




# Gowin PicoRV32 IDE 软件 参考手册

IPUG910-1.5,2023-08-18

版权所有 © 2023 广东高云半导体科技股份有限公司

**GOWIN高云**、、Gowin 以及高云均为广东高云半导体科技股份有限公司注册商标，本手册中提到的其他任何商标，其所有权利属其所有者所有。未经本公司书面许可，任何单位和个人都不得擅自摘抄、复制、翻译本档内容的部分或全部，并不得以任何形式传播。

## 免责声明

本档并未授予任何知识产权的许可，并未以明示或暗示，或以禁止发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档进行适时的更新。

## 版本信息

日期	版本	说明
2020/01/16	1.0	初始版本。
2020/03/06	1.1	<ul style="list-style-type: none"><li>● MCU 支持 Wishbone 总线接口的外部设备 GPIO;</li><li>● MCU 支持扩展 AHB 总线接口;</li><li>● MCU 支持片外 SPI-Flash 下载及运行;</li><li>● MCU 支持外部设备 SPI-Flash 读、写和擦除功能;</li><li>● MCU 支持 Hardware Stack Protection 和 Trap Stack Overflow 功能。</li></ul>
2020/06/01	1.2	<ul style="list-style-type: none"><li>● 支持 MCU 软件在线调试功能;</li><li>● 增强 MCU 内核中断处理功能;</li><li>● 优化 MCU 内核指令。</li></ul>
2021/07/16	1.3	更新 MCU 软件参考设计。
2022/01/29	1.4	<ul style="list-style-type: none"><li>● 更新 MCU 软件参考设计;</li><li>● 优化 IDE 软件选项配置;</li><li>● 完善 IDE 软件在线调试流程。</li></ul>
2023/08/18	1.5	支持 Arora V FPGA 产品。

# 目录

目录 .....	i
图目录 .....	ii
表目录 .....	iii
<b>1 GMD 软件安装与配置 .....</b>	<b>1</b>
<b>2 软件工程模板 .....</b>	<b>2</b>
2.1 创建软件工程 .....	2
2.1.1 创建软件工程 .....	2
2.1.2 选择平台类型 .....	3
2.1.3 选择编译工具链和路径 .....	3
2.1.4 建立项目工程 .....	4
2.2 配置软件工程 .....	5
2.2.1 配置 Target Processor .....	5
2.2.2 配置 Optimization .....	8
2.2.3 配置 GNU RISC-V Cross C Compiler > Includes .....	10
2.2.4 配置 GNU RISC-V Cross C Linker .....	11
2.2.5 配置 GNU RISC-V Cross Create Flash Image .....	13
2.3 编译软件工程 .....	14
2.4 下载 .....	14
2.5 软件在线调试 .....	14
2.5.1 配置软件调试等级 .....	15
2.5.2 配置调试 Flash 链接器 .....	16
2.5.3 配置软件调试选项 .....	16
2.5.4 连接调试仿真器 .....	19
2.5.5 启动软件在线调试 .....	19
<b>3 参考设计 .....</b>	<b>20</b>

# 图目录

图 2-1 创建软件工程 .....	2
图 2-2 选择平台类型 .....	3
图 2-3 选择编译工具链和路径 .....	4
图 2-4 配置 Target Processor .....	6
图 2-5 配置 Optimization .....	8
图 2-6 配置 GNU RISC-V Cross C Compiler > Includes .....	11
图 2-7 配置 GNU RISC-V Cross C Linker .....	12
图 2-8 配置 GNU RISC-V Cross Create Flash Image .....	13
图 2-9 编译软件工程 .....	14
图 2-10 配置 Debugging .....	15
图 2-11 建立软件调试配置选项 .....	16
图 2-12 配置 Main 选项卡 .....	17
图 2-13 配置 Debugger 选项卡 .....	17
图 2-14 配置 Startup 选项卡 .....	18

# 表目录

表 2-1 32 位 RISC-V 架构处理器数据类型宽度 ..... 7

# 1 GMD 软件安装与配置

GMD 软件支持 Gowin\_PicoRV32 的软件编程设计、编译、下载和调试。

高云®半导体官网提供 GMD 软件安装包下载：

<http://www.gowinsemi.com.cn/prodshow.aspx>。

GMD 软件安装与配置，及 Gowin\_PicoRV32 的调试仿真器驱动软件（Olimex Debugger Driver）安装与配置，请参考 [SUG549, GOWIN MCU Designer 用户指南](#)。


注！

建议使用 GMD V1.2。

# 2 软件工程模板

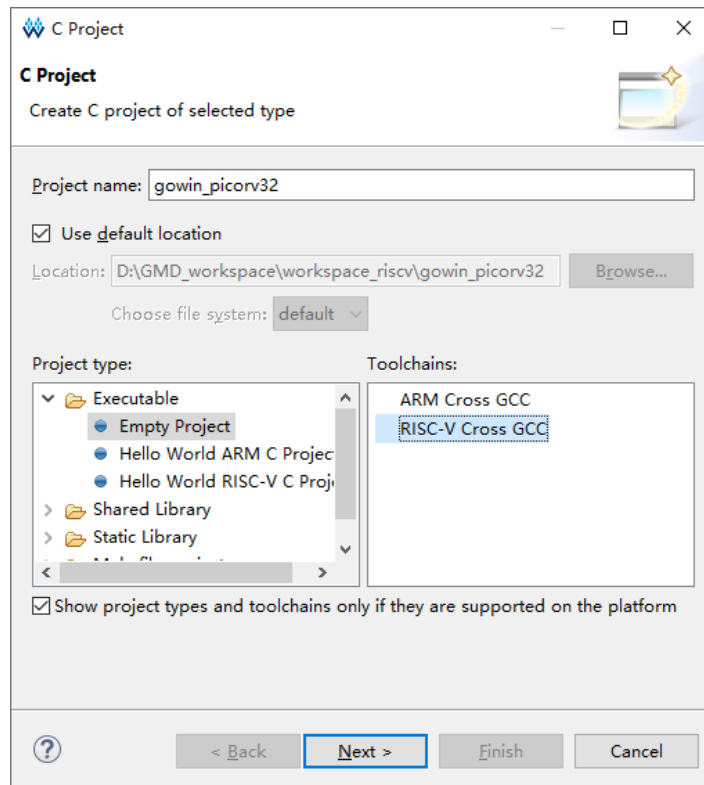
## 2.1 创建软件工程

### 2.1.1 创建软件工程

选择工具栏“New”（）或菜单栏“File > New > C Project”，如图 2-1 所示。

1. 建立项目名称和项目位置；
2. 选择项目类型“Empty Project”；
3. 选择编译工具链“RISC-V Cross GCC”；
4. 单击“Next”。

图 2-1 创建软件工程

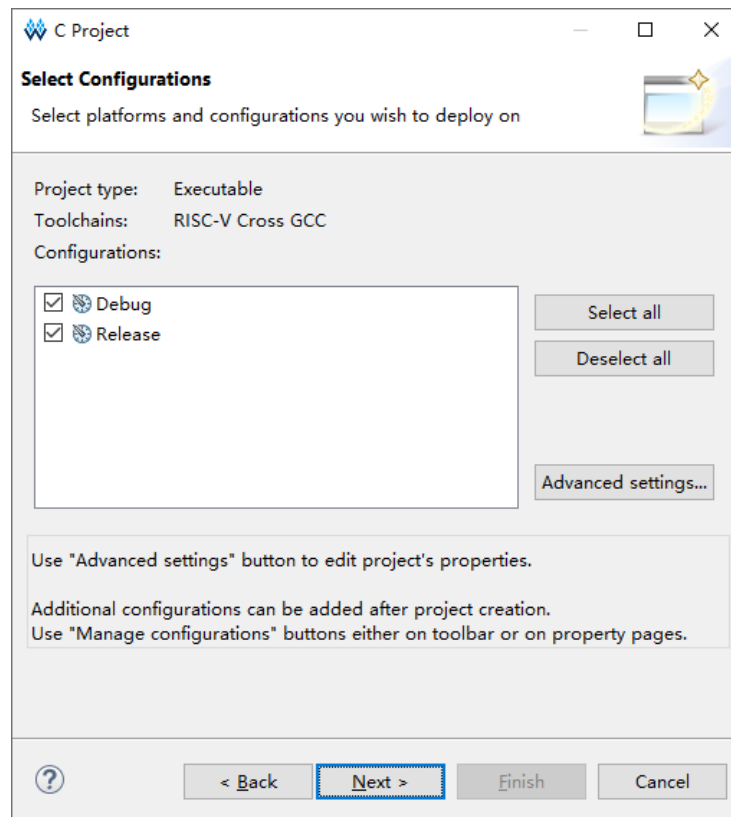




## 2.1.2 选择平台类型

选择平台类型“Debug”和“Release”，单击“Next”，如图 2-2 所示。

图 2-2 选择平台类型

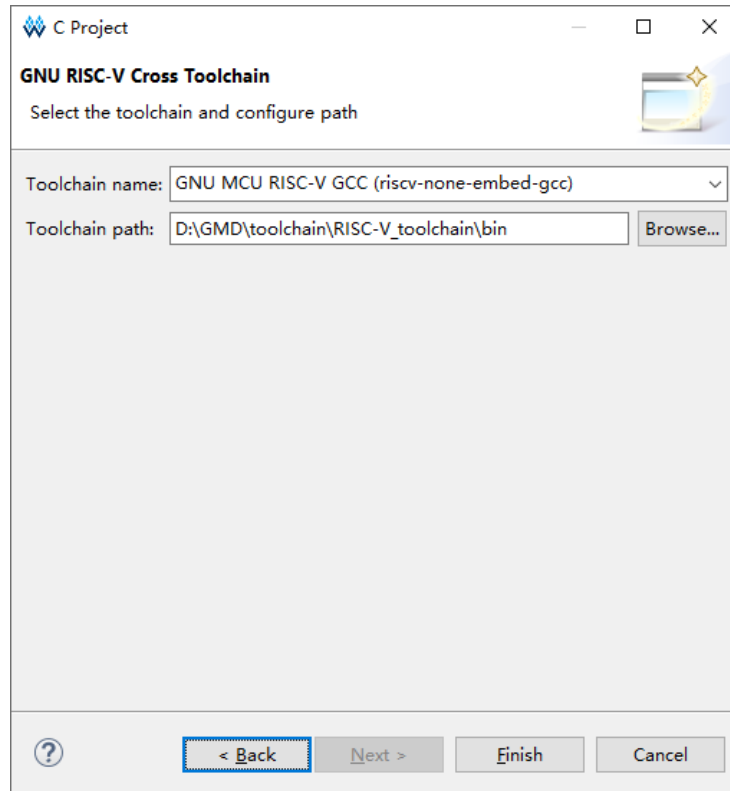


## 2.1.3 选择编译工具链和路径

选择编译工具链 `riscv-none-embed-gcc` 及其所在路径，建议默认配置 Toolchain name 和 Toolchain path，单击“Finish”，如图 2-3 所示。

如果需要使用自定义的 RISC-V 编译工具链，则请修改指定编译工具链及其路径。

图 2-3 选择编译工具链和路径



### 2.1.4 建立项目工程

完成软件工程创建后,在 **Project Explorer** 视图选择新建的项目工程,添加工程结构和代码,引入软件编程设计。

以软件开发工具包参考设计为例,软件编程设计项目工程及代码定义,如下所示:

- **bsp:** 外部设备驱动函数定义
  - simpleuart.h  
simpleuart.c: 外部设备 Simple UART 驱动函数定义
  - wbgpio.h  
wbgpio.c: 外部设备 GPIO 驱动函数定义
  - wbi2c.h  
wbi2c.c: 外部设备 I2C 驱动函数定义
  - wbspi.h  
wbspi.c: 外部设备 SPI 驱动函数定义
  - wbspiflash.h  
wbspiflash.c: 外部设备 SPI-Flash 驱动函数定义
  - wbuart.h  
wbuart.c: 外部设备 UART 驱动函数定义
- **lib:** 系统信息定义
  - firmware.h  
firmware.c: 系统信息定义

- printf.c: 重定义 printf 函数
- config.h: 用户配置启动和运行方式
- custom\_ops.S: 用户指令定义
- irq.h
  - irq.c: 中断处理程序
- loader.c: 启动方式定义
- main.c: 用户主函数
- picorv32.h: 内核寄存器、地址映射定义
- start.S: 启动引导程序定义
- sections\_debug.ld: 用于调试的链接
- sections\_xip.ld: 用于外部 SPI-Flash 运行程序指令的链接
- sections.ld: 用于内部 ITCM 运行程序指令的链接

软件编程设计项目工程如有代码更新，请在视图“Project Explorer”选定当前软件工程，右击选择“Refresh”，更新软件编程设计项目工程。

## 2.2 配置软件工程

在视图“Project Explorer”中，选择当前软件工程，右击选择“Properties > C/C++ Build > Settings > Tool Settings”选项，配置当前软件工程的参数选项。

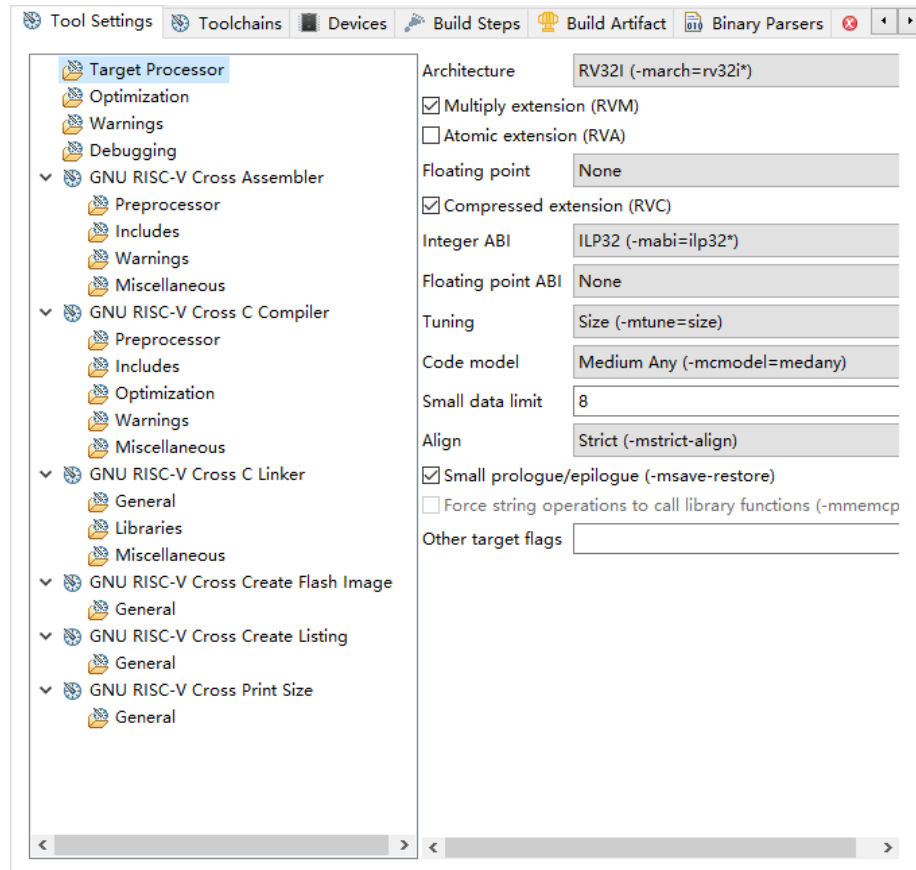
Gowin\_PicoRV32 必须配置以下参数选项：

- Target Processor
- Optimization
- GNU RISC-V Cross C Compiler
  - Includes
- GNU RISC-V Cross C Linker
  - General
- GNU RISC-V Cross Create Flash Image
  - General

### 2.2.1 配置 Target Processor

配置“Target Processor”选项，该选项配置，如图 2-4 所示。

图 2-4 配置 Target Processor



- **Architecture**  
请选择 RV32I(-march=rv32i\*), Gowin\_PicoRV32 仅支持 RISC-V32 位整型指令, 所以选择 RV32I 选项。
- **Multiply extension (RVM)**  
如果需要使用 RV32M 扩展, 请开启 Multiply extension(RVM)选项。同时在 Gowin\_PicoRV32 硬件设计 IP Core Generator 工具的 Gowin PicoRV32 CORE 参数配置中, 开启 Support RV32M Extends 选项, 否则编译包含 RISC-V 快速乘法指令时,会导致 Gowin\_PicoRV32 运行错误。
- **Atomic extension (RVA)**  
Gowin\_PicoRV32 不支持原子指令扩展, 请关闭该选项。
- **Floating point**  
Gowin\_PicoRV32 不支持浮点指令, 该选项请选择 None。
- **Compressed extension (RVC)**  
如果需要使用 RV32C 扩展, 请开启 Compressed extension (RVC)选项。同时在 Gowin\_PicoRV32 硬件设计 IP Core Generator 工具的 Gowin PicoRV32 CORE 参数配置中, 开启 Support RV32C Extends 选项, 否则编译包含 RISC-V 16 位压缩指令时,会导致 Gowin\_PicoRV32 运行错误。
- **Integer ABI**  
设置 RISC-V 目标平台所支持的 ABI 函数调用规则。Gowin\_PicoRV32

为 32 位 RISC-V 架构处理器平台，且不支持硬件浮点指令，因此该选项请选择 ILP32 (-mabi=ilp32\*)。32 位 RISC-V 架构处理器数据类型宽度，如表 2-1 所示。

**表 2-1 32 位 RISC-V 架构处理器数据类型宽度**

C 语言数据类型	32 位 RISC-V 架构数据类型宽度（单位：字节）
char	1
short	2
int	4
long	4
long long	8
void *	4
float	4
double	8
long double	16

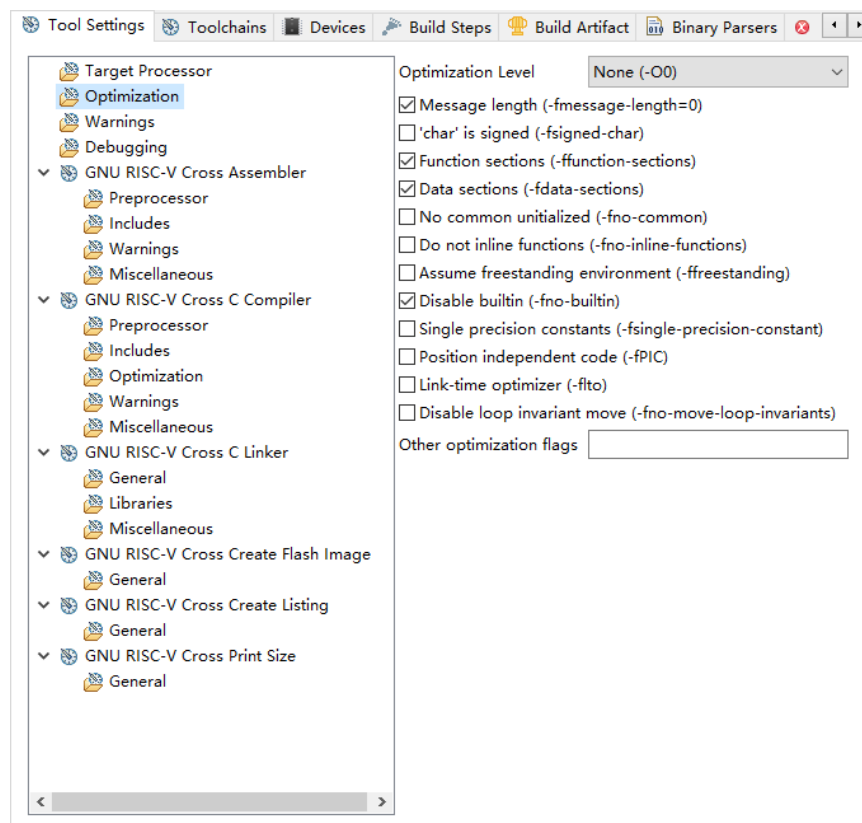
- **Tuning**  
指定编译工具链 GCC 为其调整代码性能的目标处理器名称。Gowin\_PicoRV32 不支持该选项，请选择 Size(-mtune=size)。
- **Code model**  
设置参数-mcmmodel，该参数指定程序的寻址范围，选择 Medium Any (-mcmmodel=medany)。(mcmmodel=medany)选项用于指定该程序的寻址范围可以在任意的 4GB 空间内，寻址空间不预定，应用相对灵活。
- **Small data limit**  
设置参数-msmall-data-limit，该参数指定可以放置到小数据区域的全局变量和静态变量的最大 Size (以字节为单位)。该参数设置为 8。
- **Align**  
设置是否避免产生非对齐内存访问的操作。Gowin\_PicoRV32 不支持快速非对齐访问，建议选择 Strict(-mstrict-align)。
- **Small prologue/epilogue(-msave-restore)**  
如果开启该选项，则设置为调用 Size 最小、但速度更慢的起始和返回代码的库函数。默认使用快速的内联代码。
- **Other target flags** 选项中可以手动添加如下配置：
  - **Allow use of PLTs(-mplt)**  
如果配置该选项，则使用 PLT 生成中断控制代码。
  - **Integer divide instructions(-mdiv)**  
如果配置该选项，则使用整型除法硬件指令，需要处理器支持 RV32M 指令集扩展。Gowin\_PicoRV32 支持 RV32M 指令集扩展，所以可以配置 Multiply extension(RVM)选项，同时在 Gowin\_PicoRV32 硬件设计 IP Core Generator 工具的 Gowin PicoRV32 CORE 参数设置中，开启 Support RV32M Extends 选项。

- **-mpreferred-stack-boundary=num**  
栈边界对齐为  $2num$  字节边界对齐。如果未指定，则默认值为 **24**（即 **16 字节或 128 位**）对齐。如果配置该选项，则在构建所有模块时都需要使用该选项（包括库，系统库和起始模块）。
- **-mexplicit-relocs / -mno-explicit-relocs**  
处理符号地址时，使用或不使用汇编程序重定向操作符。另一种配置是使用汇编宏，这可能会限制优化。
- **-mrelax**
- **-mno-relax**  
使用链接器松弛来减少实现符号地址所需的指令数量。默认是使用链接器的松弛。
- **-memit-attribute / -mno-emit-attribute**  
输出或不输出 RISC-V 属性信息到 ELF 对象中。该特性需要 binutils 2.32 版本。
- **-malign-data=type**  
控制 GCC 如何对齐数组、结构体、联合体等类型的变量和常量。支持的类型值为“xlen”和“natural”。其中，“xlen”使用寄存器宽度作为对齐值，“natural”使用自然对齐，默认值为“xlen”。

## 2.2.2 配置 Optimization

配置“Optimization”选项，该选项配置，如图 2-5 所示。

图 2-5 配置 Optimization



- **Optimization Level**  
通过-O 等级来设置优化等级，优化编译 Size、运行速度和编译时间等。可选择的包括-O0、-O1、-O2、-O3、-Os、-Og，-O0/-O1/-O2/-O3 优化等级逐级提高。
  - -O0: 不做优化；
  - -Os: 在-O2 的基础上，关闭导致编译 Size 增大的选项，实现对编译 Size 的最优化；
  - -Og: 在-O0 的基础上，增加一些适用于调试的编译选项，Gowin \_PicoRV32 暂不支持片上调试，因此建议采用其他优化策略。
- **Message length (-fmessage-length=n)**  
将错误信息以每行 n 个字符的形式显示在控制台窗口中。如果设置为 0，则关闭换行功能，一条错误信息显示为一行。默认为-fmessage-length=0。建议开启该选项。
- **'char' is signed (-fsigned-char)**  
设置 char 类型数据为有符号数。
- **Function sections (-ffunction-sections) / Data sections (-fdata-sections)**
  - 若目标支持任意分段，则令每个函数或数据项在输出文件中开辟单独的段，使用函数名或数据项的名称作为输出段的名称。
  - 若链接器可以执行改进指令空间中引用的局部性的优化，则可以使能该选项。
  - 若与链接器垃圾收集（链接器 `--gc-sections` 选项）一起使用，则最后生成执行文件时自动删除没有使用的函数段和数据项段，产生更小编译 Size。

**注！**

只有在能产生显著效果时才开启该选项，并配合在“GNU RISC-V Cross C Linker > General”选项中开启“Remove unused sections (-Xlinker --gc-sections)”选项，以减少编译 Size。

- **No common uninitialized (-fno-common)**  
fno-common 选项指定编译器将未初始化的全局变量放到目标文件的 BSS 段中。这将阻止链接器合并暂定定义，因此如果在多个编译单元中定义了相同的变量，则会出现多定义错误。建议关闭该选项。
- **Do not inline functions (-fno-inline-functions)**  
如果开启该选项，则除了使用 `always_inline` 属性标记的函数之外，不展开任何内联函数。关闭优化时，此为默认设置。也可使用 `noinline` 属性标记单个函数，以避免该函数的内联。
- **Assume freestanding environment (-ffreestanding)**  
断言编译目标时的一个独立环境，独立环境中可能不存在标准库，程序启动可能不是主函数。显著案例是操作系统内核。相当于-fno-hosted。
- **Disable builtin (-fno-builtin)**
  - 不识别不以“\_builtin\_”作为前缀的内置函数。受影响的函数，包括使

用 `-ansi` 或 `-std` 选项(用于严格的 ISO C 一致性)时不是内置函数的函数, 因为没有 ISO 标准含义。

- GCC 通常生成特殊代码来更有效地处理某些内置函数, 例如, 对 `alloca` 的调用可能变成直接调整堆栈的单个指令, 对 `memcpy` 的调用可能变成内联复制循环。生成的代码通常更小、更快, 但是由于函数调用不再以这样的方式出现, 不能在這些调用上设置断点, 也不能通过连接不同的库来更改函数的行为。
- 此外, 当一个函数被识别为内置函数时, GCC 可能会使用关于该函数的信息来警告调用该函数的问题, 或者生成更有效的代码, 即使生成的代码仍然包含对该函数的调用。例如, 当内建 `printf` 并且已知 `strlen` 不修改全局内存时, 会用 `-Wformat` 给出对 `printf` 的错误调用的警告。
- **Single precision constants (-fsingle-precision-constant)**  
将浮点常量作为单精度数据处理。
- **Position independent code (-fPIC)**  
如果目标机器支持, 则生成适合在共享库中使用的与位置无关的代码 (PIC)。Gowin\_PicoRV32 不支持 PIC, 建议关闭该选项。
  - **Link-time optimizer (-flto)**  
该选项运行标准的连接时间优化器。
  - 使用源代码调用时, 生成 GIMPLE (GCC 的内部表示之一)并将其写入对象文件中的特殊 ELF 节。当对象文件连接在一起时, 所有的函数体都从 ELF 部分读取并实例化, 如同是同一翻译单元的一部分。
  - 使用连接时优化器, 应在编译时和最终连接期间指定 `-flto` 和优化选项, 建议使用相同的选项编译参与相同连接的所有文件, 并在连接时指定这些选项。
- **Disable loop invariant move (-fno-move-loop-invariants)**  
选择是否取消 RTL 循环优化器中的循环不变动作传递, 若优化等级设置为 `-O1` 或更高等级 (`-Og` 除外), 自动开启 RTL 循环优化器中的循环不变动作传递。

### 2.2.3 配置 GNU RISC-V Cross C Compiler > Includes

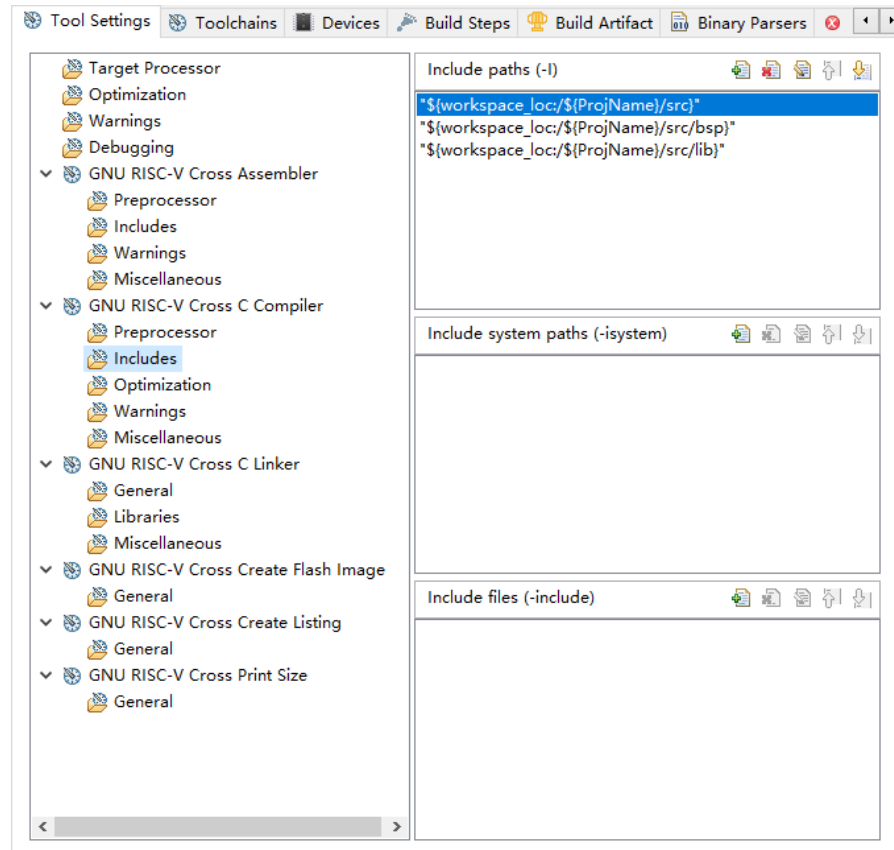
配置“GNU RISC-V Cross C Compiler > Includes > Include paths (-I)”选项, 该选项配置 C 头文件引用路径, 如图 2-6 所示。

以软件开发工具包参考设计为例, C 头文件引用路径配置, 如下所示:

- "\${workspace\_loc:\${ProjName}/src}"
- "\${workspace\_loc:\${ProjName}/src/bsp}"
- "\${workspace\_loc:\${ProjName}/src/lib}"



图 2-6 配置 GNU RISC-V Cross C Compiler &gt; Includes



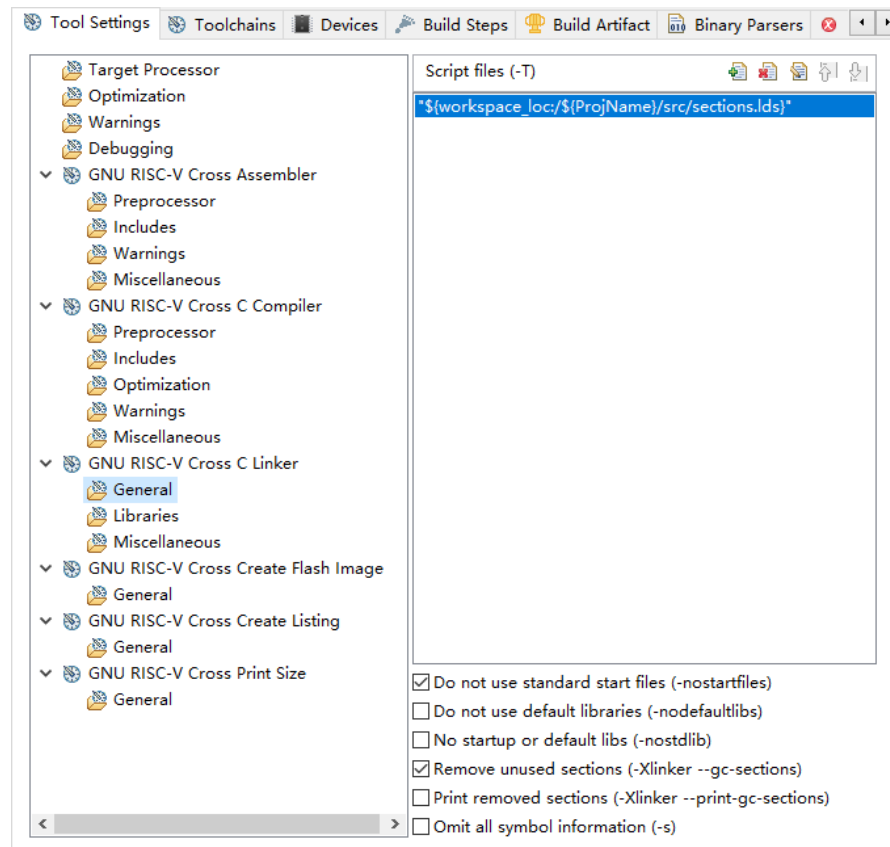
## 2.2.4 配置 GNU RISC-V Cross C Linker

配置“GNU RISC-V Cross C Linker > General > Script files (-T)”选项，该选项配置“sections.lds”、“sections\_xip.lds”或“sections\_debug.lds”作为 Gowin\_PicoRV32 Flash 链接器，如图 2-7 所示。

参照 Gowin\_PicoRV32 硬件设计 IPCore Generator 工具中“ITCM > Boot Mode”选项：

- 如果 Boot Mode 选择“MCU boot from external Flash and run in ITCM”或“MCU boot and run in ITCM”选项，则选择 sections.lds 作为 Flash 链接器，例如“\${workspace\_loc}/\${ProjName}/src/sections.lds”
- 如果 Boot Mode 选择“MCU boot and run in external Flash”选项，则选择 sections\_xip.lds 作为 Flash 链接器，例如“\${workspace\_loc}/\${ProjName}/src/sections\_xip.lds”
- 如果 Boot Mode 选择“MCU boot from external Flash and run in ITCM”或“MCU boot and run in ITCM”选项，并且执行软件在线调试，则选择 sections\_debug.lds 作为 Flash 链接器，例如“\${workspace\_loc}/\${ProjName}/src/sections\_debug.lds”

图 2-7 配置 GNU RISC-V Cross C Linker



- **Do not use standard start files (-nostartfiles)**  
设置连接时不使用标准启动文件。Gowin\_PicoRV32 必须使用自定义的起始文件，因此必须开启该选项。
- **Do not use default libraries (-nodefaultlibs)**  
设置连接时不使用标准系统库，只有被选择的库传递给链接器。指定系统库连接的选项(如-static-libgcc 或-shared-libgcc)将被忽略。正常使用标准启动文件，除非使用-nostartfiles。编译器可能会生成对 memcmp、memset、memcpy 和 memmove 的调用。
- **No startup or default libs (-nostdlib)**
  - 连接时不使用标准的系统启动文件或库。
  - 没有任何启动文件，只有用户指定的库被传递给链接器，并且指定系统库连接的选项(如-static-libgcc 或-shared-libgcc)将被忽略。
  - 请关闭该选项。
- **Remove unused sections (-Xlinker --gc-sections)**
  - 该选项删除没有调用的段。
  - 该选项配合编译器优化设置的-ffunction-sections 和-fdata-sections 选项，可在连接时删除未调用的函数和变量，进一步减小编译 Size。
- **Print removed sections (-Xlinker --print-gc-sections)**  
当开启 Remove unused sections (-Xlinker --gc-sections)时，可以同步开

启该选项，编译时输出被优化删除的段的名称，标记被优化删除的段。

- **Omit all symbol informations (-s)**

从可执行文件中删除所有符号表和重定位信息。

注！

默认请开启 “Do not use standard start files (-nostartfiles)” 和 “Remove unused sections (-Xlinker --gc-sections)” 两个选项。

## 2.2.5 配置 GNU RISC-V Cross Create Flash Image

配置 “GNU RISC-V Cross Create Flash Image > General” 选项，该选项配置，如图 2-8 所示。

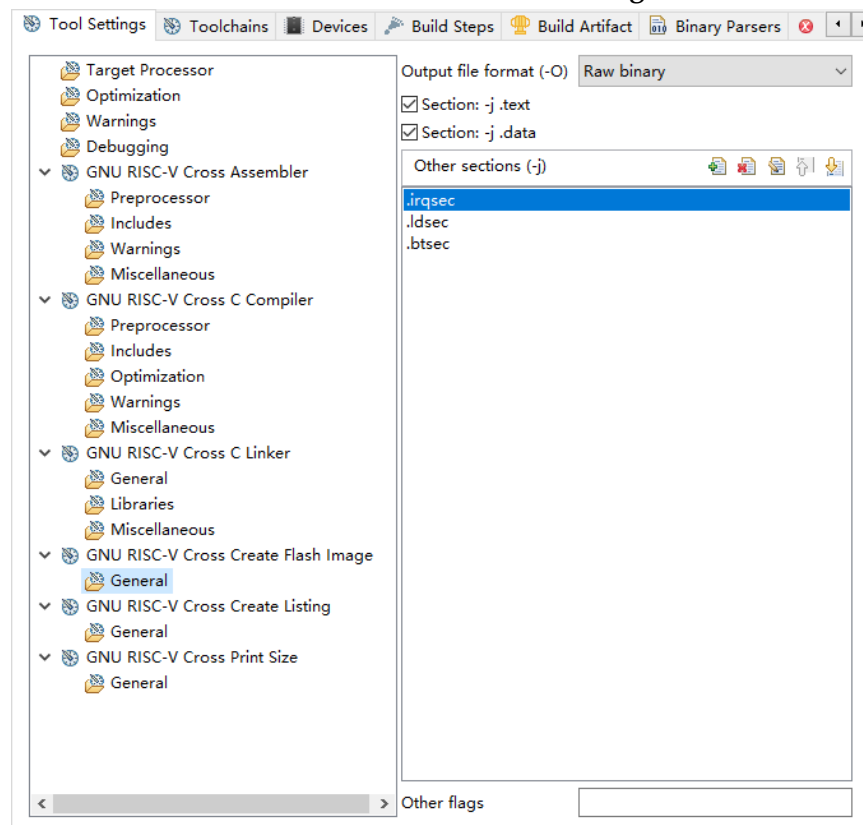
以软件开发工具包参考设计为例，该选项配置，如下所示：

- **Output file format (-O)**选项，选择输出文件格式为 **Raw binary**
- 请开启 **Section: -j .text** 选项
- 请开启 **Section: -j .data** 选项

如果在软件工程中有自定义的段，并将函数或变量映射到自定义的段中，则需在 **Other sections (-j)**中添加这些自定义的段，如 **Gowin\_PicoRV32** 自定义以下段：

- **.irqsec**
- **.ldsec**
- **.btsec**

图 2-8 配置 GNU RISC-V Cross Create Flash Image



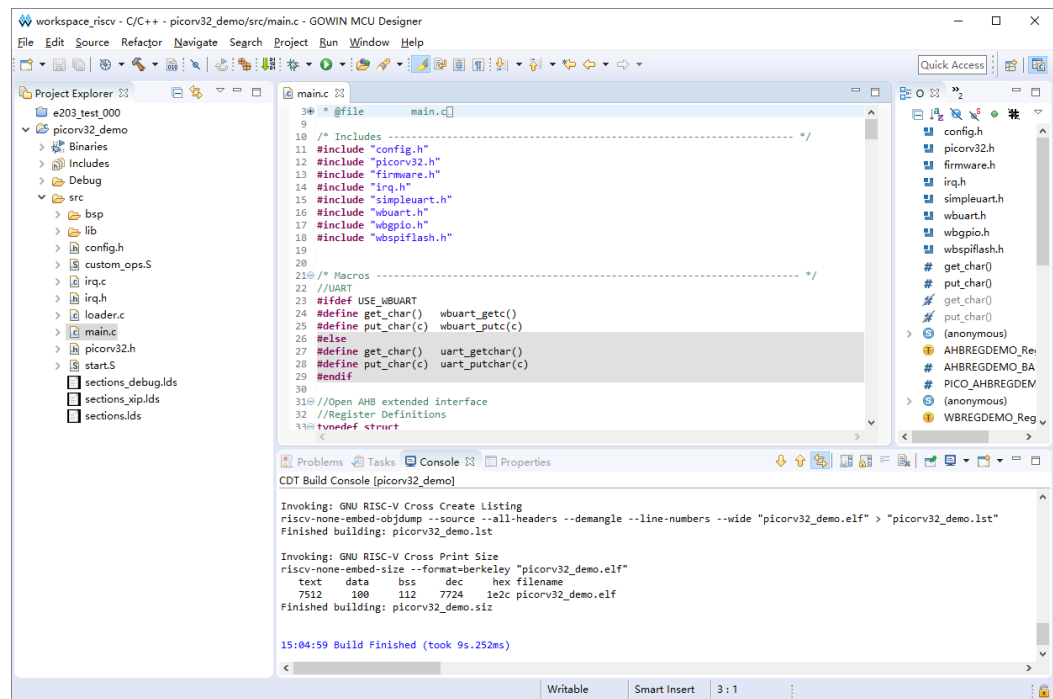
## 2.3 编译软件工程

参照 Gowin\_PicoRV32 硬件设计 IP Core Generator 工具中“ITCM > Boot Mode”配置方式，定义 config.h 启动参数宏定义 BUILD\_MODE:

- MCU boot and run in ITCM : #define BUILD\_MODE BUILD\_LOAD
- MCU boot from external Flash and run in ITCM: #define BUILD\_MODE BUILD\_BURN
- MCU boot and run in external Flash: #define BUILD\_MODE BUILD\_XIP

完成软件工程选项配置、用户配置和代码编写后，编译软件工程，单击工具栏“Build”(🔧) 或“Build All”(🏠)，或菜单栏“Project > Build Project”或“Project > Build All”，编译产生软件编程设计 Binary 文件，如图 2-9 所示。

图 2-9 编译软件工程



## 2.4 下载

完成 Gowin\_PicoRV32 软件编程设计编译后，软件编程设计下载方法，请参考 [IPUG913, Gowin PicoRV32 软件下载参考手册](#)。

## 2.5 软件在线调试

完成 Gowin\_PicoRV32 软件编程设计 Binary 文件下载后，如果用户软件应用设计出现问题，可以连接开发板与 Olimex 调试仿真器（或 RiscV 指令集架构处理器所支持的其他调试仿真器），在线调试当前软件设计。

Gowin\_PicoRV32 软件调试流程，包括：

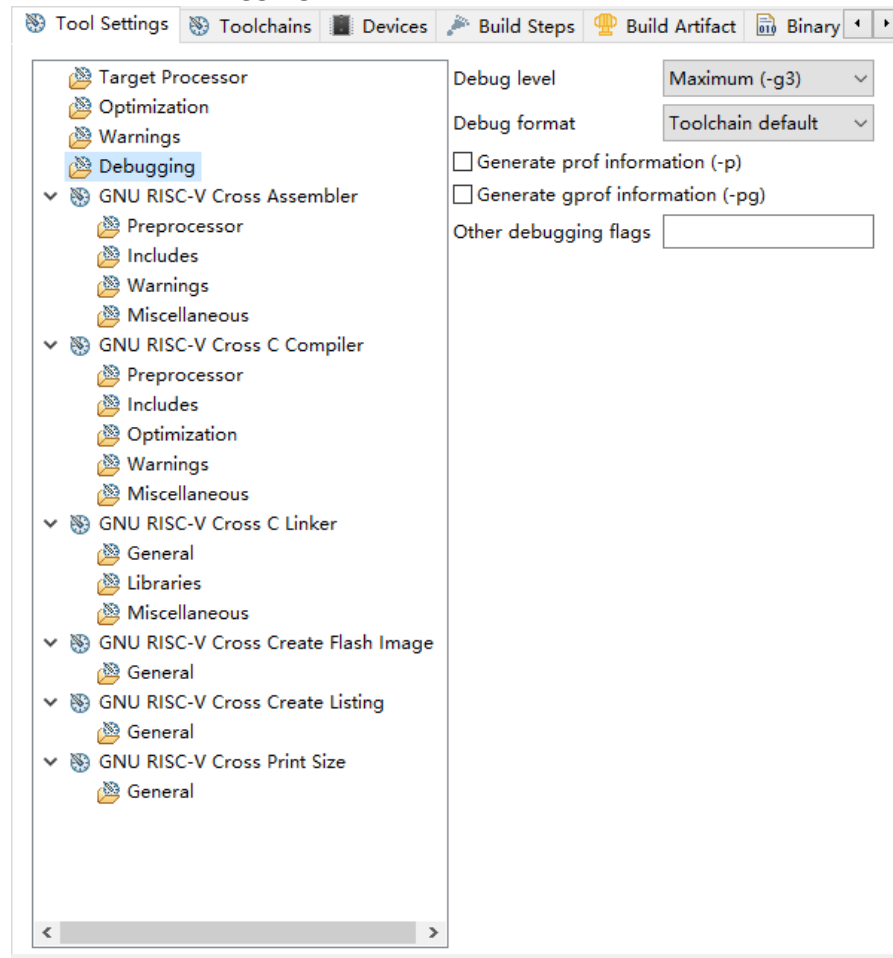
- 配置软件调试等级

- 配置调试 Flash 链接器
- 配置软件调试选项
- 连接调试仿真器
- 启动软件在线调试

### 2.5.1 配置软件调试等级

在 Project Explorer 视图中，选择当前工程，右击选择“Properties > C/C++ Build > Settings > Tool Settings > Debugging > Debug level”选项，该选项配置，如图 2-10 所示。

图 2-10 配置 Debugging



Debug level 配置，包括以下选项：

- None: 无调试等级配置
- Minimal (-g1): 最小调试等级配置
- Default (-g): 默认调试等级配置
- Maximum (-g3): 最大调试等级配置

## 2.5.2 配置调试 Flash 链接器

在线调试 Gowin\_PicoRV32 软件编程设计时，请指定调试所用的 Flash 链接器“sections\_debug.lds”，Flash 链接器配置方法请参考 [2.2.4 配置 GNU RISC-V Cross C Linker](#)。

选项“GNU RISC-V Cross C Linker > General > Script files (-T)”，修改为“\${workspace\_loc:\${ProjName}/src/sections\_debug.lds}”。

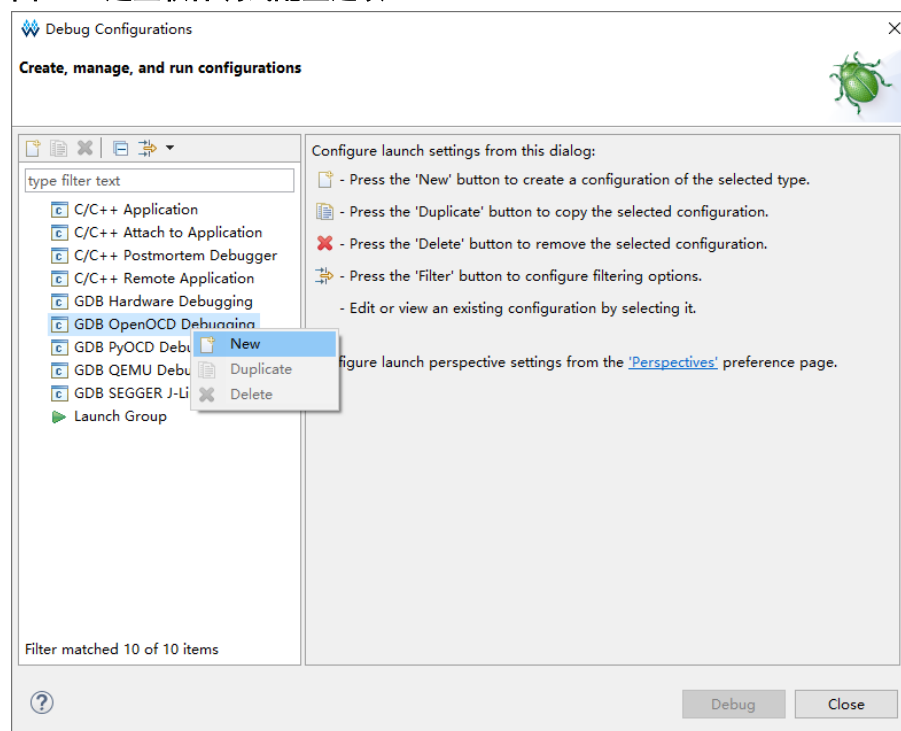
完成配置调试 Flash 链接器后，请参考 [2.3 编译软件工程](#)，重新编译产生软件编程设计 Binary 文件。

完成软件工程编译后，请参考 [2.4 下载](#)，重新下载软件编程设计 Binary 文件。

## 2.5.3 配置软件调试选项

选择菜单栏“Run > Debug Configurations > GDB OpenOCD Debugging”选项，右击选择“New”选项，建立当前工程的软件调试配置选项，如图 2-11 所示。

图 2-11 建立软件调试配置选项

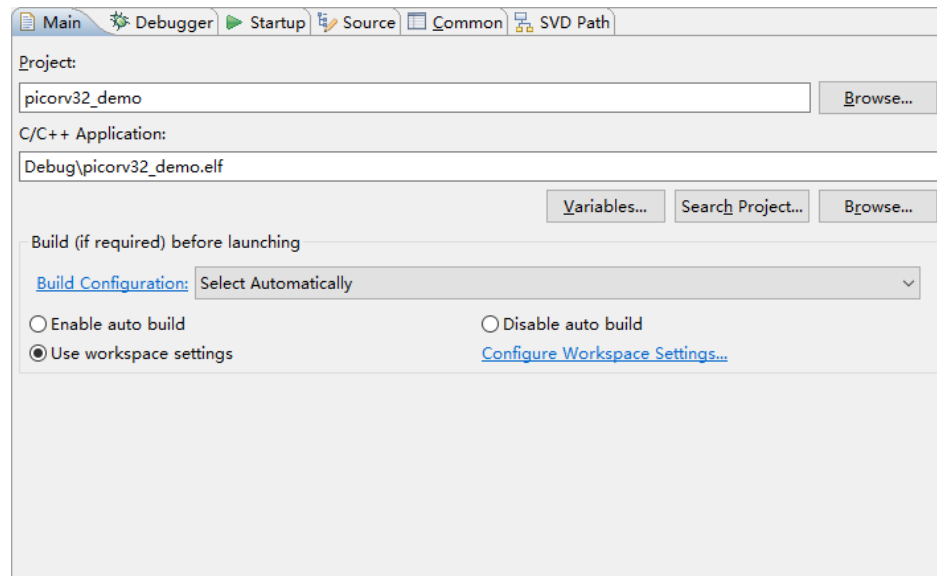


选择已建立的软件调试选项，配置 Main、Debugger、Startup 等调试选项卡。

### 配置 Main 选项卡

选择“Main”选项卡，配置当前工程的 Project 和 C/C++ Application 等选项，如图 2-12 所示。

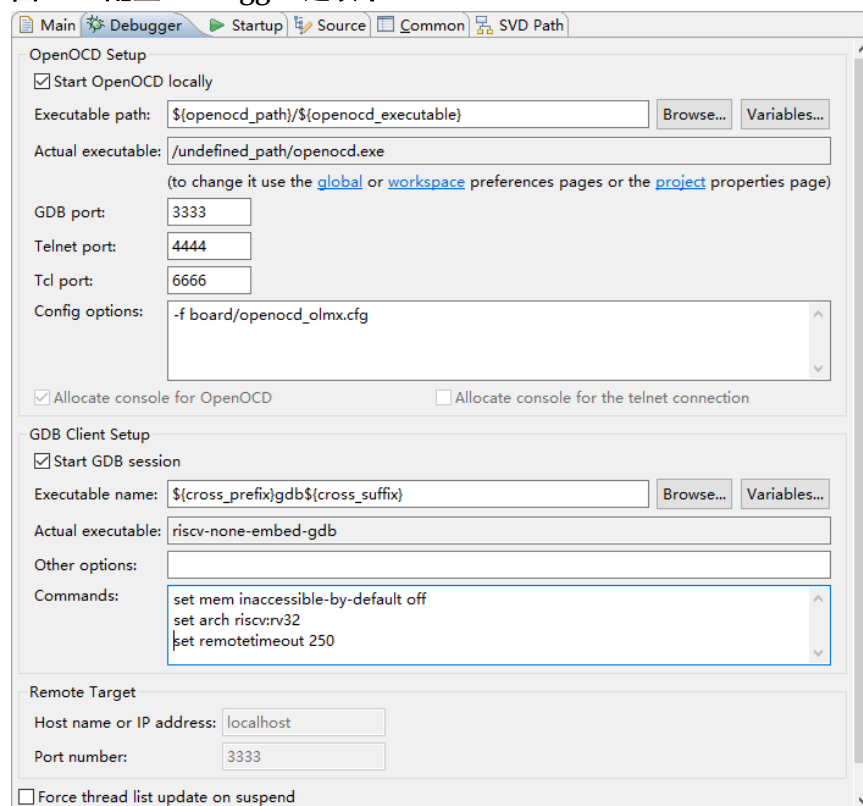
图 2-12 配置 Main 选项卡



### 配置 Debugger 选项卡

选择“Debugger”选项卡，配置 OpenOCD 和 GDB 等选项，如图 2-13 所示。

图 2-13 配置 Debugger 选项卡



- OpenOCD Config options 选项配置  
该选项配置为“-f board/openocd\_olmx.cfg”，指定所用调试仿真器配置

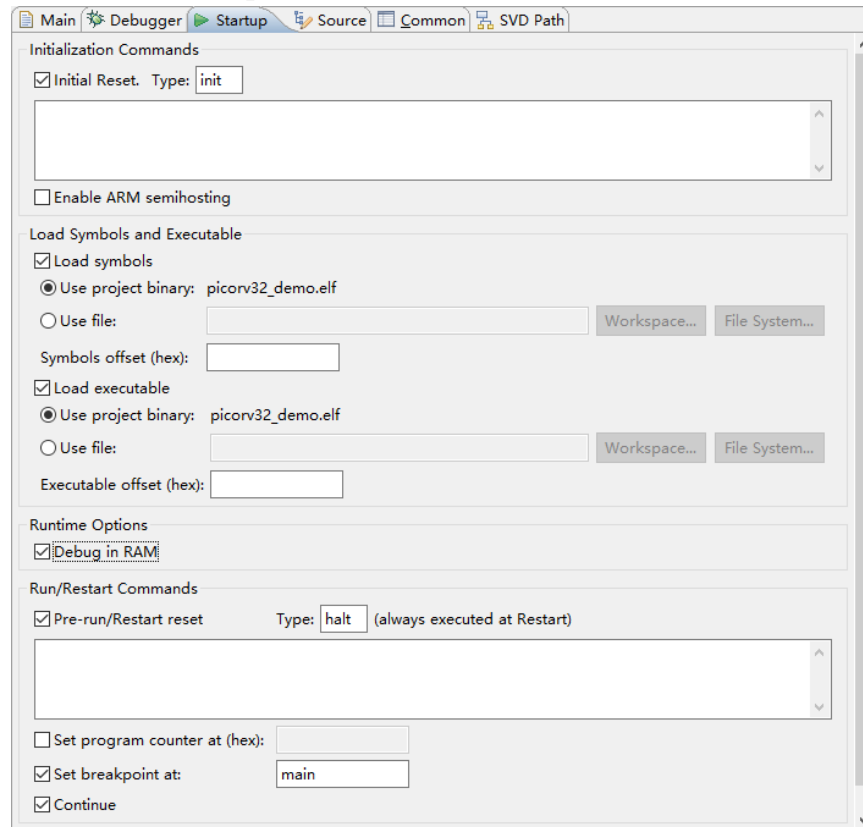
文件，GMD 软件默认使用 Olimex 仿真器。

- GDB Commands 选项配置
  - set mem inaccessible-by-default off
  - set arch riscv:rv32（指定 RiscV 指令集架构）
  - set remotetimeout 250（防止协议超时）

### 配置 Startup 选项卡

选择“Startup”选项卡，配置执行在线调试等选项，如图 2-14 所示。

图 2-14 配置 Startup 选项卡



- Initialization Commands 选项配置
  - 请开启 Initial Reset 选项，Type 类型配置为“init”，执行在线调试时，可以自动执行初始化
  - 请关闭 Enable ARM semihosting 选项，Gowin\_PicoRV32 不支持该功能
- Load Symbols and Executable 选项配置
  - 请开启 Load symbols 选项，选择“Use project binary”
  - 请开启 Load executable 选项，选择“Use project binary”选项
- Runtime Options 选项配置
 

请开启 Debug in RAM 选项，执行在线调试时，读取软件编程设计 Binary 文件，写入指令存储器 ITCM
- Run/Restart Commands 选项配置



- 请开启 Pre-run/Restart reset 选项，Type 类型配置为“halt”，执行在线调试时，可以自动暂停
- 请开启 set breakpoint at 选项，配置为“main”，执行在线调试时，可以自动在 main 主函数的第一条语句位置设置断点
- 请开启 Continue 选项，执行在线调试时，可以自动运行到断点位置，并暂停

## 2.5.4 连接调试仿真器

以 Olimex 调试仿真器为例。


连接开发板与 Olimex 调试仿真器（或 RiscV 指令集架构处理器所支持的其他调试仿真器）。

Olimex 调试仿真器，选用 Olimex arm-usb-tiny-h，链接为：  
<https://www.olimex.com/Products/ARM/JTAG/ARM-USB-TINY-H/>

Olimex 调试仿真器驱动软件安装与配置，请参考 [SUG549, GOWIN MCU Designer 用户指南](#)。

将 Olimex 调试仿真器的 TDI、TDO、TMS、TCK 引脚按照标准 JTAG 接口顺序与开发板连接，VREF 引脚连接 3.3V，4/6/8/.../20 的 GND 引脚只需连接一个即可。

## 2.5.5 启动软件在线调试

单击工具栏 Debug “”，执行软件在线调试，调试视图包括：

- 代码区：可以查看 C 代码和汇编代码，设置断点，运行程序等；
- 变量区：可以查看通用寄存器的值或程序变量的值；
- 汇编指令区：可以查看当前指令存储器 ITCM 中的指令和当前正在执行的指令；
- 控制区：可以控制在线调试的进程，设置断点等。

### 调试中断处理函数

Gowin\_PicoRV32 是面积优化的 RiscV 指令集架构处理器，中断控制系统设计简单易用，不支持硬件中断嵌套和中断优先级设置。

因此，当正在执行在线调试，并且程序处于暂停状态时（执行单步运行、断点运行等操作，使程序运行到某行或某条指令，且暂停在该位置的状态），MCU 无法响应外部中断请求。

如果需要对外部中断处理函数执行在线调试，则需要在中断处理函数中的相应位置设置断点，执行程序进入连续运行状态。

单击工具栏“Resume”，可以使程序进入连续运行状态，如果没有遇到断点，则会保持连续运行状态。在此状态下，当外部设备发出中断请求，MCU 执行中断处理函数，运行到中断处理函数中的断点位置进入暂停状态，此时可以在中断处理函数中执行单步调试、断点调试、变量查看等操作。

# 3 参考设计

Gowin\_PicoRV32 支持 GMD（已测试软件版本：V1.2）软件环境的参考设计，通过链接获取如下[参考设计](#)：

...\ref\_design\MCU\_RefDesign\picorv32\_demo

