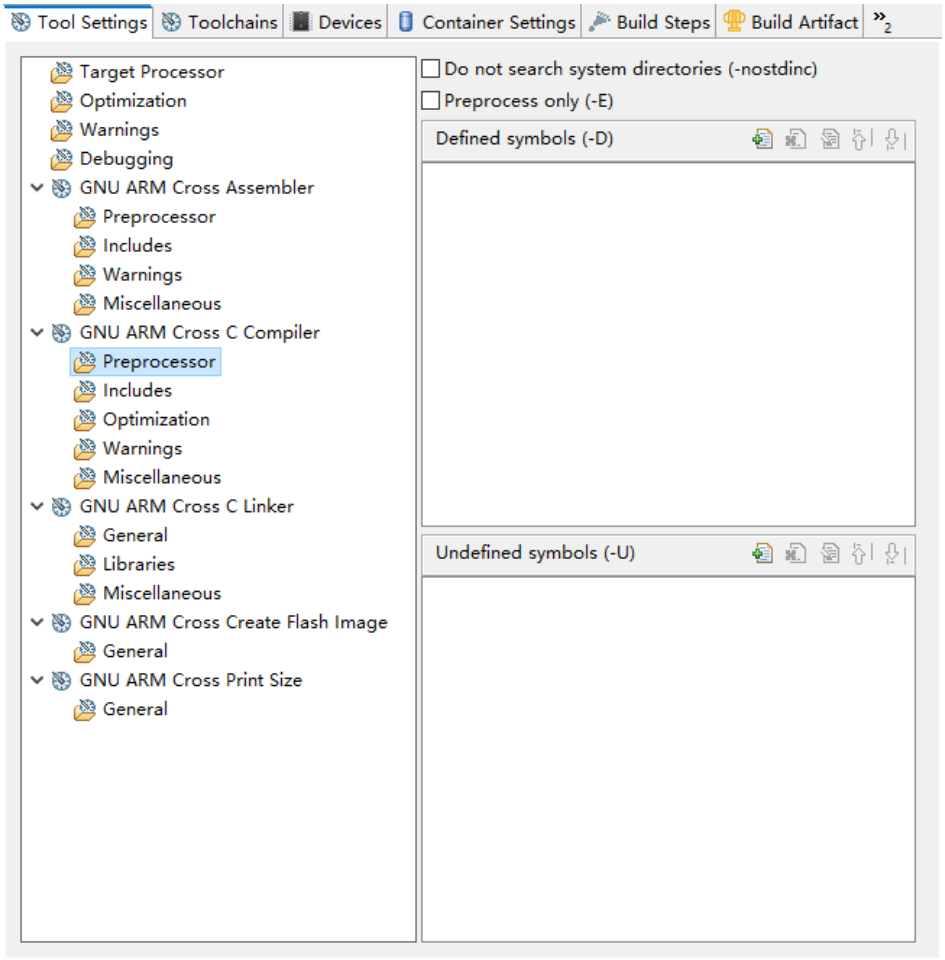
GNU ARM Cross C Compiler

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。

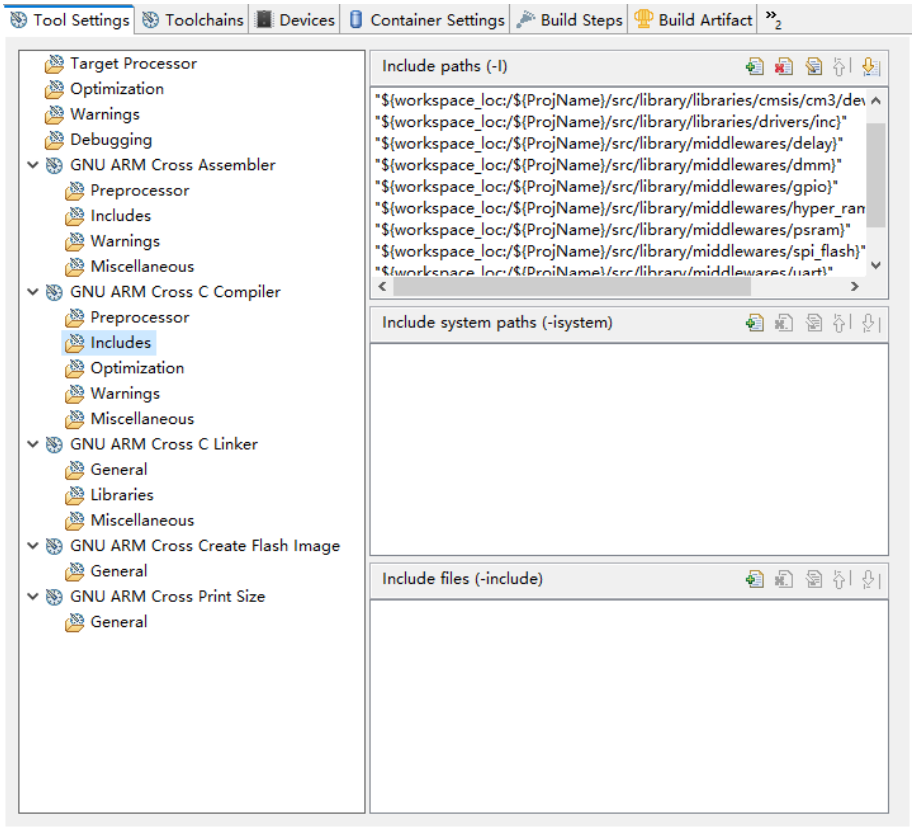
图 8-13 GNU ARM Cross C Compiler > Preprocessor



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。

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

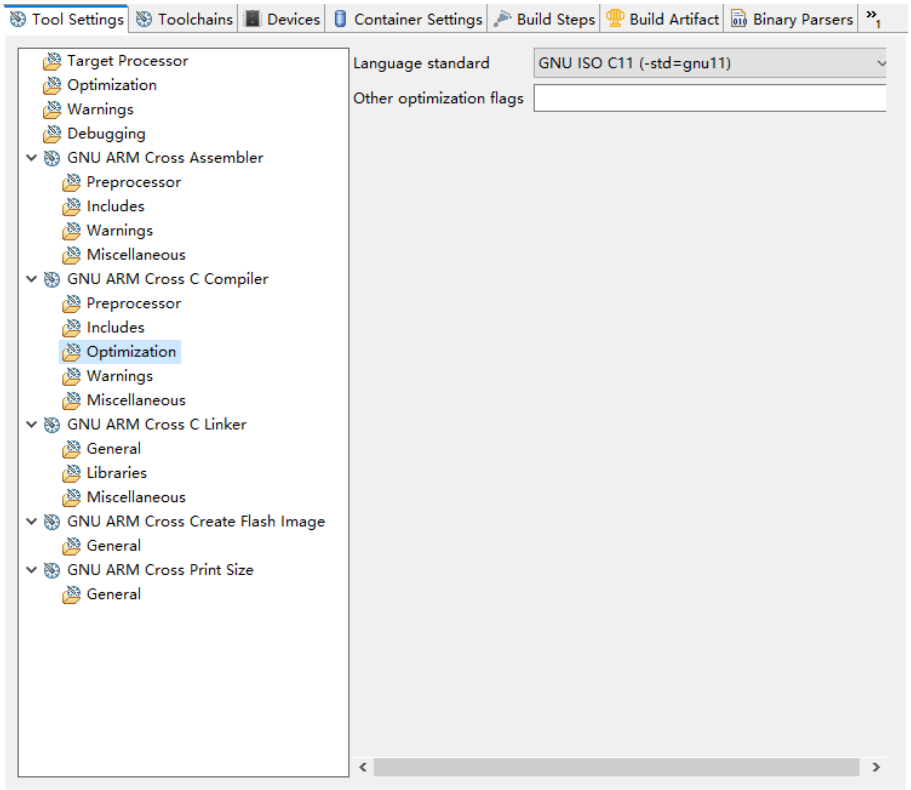
选项	设置
Include paths (-I)	“${workspace_loc:/${ProjName}/src/library/libraries/cmsis/cm3/core_support/gmd}” “${workspace_loc:/${ProjName}/src/library/libraries/cmsis/cm3/device_support}” “${workspace_loc:/${ProjName}/src/library/libraries/drivers/inc}” “${workspace_loc:/${ProjName}/src/library/middlewares/delay}” “${workspace_loc:/${ProjName}/src/library/middlewares/dmm}” “${workspace_loc:/${ProjName}/src/library/middlewares/gpio}” “${workspace_loc:/${ProjName}/src/library/middlewares/hyper_ram}” “${workspace_loc:/${ProjName}/src/library/middlewares/psram}”

选项	设置
	“${workspace_loc:/${ProjName}/src/library/middlewares/spi_flash}” “${workspace_loc:/${ProjName}/src/library/middlewares/uart}” “${workspace_loc:/${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



选项设置如表 8-6 所示。

表 8-6 GNU ARM Cross C Compiler > Optimization

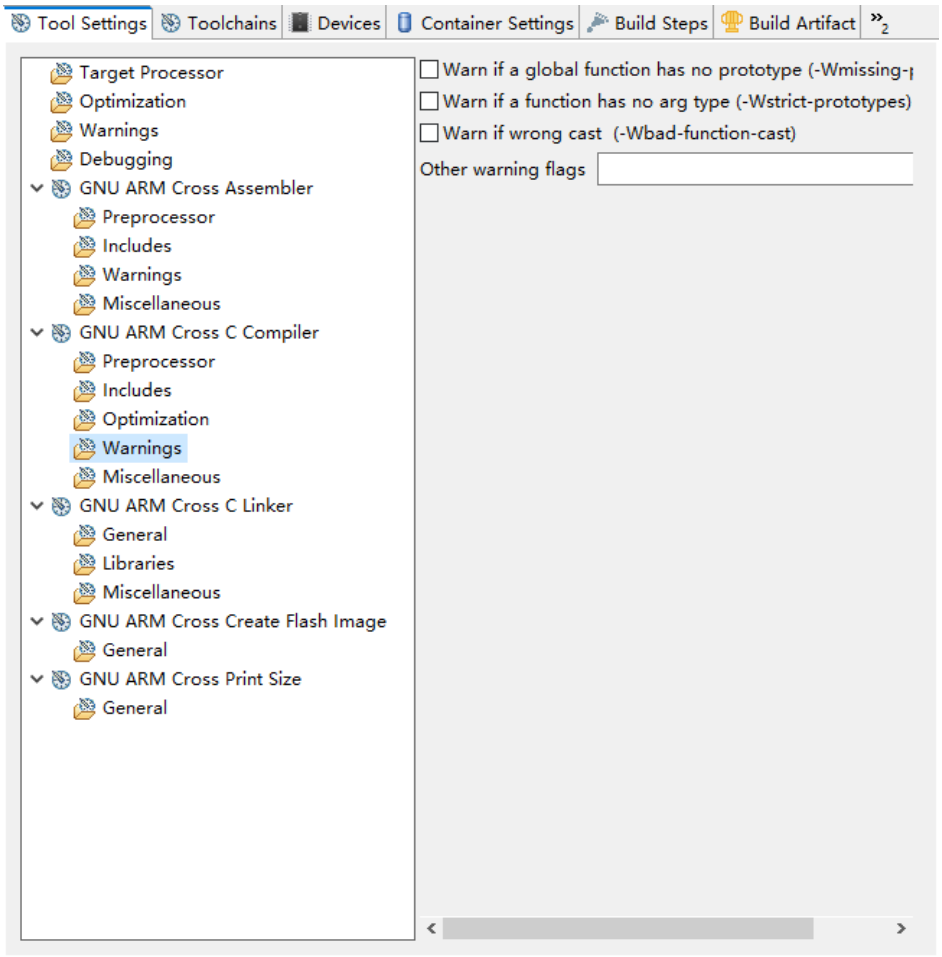
选项	设置
Language standard	GNU ISO C11 (-std=gnu11)

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。

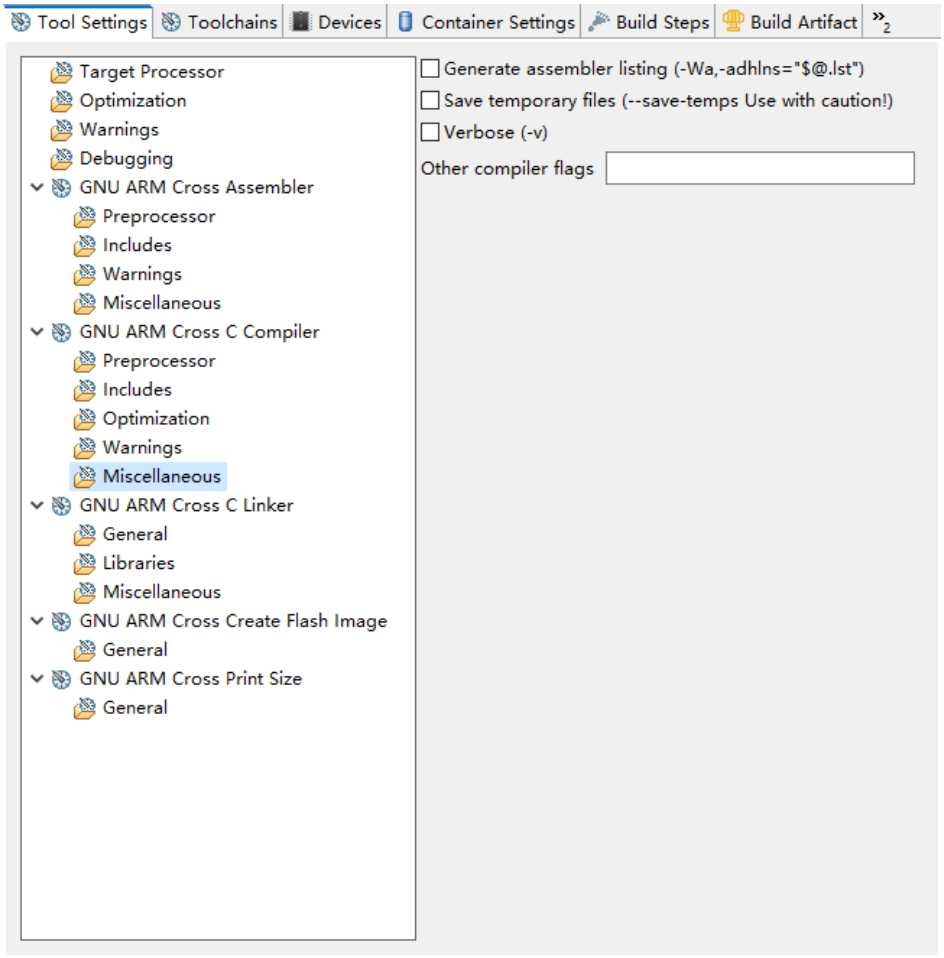
图 8-16 GNU ARM Cross C Compiler > Warnings



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



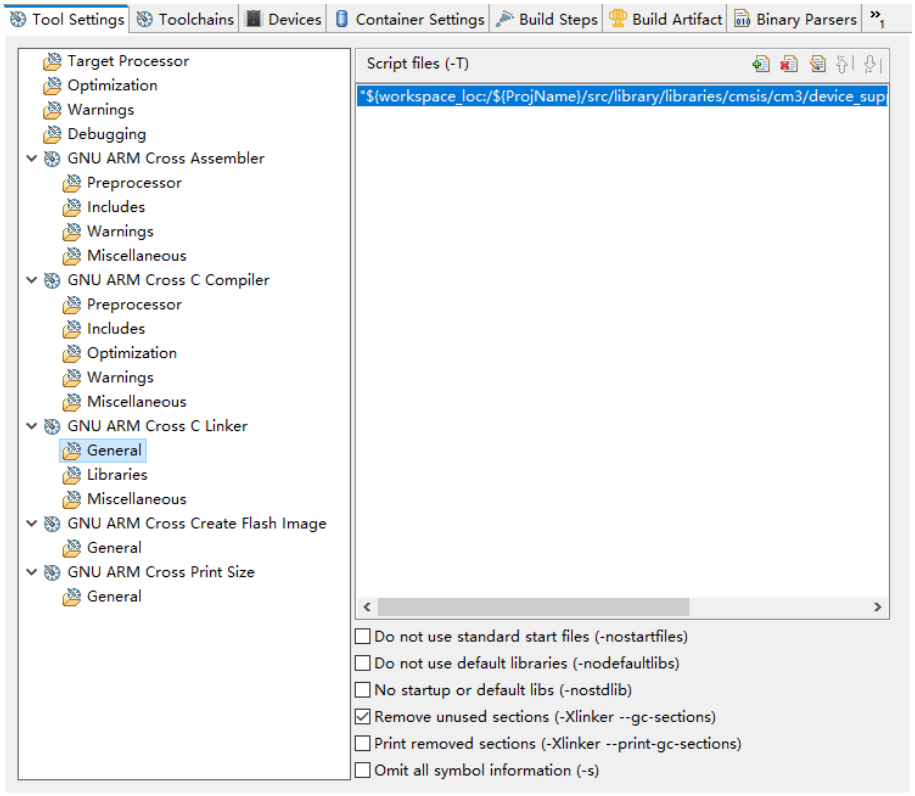
GNU ARM Cross C Linker

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。

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

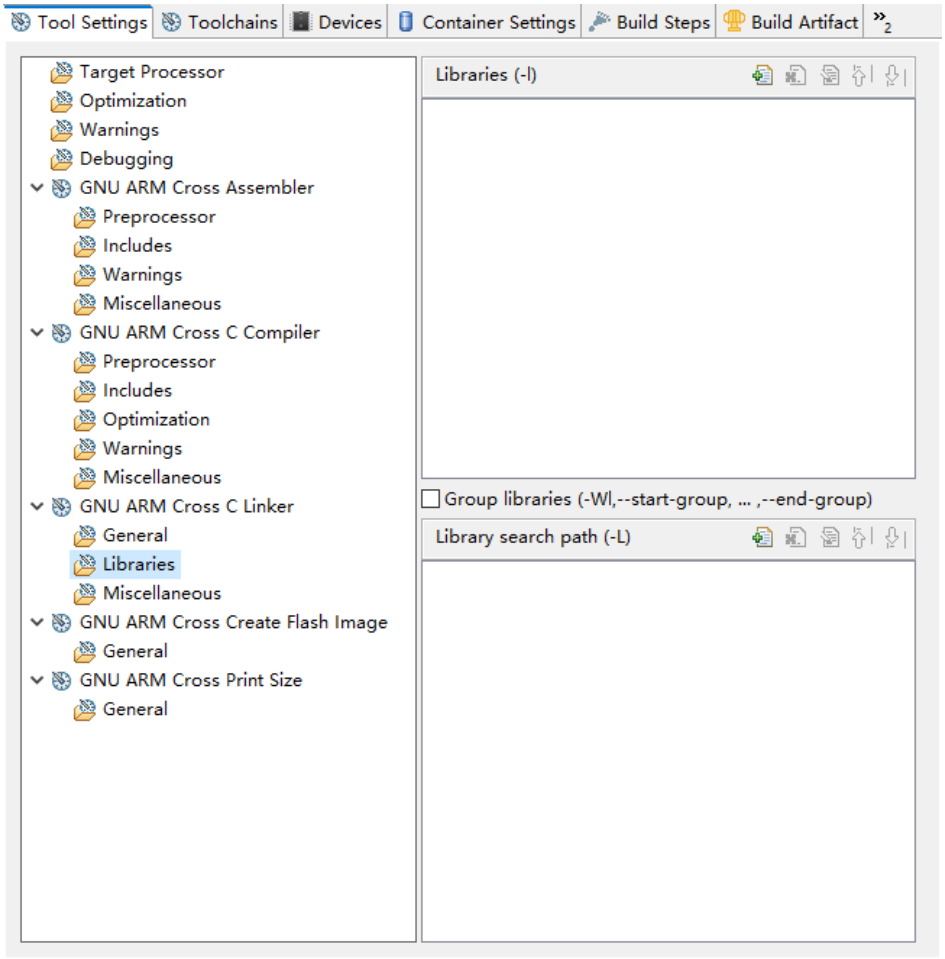
选项	设置
Script files (-T)	“${workspace_loc:/${ProjName}/src/library/libraries/cmsis/cm3/device_support/startup/gmd/linker/gw1ns4c_flash.ld}”
Do not use standard start files (-nostartfiles)	-
Do not use default libraries (-nodefaultlibs)	-
No startup or default libs (-nostdlib)	-
Remove unused sections (-Xlinker --gc-sections)	√
Print removed sections (-Xlinker --print-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 (-l)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。

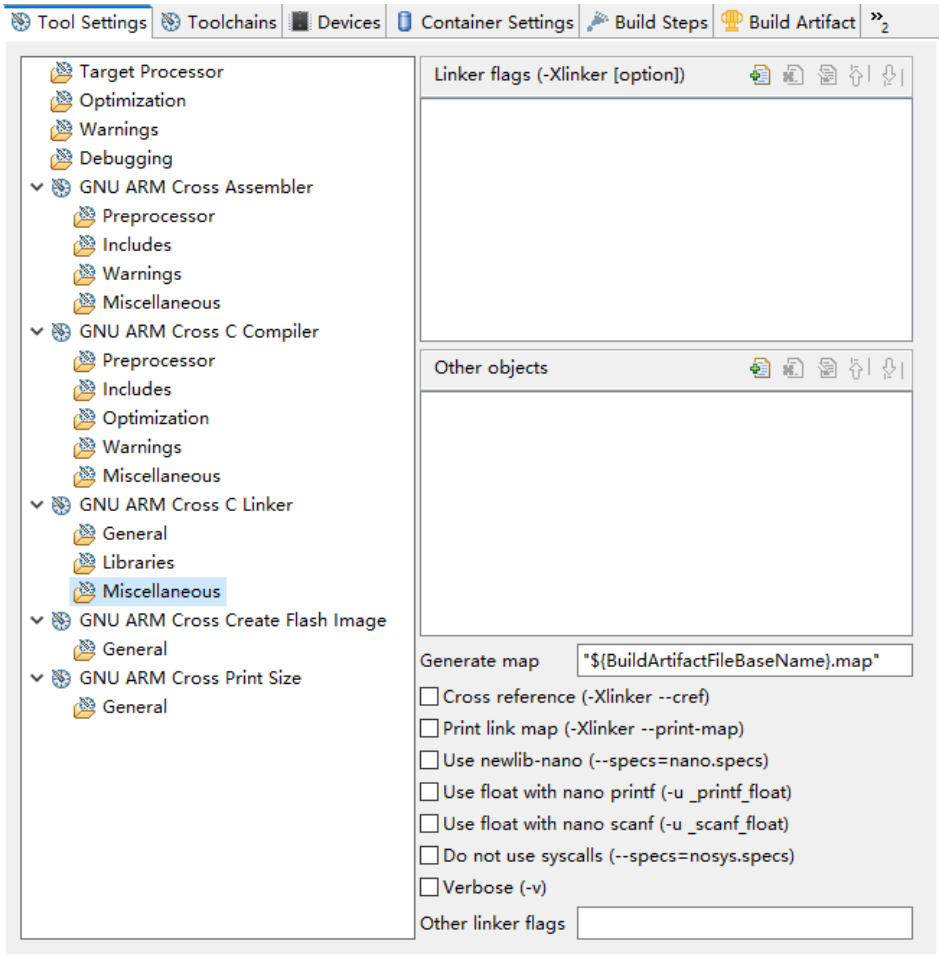
图 8-19 GNU ARM Cross C Linker > Libraries



3. Miscellaneous

按需设置或填写额外的链接选项如图 8-20 所示。

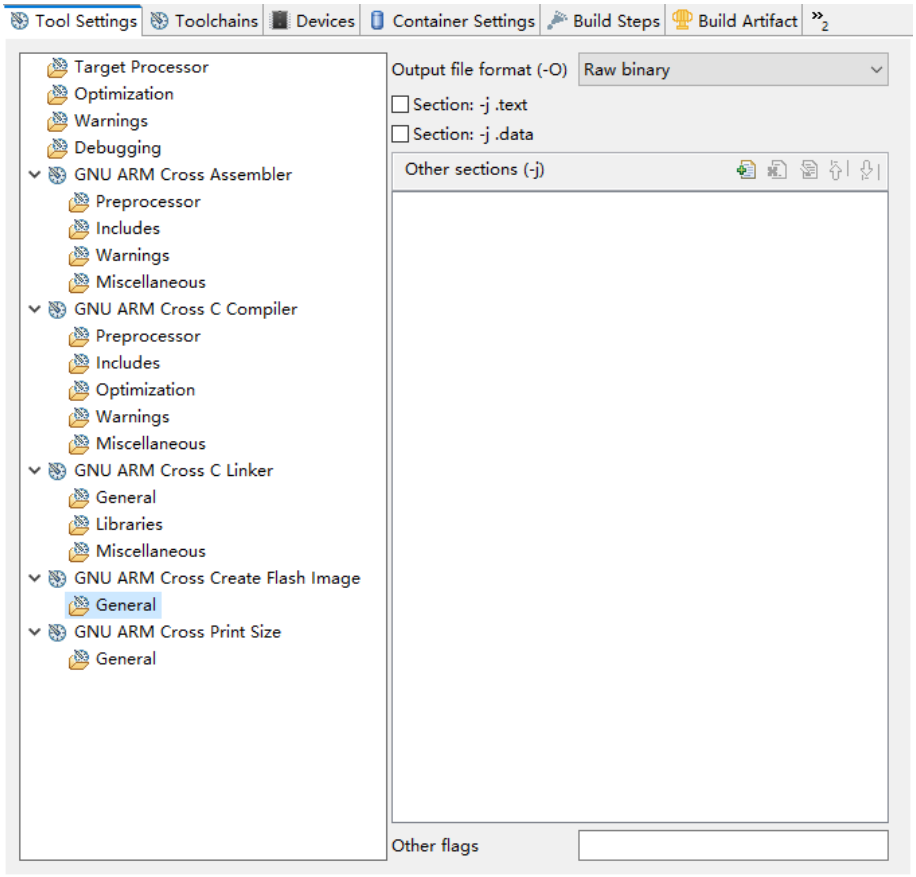
图 8-20 GNU ARM Cross C Linker > Miscellaneous



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。

图 8-21 GNU ARM Cross Create Flash Image



选项设置如表 8-8 所示。

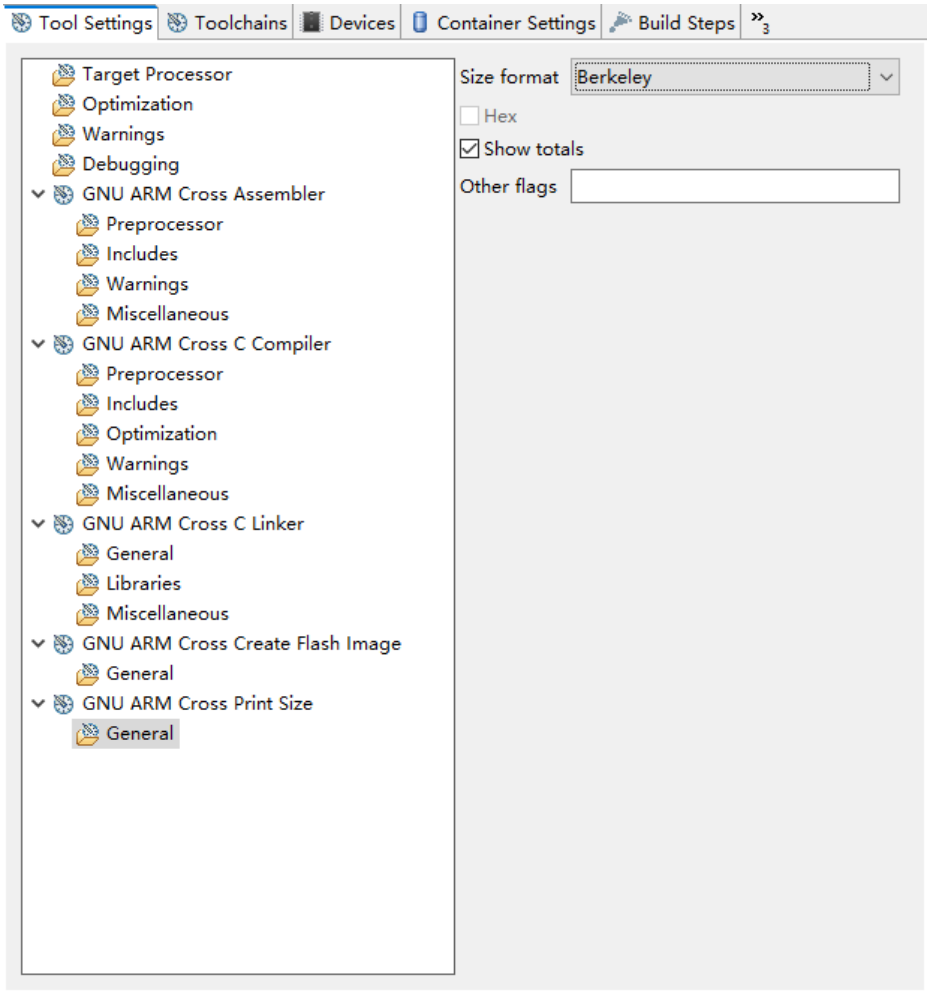
表 8-8 GNU ARM Cross Create Flash Image

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	√
Other flags	-

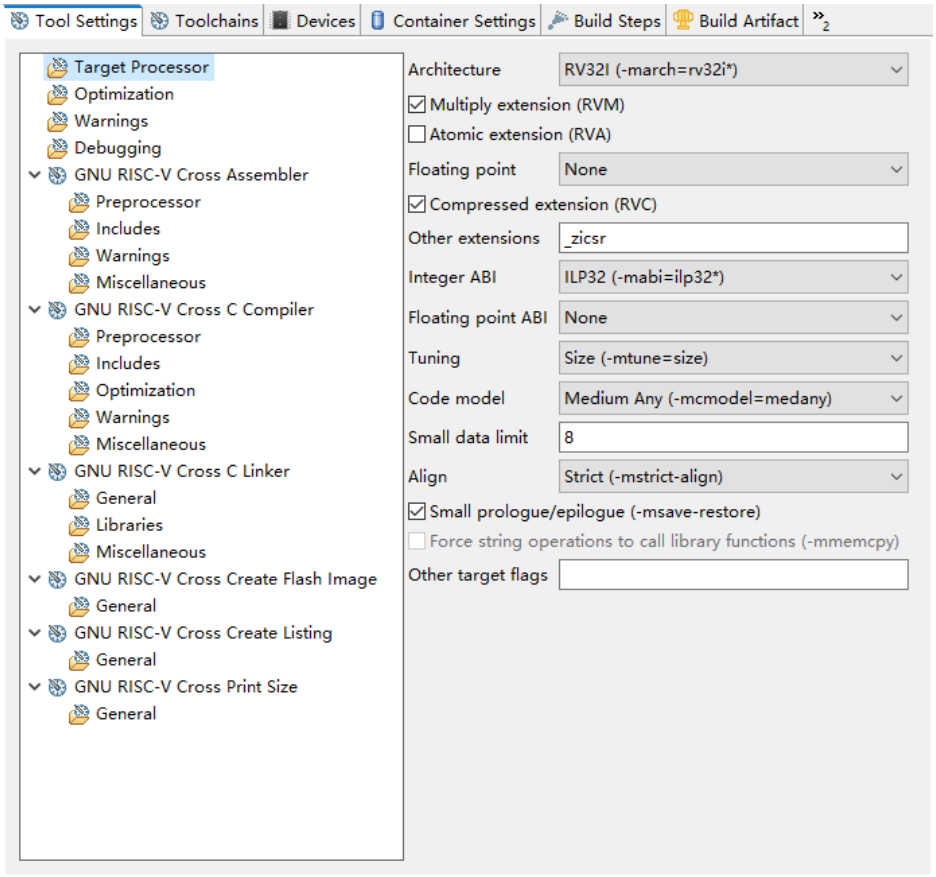
8.2.2 RISC-V MCU 工程设置

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。

图 8-23 Target Processor



选项设置如表 8-10 所示。

表 8-10 Target Processor

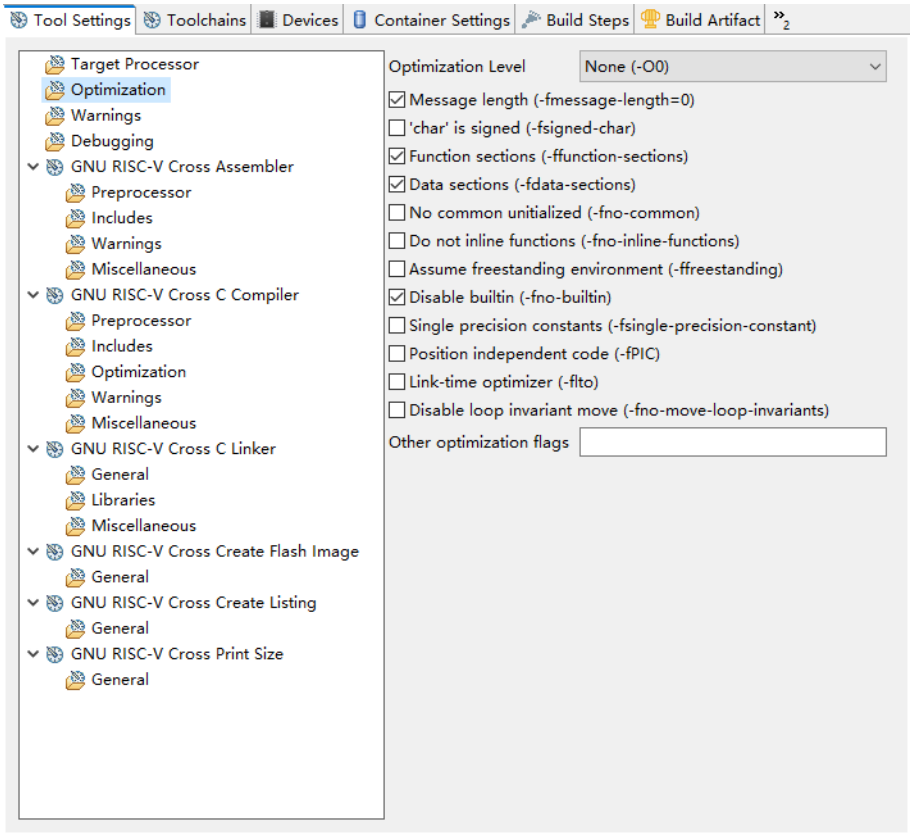
选项	设置	说明
Architecture	RV32I (-march = rv32i)	RISC-V 架构
Multiply extension (RVM)	√	乘除法指令
Atomic extension (RVA)	-	原子指令
Floating point	None	浮点指令
Compressed extension (RVC)	√	压缩指令
Other extensions	_zicsr	额外的扩展指令
Integer ABI	ILP32 (-mabi = ilp32*)	整数运算的 ABI
Floating point ABI	None	浮点运算的 ABI
Tuning	Size (-mtune = 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)	√	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



选项设置如表 8-11 所示。

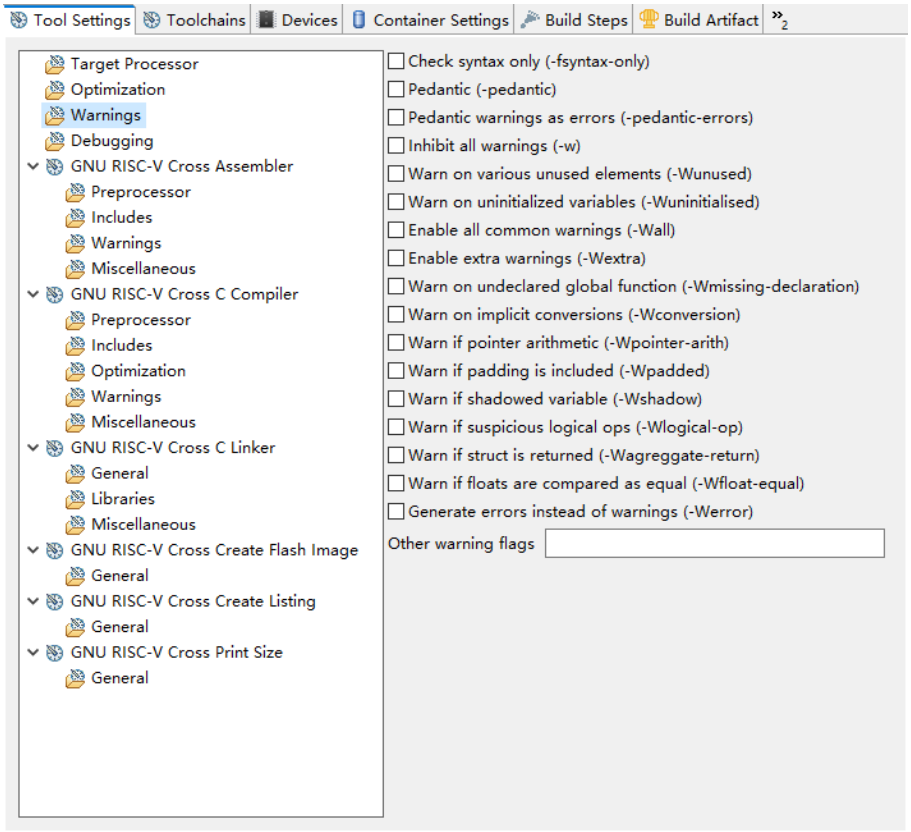
表 8-11 Optimization

选项	设置
Optimization Level	None (-O0)
Message length (-fmessage-length = 0)	√
‘char’ is signed (-fsigned-char)	-
Function sections (-ffunction-sections)	√
Data sections (-fdata-sections)	√
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。

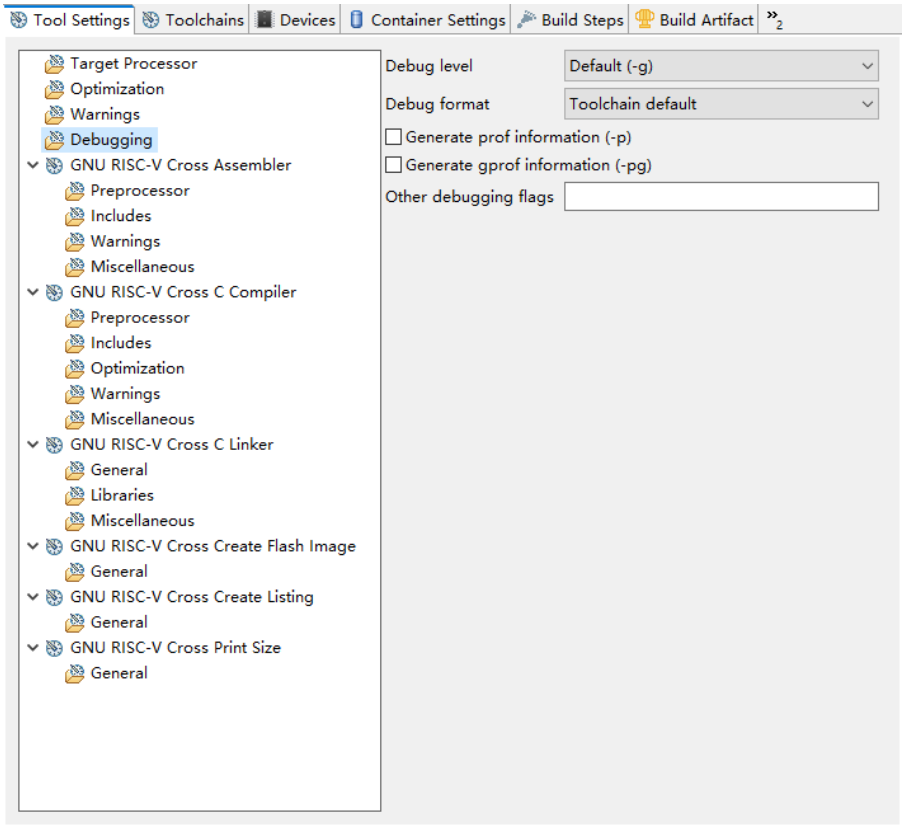
图 8-25 Warnings



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。

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	-

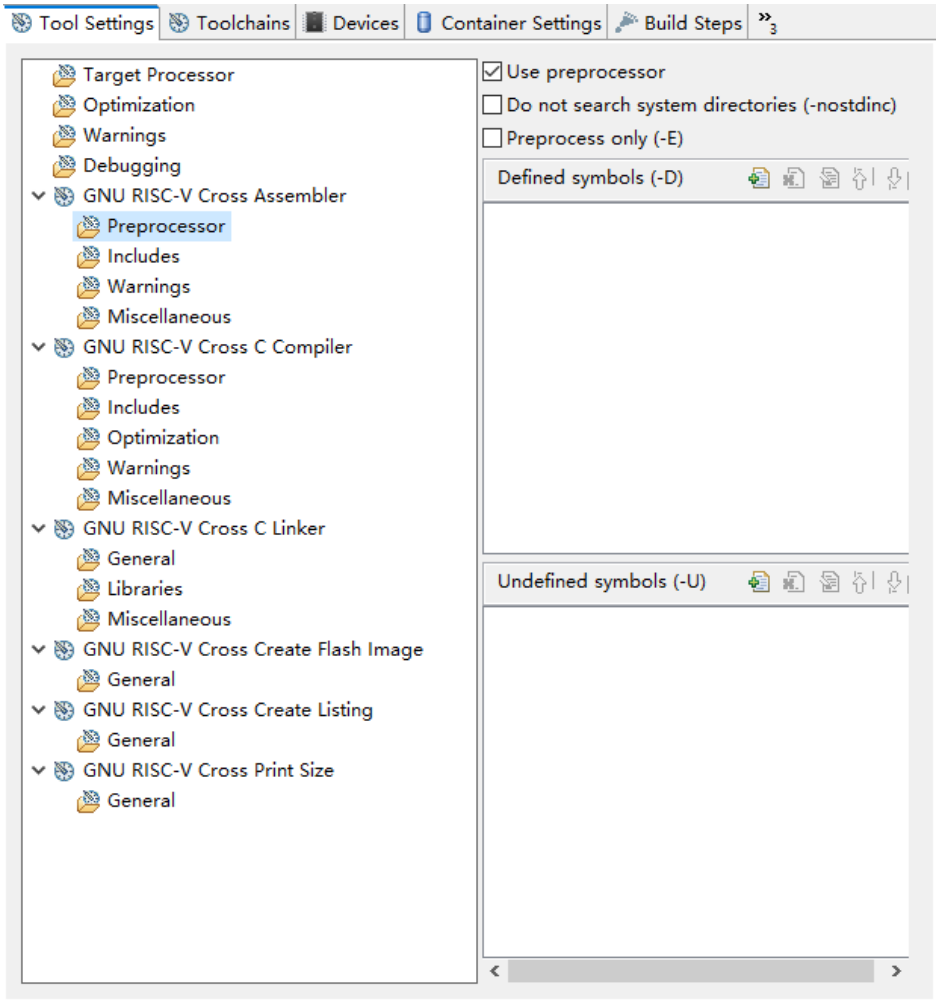
GNU RISC-V Cross Assembler

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。

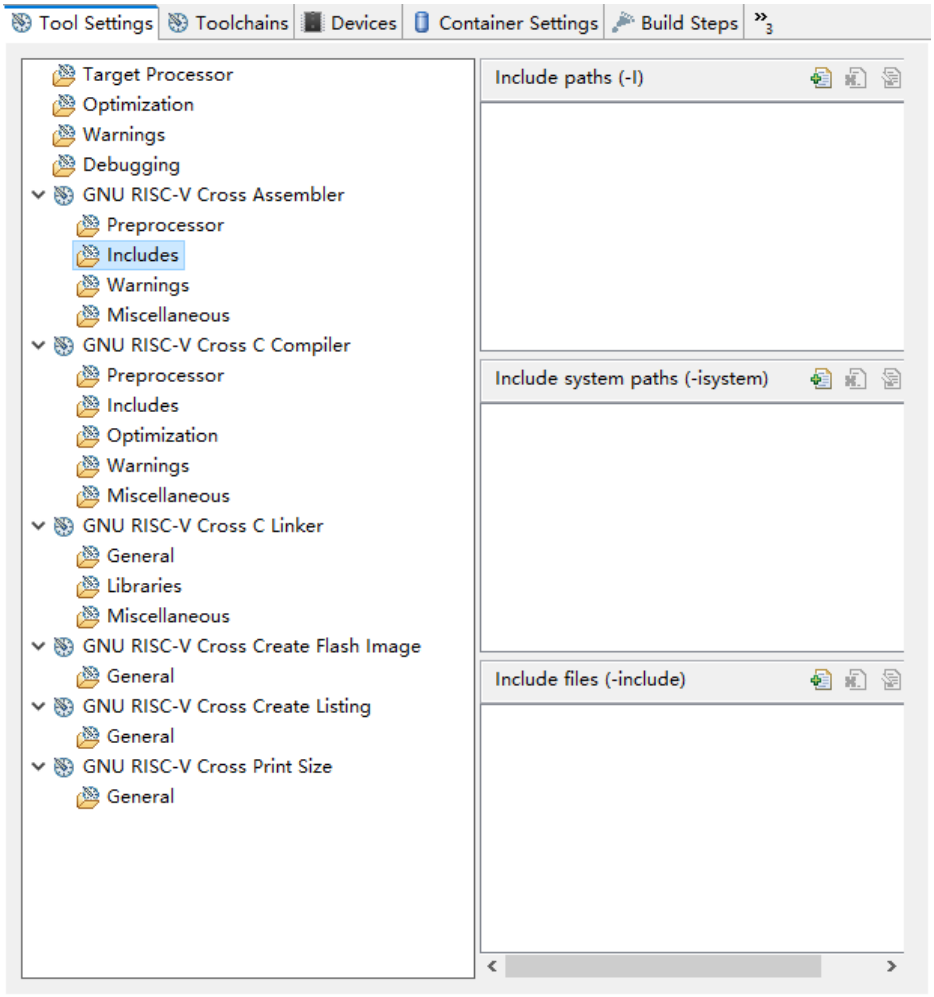
图 8-27 GNU RISC-V Cross Assembler > Preprocessor



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 8-28 所示。

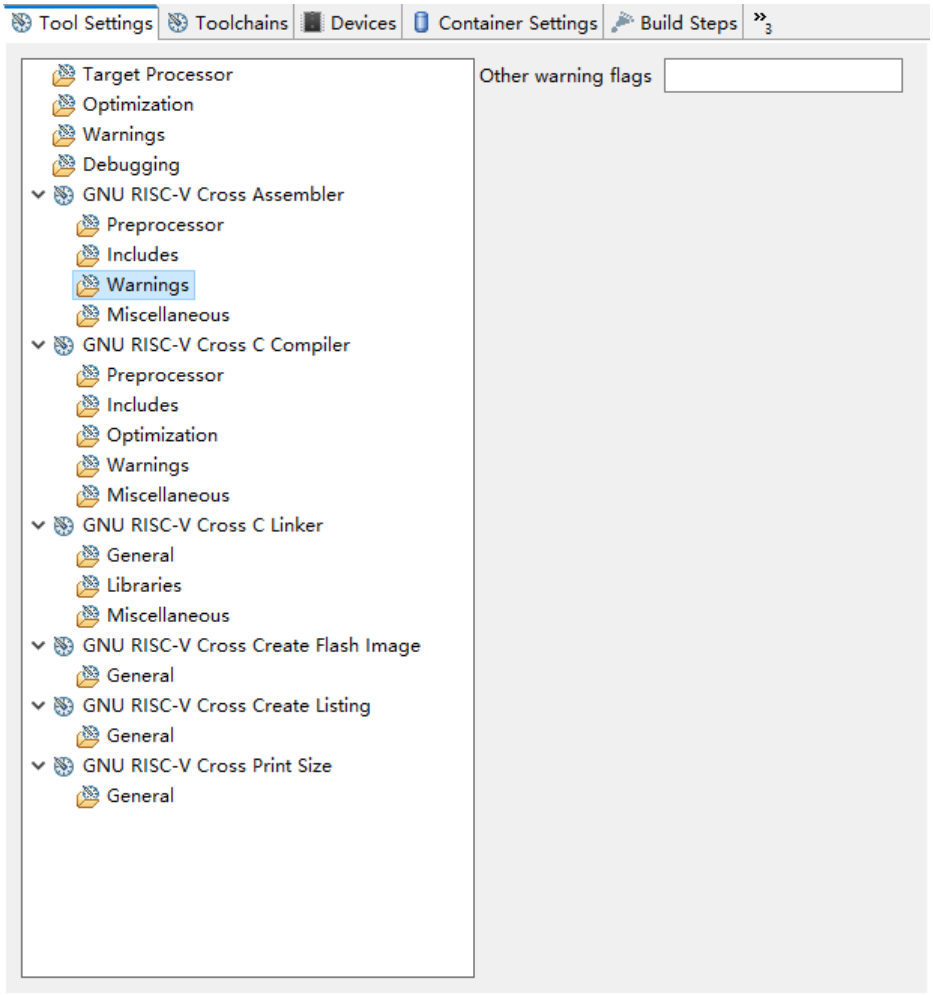
图 8-28 GNU RISC-V Cross Assembler > Includes



3. Warnings

按需填写额外的警告选项如图 8-29 所示。

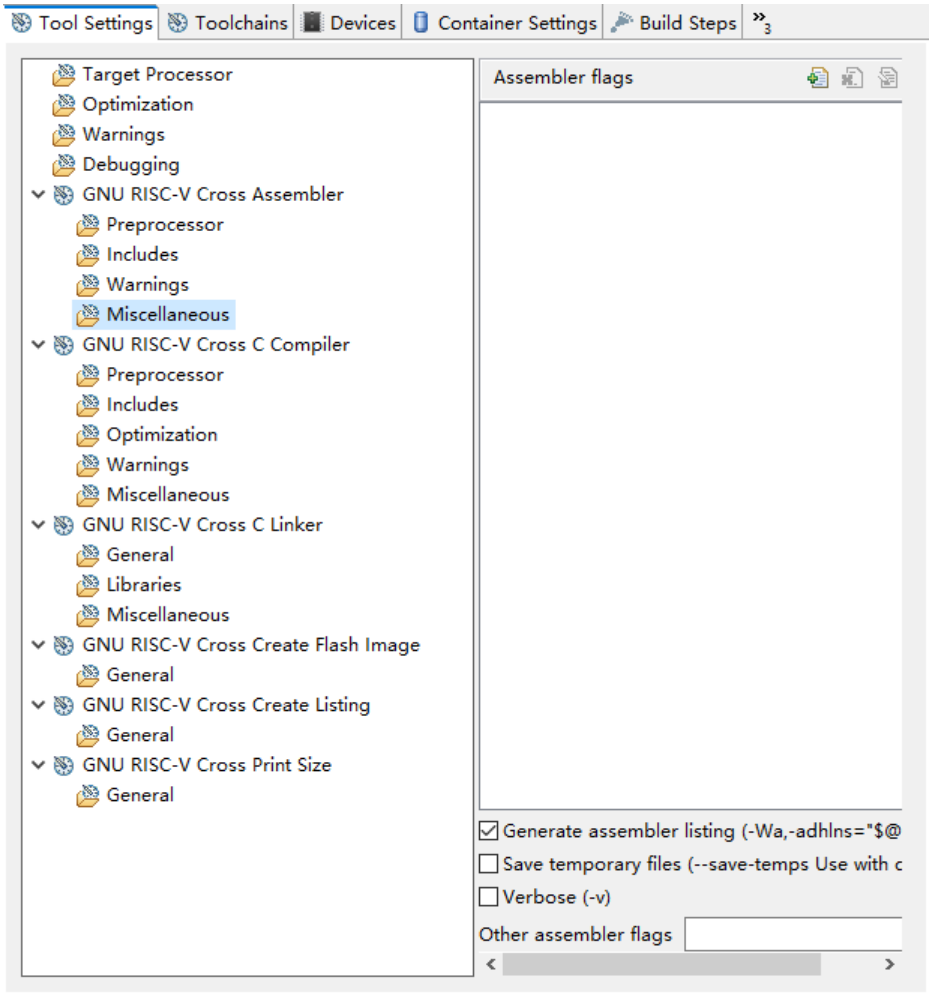
图 8-29 GNU RISC-V Cross Assembler > Warnings



4. Miscellaneous

按需填写额外的编译器选项如图 8-30 所示。

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,-adhlns=”$@.lst”)	√
Save temporary files (--save-temps Use with caution!)	-
Verbose (-v)	-
Other assembler flags	-

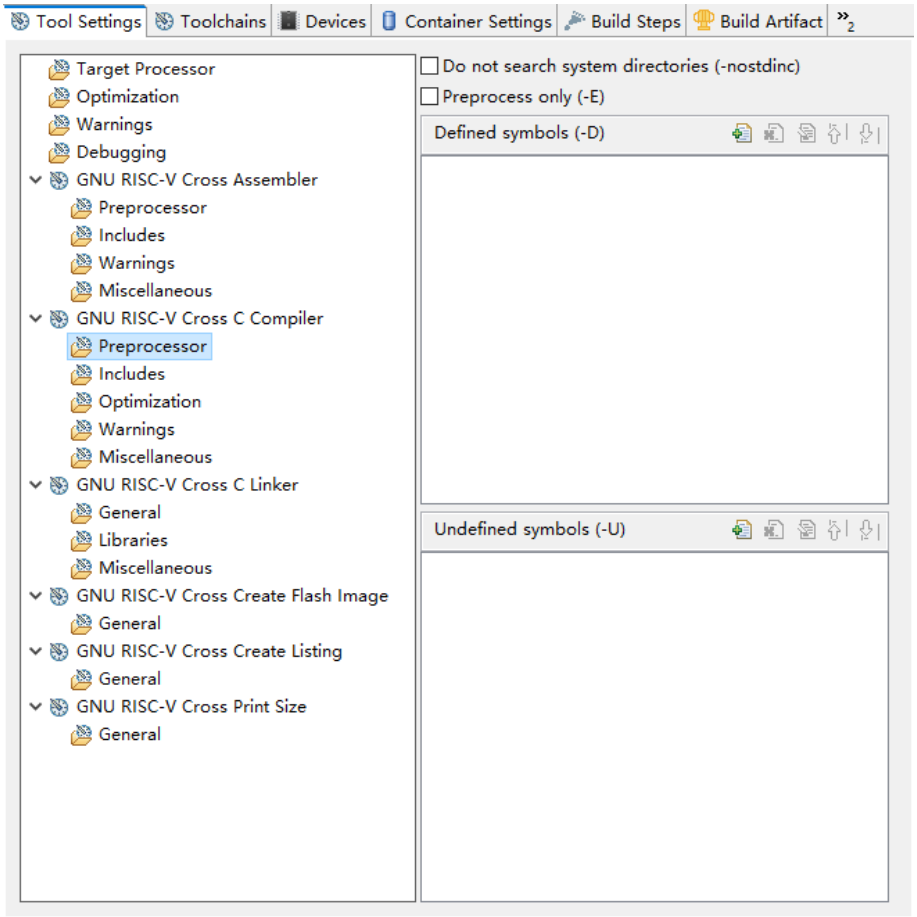
GNU RISC-V Cross C Compiler

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。

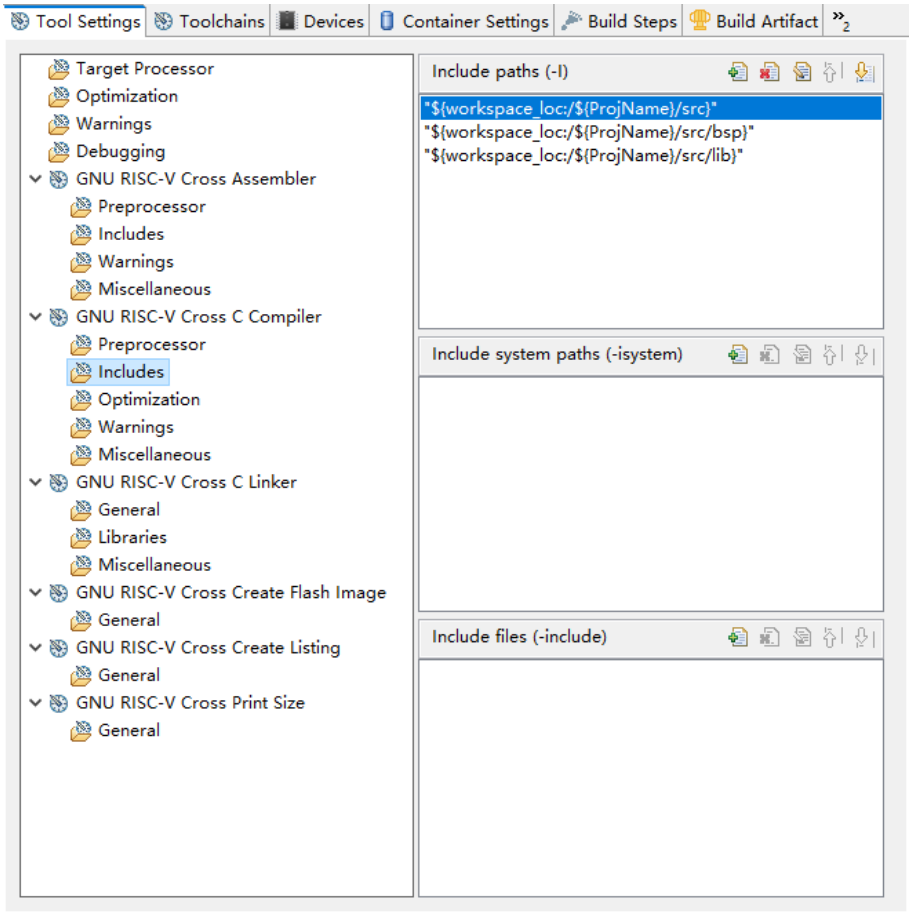
图 8-31 GNU RISC-V Cross C Compiler > Preprocessor



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。

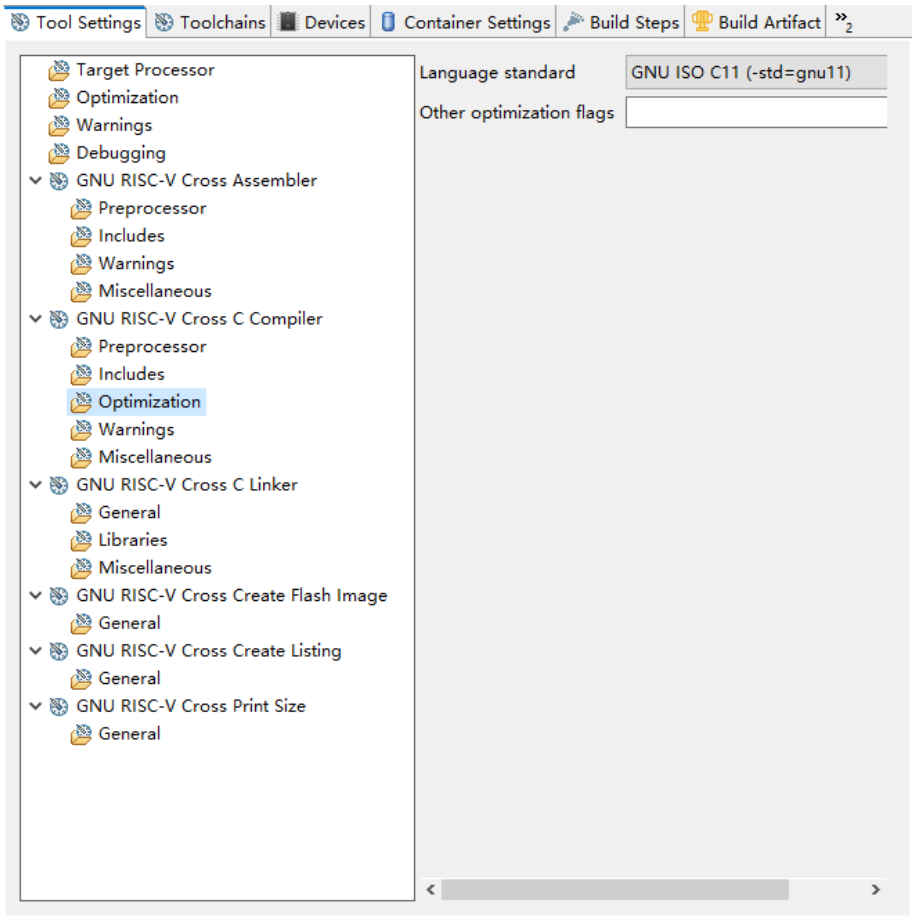
表 8-14 GNU RISC-V Cross C Compiler > Includes

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。

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



选项设置如表 8-15 所示。

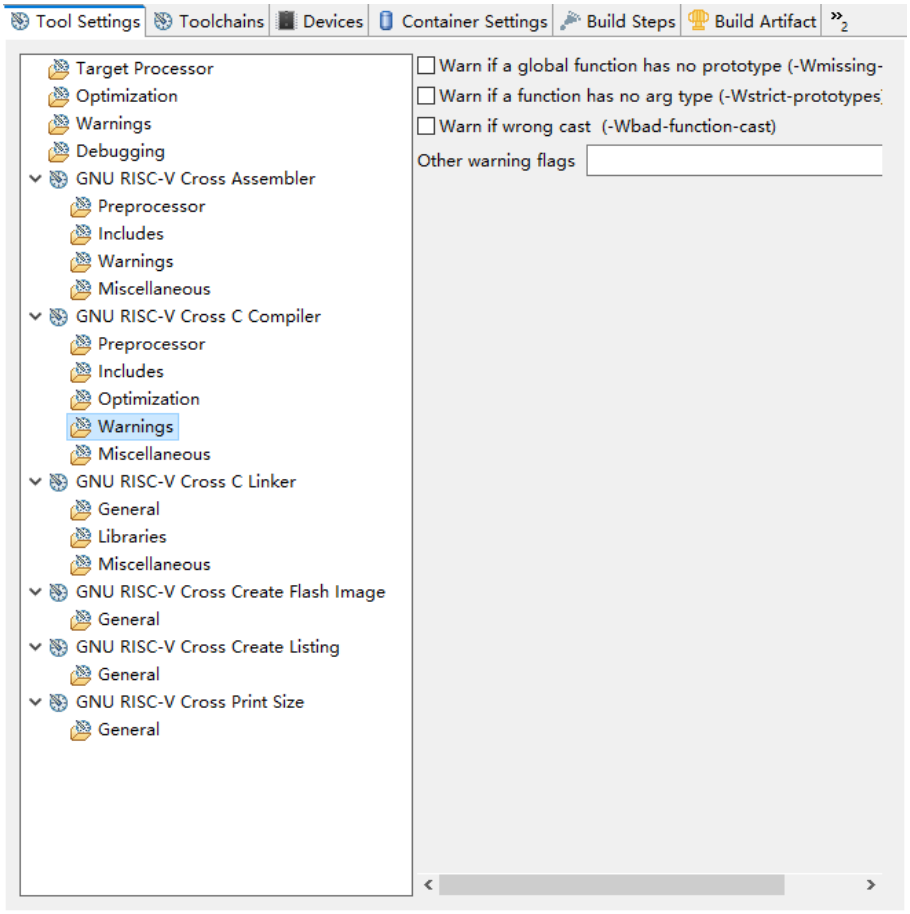
表 8-15 GNU RISC-V Cross C Compiler > Optimization

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。

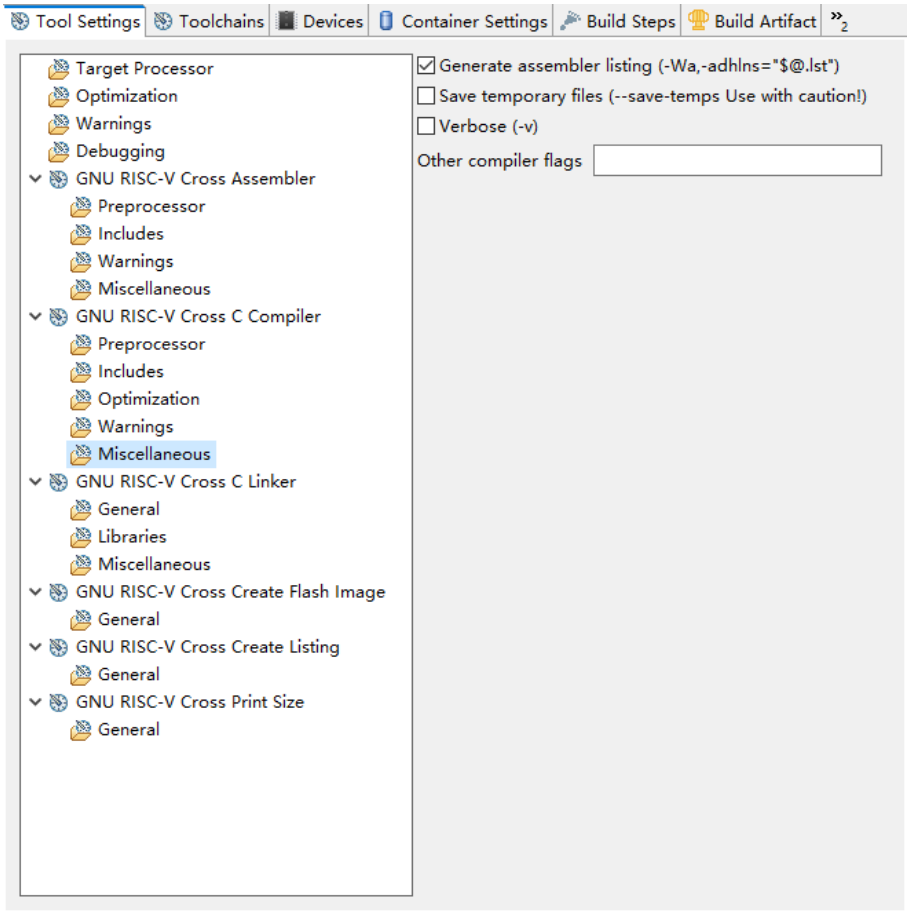
图 8-34 GNU RISC-V Cross C Compiler > Warnings



5. Miscellaneous

按需填写额外的编译器选项如图 8-35 所示。

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,-adhlns="$@.lst")	√
Save temporary files (--save-temps Use with caution)	-
Verbose (-v)	-
Other compiler flags	-

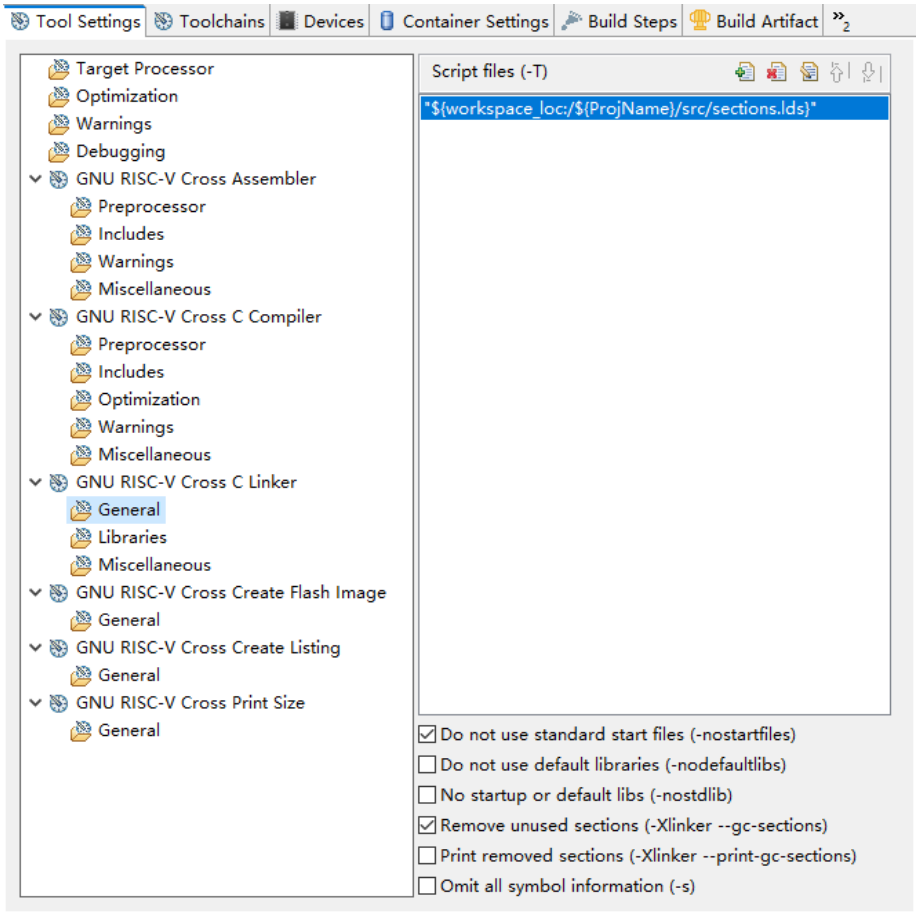
GNU RISC-V Cross C Linker

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。

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



选项设置如表 8-17 所示。

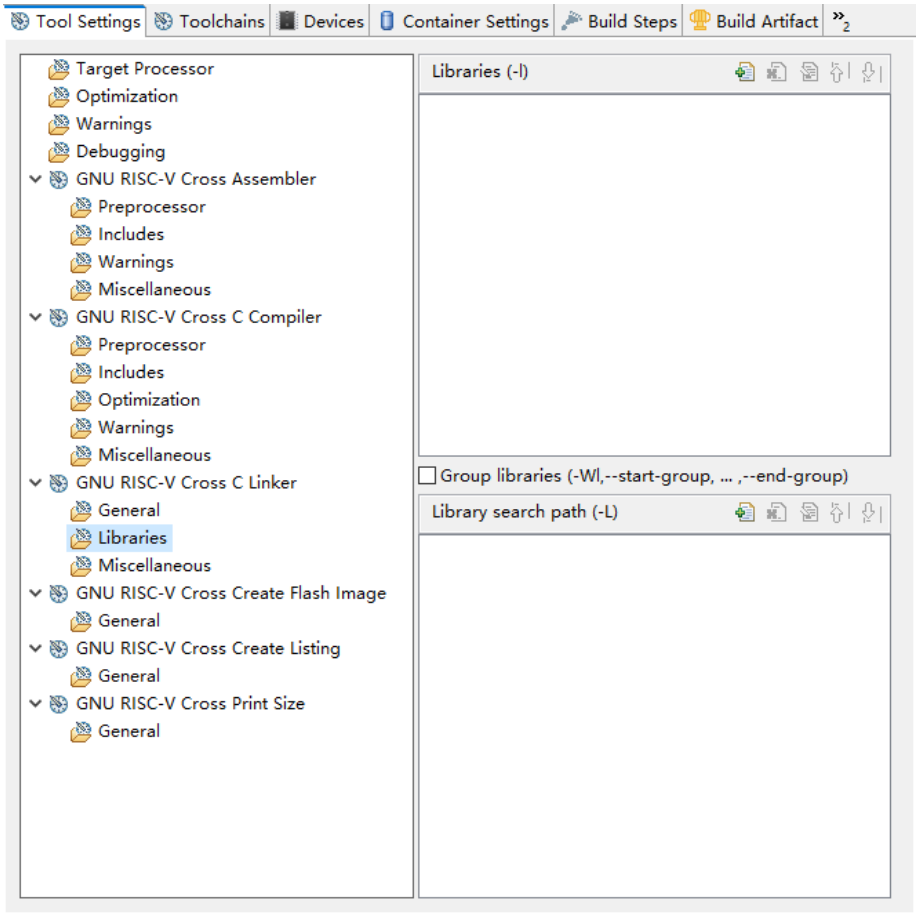
表 8-17 GNU RISC-V Cross C Linker > General

选项	设置
Script files (-T)	“${workspace_loc:/${ProjName}/src/sections.lds}”
Do not use standard start files (-nostartfiles)	√
Do not use default libraries (-nodefaultlibs)	-
No startup or default libs (-nostdlib)	-
Remove unused sections (-Xlinker --gc-sections)	√
Print removed sections (-Xlinker --print-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 (-l)，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。

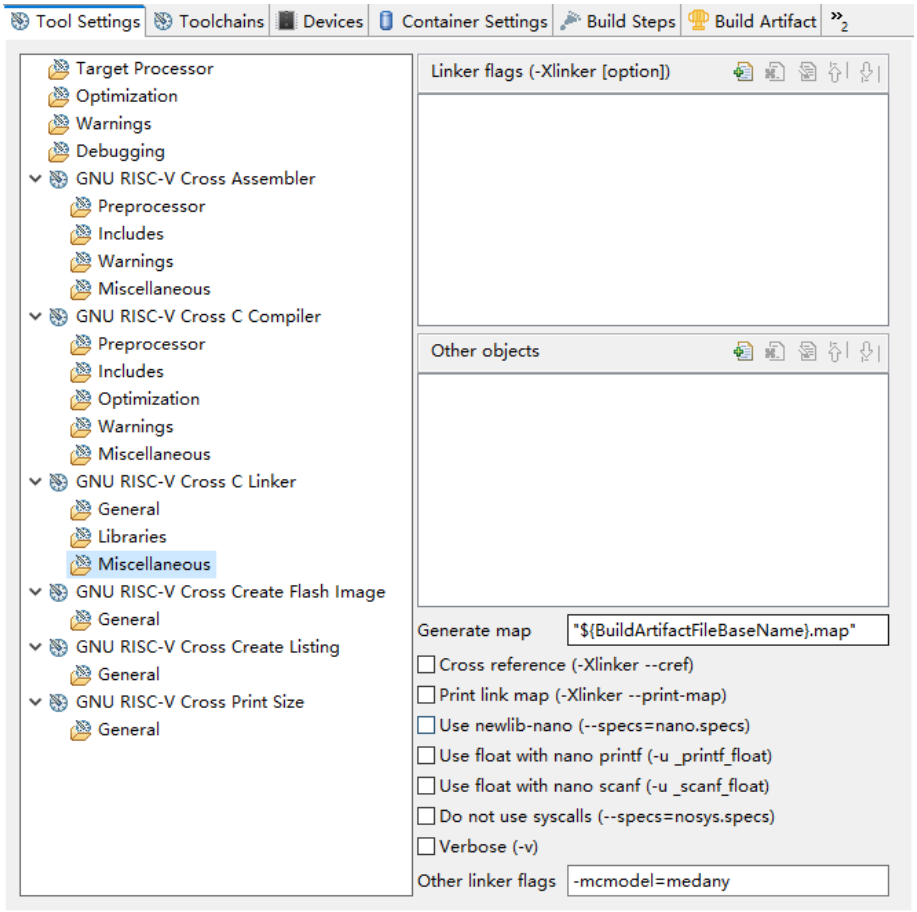
图 8-37 GNU RISC-V Cross C Linker > Libraries



3. Miscellaneous

按需设置或填写额外的链接选项如图 8-38 所示。

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



选项设置如表 8-18 所示。

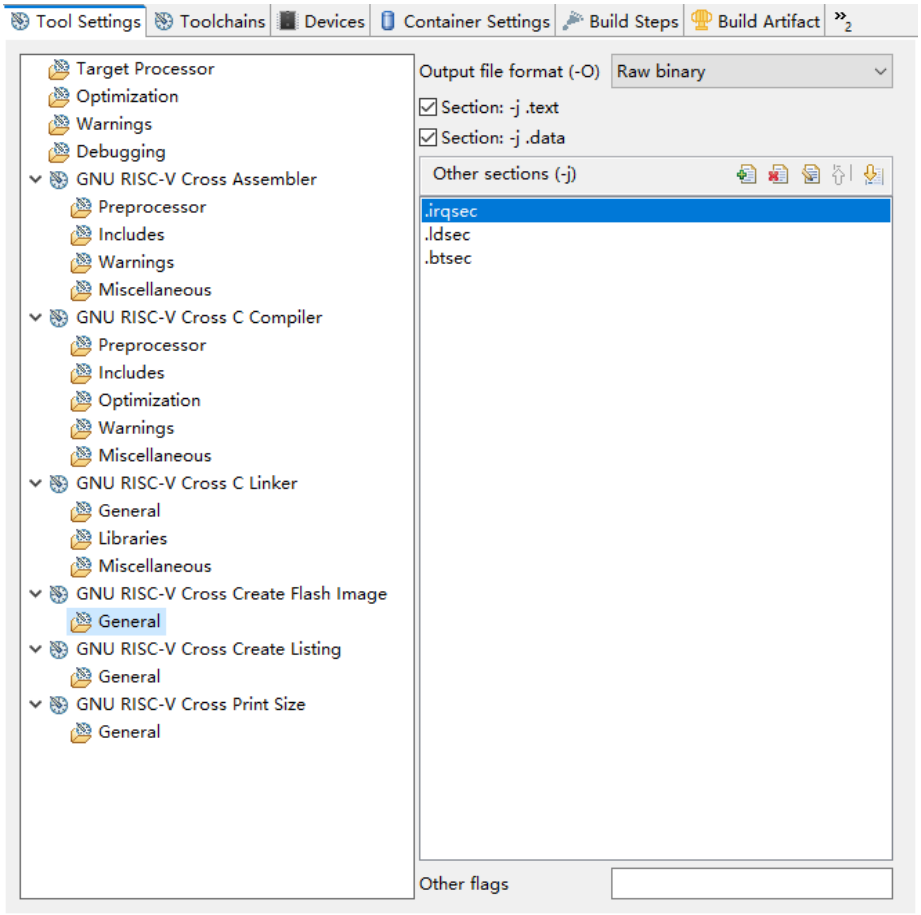
表 8-18 GNU RISC-V Cross C Linker > Miscellaneous

选项	设置
Linker flags (-Xlinker [option])	-
Other objects	-
Cross reference (-Xlinker --cref)	-
Print link map (-Xlinker --print-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



选项设置如表 8-19 所示。

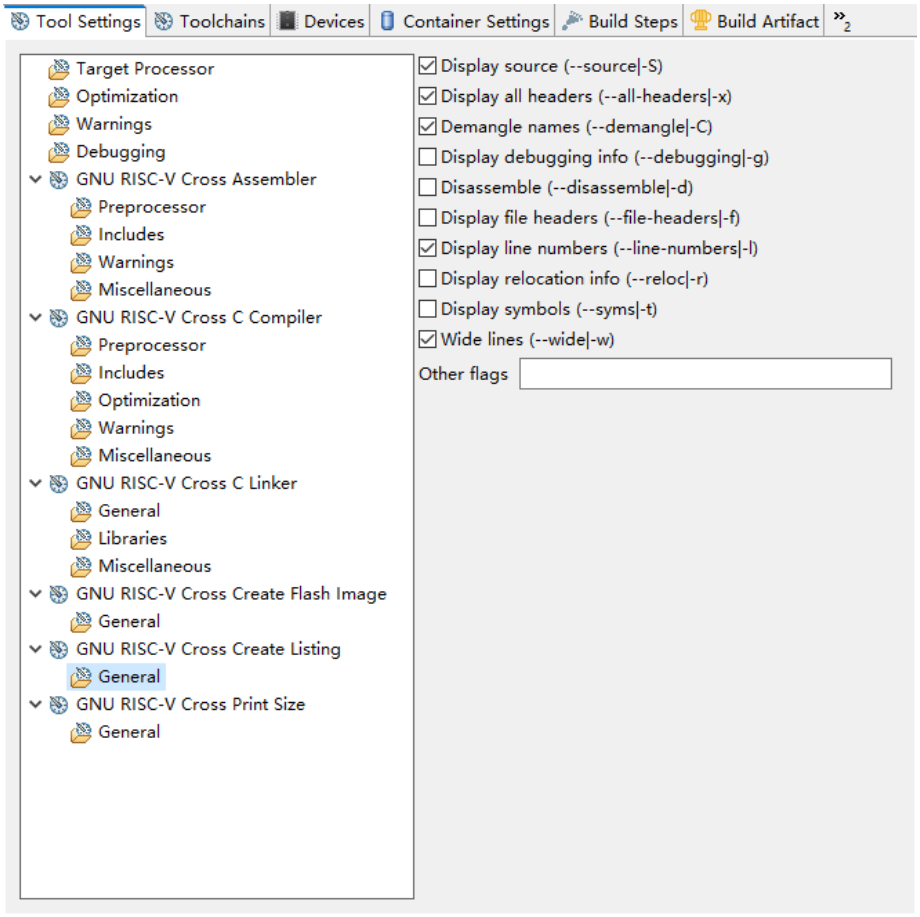
表 8-19 GNU RISC-V Cross Create Flash Image

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



选项设置如表 8-20 所示。

表 8-20 GNU RISC-V Cross Create Listing

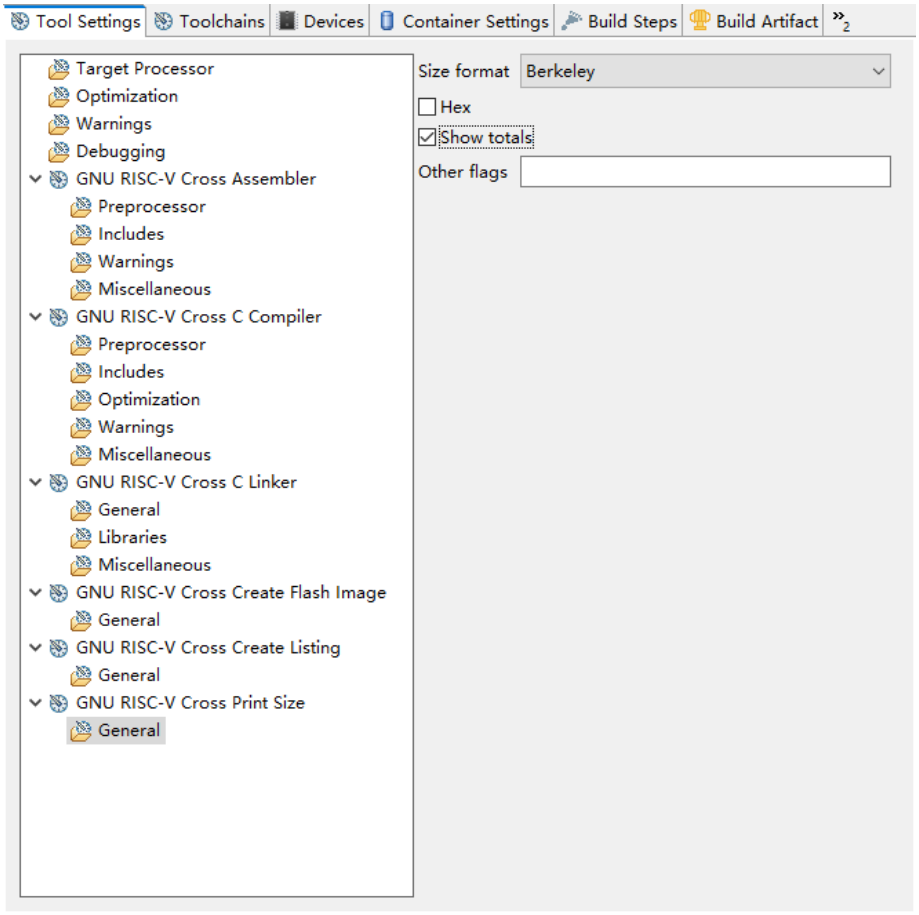
选项	设置
Display source (--source|-S)	√
Display all headers (--all-headers|-x)	√
Demangle names (-demangle|-C)	√
Display debugging info (--debugging|-g)	-
Disassemble (--disassemble|-d)	-
Display file headers (--file-headers|-f)	-
Display line numbers (--line-numbers|-l)	√
Display relocation info (--reloc|-r)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

选项	设置
Display symbols (--syms|-t)	-
Wide lines (--wide|-w)	√
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



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
Hex	-

选项	设置
Show totals	√
Other flags	-

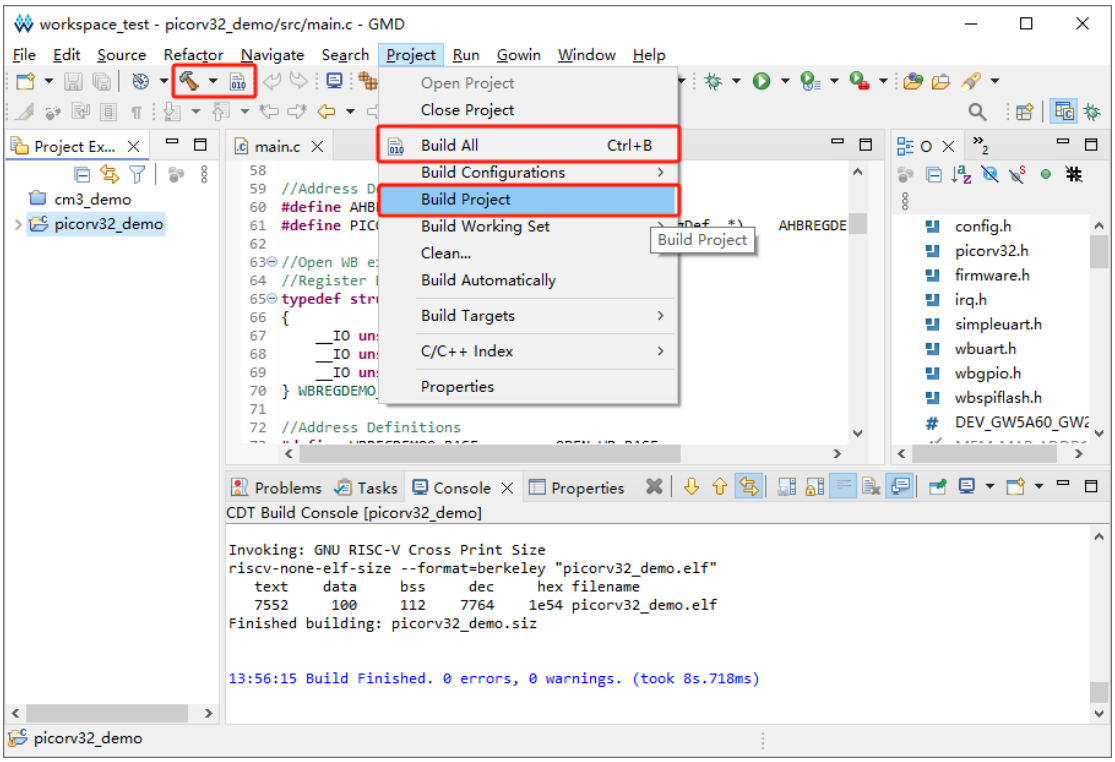
8.3 编译工程

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”（ ）
- 选择菜单栏“Project > Build All”
- 选择菜单栏“Project > Build Project”

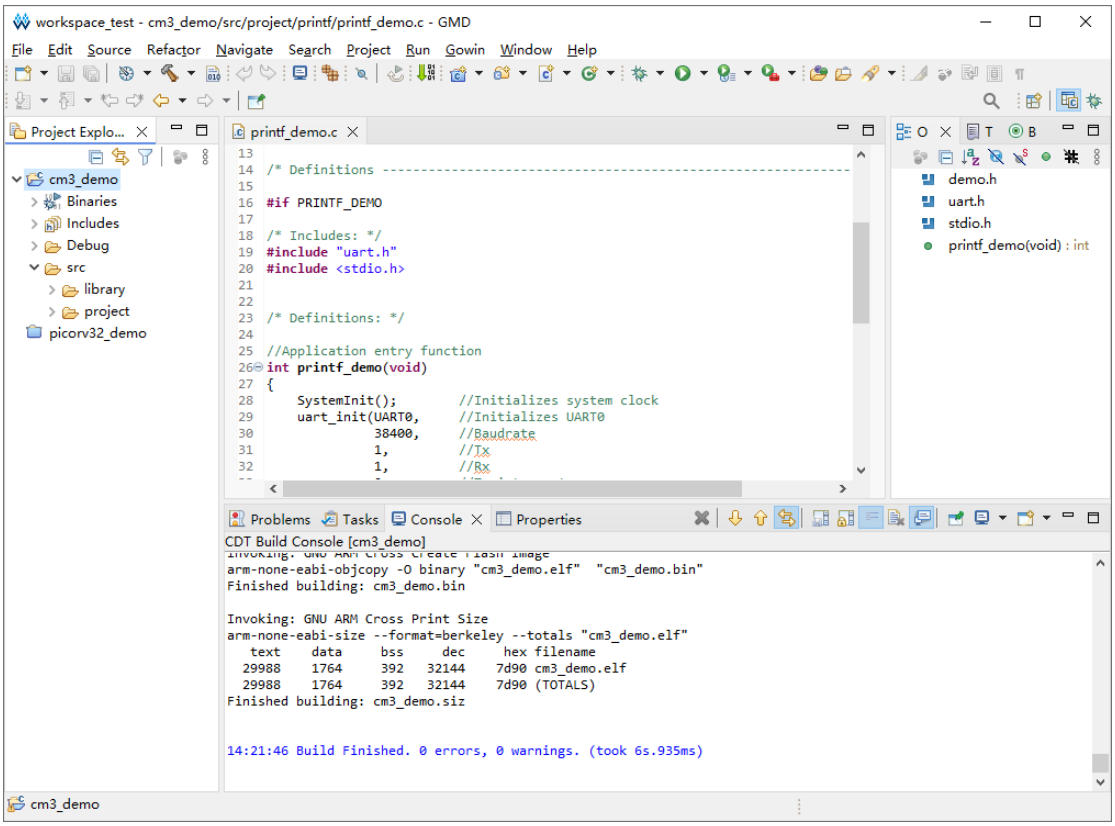
图 8-42 Build



8.3.2 编译工程

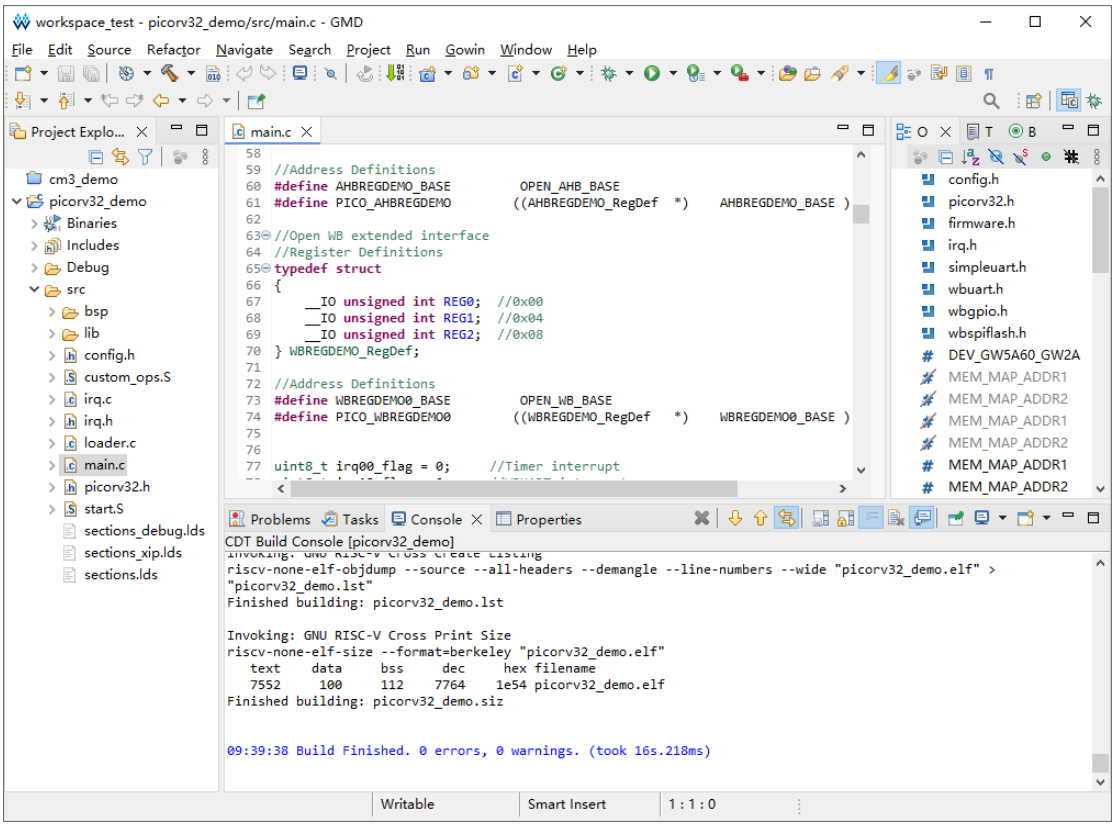
Arm MCU 工程编译示例如图 8-43 所示。

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。

图 8-44 RISC-V MCU 工程编译示例

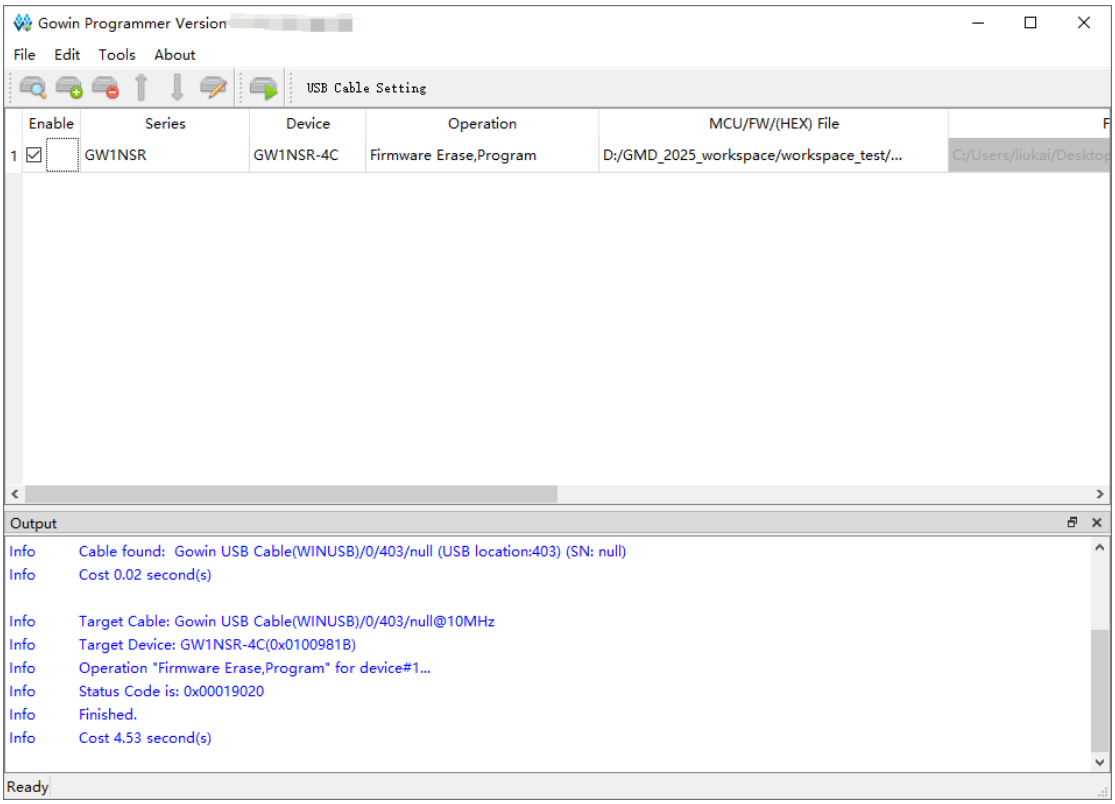


9 下载工程

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。

图 9-1 下载工具

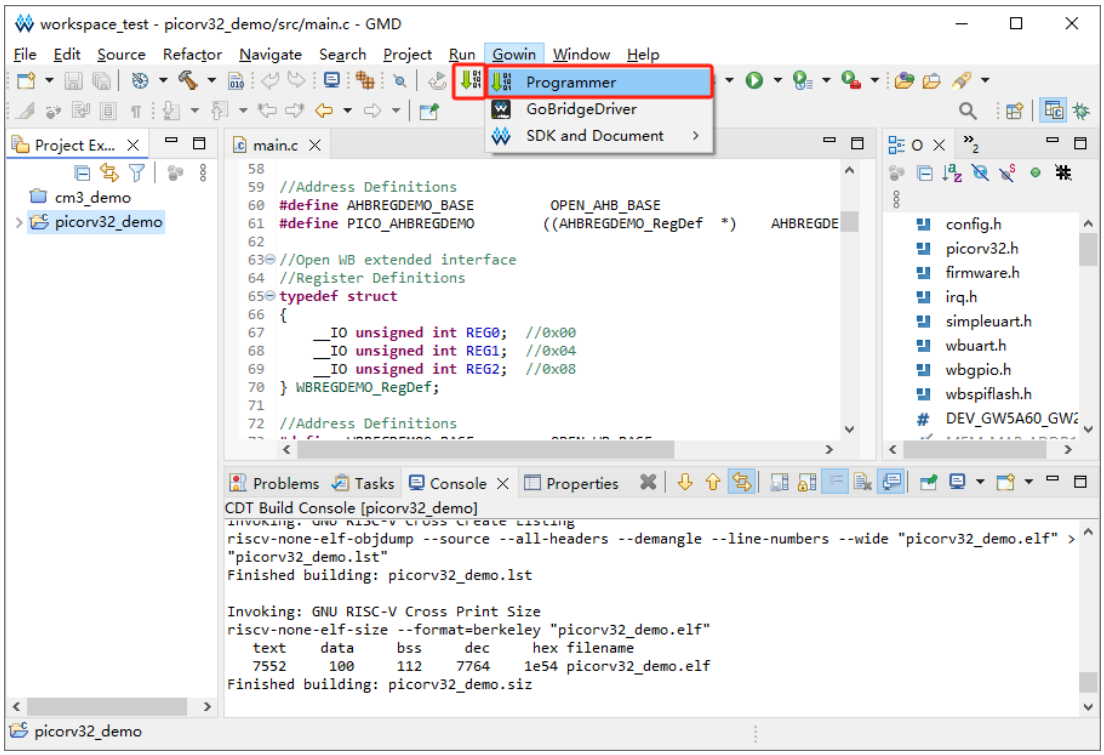


9.2 下载方法

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



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

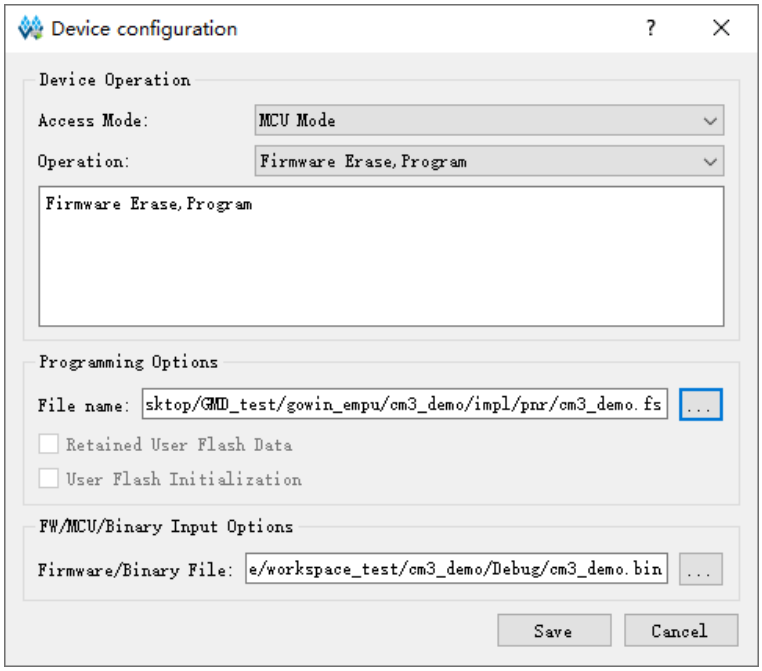


表 9-1 下载设置

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

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

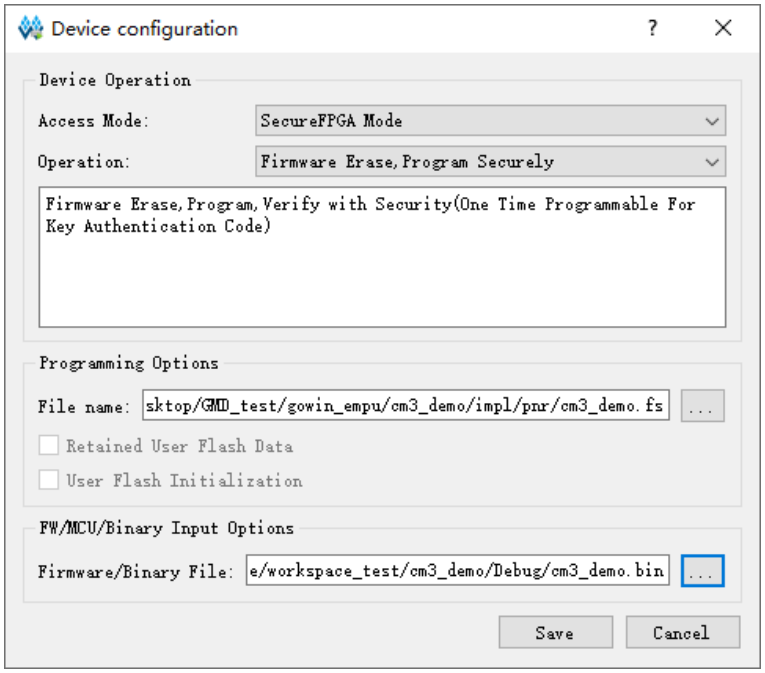


表 9-2 下载设置

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。

图 9-5 下载设置

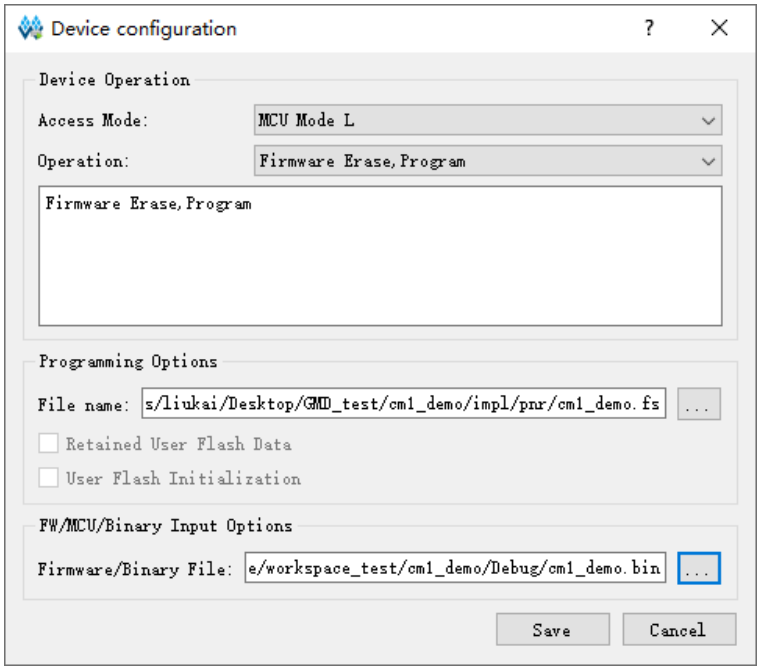


表 9-3 下载设置

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

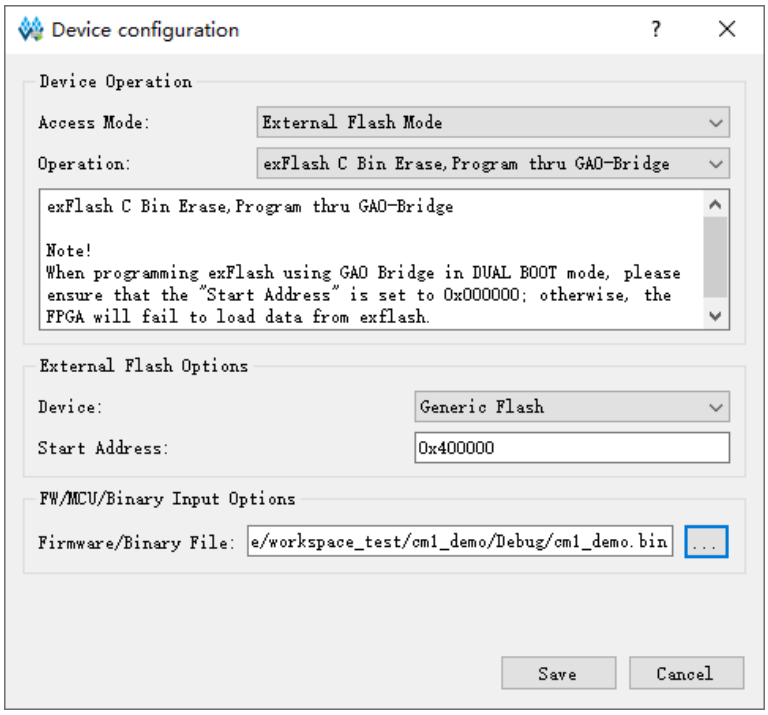


表 9-4 下载设置

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

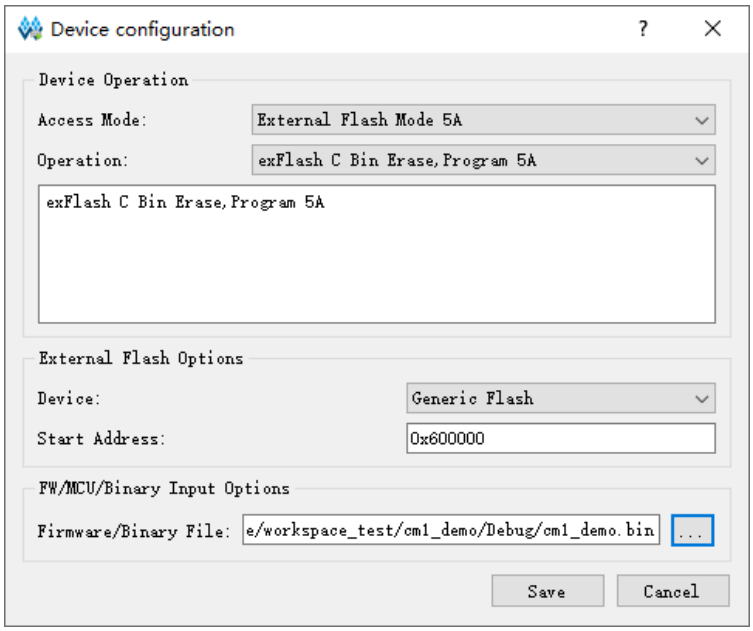


表 9-5 下载设置

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-4C)	GDB OpenOCD Debugging	GWU2X
		GWUSB (dual FTDI)
		GWUSB (single FTDI)

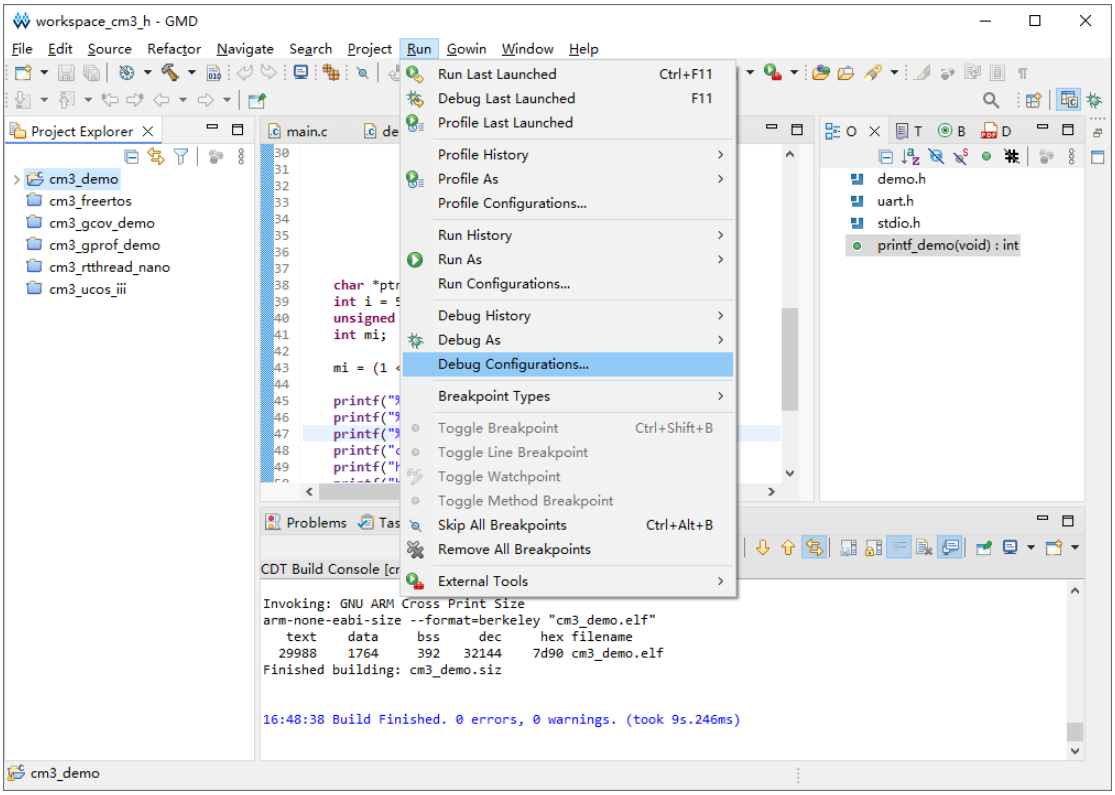
MCU 产品	调试模式	调试仿真器
		J-Link (WinUSB driver)
	GDB SEGGER J-Link Debugging	J-Link (Segger J-Link driver)
	GDB QEMU arm Debugging	-
Gowin_EMPU_M1	GDB OpenOCD Debugging	GWU2X
		GWUSB (dual FTDI)
		GWUSB (single FTDI)
		J-Link (WinUSB driver)
	GDB SEGGER J-Link Debugging	J-Link (Segger J-Link driver)
Gowin_EMPU(GW5AS-25)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

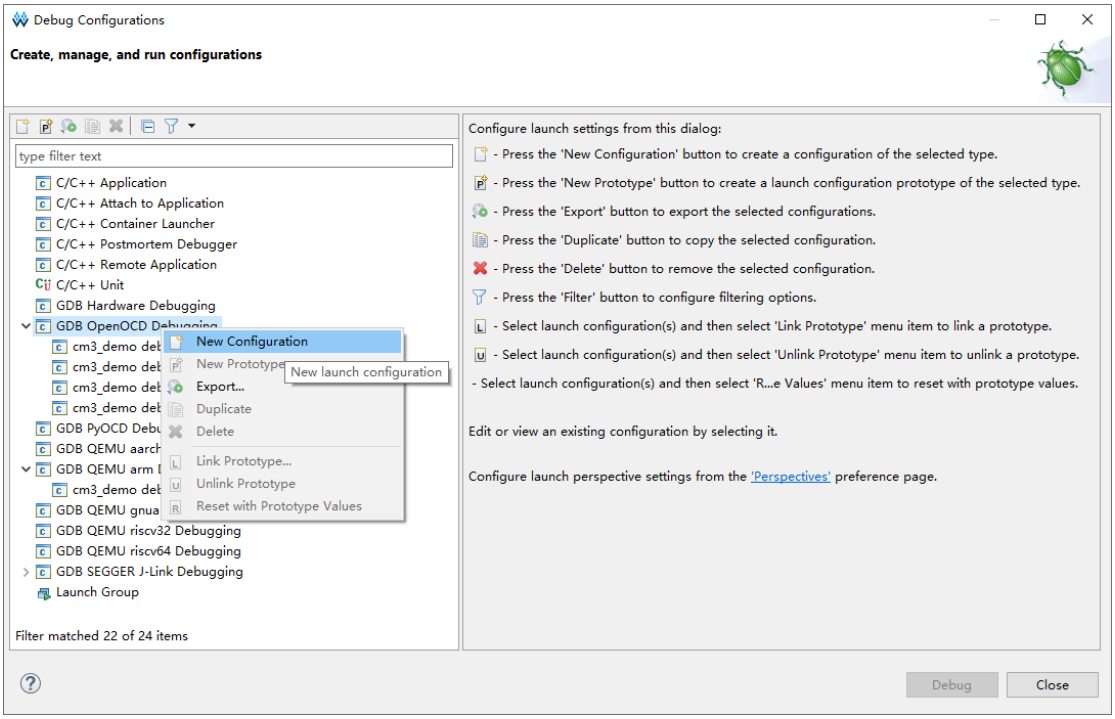
图 10-1 Debug



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



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

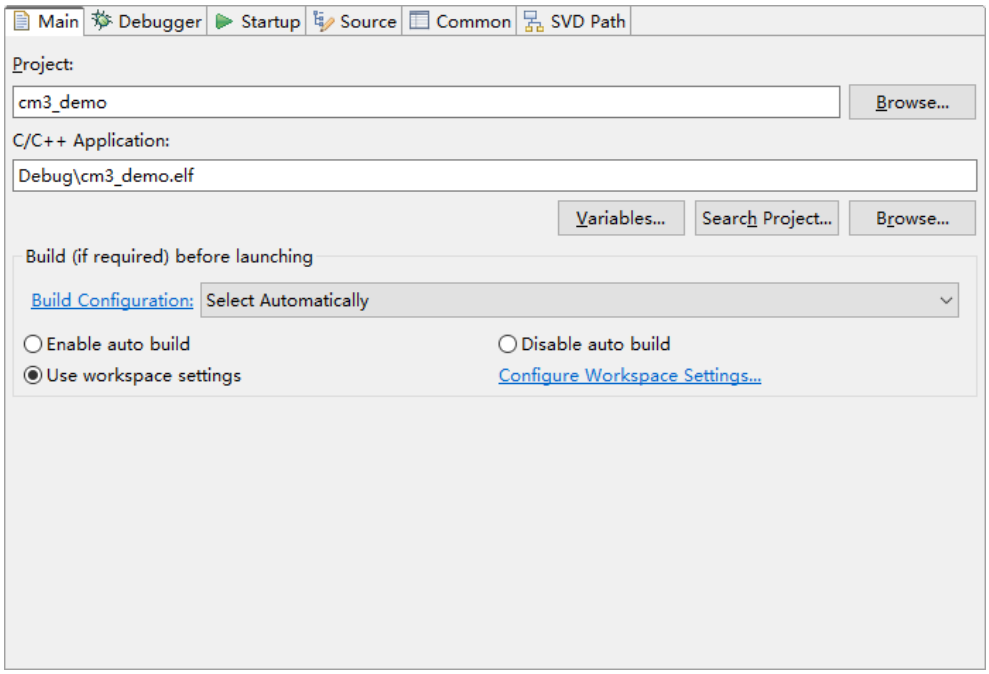
例如 Gowin_EMPU(GW1NS-4C)支持 GDB OpenOCD Debugging、GDB SEGGER J-Link Debugging、GDB QEMU arm Debugging 调试模式，如表 10-1 所示。

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



2. Debugger 视图

Debugger 视图中的调试设置如图 10-4 和表 10-2 所示。

图 10-4 Debugger 视图

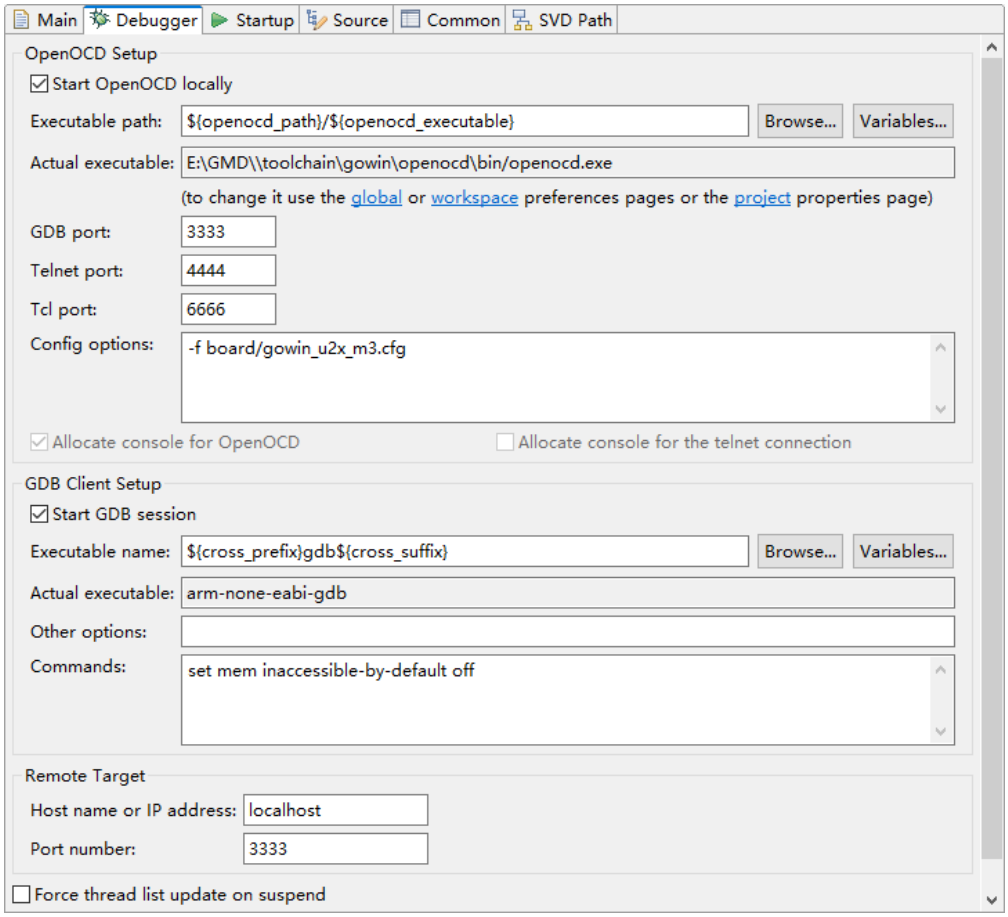


表 10-2 Debugger 视图

选项	设置
Start OpenOCD locally	√
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	√
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-4C)	GWU2X	gowin_u2x_m3.cfg
	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

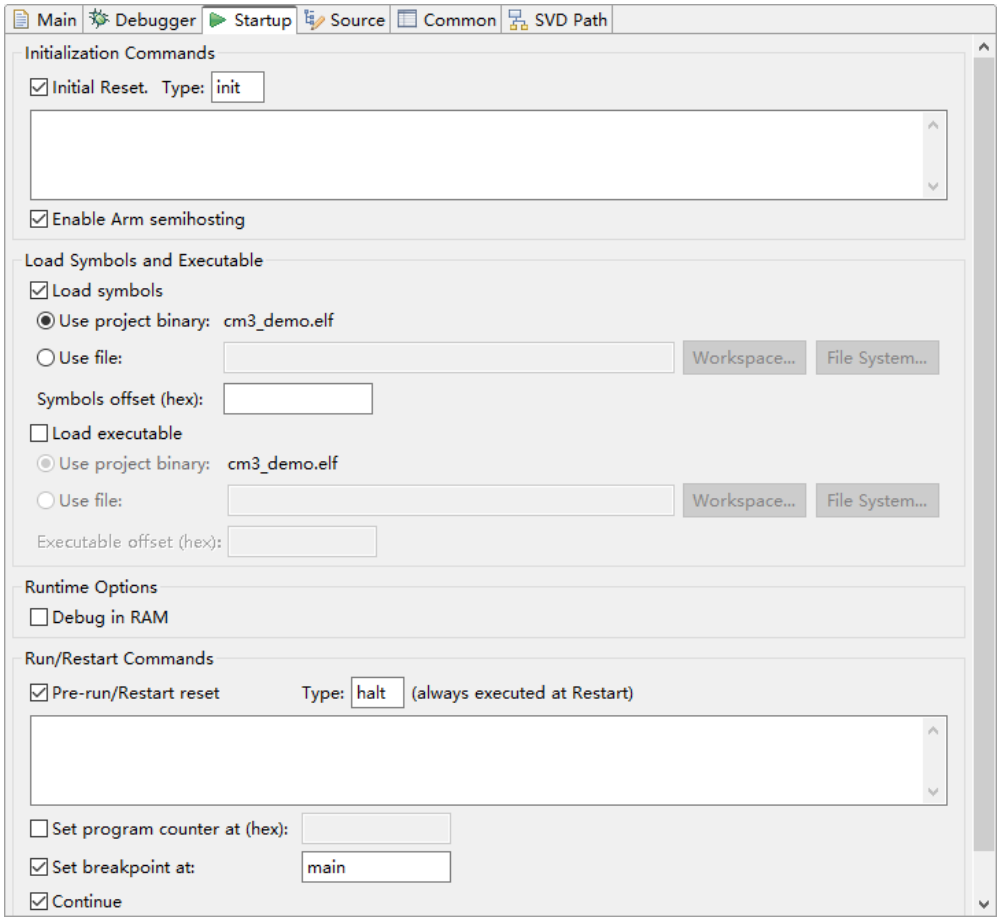


表 10-4 Startup 视图

选项	设置
Initial Reset.	√
Type	init
Enable Arm semihosting	√

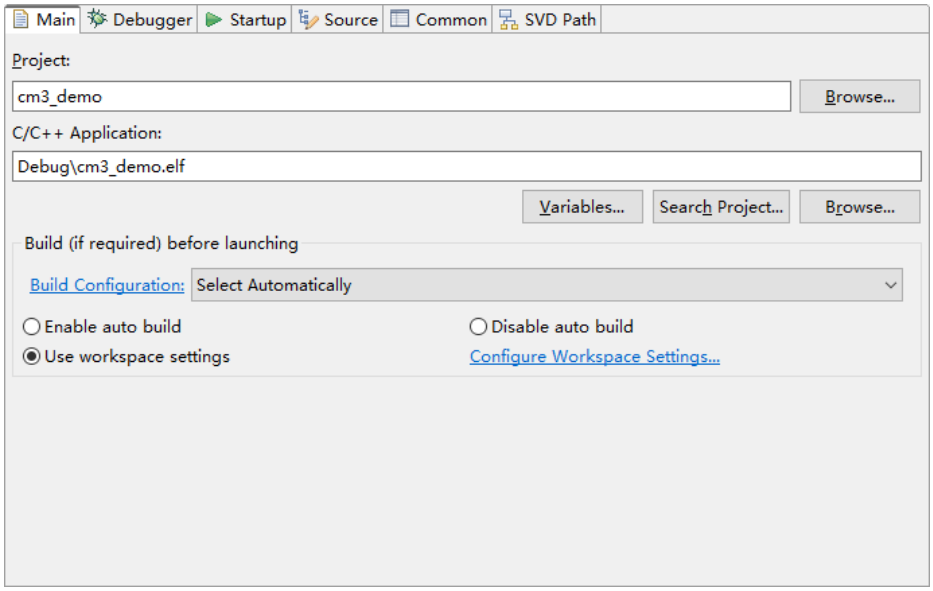
选项	设置
Load symbols	√
Load executable	×
Debug in RAM	×
Pre-run/Restart reset	√
Type	halt
Set program counter at (hex)	-
Set breakpoint at	main
Continue	√

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



2. Debugger 视图

Debugger 视图中的调试设置如图 10-7 和表 10-5 所示。

图 10-7 Debugger 视图

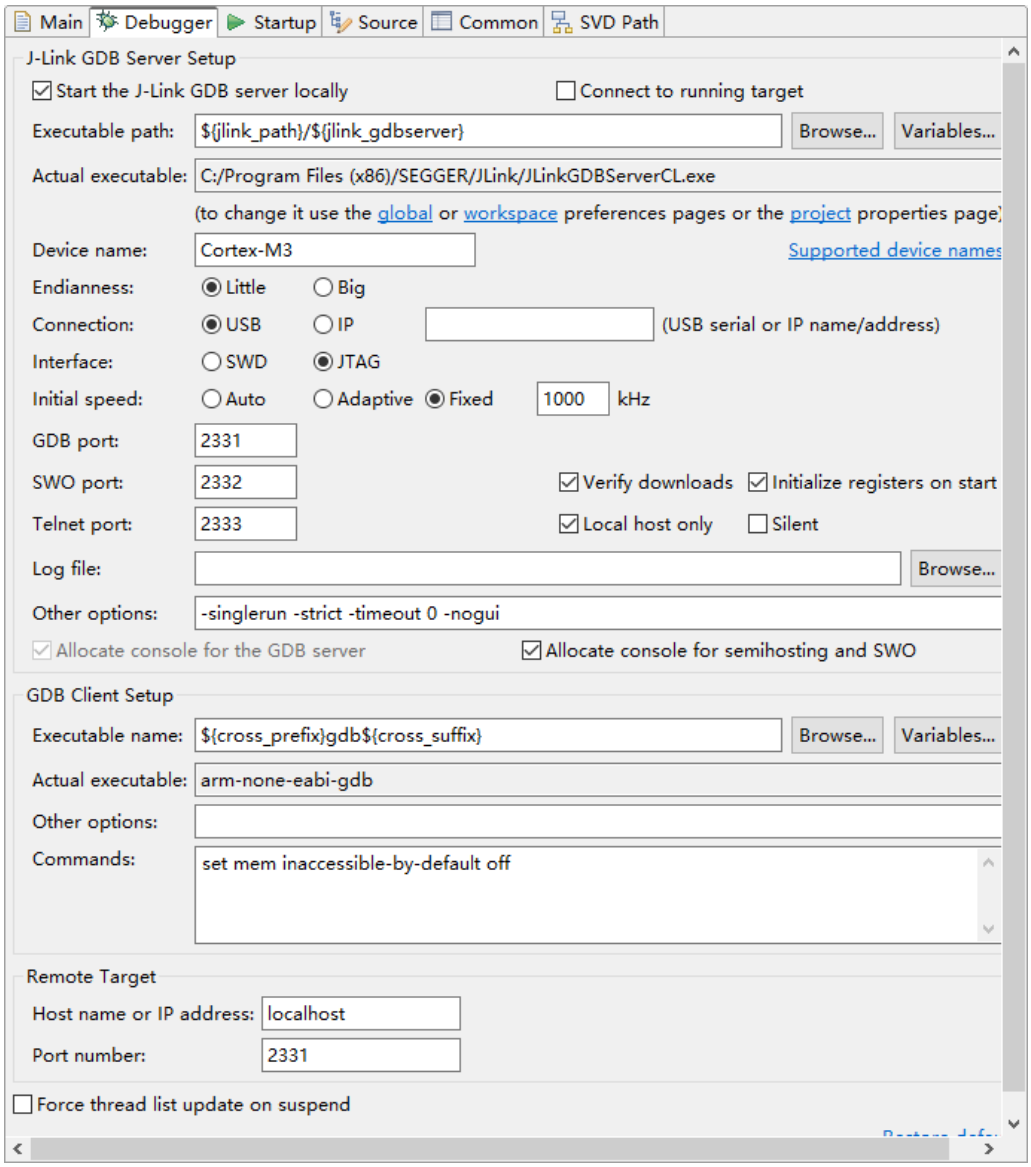


表 10-5 Debugger 视图

选项	设置
Start the J-Link GDB server locally	√
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

选项	设置
Initial speed	Fixed
GDB port	2331
Verify downloads	√
Initialize registers on start	√
Local host only	√
Silent	-
Other options	-singlerun -strict -timeout 0 -nogui
Allocate console for semihosting and SWO	√
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。

图 10-8 Startup 视图

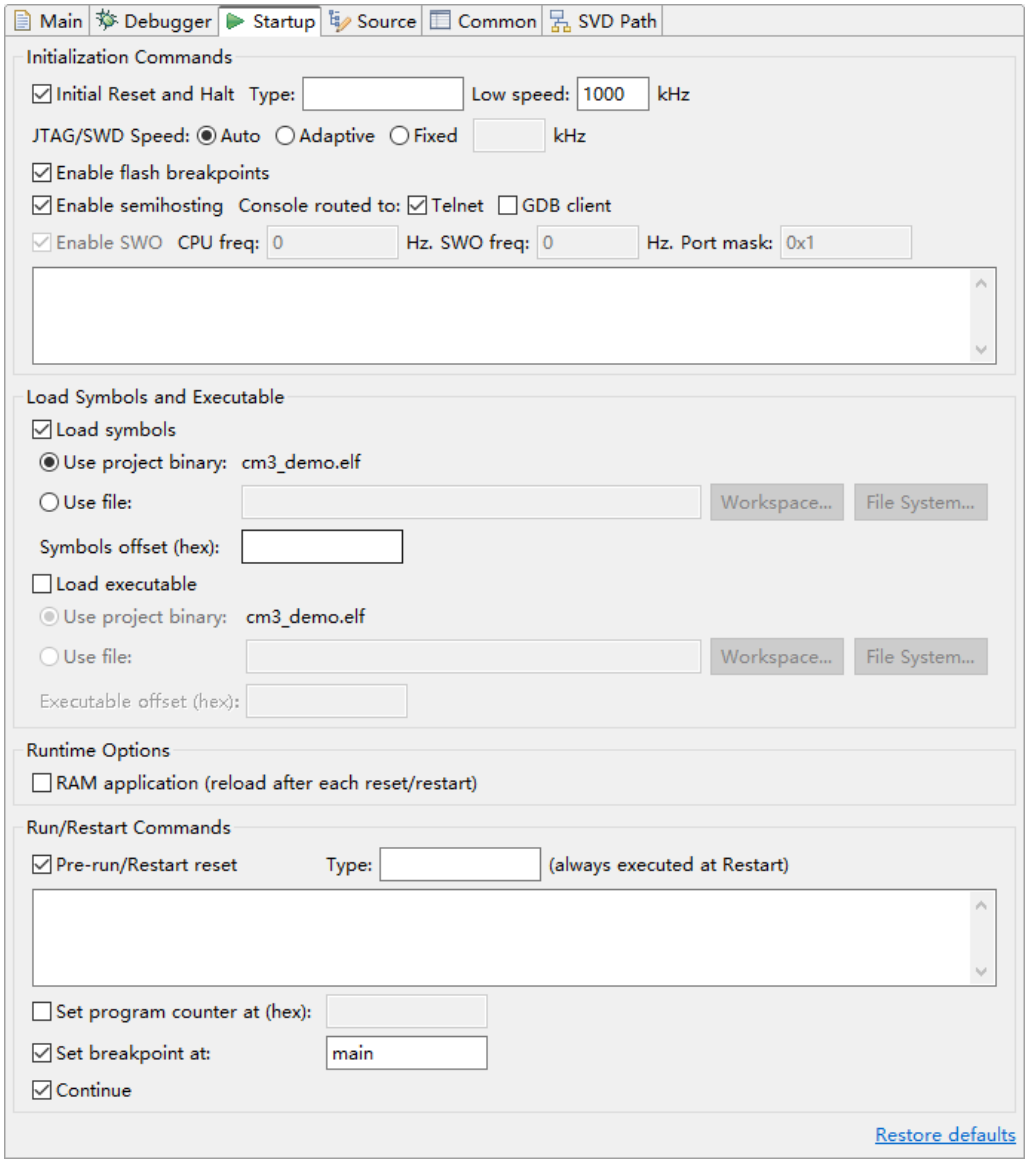


表 10-7 Startup 视图

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

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

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

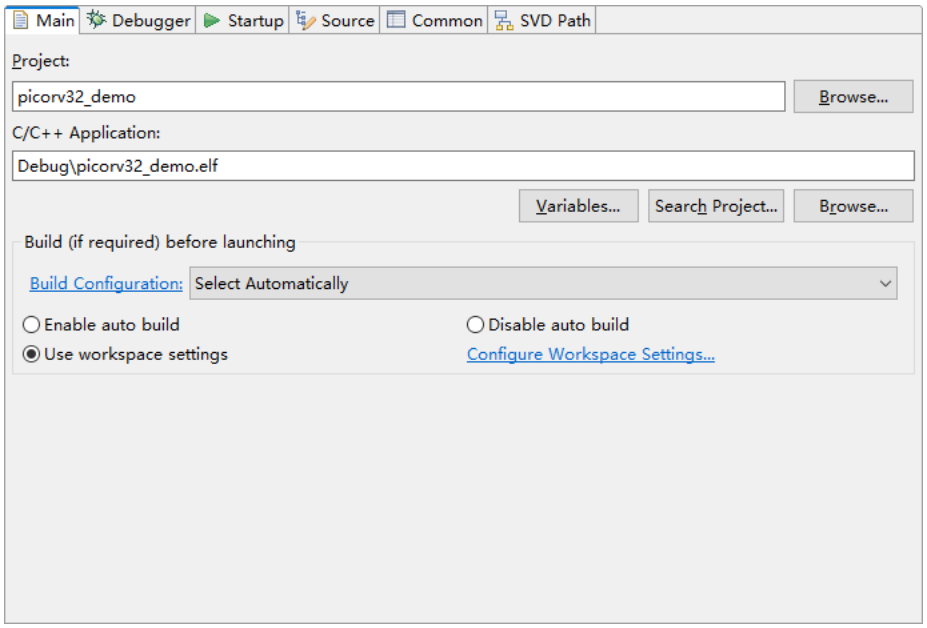
例如 Gowin_PicoRV32 支持 GDB OpenOCD Debugging、GDB QEMU riscv32 Debugging 调试模式，如表 10-1 所示。

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



2. Debugger 视图

Debugger 视图中的调试设置如图 10-10 和表 10-8 所示。

图 10-10 Debugger 视图

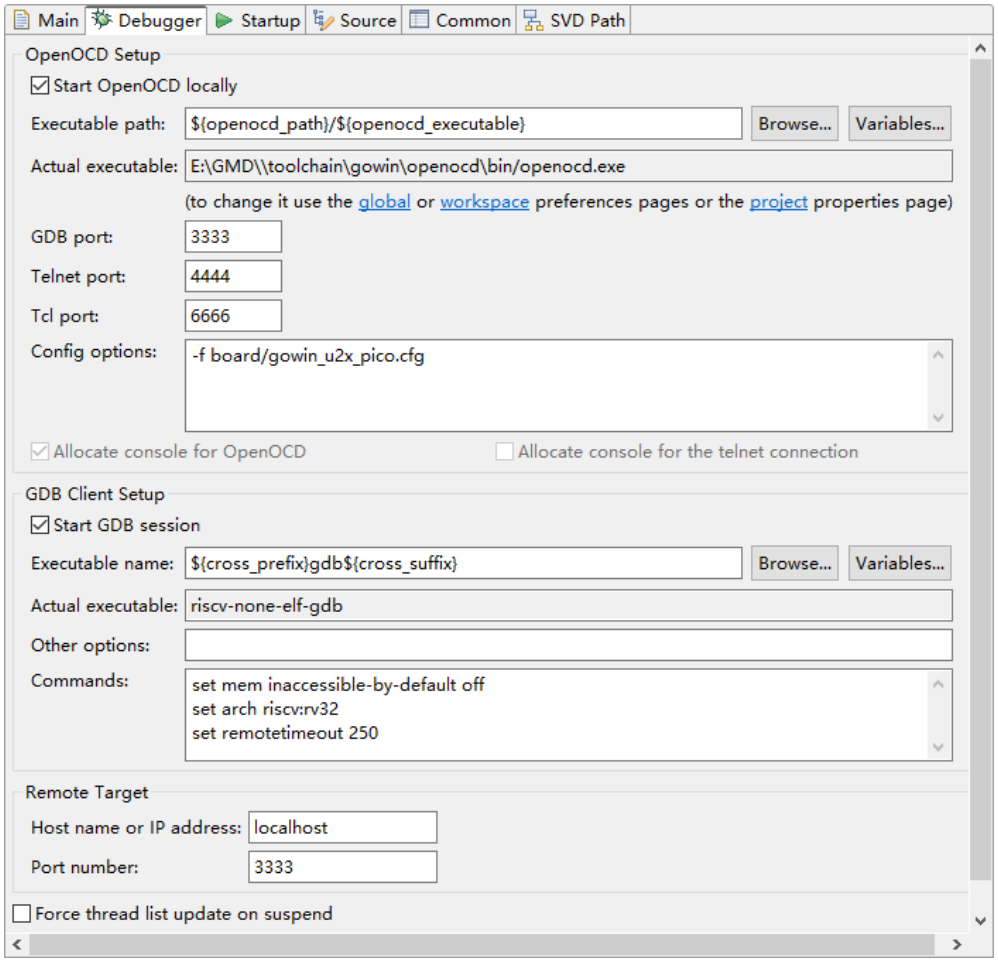


表 10-8 Debugger 视图

选项	设置
Start OpenOCD locally	√
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	√
Commands	set mem inaccessible-by-default off set arch riscv:rv32 set remotetimeout 250

表 10-9 OpenOCD 配置文件

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

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

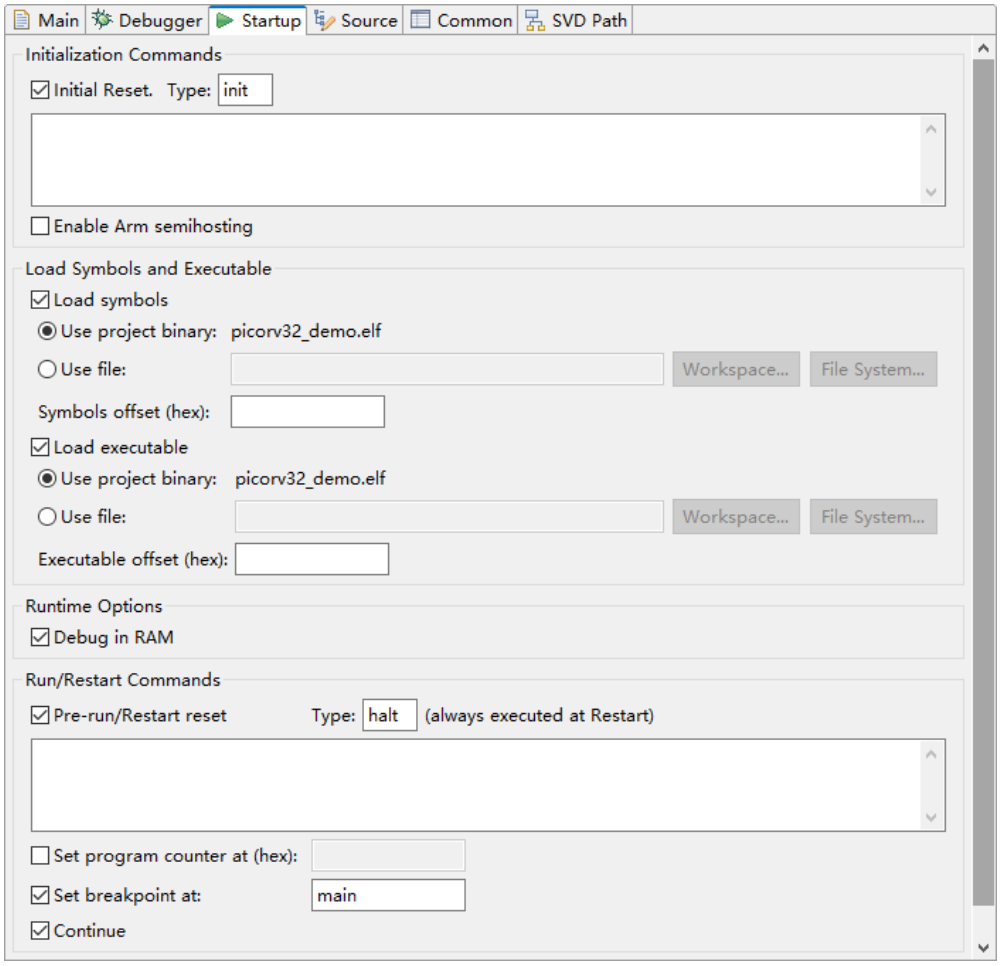


表 10-10 Startup 视图

选项	设置
Initial Reset.	√

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

GDB QEMU riscv32 Debugging

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

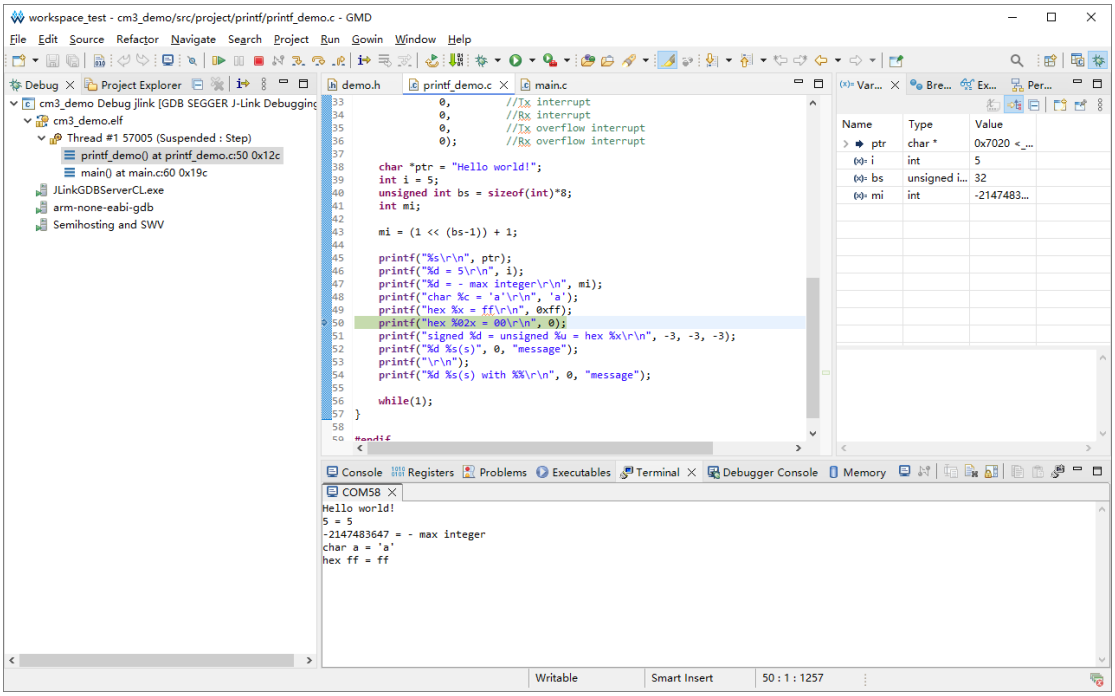
10.6 运行工程

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。

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。

表 10-11 调试功能选项

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（ ）	进入反汇编视图，单步运行汇编指令

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…

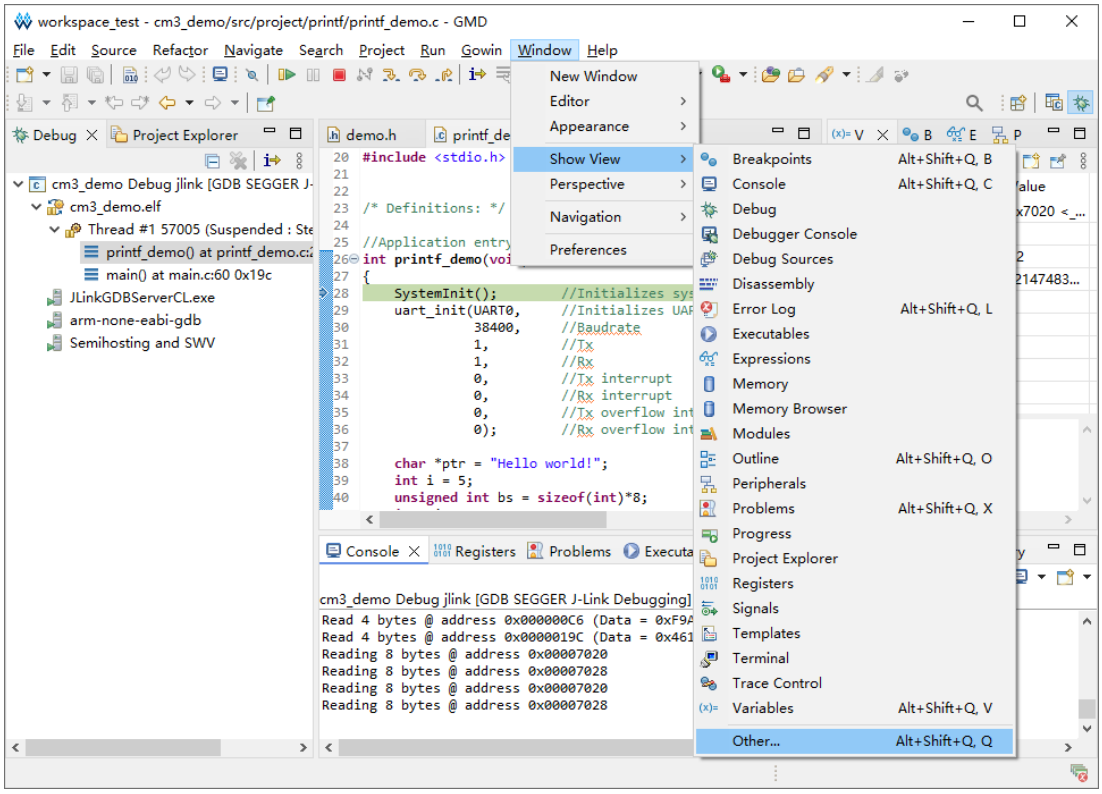
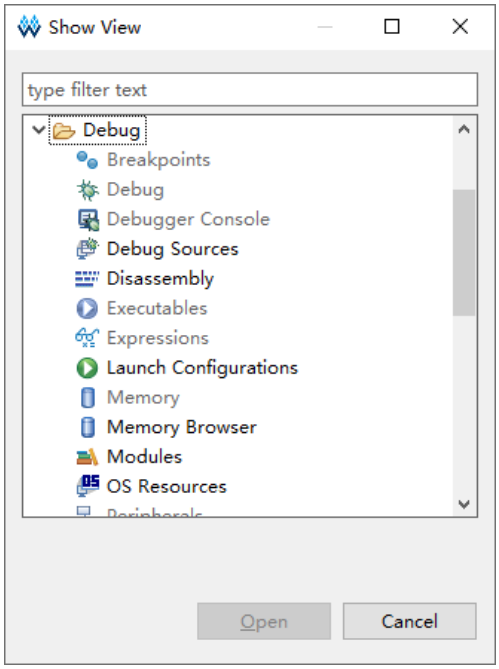


图 10-14 Debug View

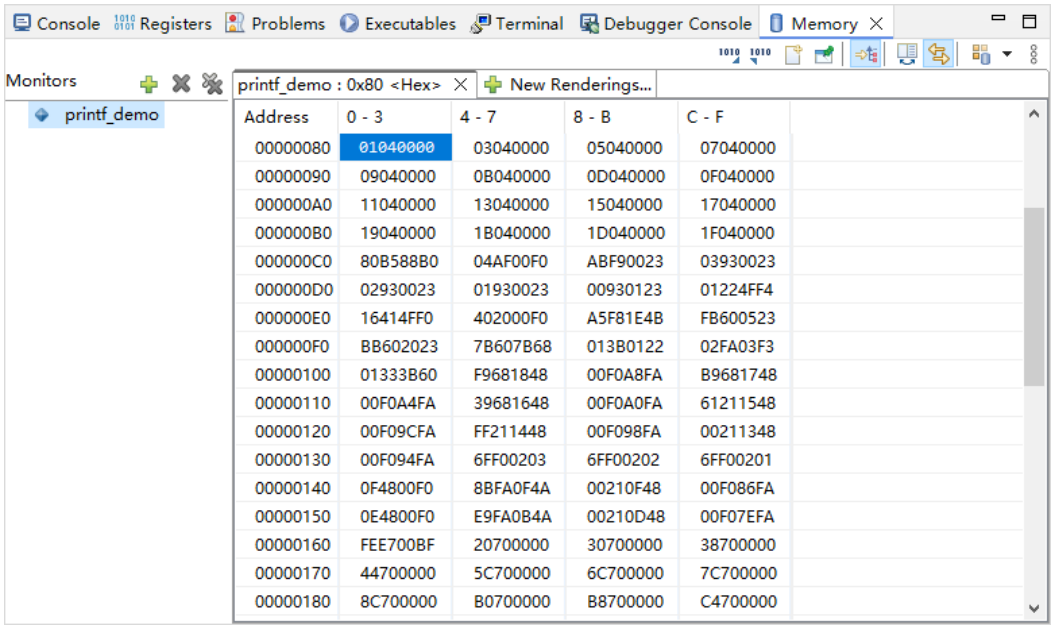


一些常用的调试视图如下所述。

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。

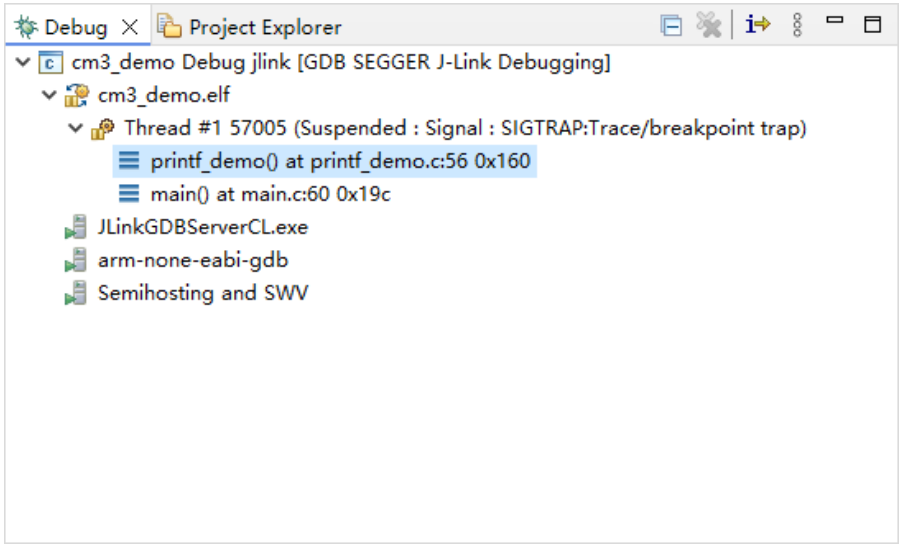
图 10-15 内存视图



建立内存监视器

1. 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



2. 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

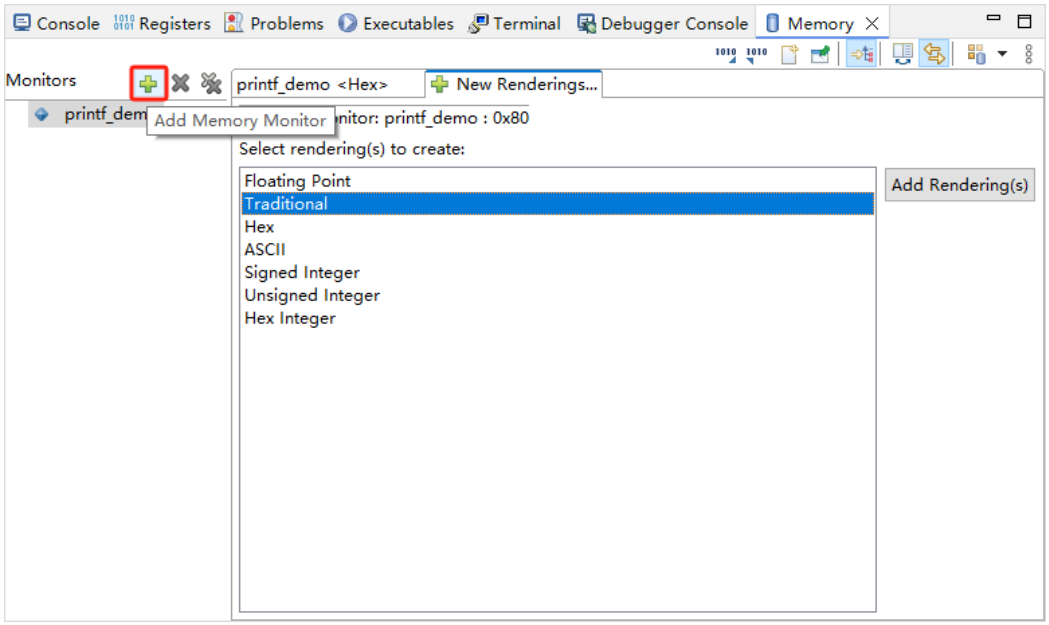
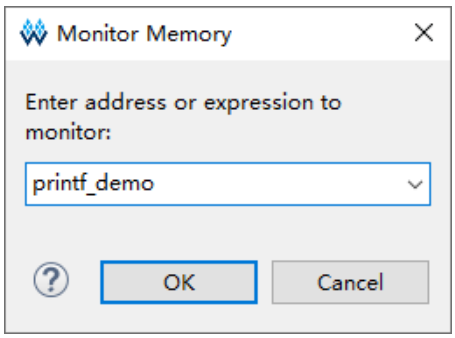
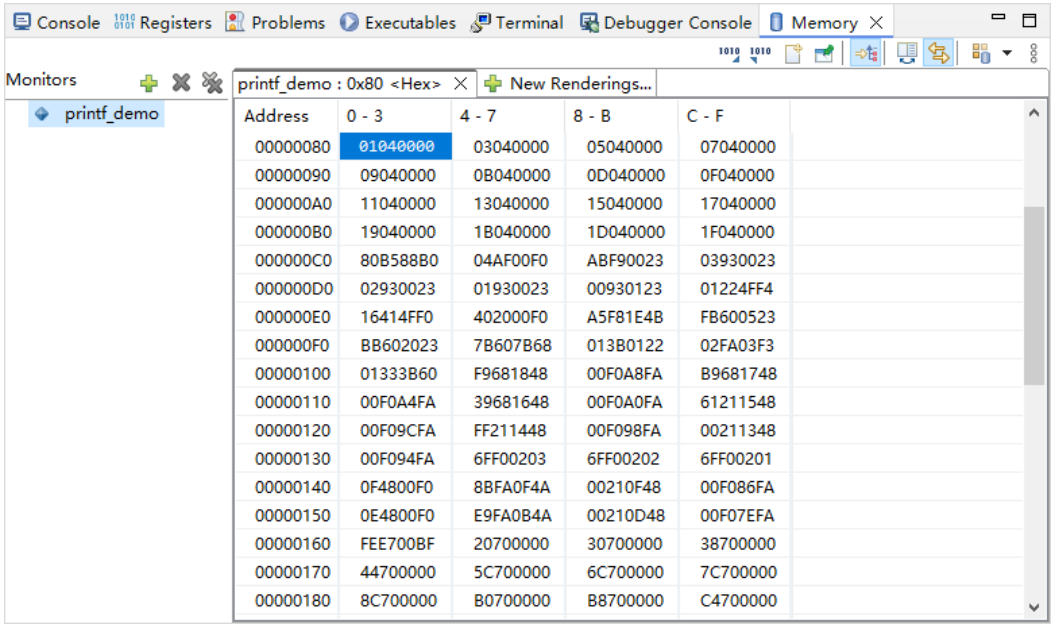


图 10-18 Monitor Memory



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。

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。

图 10-20 创建浮点格式的内存渲染器

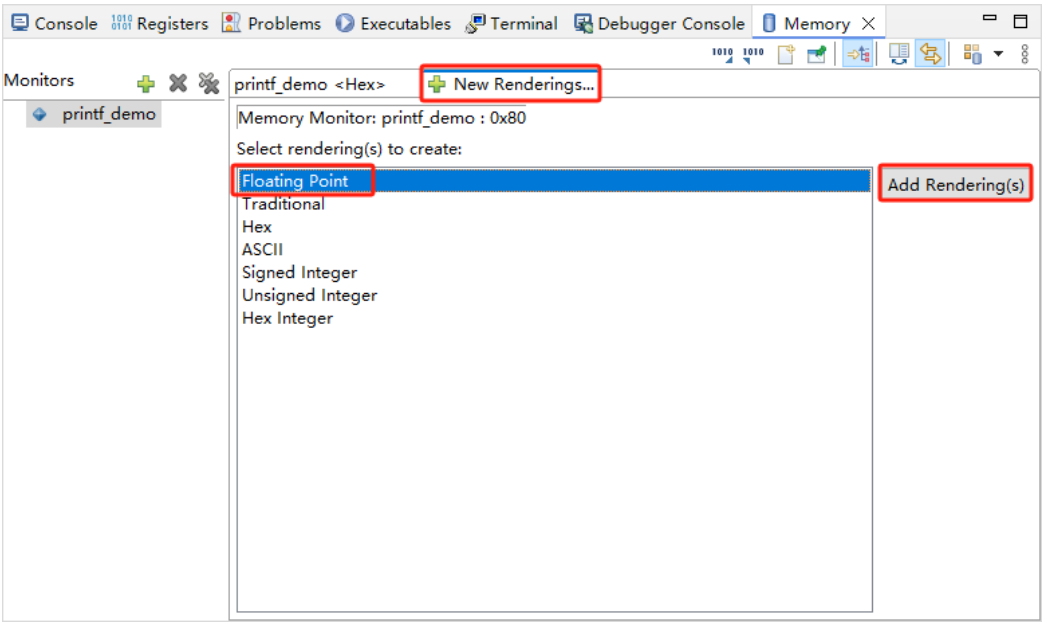
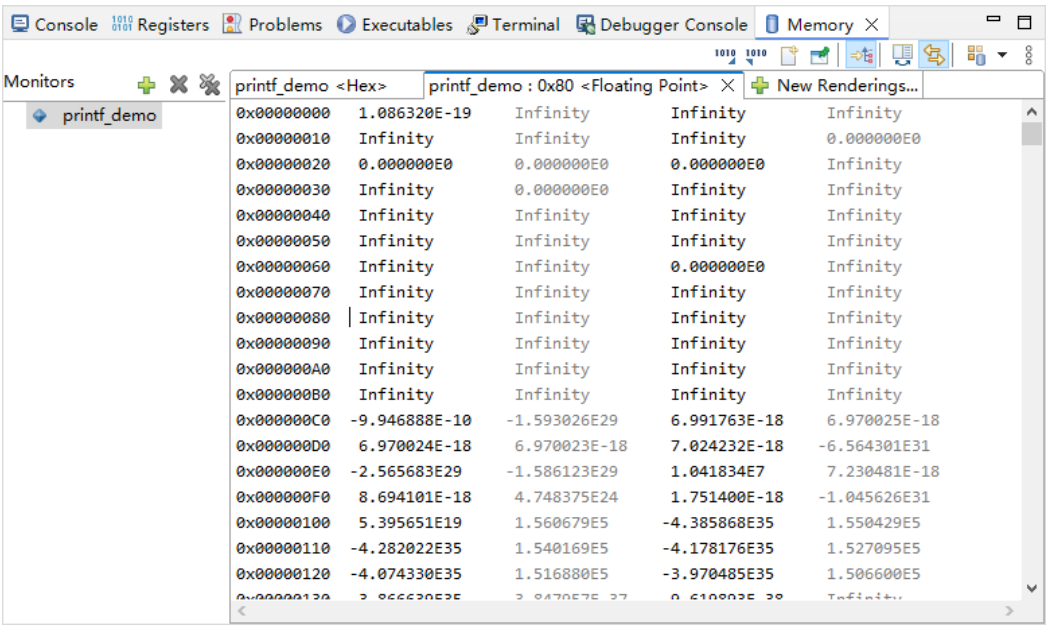
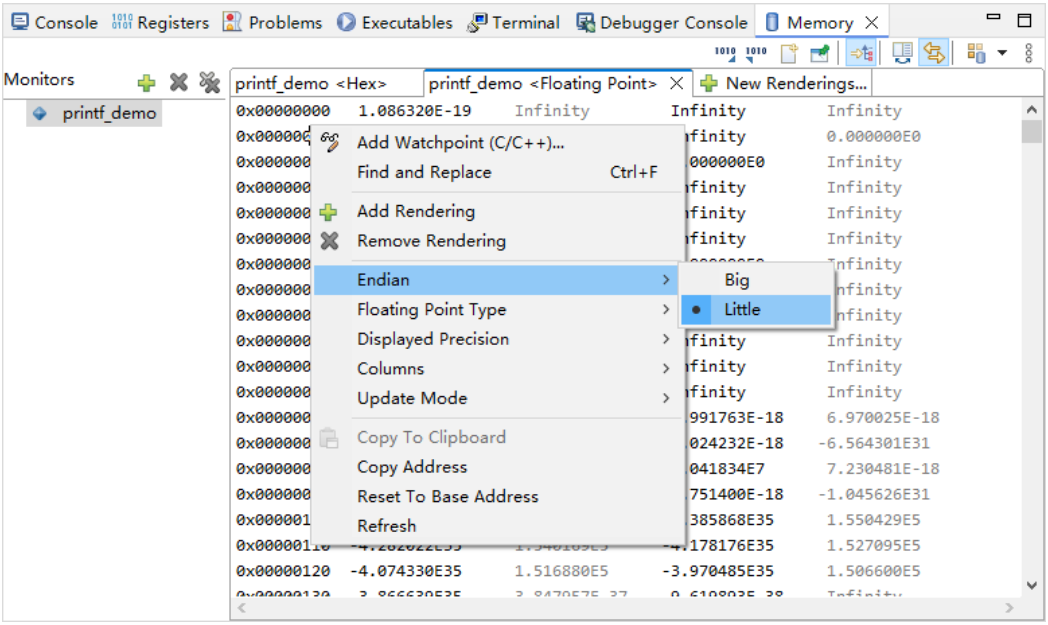


图 10-21 浮点格式的内存渲染器



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。

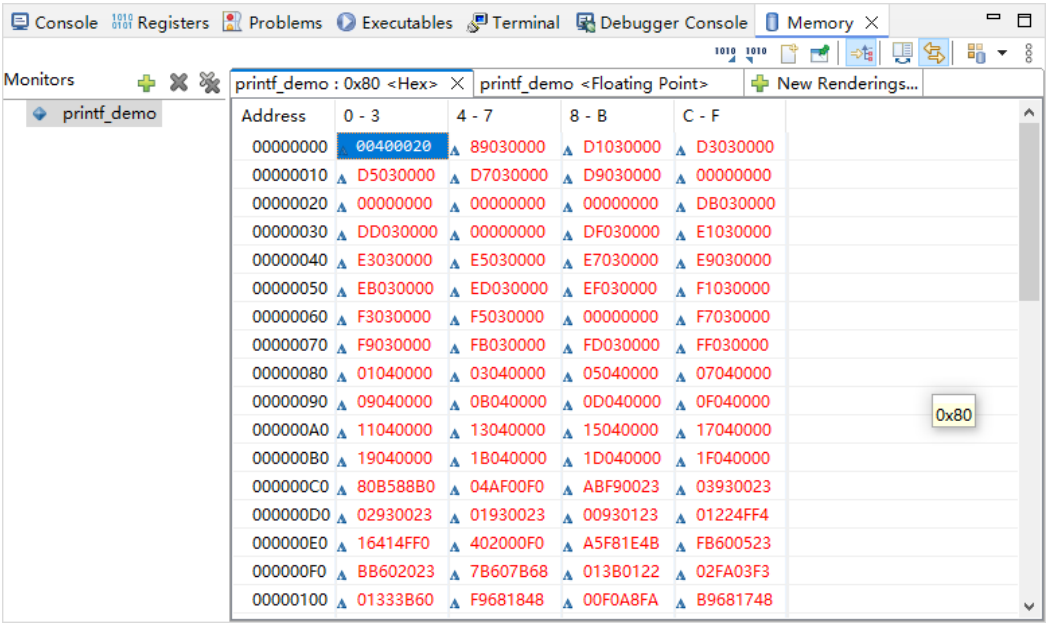
图 10-22 设置字节顺序



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。

图 10-23 存储内容显示

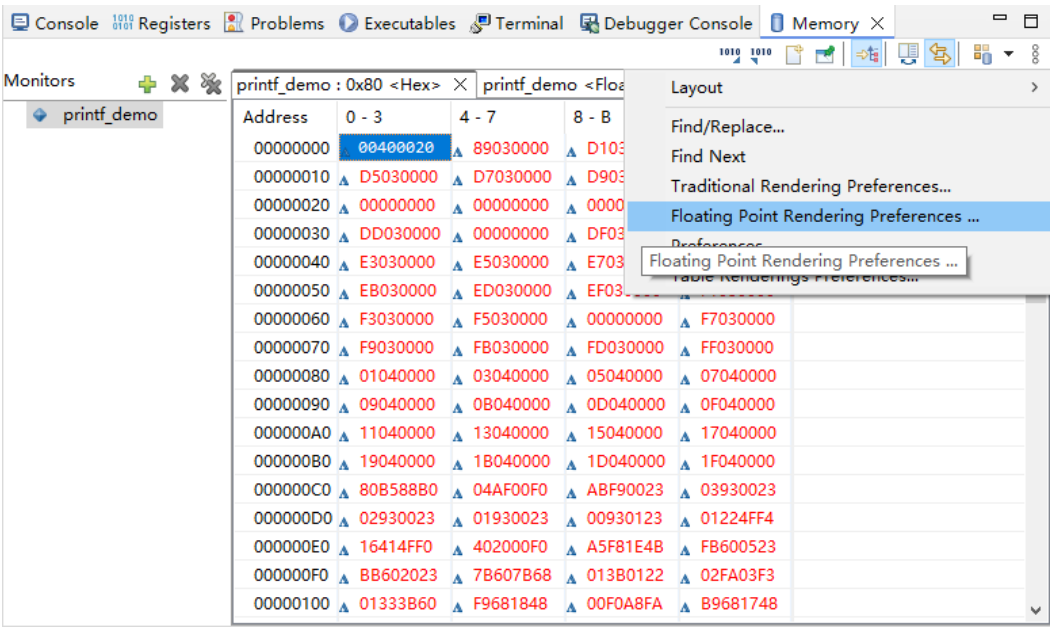


设置渲染效果

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。

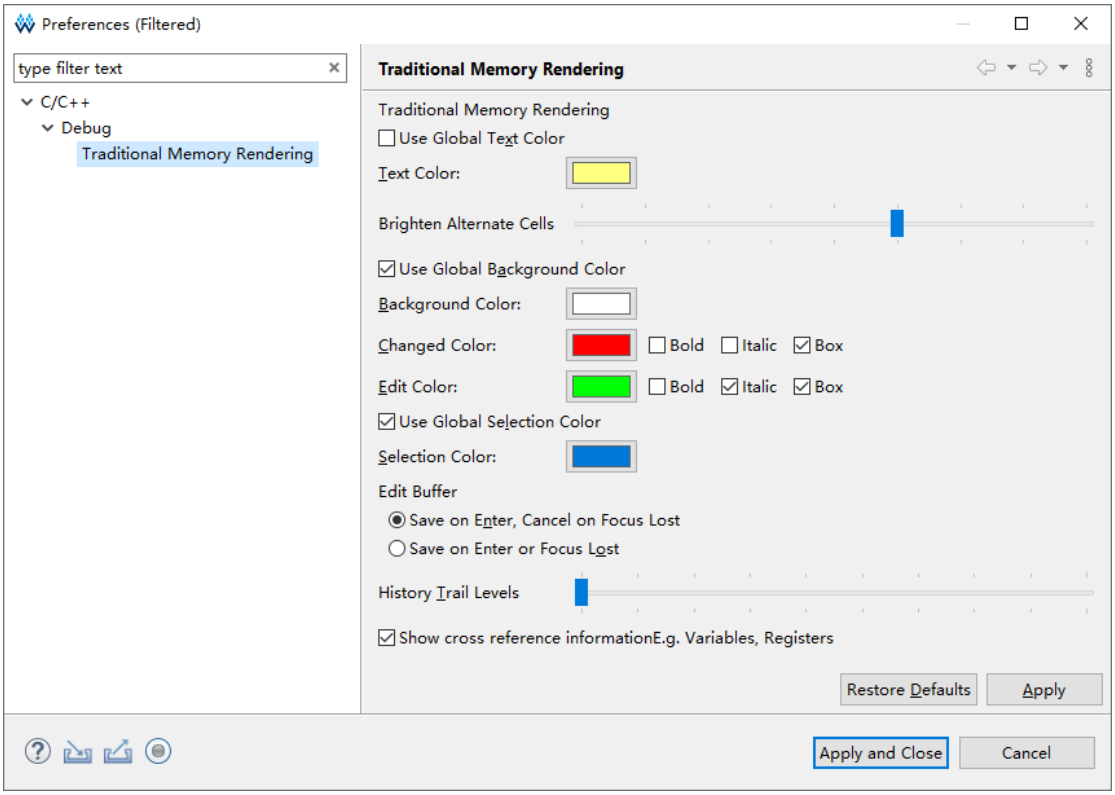
1. 内存视图中，选择“View Menu”（ ），弹出的下拉菜单中可以选择“Traditional Rendering Preferences…”或“Floating Point Rendering Preferences…”，分别设置传统格式或浮点格式的内存渲染器的渲染效果，如图 10-24 所示。

图 10-24 设置渲染效果



2. 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。

图 10-25 设置内存渲染效果

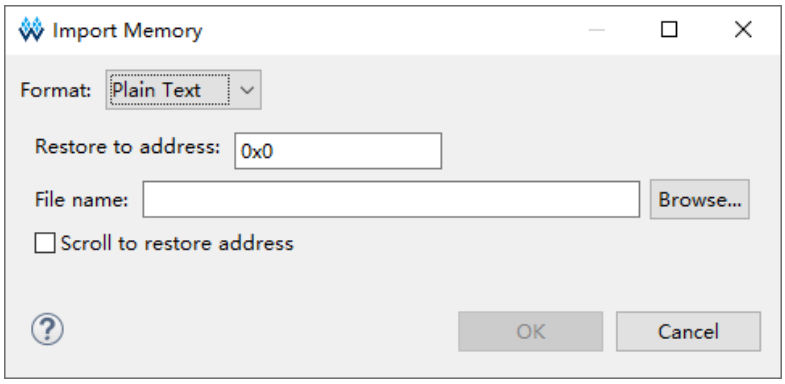


视图工具栏功能

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



- Export（ ）：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

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。

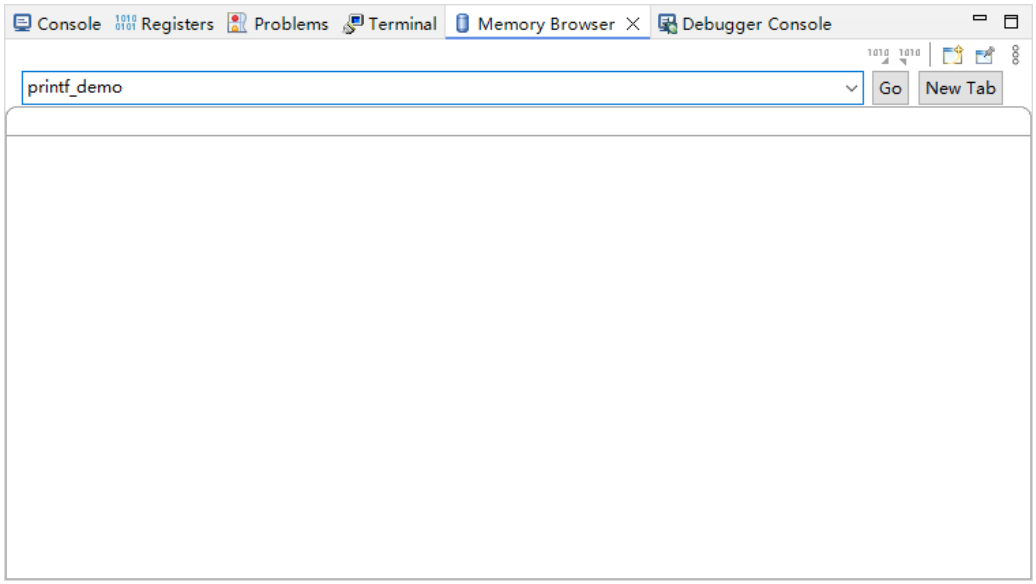
10.7.2 内存浏览器视图

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

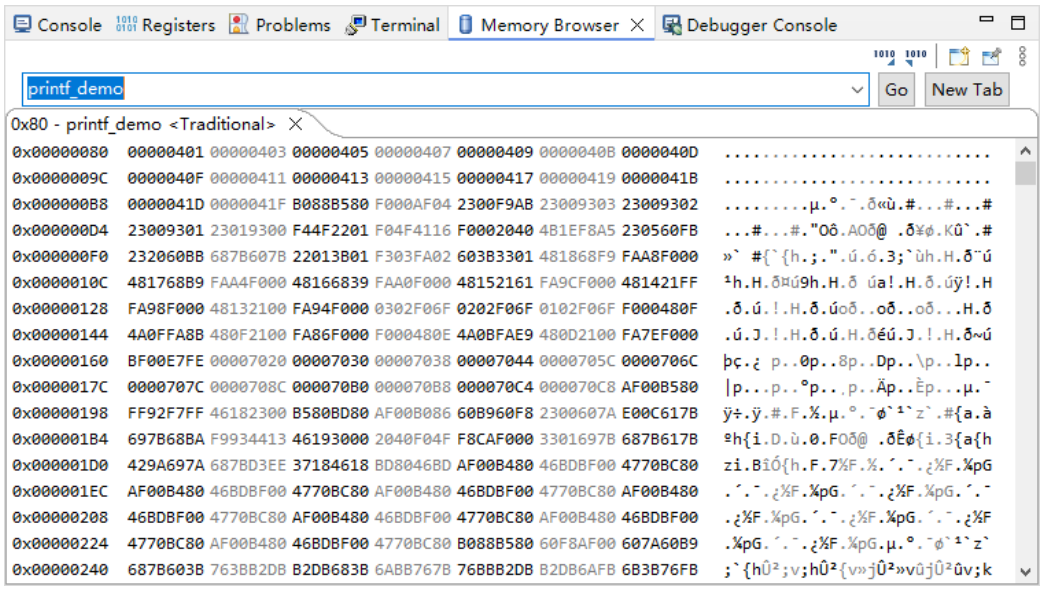
1. 程序运行暂停后，在内存浏览器视图的文本框中输入想要查看的内存地址或表达式，选择“Go”，建立一个内存监视器，如图 10-28 所示。

图 10-28 内存浏览器视图



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。

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。

图 10-30 建立浮点格式的内存渲染器

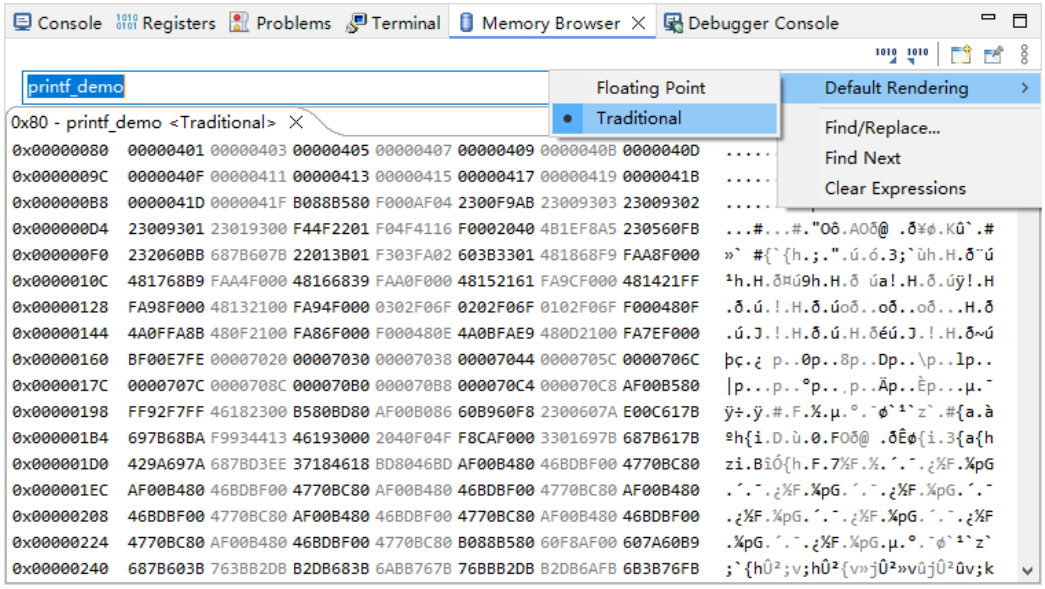
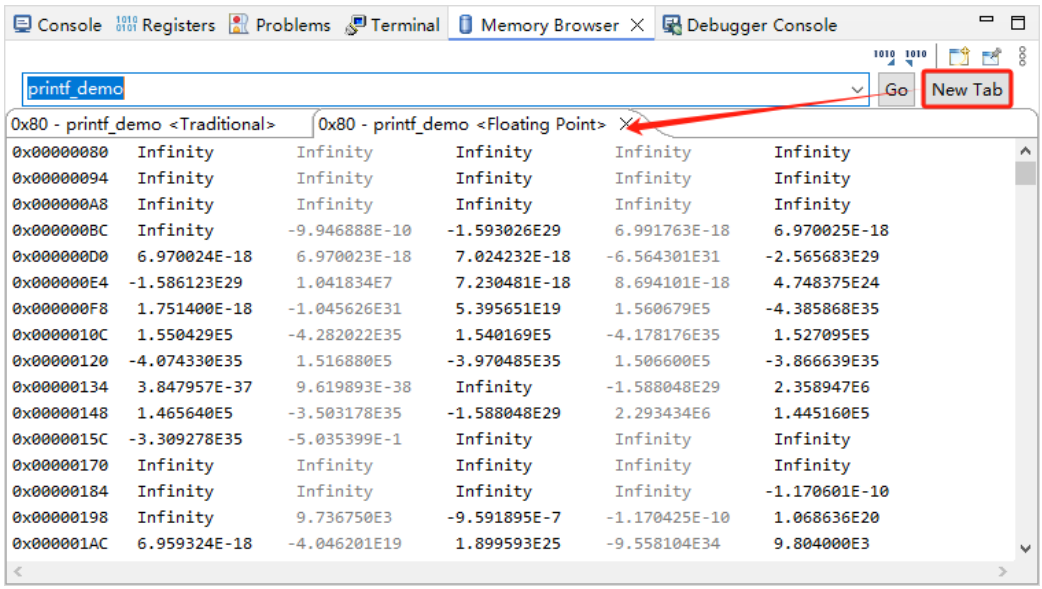
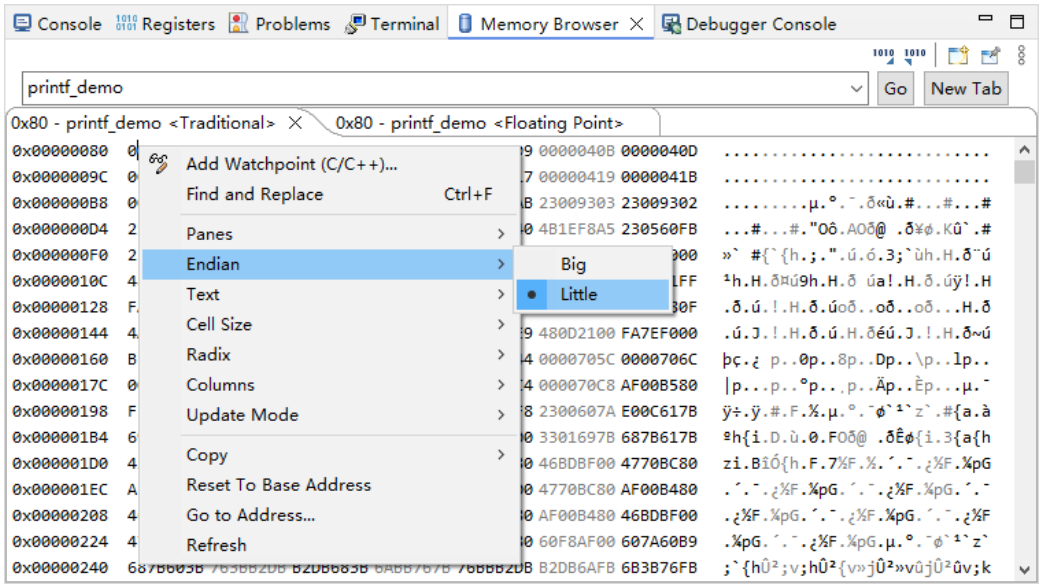


图 10-31 浮点格式的内存渲染器



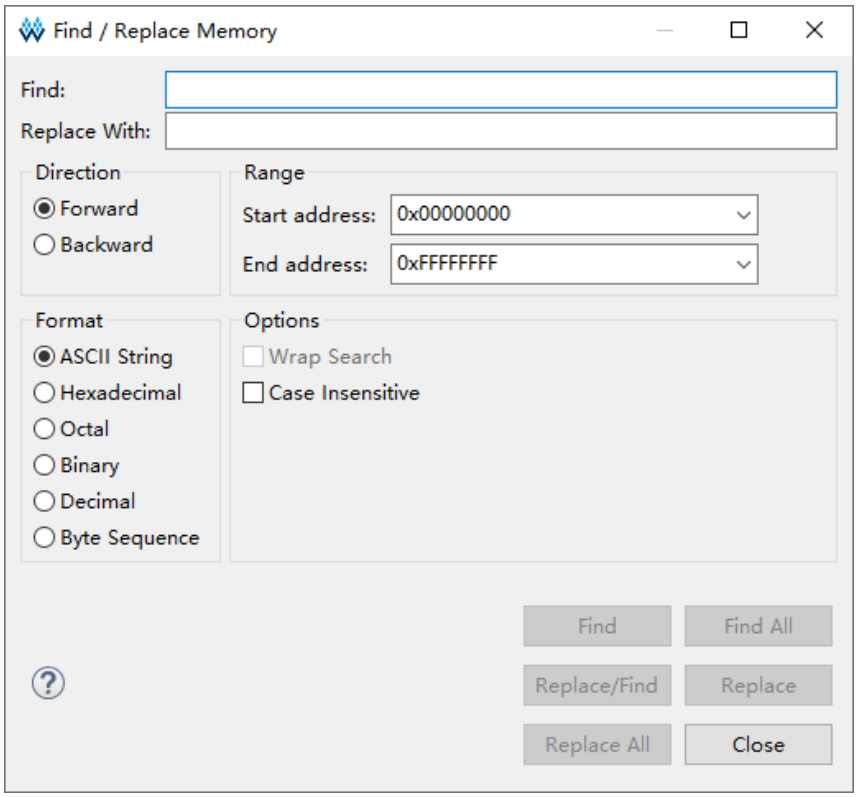
3. 右键内存渲染器的任意位置，弹出的下拉菜单中选择“Endian”，可以修改数据显示的字节顺序或可寻址大小，如图 10-32 所示。

图 10-32 设置字节顺序



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。

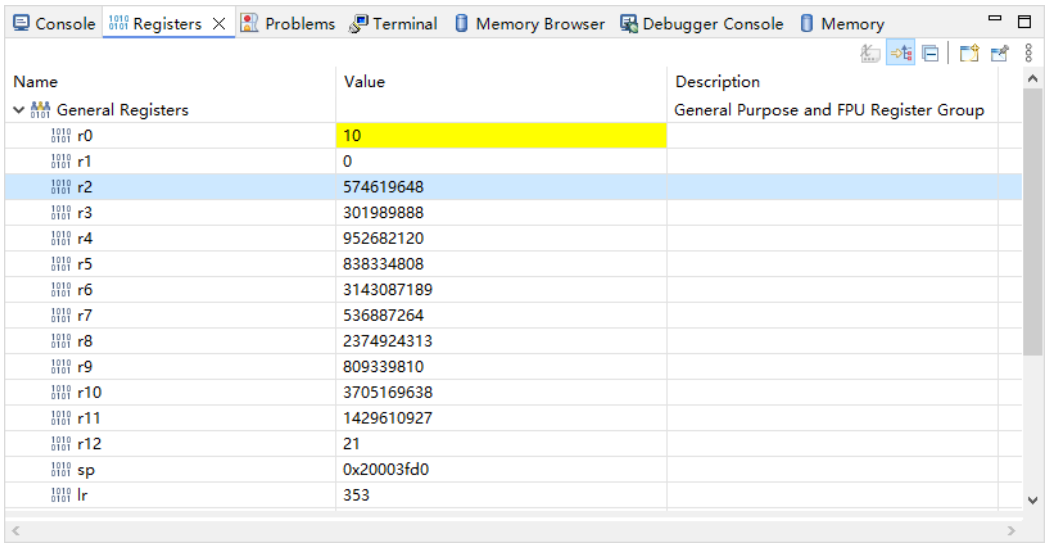
图 10-33 Find/Replace



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。

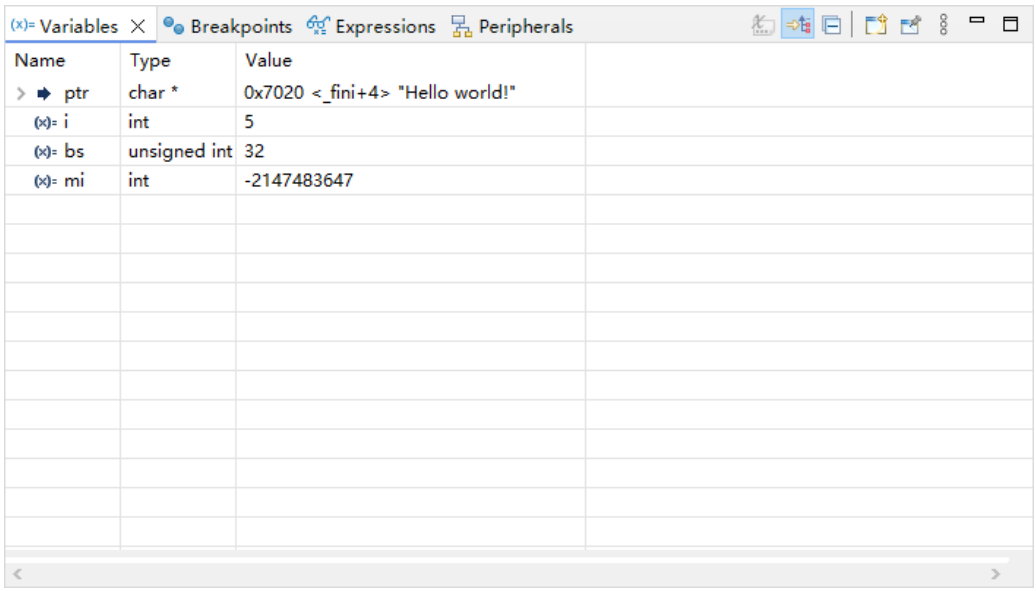
图 10-34 寄存器视图



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。

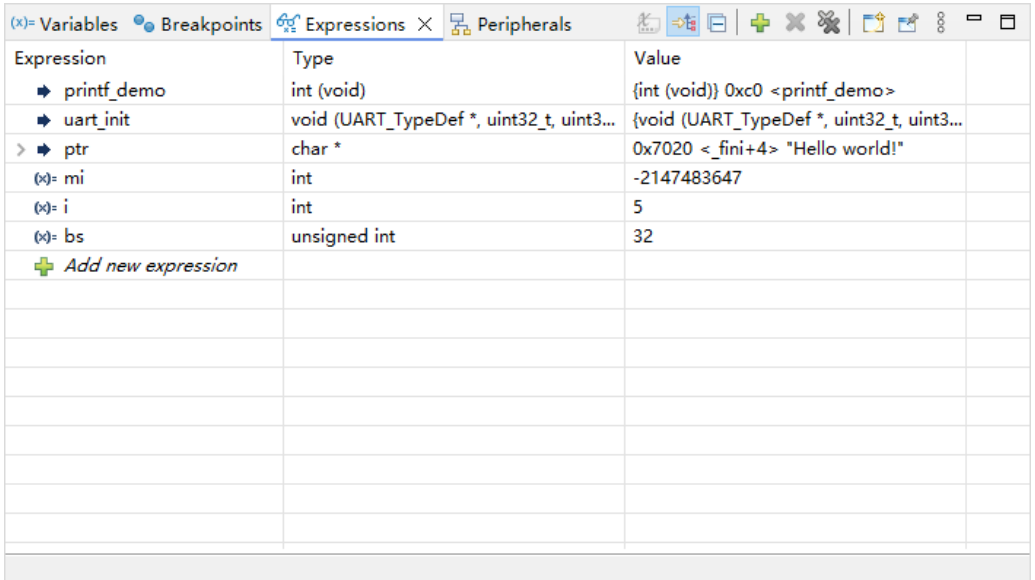
图 10-35 变量视图



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。

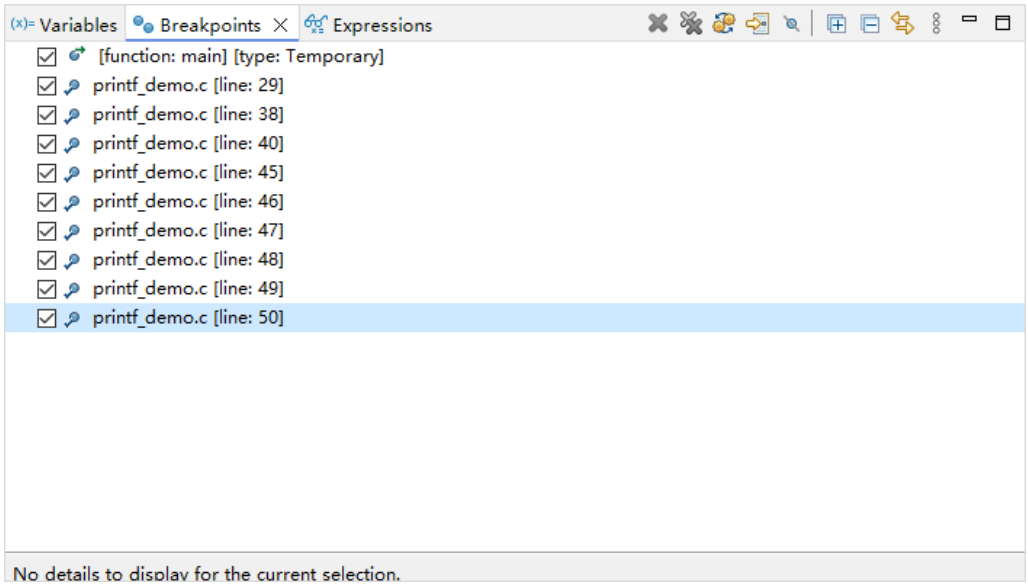
图 10-36 表达式视图



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。

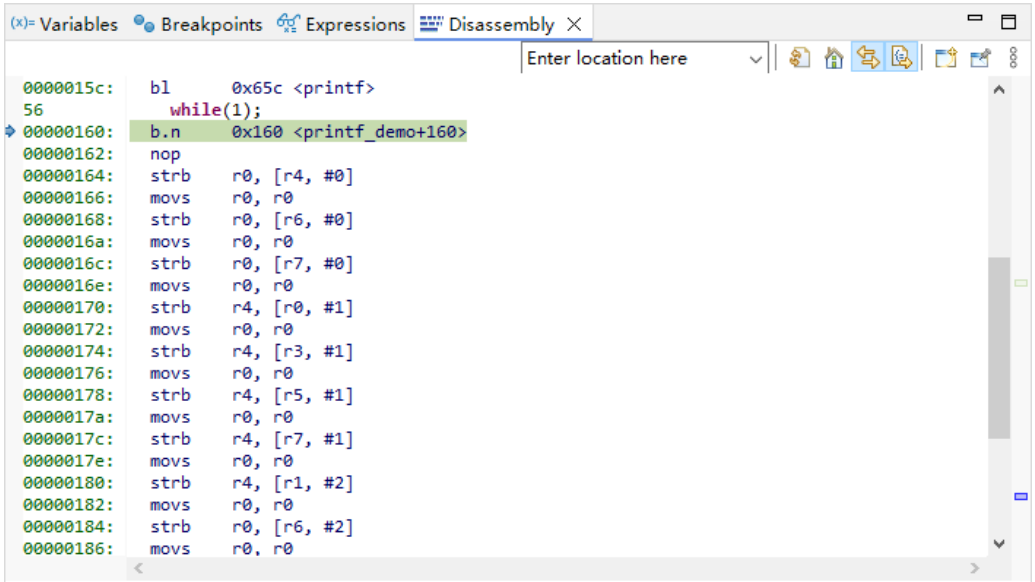
图 10-37 断点视图



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。

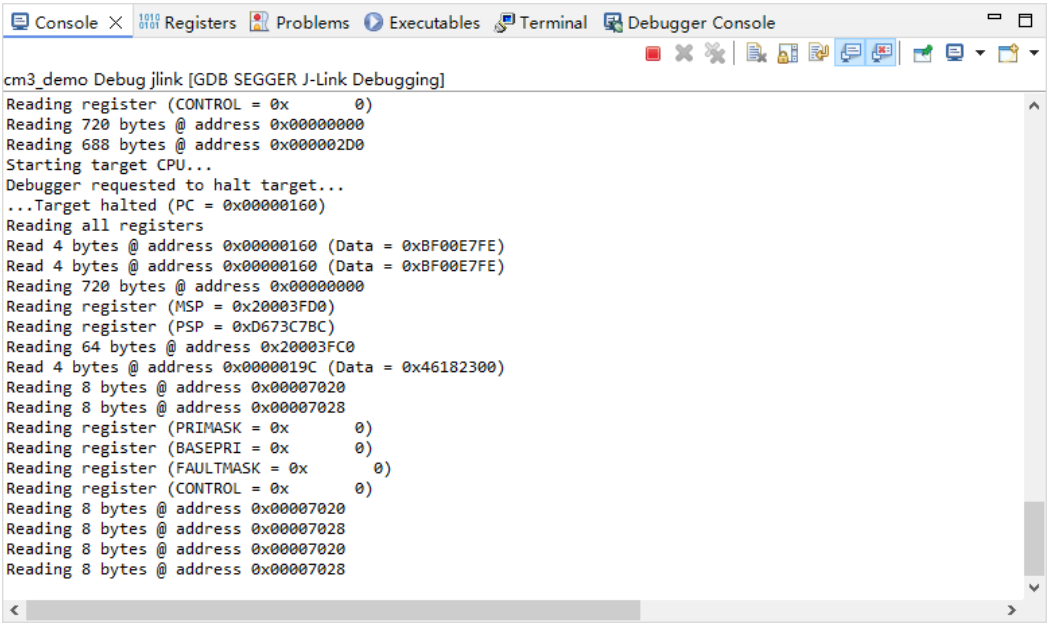
图 10-38 反汇编视图



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



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。

表 11-1 已支持的外设功能

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

11.2 QEMU 使用方法

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

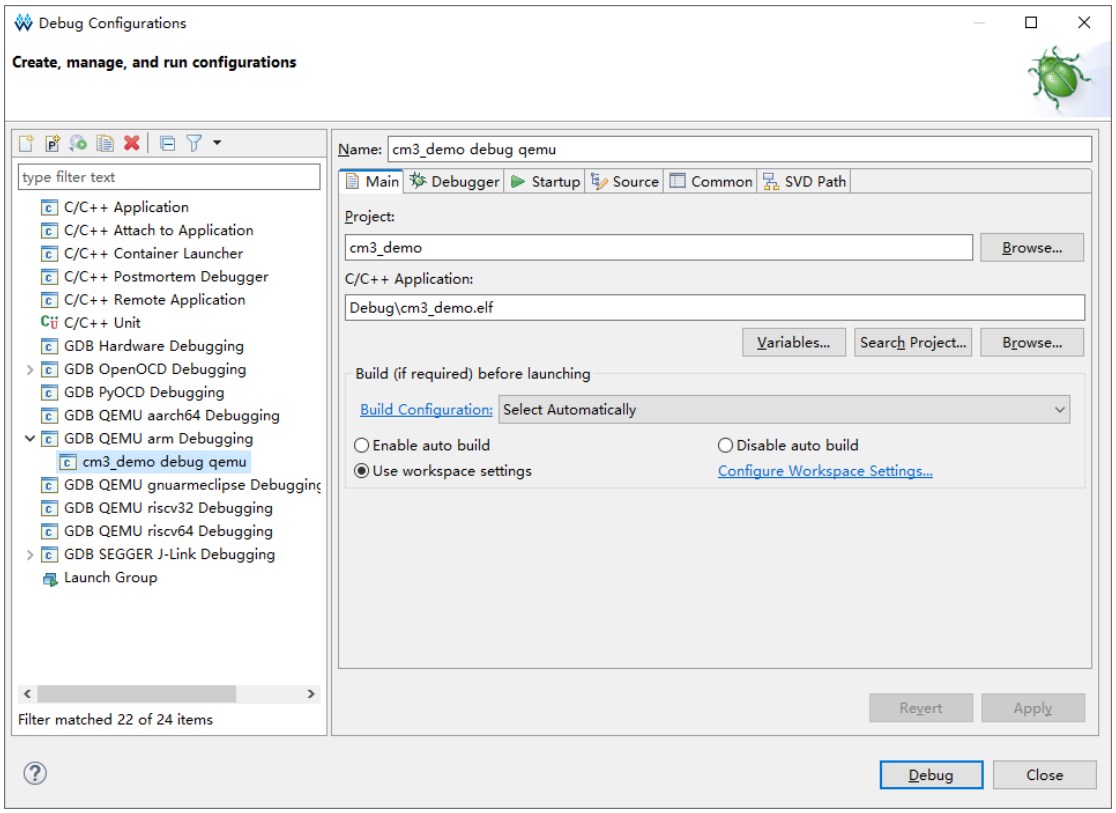
- 选定当前工程，右键选择“Debug As > Debug Configurations…”
- 选择工具栏“Debug > Debug Configurations…”（  ）
- 选择菜单栏“Run > Debug Configurations…”

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。

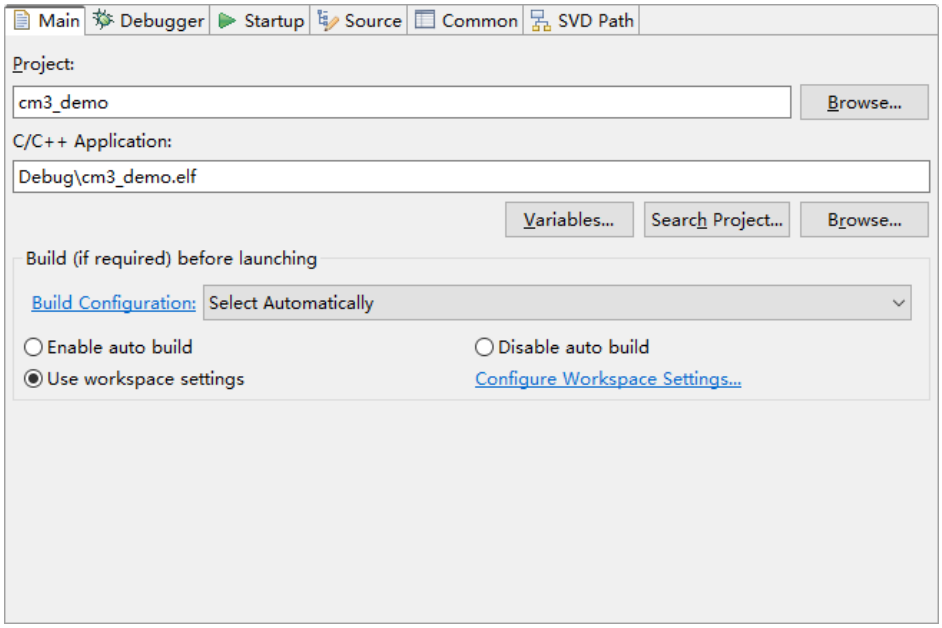
图 11-1 GDB QEMU arm Debugging



1. Main 视图

Main 视图中的 QEMU 设置如图 11-2 所示。

图 11-2 Main 视图



2. Debugger 视图

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

图 11-3 Debugger 视图

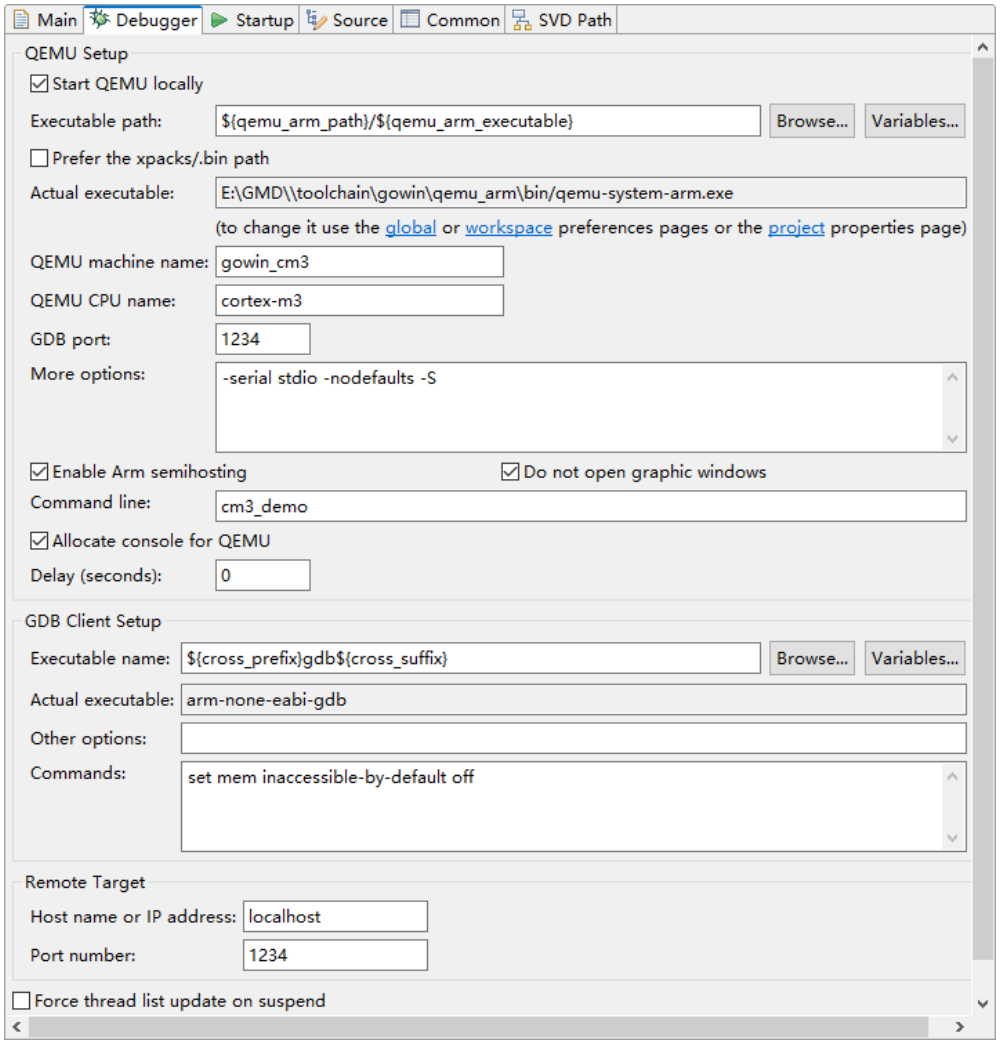


表 11-2 Debugger 视图

选项	设置
Start QEMU locally	√
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	√
Do not open graphic windows	√
Allocate console for QEMU	√
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

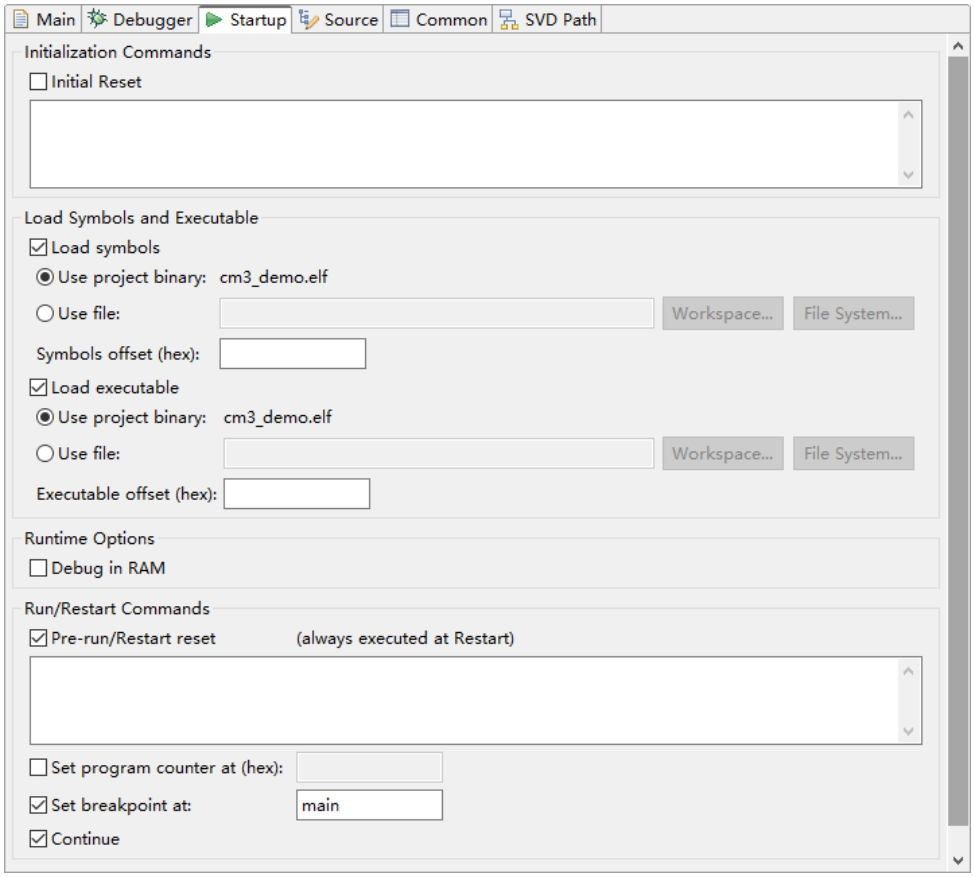


表 11-4 Startup 视图

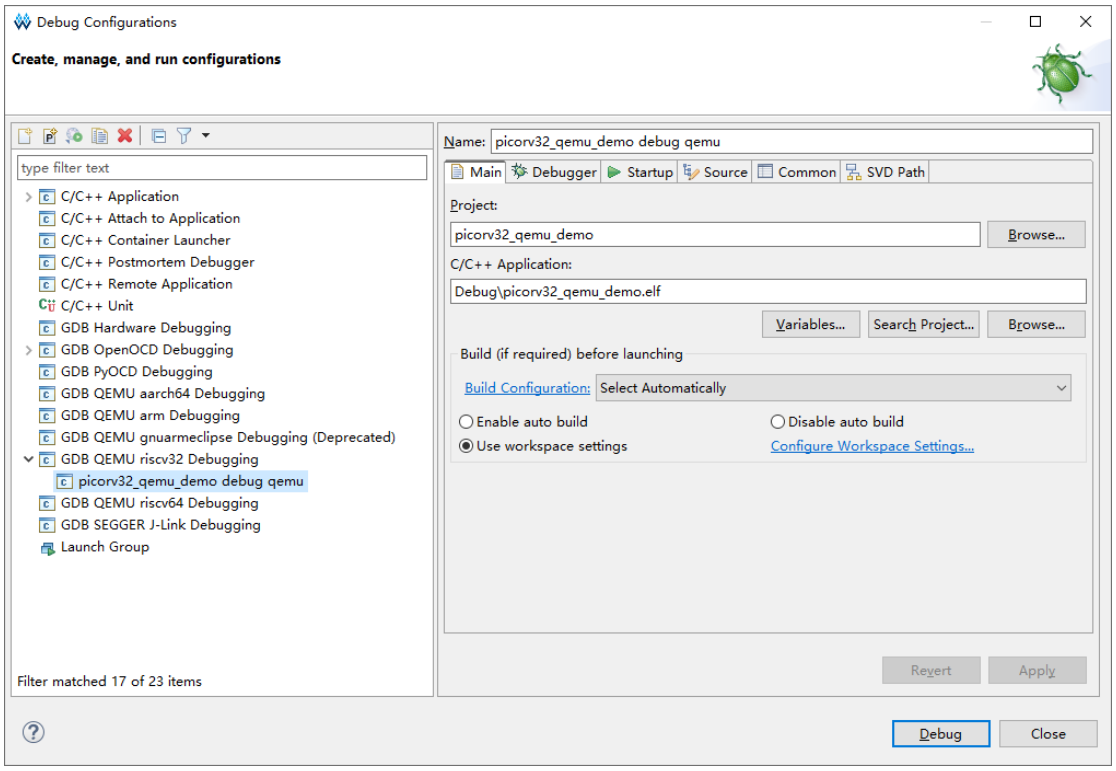
选项	设置
Initial Reset.	×
Load symbols	√
Load executable	√
Debug in RAM	×
Pre-run/Restart reset	√
Set program counter at (hex)	-
Set breakpoint at	main

选项	设置
Continue	√

11.2.2 RISC-V MCU QEMU

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

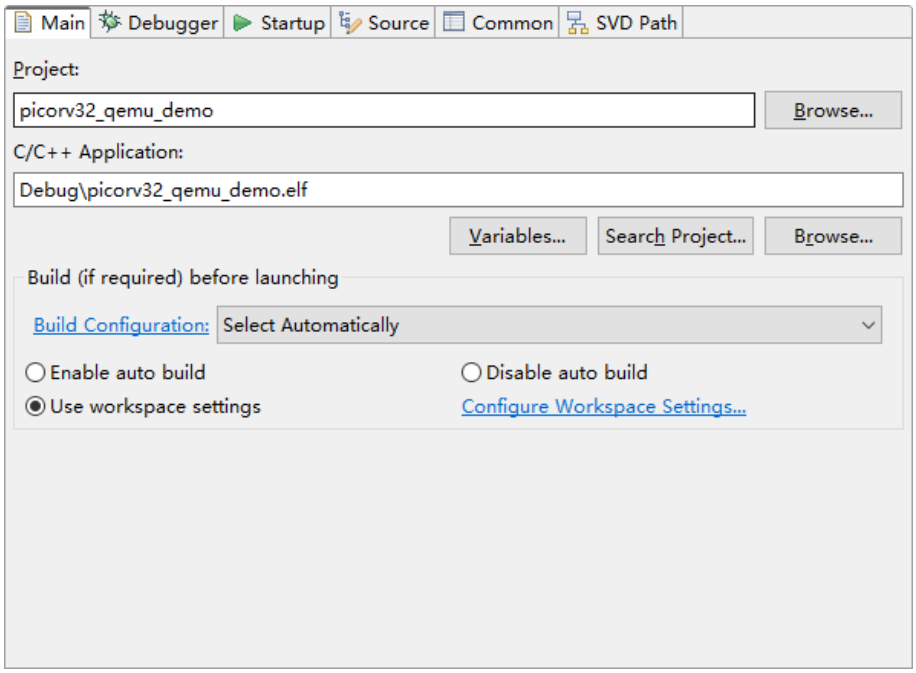
图 11-5 GDB QEMU riscv32 Debugging



1. Main 视图

Main 视图中的 QEMU 设置如图 11-6 所示。

图 11-6 Main 视图



2. Debugger 视图

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

图 11-7 Debugger 视图

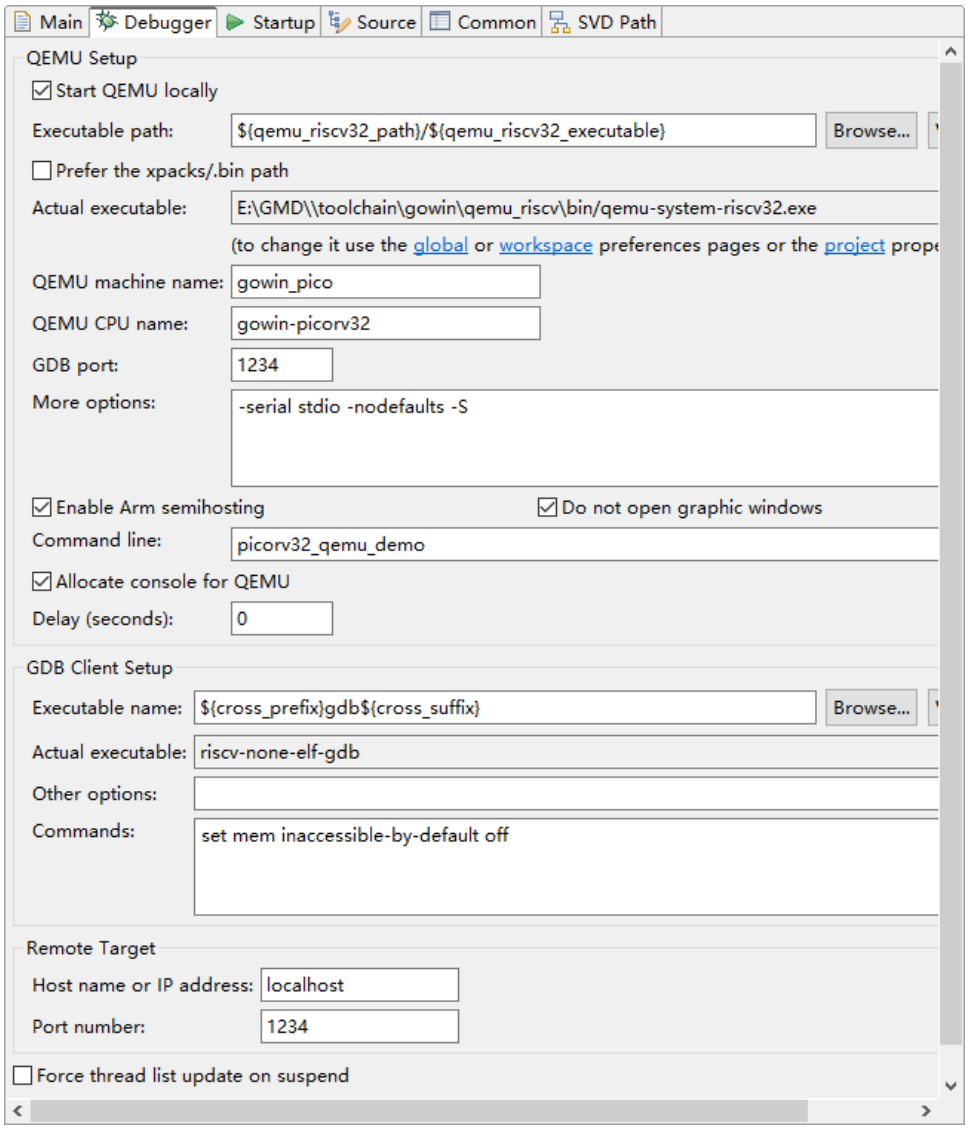


表 11-5 Debugger 视图

选项	设置
Start QEMU locally	√
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	√
Do not open graphic windows	√
Allocate console for QEMU	√
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

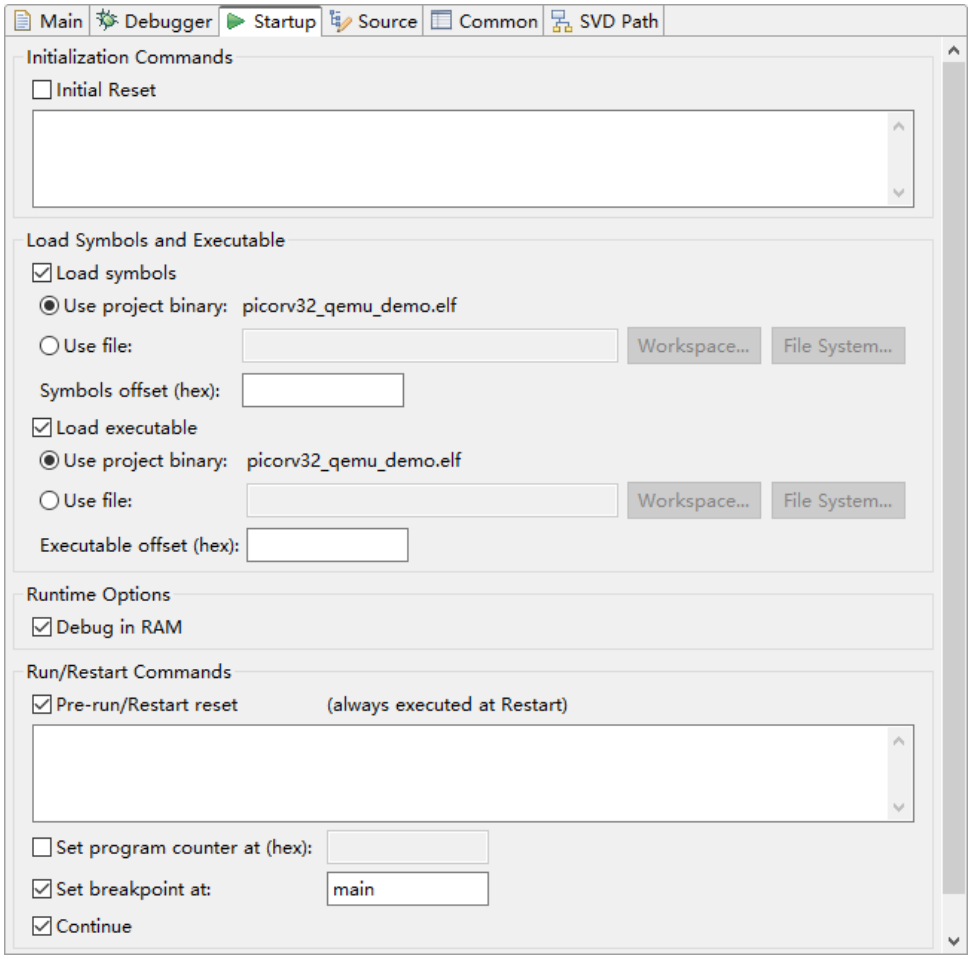


表 11-7 Startup 视图

选项	设置
Initial Reset.	×
Load symbols	√

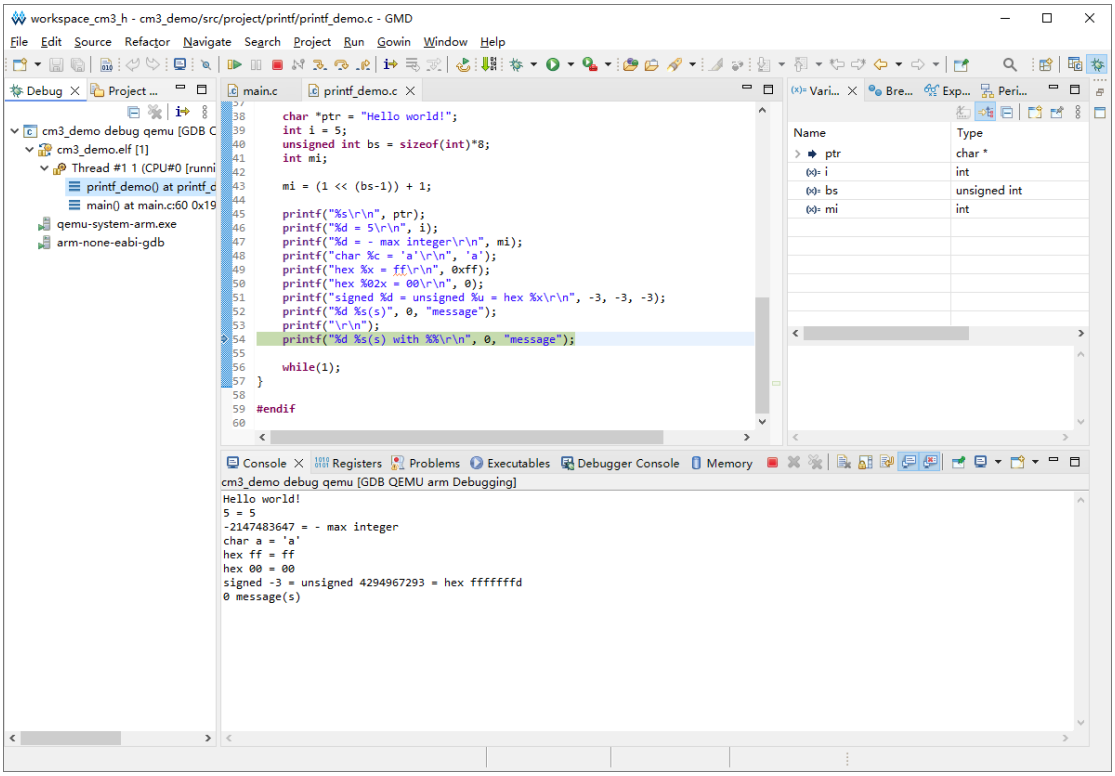
选项	设置
Load executable	√
Debug in RAM	√
Pre-run/Restart reset	√
Set program counter at (hex)	-
Set breakpoint at	main
Continue	√

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

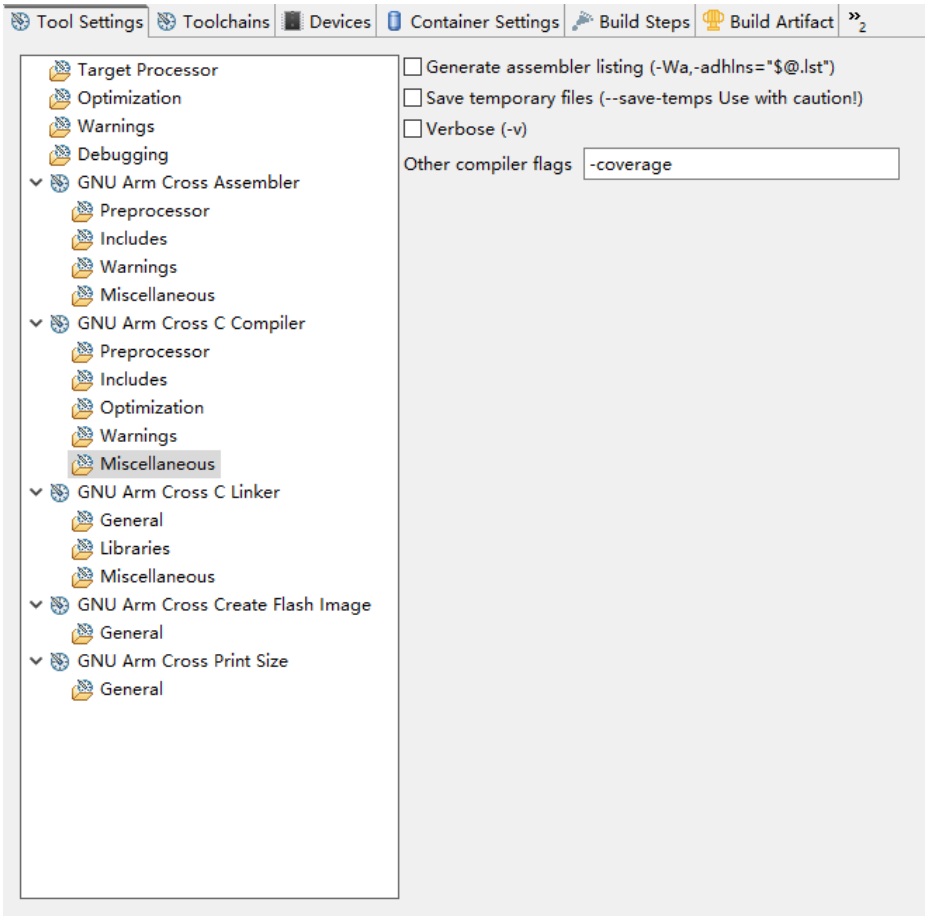
工程创建后，对想要进行代码分析的文件或文件夹设置一个额外的编译选项-coverage，然后编译工程。

- 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。

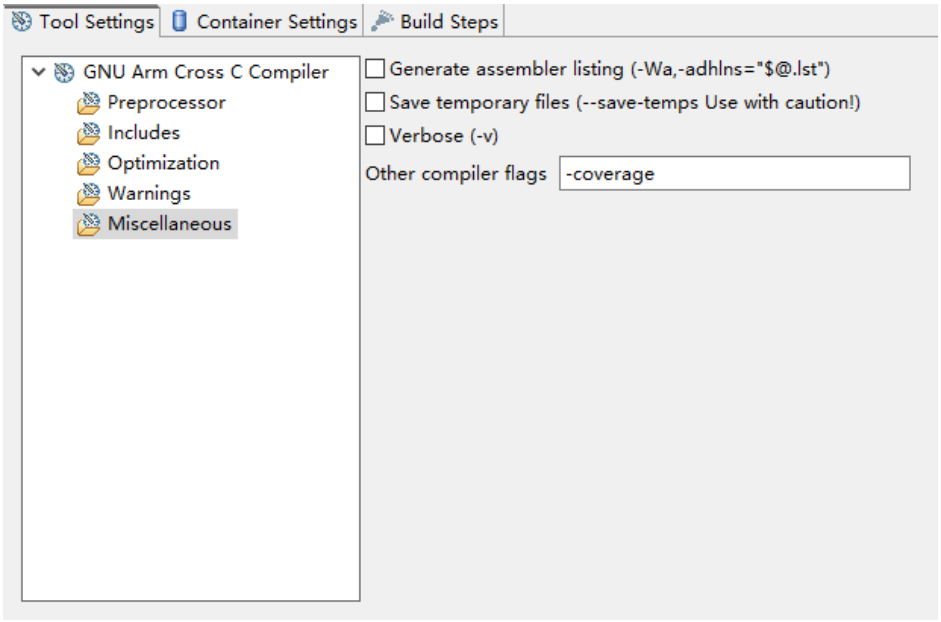
图 12-1 指定所有源代码



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。

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。

图 12-3 设置 Linux Tools Path

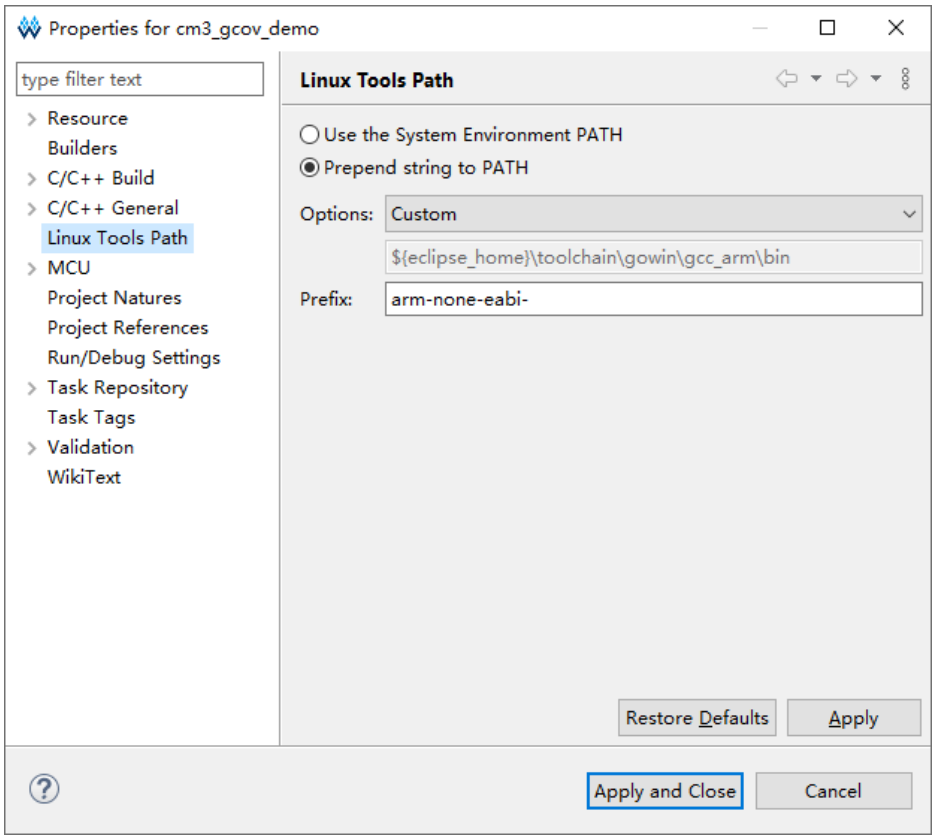
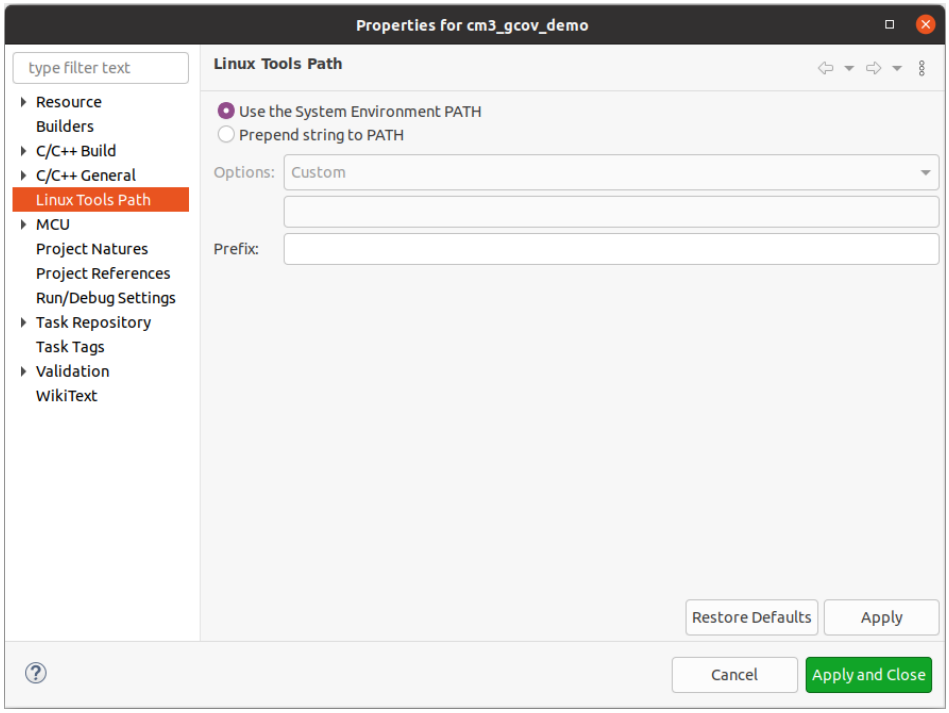


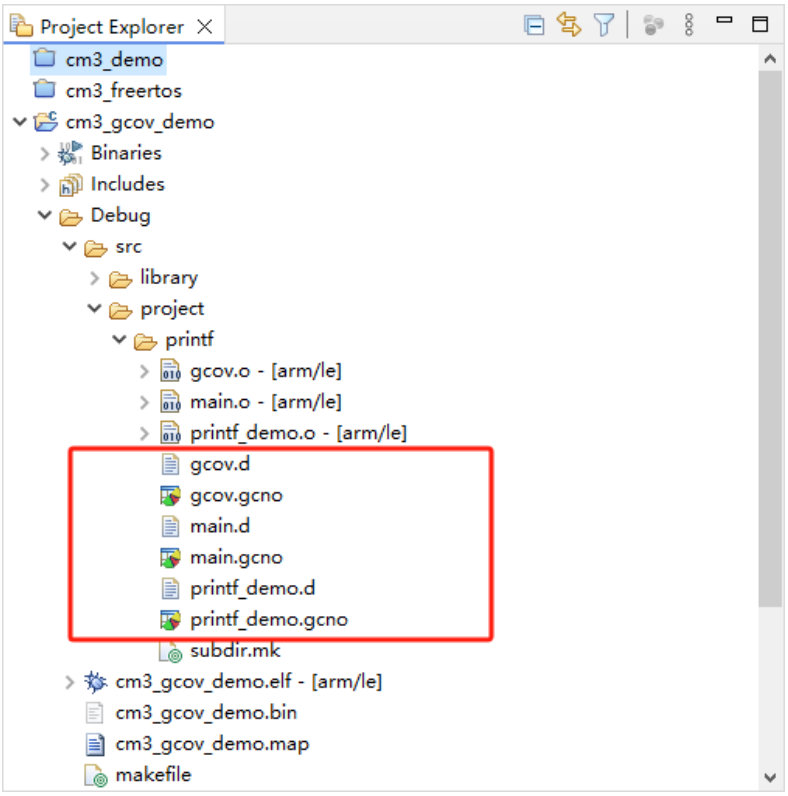
图 12-4 设置 Linux Tools Path



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



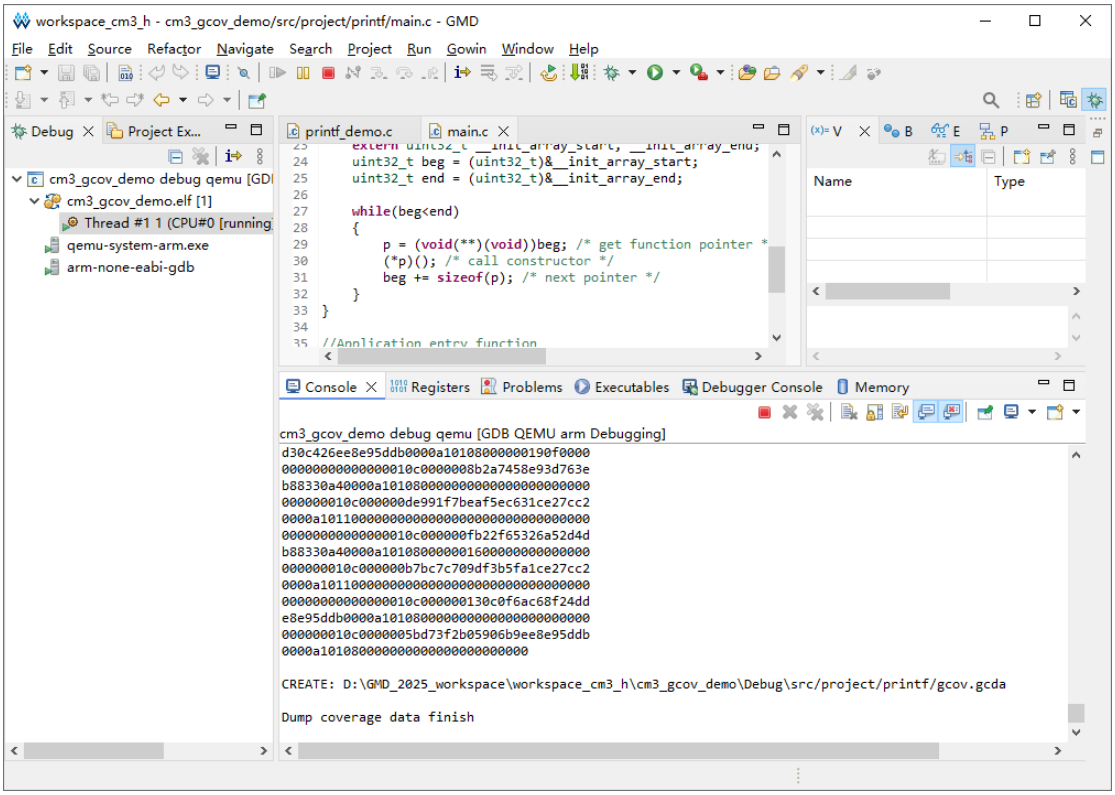
12.2.4 运行工程及使用 Code Coverage 功能

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：

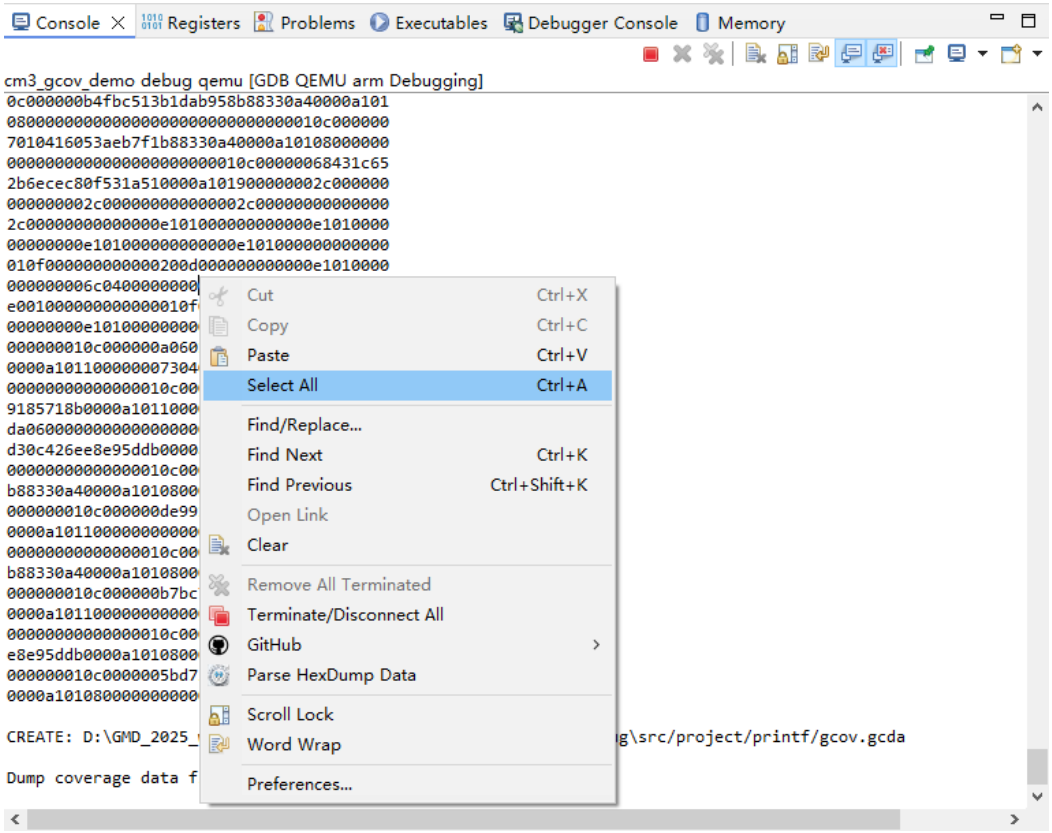
1. “Debug As > Debug Configurations… > GDB QEMU arm Debugging”，在 QEMU 中启动调试运行，运行工程并在控制台中查看 Profiling 信息，如图 12-6 所示。

图 12-6 运行工程并查看 Profiling 信息



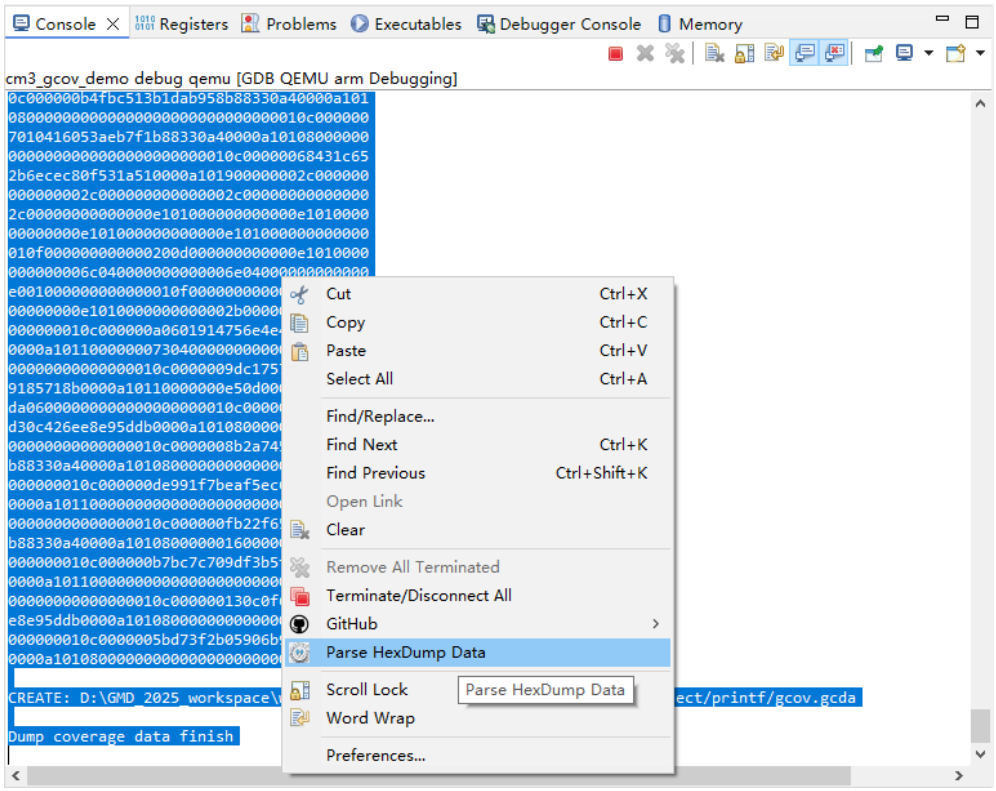
2. 输出的 Profiling 信息需要解析后 GMD 软件才可以正确读取，在控制台内任意位置右键，在弹出的下拉菜单中选择“Select All”，选中所有的输出信息，如图 12-7 所示。

图 12-7 Select All



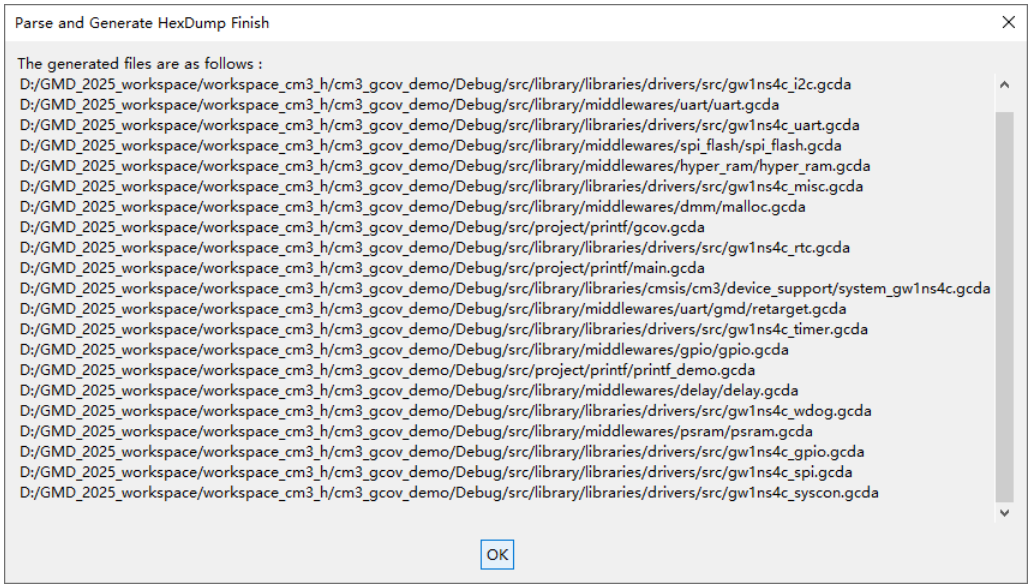
3. 然后在控制台中任意位置再次右键，选择“Parse HexDump Data”，如图 12-8 所示。

图 12-8 Parse HexDump Data



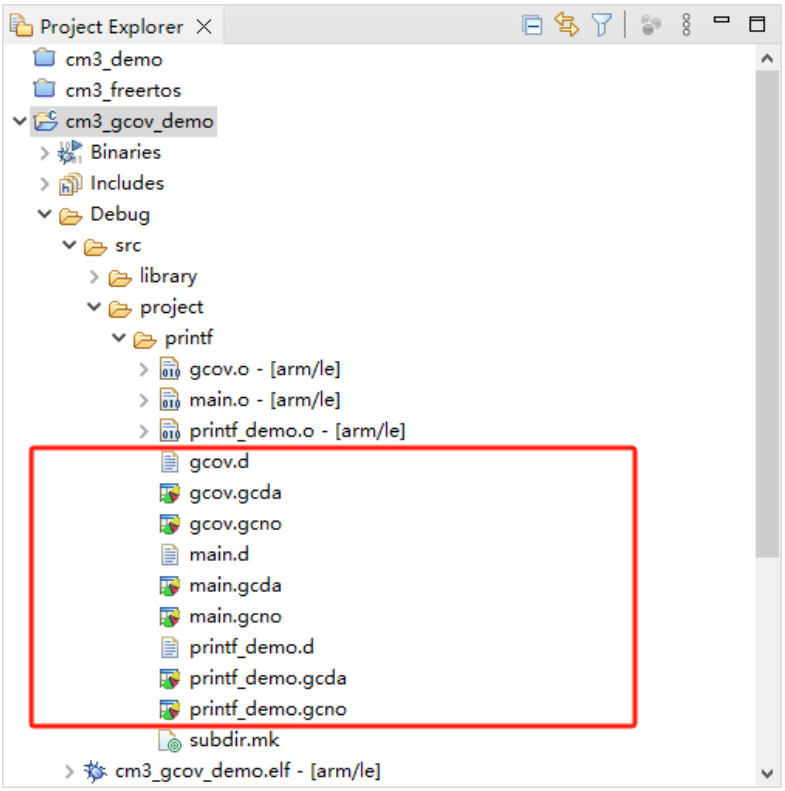
4. 此时 GMD 软件会对输出的文件进行分析，并将结果分别保存在对应的文件中，如图 12-9 所示。

图 12-9 解析 Profiling 信息并生成对应文件



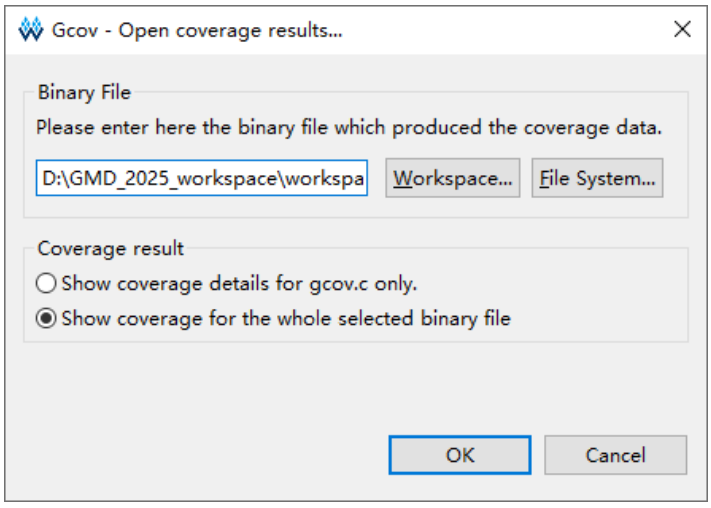
5. 再次查看工程的 Debug 文件夹，可以看到产生了对应的.gcda 文件，如图 12-10 所示。

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

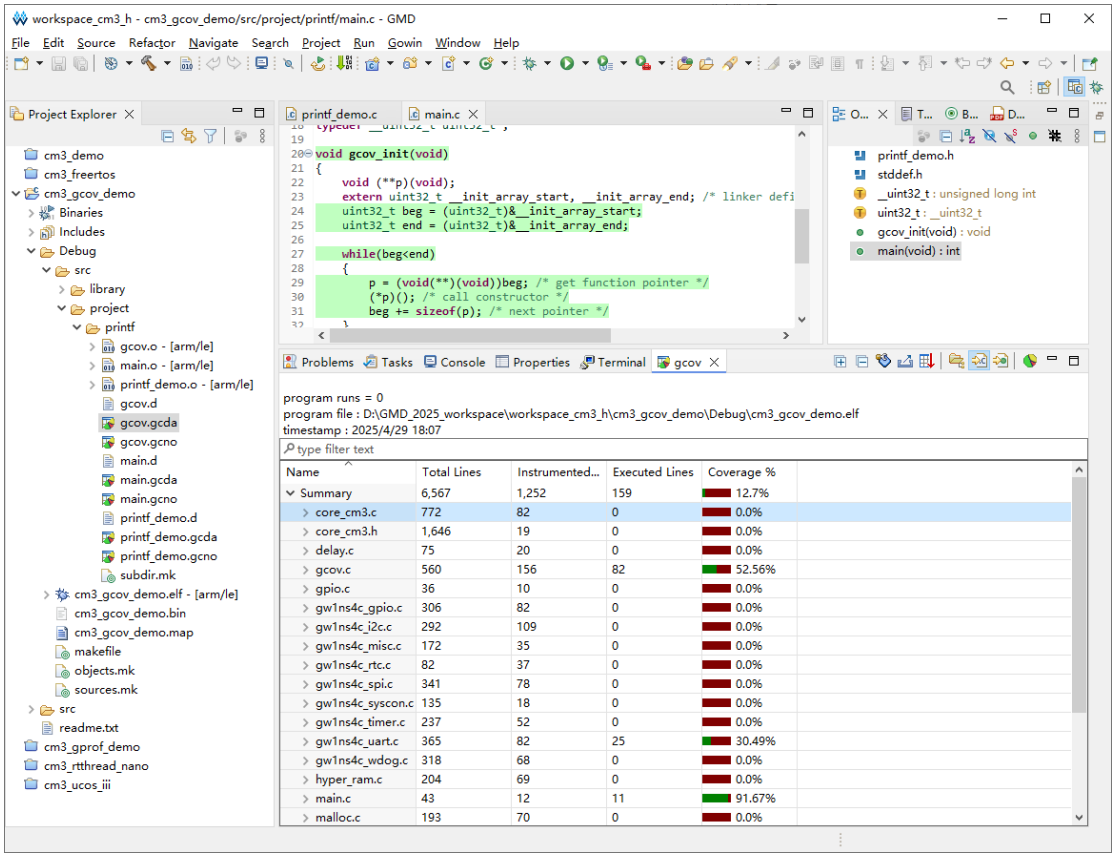


7. 然后可以看到对应用程序的分析结果，在结果中显示了某个文件或某个方法在程序运行过程中是否执行到，以及代码执行覆盖比等数据。双击 Gcov 中的某一行，GMD 软件会自动打开对应的文件，并对文件中的代码着色，绿色表示在程序运行过程中有被执行到，红色表示在程序运行

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



Gcov 工具相关功能如表 12-1 所示。

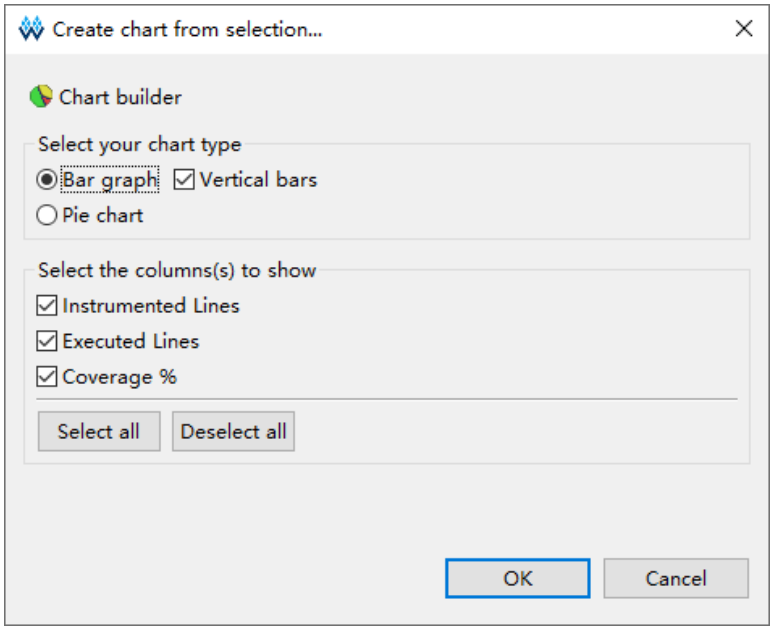
表 12-1 Gcov 功能

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…”（ ），如图 12-13 所示。

图 12-13 Create chat…



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

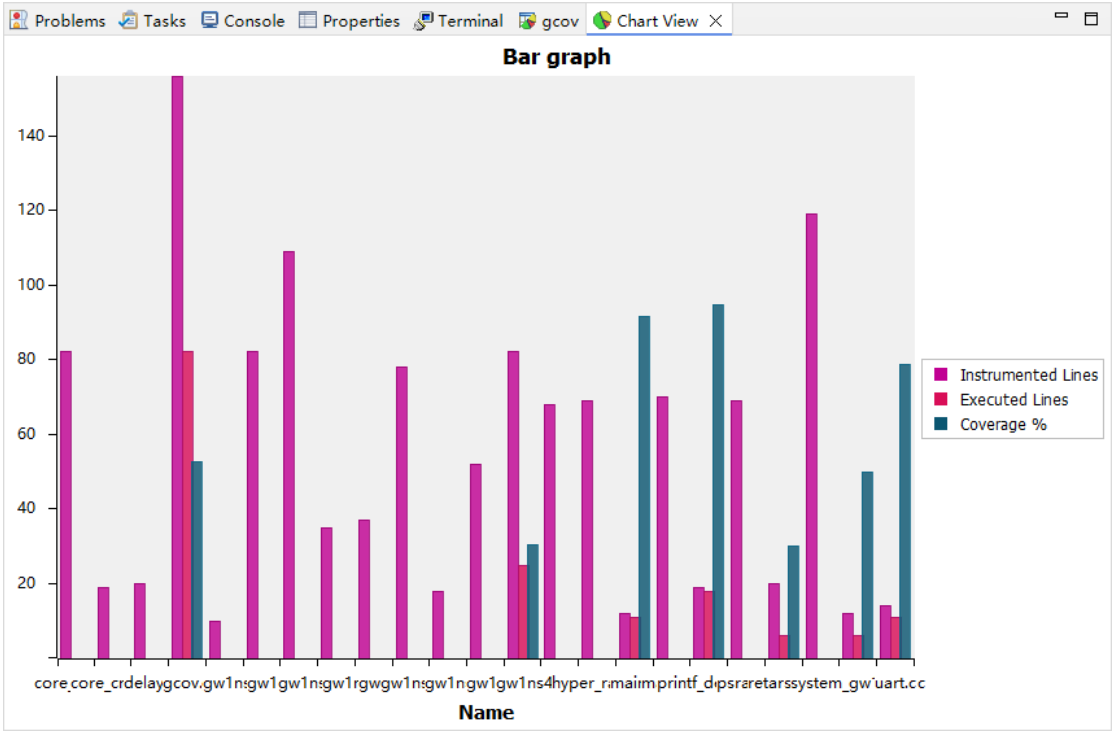
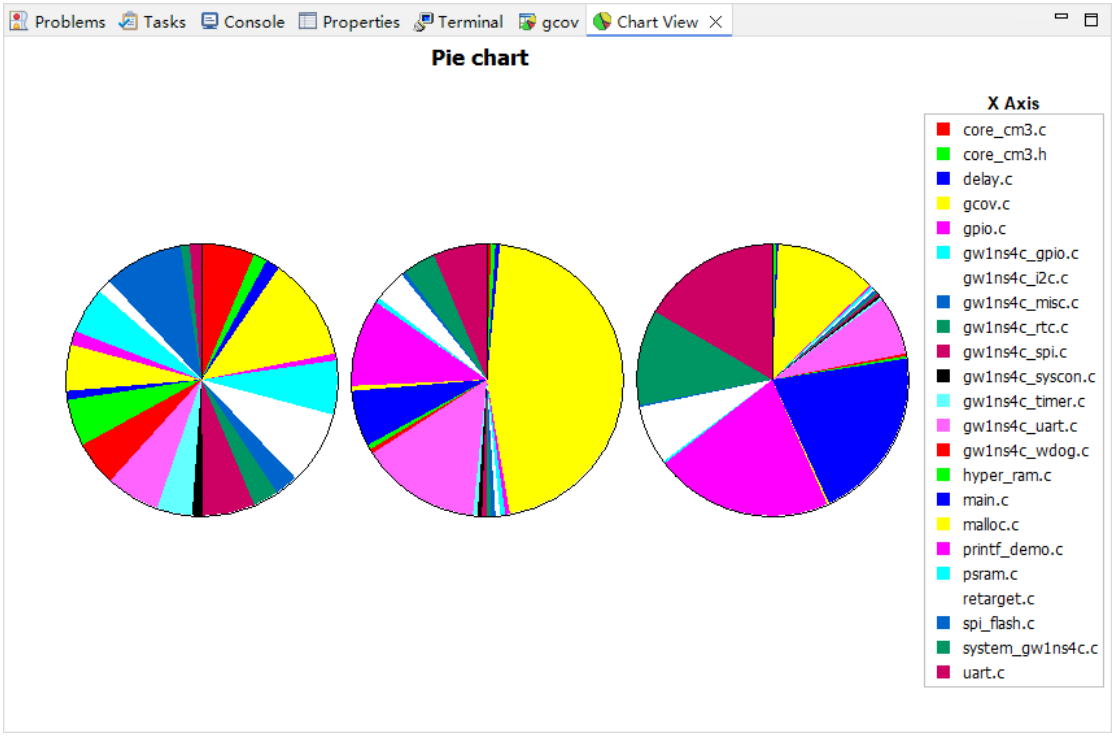


图 12-15 饼状图



13 Profiling 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

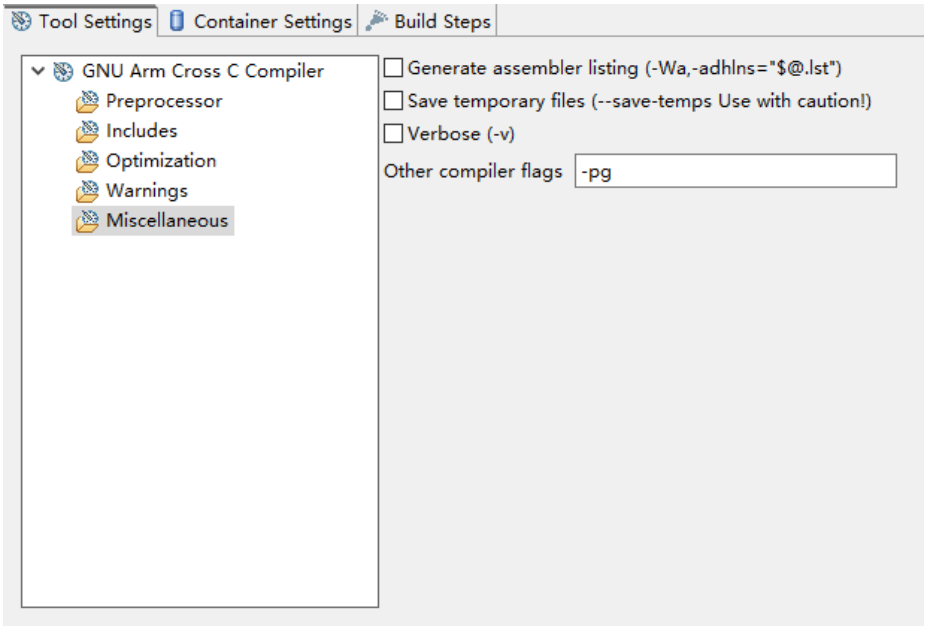
13.2.2 设置相关选项

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



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。

图 13-2 设置 Linux Tools Path

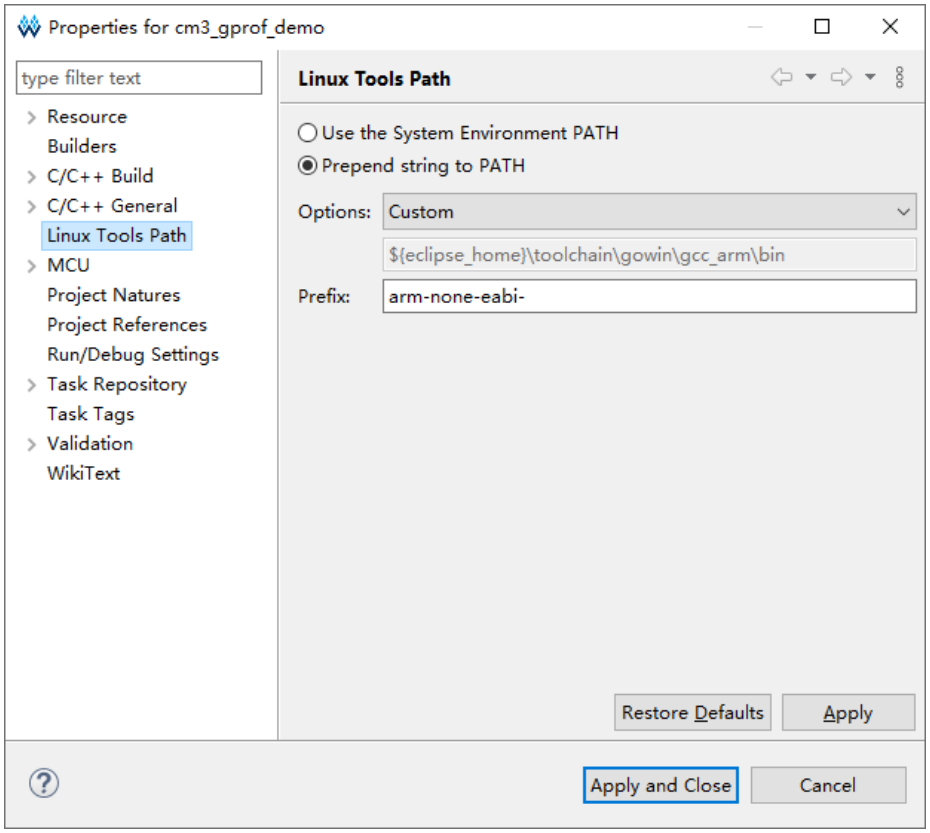
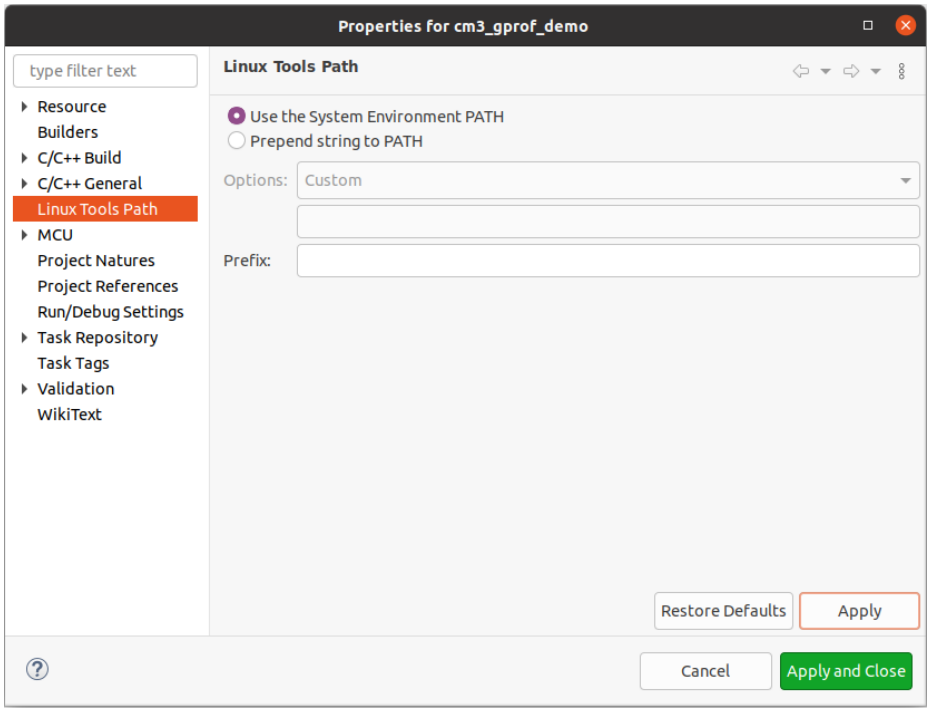


图 13-3 设置 Linux Tools Path



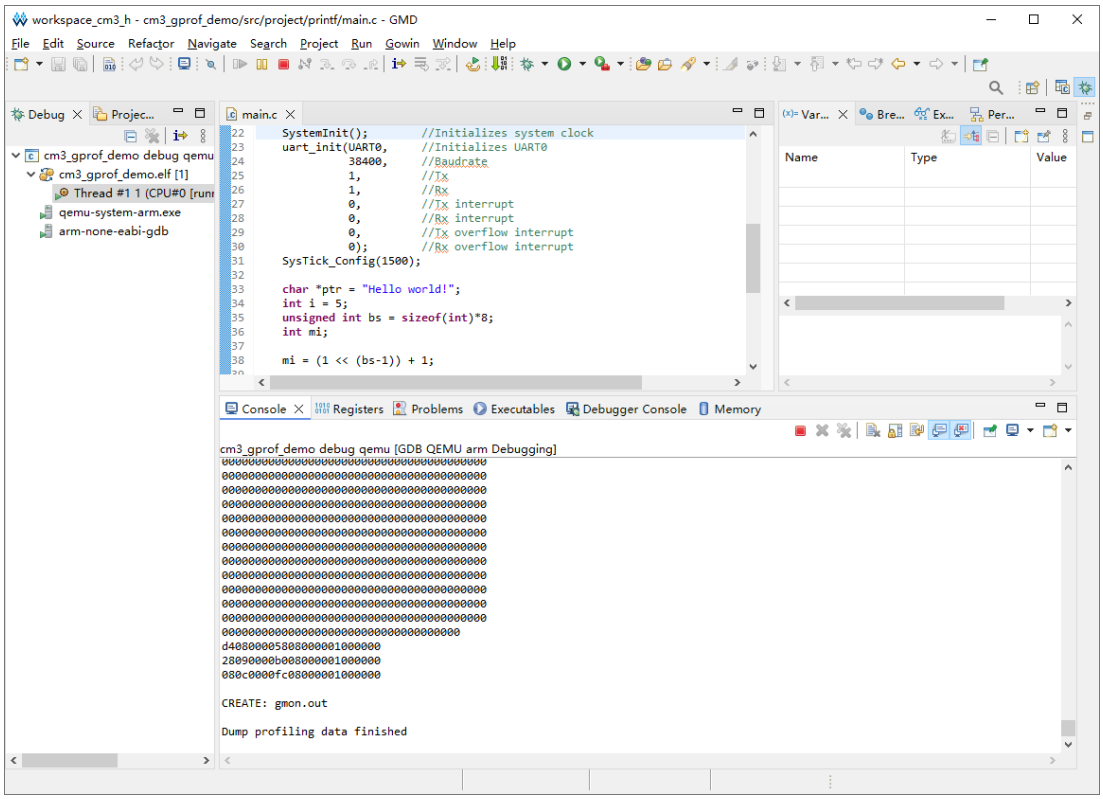
13.2.3 运行工程及使用 Profiling 功能

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：

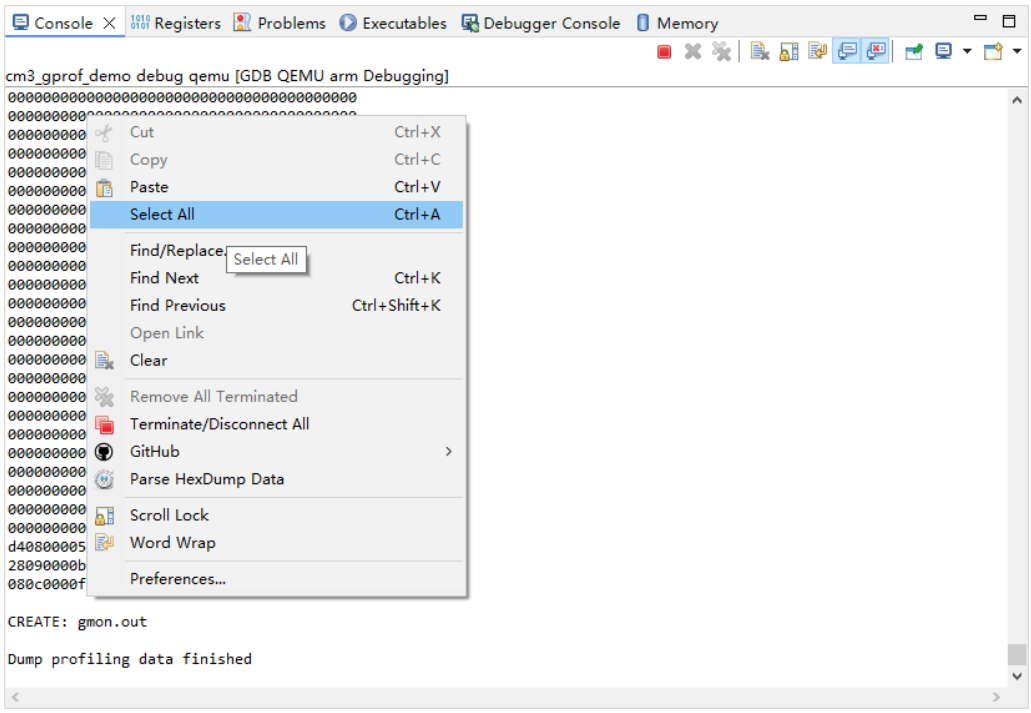
1. “Debug As > Debug Configurations… > GDB QEMU arm Debugging”，在 QEMU 中启动调试运行，运行工程并在控制台中查看 Profiling 信息，如图 13-4 所示。

图 13-4 运行工程并查看 Profiling 信息



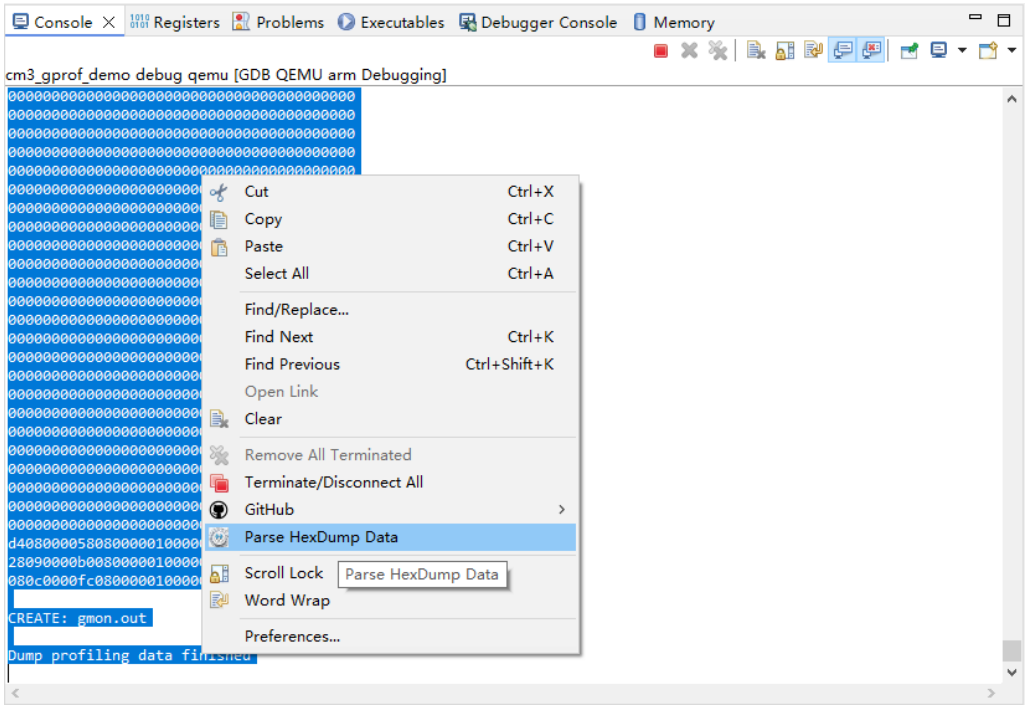
2. 输出的 Profiling 信息需要解析后 GMD 软件才可以正确读取，在控制台内任意位置右键，在弹出的下拉菜单中选择“Select All”，选中所有的输出信息，如图 13-5 所示。

图 13-5 Select All



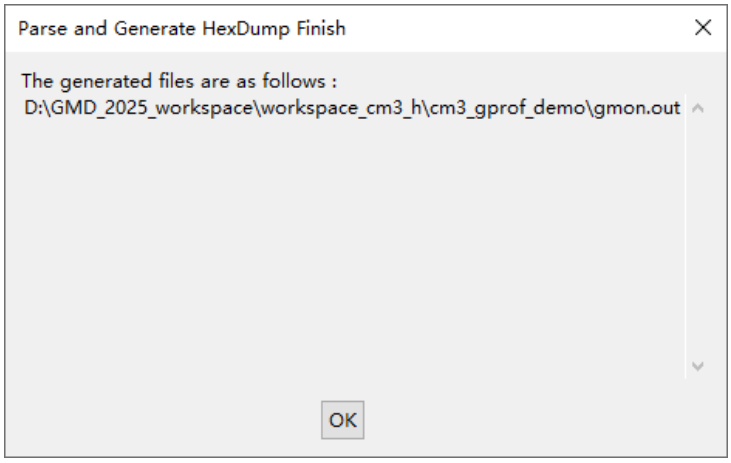
3. 然后，在控制台中任意位置再次右键，选择“Parse HexDump Data”，如图 13-6 所示。

图 13-6 Parse HexDump Data



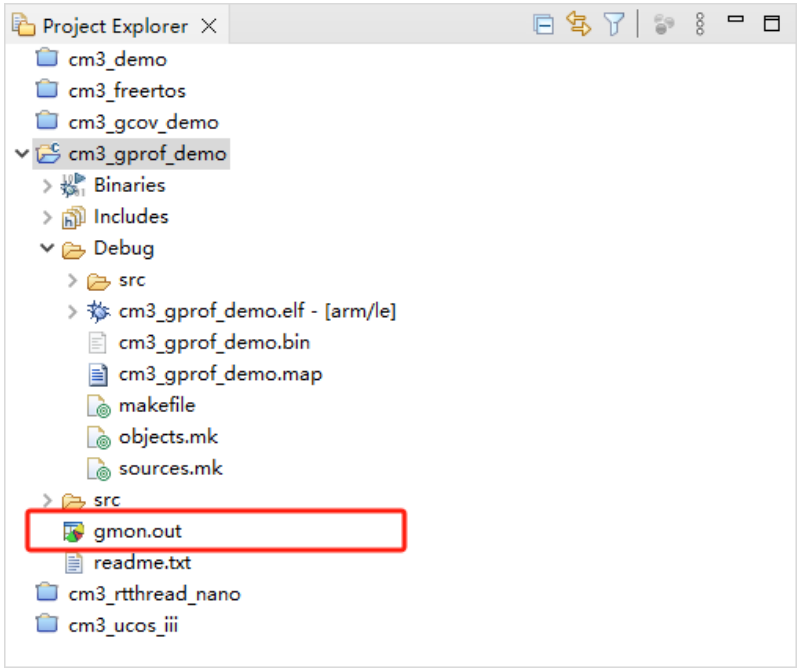
4. 此时 GMD 软件会对输出的文件进行分析，并将结果分别保存在对应的文件中，如图 13-7 所示。

图 13-7 解析 Profiling 信息并生成对应文件



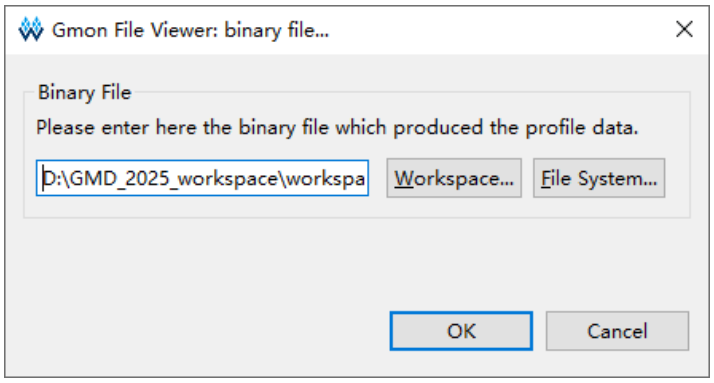
5. 查看工程的 Debug 文件夹，可以看到产生了对应的 gmon.out 文件，如图 13-8 所示。

图 13-8 查看生成的 gmon.out 文件



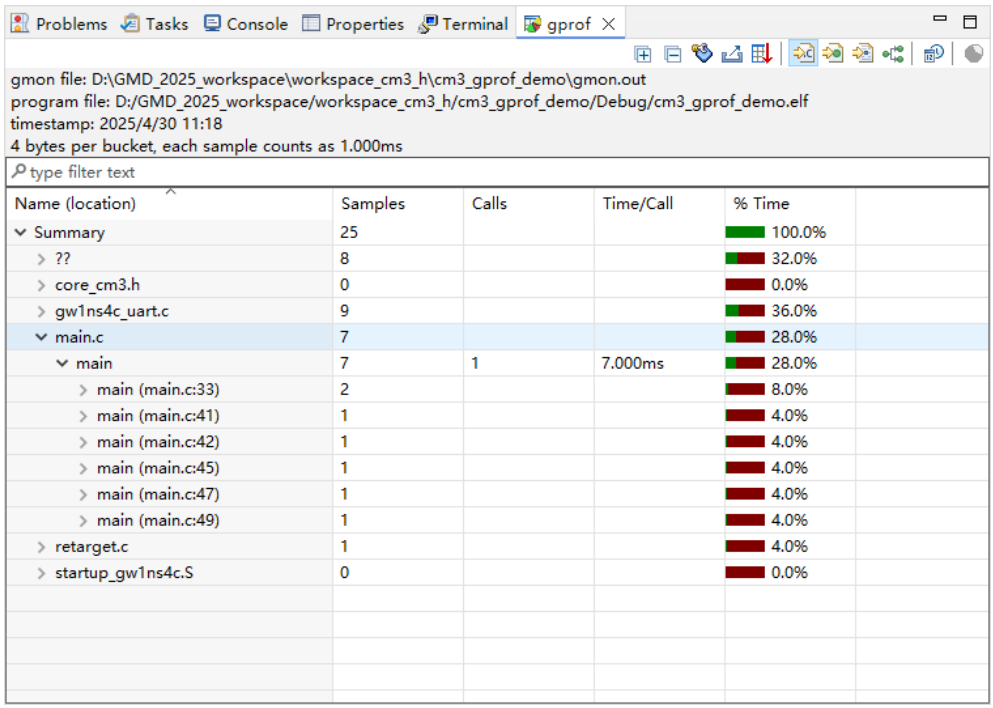
6. 双击 gmon.out 文件，打开 Gcov 工具，如图 13-9 所示。

图 13-9 打开 gmon.out 文件



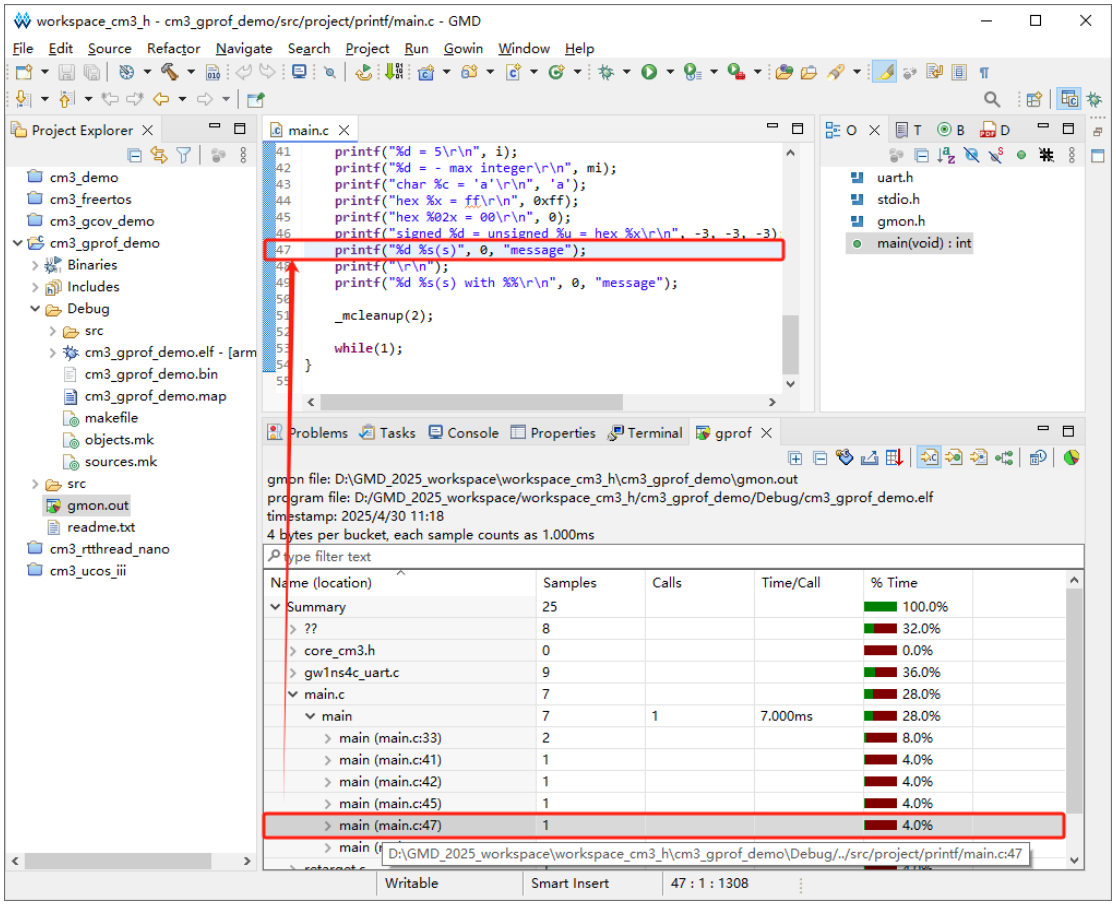
7. Gprof 工具启动，就可以看到对应用程序的分析结果，显示了文件、方法的调用关系等，如图 13-10 所示。

图 13-10 Gprof



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。

图 13-11 查看 Profiling 信息



Gprof 工具相关功能如表 13-1 所示。

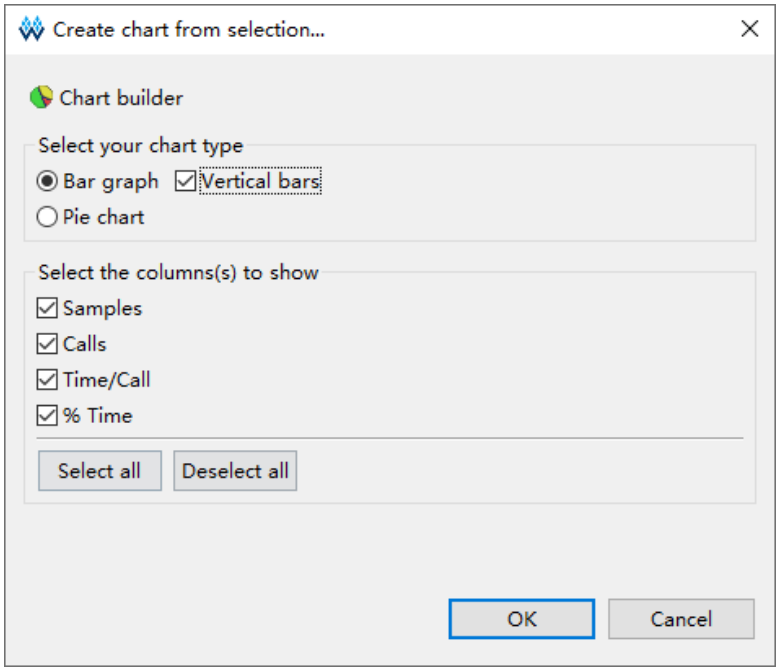
表 13-1 Gprof 功能

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…”（ ），如图 13-12 所示。

图 13-12 Create chat…



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

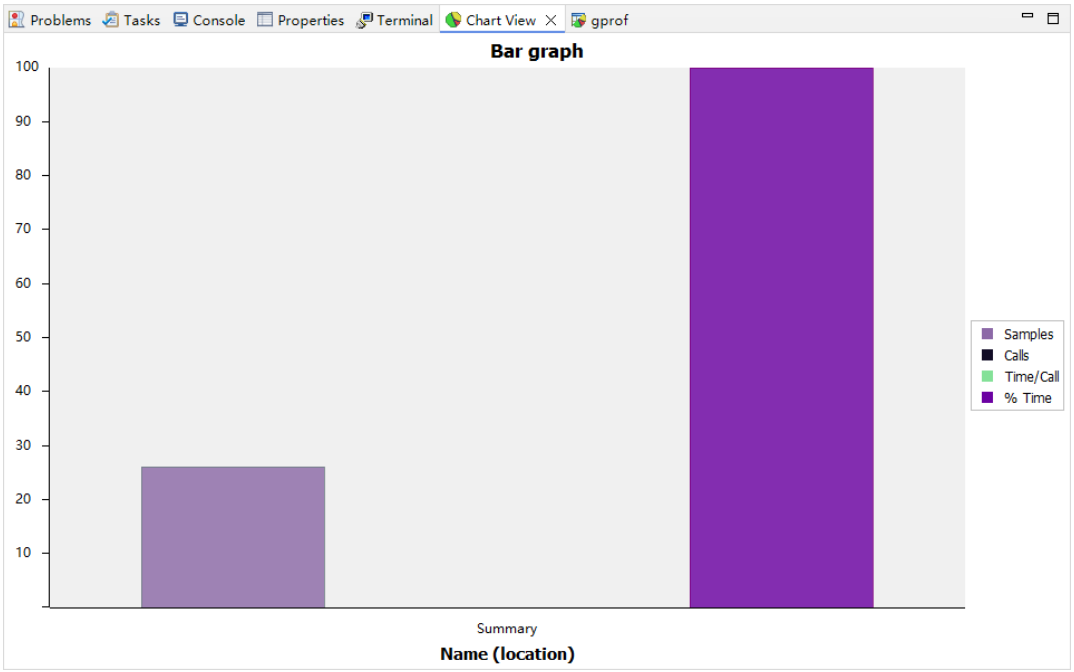
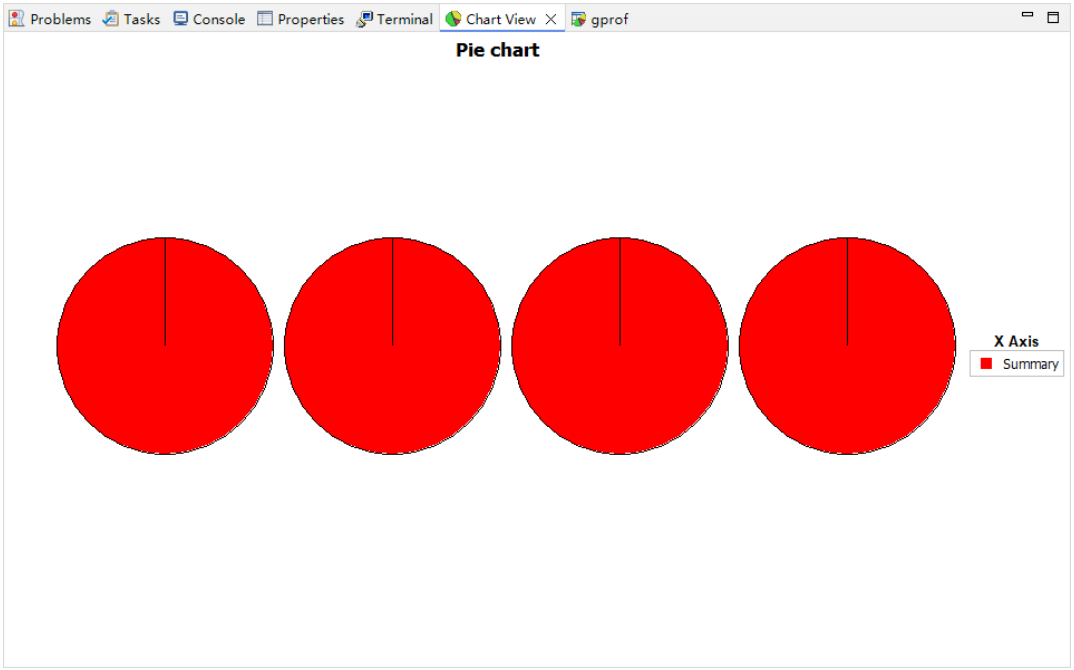


图 13-14 饼状图




GOWIN高云
智慧逻辑 定制未来