

Gowin PicoRV32 IDE 软件参考手册

IPUG910-1.0,2020-01-16

版权所有[©]2020 广东高云半导体科技股份有限公司

未经本公司书面许可,任何单位和个人都不得擅自摘抄、复制、翻译本文档内容的部分或全部,并不得以任何形式传播。

免责声明

本文档并未授予任何知识产权的许可,并未以明示或暗示,或以禁止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
2020/01/16	1.0	初始版本。

目录

目录i
图目录ii
表目录iii
1 GMD 安装与配置1
2 软件编程模板
2.1 模板工程创建2
2.1.1 工程创建2
2.1.2 平台类型配置
2.1.3 编译工具链和路径配置
2.1.4 导入软件编程设计4
2.2 模板工程配置
2.2.1 配置 Target Processor4
2.2.2 配置 Optimization
2.2.3 配置 GNU RISC-V Cross Assembler Includes11
2.2.4 配置 GNU RISC-V Cross C Compiler Includes11
2.2.5 配置 GNU RISC-V Cross C Linker12
2.2.6 配置 GNU RISC-V Cross Create Flash Image14
2.3 编译15
2.4 下载16
3 参考设计17

i

图目录

图 2-1 工程创建	2
图 2-2 平台类型配置	3
图 2-3 编译工具链和路径配置	4
图 2-4 配置 Target Processor	5
图 2-5 配置 Optimization	8
图 2-6 配置 GNU RISC-V Cross Assembler Includes	11
图 2-7 配置 GNU RISC-V Cross C Compiler Includes	12
图 2-8 配置 GNU RISC-V Cross C Linker	13
图 2-9 配置 GNU RISC-V Cross Create Flash Image	15
图 2-10 编译	16

表目录

表 2-1 32 位 RISC-V 架构处理器数据类型等	宽度6
------------------------------	-----

1GMD 安装与配置

GOWIN MCU Designer 编译软件支持 Gowin_PicoRV32 的软件编程设计、编译和下载。

高云半导体官网提供 GOWIN MCU Designer 软件安装包下载: http://www.gowinsemi.com.cn/prodshow.aspx。

GOWIN MCU Designer 软件安装与配置,请参考 <u>SUG549</u>, GOWIN MCU Designer 用户指南。

2 软件编程模板

2.1 模板工程创建

2.1.1 工程创建

选择菜单栏 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. 选择工具链 RISC-V Cross GCC;
- 4. 单击 "Next"。

图 2-1 工程创建

😡 C Project	
C Project Create C project of selected type	
Project name: picoRV32_demo	
✓ Use <u>d</u> efault location	
Location: C:\GMD\workspace\picoRV32	demo B <u>r</u> owse
Choose file system: default 💌	
Project type:	Toolchains:
▲ 🔁 Executable	ARM Cross GCC
Empty Project Hello World ARM C Project	RISC-V Cross GCC
 Hello World RISC-V C Project 	
Shared Library	
Static Library	
Makefile project	
Show project types and toolchains on	y if they are supported on the platform
	xt > Einish Cancel

2.1.2 平台类型配置

平台类型配置,选择 Debug 和 Release,单击"Next",如图 2-2 所示。

图 2-2 平台类型配置

🔆 C Project		
Select Configurations Select platforms and configurations you wish to deploy on		
Project type: Executable Toolchains: RISC-V Cross GCC Configurations:		
 ✓ So Debug ✓ So Release 	Select all Deselect all	
	Advanced settings	
Use "Advanced settings" button to edit project's properties. Additional configurations can be added after project creation. Use "Manage configurations" buttons either on toolbar or on property pages.		
(?) < Back Next > Fin	ish Cancel	

2.1.3 编译工具链和路径配置

选择交叉编译工具链 riscv-none-embed-gcc 和其所在路径,单击 "Finish",完成 Gowin_PicoRV32 软件编程模板工程的创建,如图 2-3 所示。

默认配置为 GOWIN MCU Designer 软件中 RISC-V 交叉编译链安装路 径,如需自定义 RISC-V 交叉工具链,则请手动修改指定工具链位置。

🔀 C Project		_ D X
GNU RISC-V Cross Toolchain Select the toolchain and configure path		
Toolchain name:	GNU MCU RISC-V GCC (riscv-none-embed-gcc)	-
Toolchain path:	C:\GMD\toolchain\RISC-V_toolchain\bin	Browse
]
(?)	< <u>B</u> ack <u>N</u> ext > <u>Finish</u>	Cancel

图 2-3 编译工具链和路径配置

2.1.4 导入软件编程设计

完成 Gowin_PicoRV32 软件编程新工程创建后,选择工作空间 workspace 下新建的项目工程,导入软件编程设计。

以 GMD_RefDesign 参考设计为例,软件编程设计目录及代码定义如下 所示。

- STARTUP: Gowin_PicoRV32 引导启动文件
- SYSTEM: Gowin_PicoRV32 内核及系统定义
- PERIPHERAL: 外部设备驱动函数库
- sections.lds: Flash 连接器脚本

软件编程设计目录下如有代码更新,请在 Project Explorer 当前工程中 右键选择 "Refresh",更新 GMD 工程模板中的文件及代码。

2.2 模板工程配置

GOWIN MCU Designer 中,在 Project Explorer 选择当前工程,右键选择 Properties,选择选项卡 C/C++ Build,选择 Setting 选项,配置 Gowin_PicoRV32 模板工程的参数选项。

2.2.1 配置 Target Processor

选择 Target Processor,如图 2-4 所示。

🛞 Tool Settings 🛞 Toolchains 📕 Devices 🌶	🕅 Build Steps 🚇 E	Build Artifact 🗟 Binary Pa 🔹 🕨
🖄 Target Processor	Architecture	RV32I (-march=rv32i*)
🖄 Optimization	Multiply extension (RVM)	
2 Warnings	Atomic extension	n (RVA)
Debugging	Electing point	Nere
GNU RISC-V Cross Assembler	rioating point	INDITE
Preprocessor	Compressed ext	ension (RVC)
🖉 Includes	Integer ABI	ILP32 (-mabi=ilp32*)
🖉 Warnings	Flantine entite ADI	
Miscellaneous	Floating point ABI	INONE
GNU RISC-V Cross C Compiler	Tuning	Size (-mtune=size)
Preprocessor	Codo model	Toolshain dafault
Includes	code model	
Optimization	Small data limit	8
Warnings	Alian	Toolchain default (-mtune)
GINU RISC-V Cross C Linker	Small prologue/	epilogue (-msave-restore)
General	Force string ope	rations to call library functions
	Other target flags	-mplt -mfdiv -mdiv -mno-relax
GNU RISC-V Cross Create Flash Image		
GNU RISC-V Cross Print Size		
😕 General		

图 2-4 配置 Target Processor

• Architecture

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,则请使能 Multiply extension(RVM)选项。

同时在 Gowin PicoRV32 硬件设计 IPCore Genrator 的参数配置中, 使能 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,因此禁用该选项。

• Floating point

Gowin PicoRV32 不支持浮点指令,因此设置该选项为 None。

Compressed extension (RVC)

如果需要使用 RV32C 扩展,则请使能 Compressed extension (RVC) 选项。

同时在 Gowin PicoRV32 硬件设计 IPCore Generator 的参数配置中, 使能 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,因此配置该选项为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。

该选项不支持 Gowin PicoRV32,因此配置为 Size(-mtune=size)。

• Code model

设置参数-mcmodel,该参数指定程序的寻址范围。在 RV32 架构中,寻址空间为 4GB,因此-mcmodel 选项的值对编译结果没有影响,选择 Toolchain default 即可满足一般需求。

(-mcmodel=medany)选项用于指定该程序的寻址范围为 -2GB~+2GB,由于地址空间均为正数,因此-2GB 是寻址空间中最 高的 2GB 地址空间。 (-mcmodel=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,因此建议设置为 Strict(-mstrict-align)。

• Small prologue/epilogue(-msave-restore)

如果使能该选项,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)

如果使能该选项,则使用 PLT 生成中断控制代码,否则不使用 PLT。

• Floating-point divide/sqrt instructions(-mfdiv)

如果使能该选项,则使用硬件浮点除法和开方指令,需要处理器支持 RV32F 和 RV32D 指令集扩展和浮点寄存器。

Gowin PicoRV32 不支持 RV32F 和 RV32D 指令集扩展,因此禁用 该选项。

Integer divide instructions(-mdiv)

如果使能选项,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,所以可以使能 Multiply extension(RVM)选项,同时在 Gowin PicoRV32 硬件设计 IPCore Generator 的参数设置中,使能 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-exlicit-relocs

处理符号地址时,使用或不使用汇编程序重定位操作符。

另一种配置是使用汇编宏,这可能会限制优化。

- -mrelax
- -mno-relax

使用连接器松弛来减少实现符号地址所需的指令数量。 默认是使用连接器的松弛。

• -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

选择 Optimization, 配置优化选项, 如图 2-5 所示。

图 2-5 配置 Optimization

🛞 Tool Settings 🛞 Toolchains 📕 Devices 🌶	ဳ Build Steps 🚇 Build Artifact 🗟 Binary Pa 🔹 🕨
 Target Processor Optimization Wornings Debugging S GNU RISC-V Cross Assembler Preprocessor Includes Warnings Miscellaneous S GNU RISC-V Cross C Compiler Preprocessor Includes Optimization Varnings Includes Optimization Warnings Includes Optimization S GNU RISC-V Cross C Compiler Preprocessor Includes Optimization Warnings Miscellaneous 4 S GNU RISC-V Cross C Linker General Libraries Miscellaneous 4 S GNU RISC-V Cross Create Flash Image General 	Optimization Level Optimize most (-O3) Image: Message length (-fmessage-length=0) 'char' is signed (-fsigned-char) Image: Function sections (-ffunction-sections) Image: Function sections (-ffunction-sections) Image: Data sections (-fdata-sections) Image: Provide the section of
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。

- -**OO:**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

-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。

建议禁用该选项。

'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。

若连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。

若与连接器垃圾收集(连接器 –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。

建议

只有在能产生显著效果时才使能该选项。当指定该选项时,编译器和连接器将创 建更大 Size 的对象和可执行文件,降低编译速度,影响代码生成。该选项放置编 译器和汇编器使用翻译单元内的相对位置进行优化,因为这些位置在连接之前是 未知的。这种优化的一个案例是将调用放宽为短调用指令。建议禁用该选项,并 在连接器设置中使能-gc-sections选项,以减少编译 Size。

No common unitialized (-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。这将阻止连接器合并暂定定义,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。

建议禁用该选项。

• Do not inline functions (-fno-inline-functions)

若使能该选项,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,不能在这些调用上设置断点,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,因此禁用该选项。

• 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。

要使用连接时优化器,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 Assembler Includes

选择 GNU RISC-V Cross Assembler 下的 Includes 选项, 配置汇编文件 路径, 如图 2-6 所示。

以 GMD_RefDesing 参考设计为例,汇编文件路径配置如下所示。

"\${workspace_loc:/\${ProjName}/STARTUP}"

图 2-6 配置 GNU RISC-V Cross Assembler Includes		
🛞 Tool Settings 🚷 Toolchains 📕 Devices 🌶	🎙 Build Steps 🙅 Build Artifact 🗟 Binary Pa 💶 🕨	
 Target Processor Optimization Warnings Debugging S GNU RISC-V Cross Assembler Preprocessor Includes Warnings Warnings Miscellaneous 	Include paths (-I) 🕢 🕢 🕼 🦌 *\${workspace_loc:/\${ProjName}/STARTUP}*	
 Sourcesson Preprocesson Includes Optimization Warnings Warnings Miscellaneous Sourcesson Sourcesson Miscellaneous Libraries Libraries Miscellaneous Sourcesson 	Include system paths (-isystem)	
 General MOU RISC-V Cross Print Size General 	Include files (-include) 🛃 🔬 🛜	

2.2.4 配置 GNU RISC-V Cross C Compiler Includes

选择 GNU RISC-V Cross C Compiler 下的 Includes 选项,配置 C 头文件路径,如图 2-7 所示。

以 GMD_RefDesing 参考设计为例,汇编文件路径配置如下所示。

"\${workspace_loc:/\${ProjName}/PERIPHERAL/Includes}"

"\${workspace_loc:/\${ProjName}/SYSTEM}"

"\${workspace_loc:/\${ProjName}/USER}"

🛞 Tool Settings 🛞 Toolchains 🔳 Devices 🏓	🖱 Build Steps 🙅 Build Artifact 🗟 Binary Pa 🔹 🕨
 Target Processor Optimization Warnings Debugging GNU RISC-V Cross Assembler Preprocessor Includes Warnings Miscellaneous 	Include paths (-I) *{workspace_loc:/\${ProjName}/PERIPHERAL/Inclu *{workspace_loc:/{ProjName}/SYSTEM}" *{workspace_loc:/{ProjName}/USER}"
Preprocessor Preprocessor Preprocessor Includes Optimization Warnings Wiscellaneous Miscellaneous	Include system paths (-isystem) 🗐 🗐 🦌
 A state of cross of clinical A second cross of clinical<td>Include files (-include)</td>	Include files (-include)
🖄 General	

图 2-7 配置 GNU RISC-V Cross C Compiler Includes

2.2.5 配置 GNU RISC-V Cross C Linker

选择 GNU Cross RISC-V C Linker 下的 General 选项,配置 Gowin_PicoRV32 Flash 链接脚本 sections.lds,如图 2-8 所示。

以 GMD_RefDesign 参考设计为例, Flash 链接脚本如下所示。

"\${workspace_loc:/\${ProjName}/sections.lds}"

如果 Gowin_PicoRV32 软件编程设计使用片外 SPI-Flash 启动方式,请 在 sections.lds 中修改 Flash 起始地址"FLASH ORIGIN"为 0x01000080。

如果 Gowin_PicoRV32 软件编程设计使用 ILM 启动方式,请在 sections.lds 中修改 Flash 起始地址"FLASH ORIGIN"为 0x01000000。

🛞 Tool Settings 🛞 Toolchains 📗 Devices 🏓	🕅 Build Steps 🚇 Build Artifact 🗟 Binary Pa 🔹 🕨
🖄 Target Processor	Script files (-T) 🛛 🔮 🙀 🖗 🖓
🖄 Optimization	"\${workspace loc:/\${ProiName}/sections.lds}"
🖄 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 Print Size	
🖄 General	
	☑ Do not use standard start files (-nostartfiles)
	Do not use default libraries (-nodefaultlibs)
	No startup or default libs (-nostdlib)
	Remove unused sections (-Xlinkergc-sections
	Print removed sections (-Xlinkerprint-gc-secti
	Omit all symbol information (-s)

图 2-8 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起始文件。

Gowin PicoRV32 必须使用自定义的起始文件,因此必须使能该选项。

• 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连接器。 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连接器,并且指定系统库连接的选项(如-static-libgcc或-shared-libgcc)将被忽略。

该选项设置为禁用状态。

• 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)

当使能 Remove unused sections (-Xlinker –gc-sections)时,可以使能该选项,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。

2.2.6 配置 GNU RISC-V Cross Create Flash Image

选择 GNU RISC-V Cross Create Flash Image 中的 General 选项, 配置 输出文件格式为 RAW binary, 使能 Section: -j .text 和 Section: -j .data 选项, 如图 2-9 所示。

如果在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。



图 2-9 配置 GNU RISC-V Cross Create Flash Image

2.3 编译

完成模板工程配置和代码编写后,编译工程,单击工具栏编译按钮 5,编译生成 Gowin_PicoRV32 二进制 BIN 文件,如图 2-10 所示。



/* .text sections (code) */
/* .text* sections (code) */
/* .rodata sections (constants, strings, etc.) */
/* .rodata* sections (constants, strings, etc.) */
/* .rodata* sections (constants, strings, etc.) */
/* .rodata* sections (constants, strings, etc.) */

.

19 ł

20

21

. = ALIGN(4); *(.text) *(.text*) *(.rodata) *(.rodata*) *(.srodata) *(.srodata*) . = ALIGN(4);

🖹 Problems 🧔 Tasks 📮 Console 🛛

CDT Build Console [pv32_wbuart]

Invoking: GNU RISC-V Cross Print Size

riscv-none-embed-size --format-berkeley "pv32_wbuart.elf" text data bss dec hex filename 4836 4 0 4840 12e8 pv32_wbuart.elf Finished building: pv32_wbuart.siz

= ALIGN(4);

📋 cm3 timer

cm3_uart 📋 cm3_uart0_int 📋 cm3_ucos_iii

📋 cm3_wdog 📋 DigitalSeg

i free_rtos 📋 i2c int_priority 📋 keyscan

📋 lcd

📋 led 📋 mm ⊿ 🖾 pv32_wbuart

Binaries

Includes

👂 📂 Debug PERIPHERAL

STARTUP SYSTEM 🖻 🗁 USER sections.lds

📋 retarget 📋 spi

i systick

📋 timer

📋 ucos_iii i watchdog

📋 uart 📋 uart0 int 📋 uartes

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。

↓ ☆ 🗐 📰 🔐 = 🚉 | 🛃 = 🖬 + 😁 = 🗉



Gowin_PicoRV32 支持 GOWIN MCU Designer 软件环境的参考设计: Gowin_PicoRV32\ref_design\MCU_RefDesign\GMD_RefDesign。

