

Gowin_EMPU(GW1NS-4C)解决方案



IPUG1013-1.0,2023-02-03

版本信息

日期	版本	说明
2023/02/03	1.0	初始版本。

版权所有 © 2023 广东高云半导体科技股份有限公司

GO₩**IN**高云、₩、云源、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,其所有权利属其拥有者所有。未经本公司书面许可,任何单位和个人都不得擅自摘抄、复制、翻译本文档内容的部分或全部,并不得以任何形式传播。

免责声明

本文档并未授予任何知识产权的许可,并未以明示或暗示,或以禁止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。

目录

目录i
图目录iii
表目录iv
1 关于本手册1
1.1 手册内容1
1.2 相关文档1
1.3 术语、缩略语1
1.4 技术支持与反馈2
2 嵌入式实时操作系统
2.1 uC/OS-III
2.1.1 特征
2.1.2 版本
2.1.3 配置
2.1.4 参考设计4
2.2 FreeRTOS
2.2.1 特征
2.2.2 版本
2.2.3 配置
2.2.4 参考设计5
2.3 RT-Thread Nano
2.3.1 特征5
2.3.2 版本
2.3.3 配置
2.3.4 参考设计6
3 SRAM 中运行内嵌 SPI-Flash 代码7

i

	3.1 关于本方案	7
	3.2 参考设计	7
	3.2.1 硬件参考设计	7
	3.2.2 软件参考设计	7
	3.3 软件配置方法	
	3.3.1 Target Memory 配置	
	3.3.2 SRAM 代码归纳	9
	3.3.3 SRAM 代码配置	9
	3.3.4 SCT 文件配置	11
	3.3.5 主程序代码配置	
	3.4 下载方法	
	3.4.1 提取 ER_ROM1.bin 数据	
	3.4.2 下载内嵌 SPI-Flash	
	3.4.3 下载主程序	
4 5	SRAM 中运行内嵌 User-Flash 代码	15
	4.1 关于本方案	15
	4.2 参考设计	15
	4.2.1 硬件参考设计	15
	4.2.2 软件参考设计	15
	4.3 软件配置	15
	4.3.1 SRAM 代码归纳	15
	4.3.2 SRAM 代码配置	

图目录

图 3-1 配置"Options for Target > Target"	8
图 3-2 ram_func.c 示例	9
图 3-3 选择"Options for File 'ram_func.c'…"	10
图 3-4 配置"Memory Assignment > Code/Const"	11
图 3-5 配置"Options for Target > Linker"	12
图 3-6 修改 SCT 文件	12
图 3-7 Programmer 配置	14
图 4-1 ram_func.c 示例	16
图 4-2 选择"Options for File 'ram_func.c'…"	16
图 4-3 配置"Memory Assignment > Code/Const"	17

表目录

表 1-1 术语、缩略语	
--------------	--

1 关于本手册

1.1 手册内容

针对 Gowin_EMPU(GW1NS-4C)实际应用场景以及已知问题,本手册 提出一系列解决方案,包括嵌入式实时操作系统的解决方案(RTOS),内 嵌 SPI-Flash 的代码在数据存储器 SRAM 中运行的解决方案(Running in SRAM from Embedded SPI-Flash),指令存储器 User-Flash 的代码在数据 存储器 SRAM 中运行的解决方案(Running in SRAM from Embedded User-Flash。

1.2 相关文档

通过登录高云[®]半导体网站 <u>www.gowinsemi.com</u>可以下载、查看以下 相关文档。

- <u>SUG100</u>, Gowin 云源软件用户指南
- IPUG931, Gowin_EMPU(GW1NS-4C)软件编程参考手册
- IPUG932, Gowin_EMPU(GW1NS-4C)硬件设计参考手册

1.3 术语、缩略语

表 1-1 中列出了本手册中出现的相关术语、缩略语及相关释义。

表 1-1 术语、缩略语

术语、缩	略语	全称	含义
RTOS		Real Time Operating System	实时操作系统
SPI		Serial Peripheral Interface	串行外设接口
SRAM		Static Random Access Memory	静态随机存取存储器

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或建议, 可直接与公司联系:

网址: <u>www.gowinsemi.com.cn</u>

E-mail: support@gowinsemi.com

Tel: +86 755 8262 0391

2 嵌入式实时操作系统

Gowin_EMPU(GW1NS-4C)已移植支持 uC/OS-III、FreeRTOS 和 RT-Thread Nano 嵌入式实时操作系统。

2.1 uC/OS-III

2.1.1 特征

- uC/OS-III 是一个可扩展的,可固化的,抢占式的实时内核,管理的任 务个数不受限制;
- uC/OS-III 是第三代内核,提供了现代实时内核所期望的功能,包括资源管理、同步、任务间通信等;
- uC/OS-III 提供了很多其它实时内核所没有的特性,比如能在运行时测量运行性能,直接发送信号或消息给任务,任务能同时等待多个信号量和消息队列;
- Gowin_EMPU(GW1NS-4C)已成功移植 uC/OS-III 参考设计, uC/OS-III 源代码请在 Micrium 官网 <u>http://www.micrium.com</u>下载。

2.1.2 版本

Gowin_EMPU(GW1NS-4C)参考设计使用的 uC/OS-III 版本为 V3.03.00。

2.1.3 配置

- 用户可以通过修改 UCOSIII_CONFIG\os_cfg.h 和 os_cfg_app.h 来配置 uC/OS-III;
- 用户可以通过修改 UCOS_BSP\bsp.c 和 bsp.h 来支持所用开发板。

2.1.4 参考设计

点击如下链接获取 uC/OS-III RTOS 硬件参考设计和软件参考设计:

cdn.gowinsemi.com.cn/Gowin_EMPU(GW1NS-4C)_V1.2.zip

硬件参考设计

Gowin_EMPU(GW1NS-4C)支持高云半导体云源[®]软件(V1.9.8.10及以上版本)的 uC/OS-III 硬件参考设计:

solution\rtos\ref_design\FPGA_RefDesign\DK_START_GW1NSR4C_ QN48P_V1.1\gowin_empu

软件参考设计

Gowin_EMPU(GW1NS-4C)支持 ARM Keil MDK(V5.26 及以上版本) 和 GOWIN MCU Designer(V1.1 及以上版本)软件的 uC/OS-III 软件编程 参考设计:

- solution\rtos\ref_design\MCU_RefDesign\Keil_RefDesign\ucos_iii
- solution\rtos\ref_design\MCU_RefDesign\GMD_RefDesign\cm3_ucos _iii

2.2 FreeRTOS

2.2.1 特征

- FreeRTOS 是一个轻量级的实时操作系统;
- FreeRTOS 作为一个轻量级的操作系统,功能包括:任务管理、时间管理、信号量、消息队列、内存管理、记录功能、软件定时器、协程等,可基本满足较小系统的需要;
- FreeRTOS 操作系统是完全免费的操作系统,具有源码公开、可移植、 可裁减、调度策略灵活的特点;
- Gowin_EMPU(GW1NS-4C)已成功移植 FreeRTOS 参考设计, FreeRTOS 源代码可在 FreeRTOS 官网 <u>http://www.FreeRTOS.org</u>下载。

2.2.2 版本

Gowin_EMPU(GW1NS-4C)参考设计使用的 FreeRTOS 版本为 V10.2.1。

2.2.3 配置

用户可以通过修改 include\FreeRTOSConfig.h 来配置 FreeRTOS。

2.2.4 参考设计

点击如下链接获取 FreeRTOS RTOS 硬件参考设计和软件参考设计:

cdn.gowinsemi.com.cn/Gowin_EMPU(GW1NS-4C)_V1.2.zip

硬件参考设计

Gowin_EMPU(GW1NS-4C)支持云源软件(V1.9.8.10及以上版本)的 FreeRTOS 硬件参考设计:

solution\rtos\ref_design\FPGA_RefDesign\DK_START_GW1NSR4C_ QN48P_V1.1\gowin_empu

软件参考设计

Gowin_EMPU(GW1NS-4C)支持 ARM Keil MDK(V5.26 及以上版本) 和 GOWIN MCU Designer(V1.1 及以上版本)软件的 FreeRTOS 软件编 程参考设计:

- solution\rtos\ref_design\MCU_RefDesign\Keil_RefDesign\free_rtos
- solution\rtos\ref_design\MCU_RefDesign\GMD_RefDesign\cm3_free_r tos

2.3 RT-Thread Nano

2.3.1 特征

- RT-Thread Nano 是一个极简版的硬实时内核,由C语言开发,采用面向对象的编程思想,具有良好的代码风格,是一款可裁剪、抢占式实时多任务的RTOS;
- 其内存资源占用极小,功能包括任务处理、软件定时器、信号量、邮箱 和实时调度等相对完整的实时操作系统特性;
- 实时操作系统内核及所有开源组件可以免费在商业产品中使用,不需要 公布应用程序源码,没有潜在商业风险;
- RT-Thread Nano 源代码请在 RT-Thread 官网 <u>https://www.rt-thread.org</u> 下载。

2.3.2 版本

Gowin_EMPU(GW1NS-4C)参考设计使用的 RT-Thread Nano 版本为 V3.1.5。

2.3.3 配置

- 用户可以通过修改 bsp\empu_m3\rtconfig.h 来配置 RT-Thread Nano
- 用户可以通过修改 bsp\empu_m3\drivers\board.c 来支持所用开发板

2.3.4 参考设计

点击如下链接获取 RT-Thread Nano RTOS 硬件参考设计和软件参考设计:

cdn.gowinsemi.com.cn/Gowin EMPU(GW1NS-4C) V1.2.zip

硬件参考设计

Gowin_EMPU(GW1NS-4C)支持云源软件(V1.9.8.10及以上版本)的 RT-Thread Nano 硬件参考设计:

solution\rtos\ref_design\FPGA_RefDesign\DK_START_GW1NSR4C_ QN48P_V1.1\gowin_empu

软件参考设计

Gowin_EMPU(GW1NS-4C)支持 ARM Keil MDK(V5.26 及以上版本) 软件的 RT-Thread Nano 软件编程参考设计:

 $solution\rtos\ref_design\MCU_RefDesign\Keil_RefDesign\rt_thread_n$ and

3 SRAM 中运行内嵌 SPI-Flash 代码

3.1 关于本方案

Gowin_EMPU(GW1NS-4C)的代码是下载到指令存储器 FLASH,执行 也在指令存储器 FLASH 中,如果指令存储器 FLASH 的空间不能满足用户 代码容量的需求,例如 Gowin_EMPU(GW1NS-4C)指令存储器 FLASH 为 32KB,如果用户代码超过 32KB,在有内嵌 SPI-FLASH 的器件,例如 GW1NSR-4C QN48G 中,我们可以将部分代码下载到内嵌 SPI-FLASH, 运行时借用部分数据存储器 SRAM 的空间来执行这部分代码。

3.2 参考设计

点击如下链接获取 Running in SRAM from Embedded SPI-Flash 的硬 件参考设计和软件参考设计: <u>cdn.gowinsemi.com.cn/Gowin EMPU(GW1NS-4C) V1.2.zip</u>

3.2.1 硬件参考设计

Gowin_EMPU(GW1NS-4C)支持云源软件(V1.9.8.10 及以上版本)的 硬件参考设计:

solution\running_in_sram_from_emb_spiflash\ref_design\FPGA_RefD esign\DK_START_GW1NSR4C_QN48G_V1.1\gowin_empu_spinorflash

3.2.2 软件参考设计

Gowin_EMPU(GW1NS-4C)支持 ARM Keil MDK(V5.26 及以上版本)

软件的软件编程参考设计:

solution\running_in_sram_from_emb_spiflash\ref_design\MCU_RefD esign\retarget

3.3 软件配置方法

3.3.1 Target Memory 配置

配置 "Options for Target > Target",选择以及定义 off-chip ROM1、 on-chip IROM1 和 on-chip IRAM1 选项,如图 3-1 所示。

图 3-1 配置 "Options for Target > Tar	rget"
------------------------------------	-------

🕅 Options for Target 'retarget' X						
Device Target Output Listing User C/C++ Asm Linker Debug Utilities						
ARM ARMCM3	Code Generation					
<u>X</u> tal (MHz): 12.0	ARM Compiler: Use default compiler version 5					
Operating system: None	Use Cross-Module Optimization					
System Viewer File:	Use MicroLIB 🛛 🗍 Big Endian					
Use Custom File Post (Only Manage Areas	- Pood (Wite Memory Arrow					
default off-chip Start Size Startup	default off-chip Start Size Nolnit					
▼ ROM1: 0x40002400 0x2000 C	□ RAM1: □ □					
ROM2:	□ RAM2: □					
□ ROM3: □ O	□ RAM3: □					
on-chip	on-chip					
IROM1: 0x0 0x8000 €	✓ IRAM1: 0x2000000 0x2000 □					
IROM2:	IRAM2:					
OK Cancel Defaults Help						

ROM1 配置

• Start: 0x40002400

硬件参考设计中 APB Master [1]的起始地址,即 SPI-Flash controller 的起始地址。

• Size: 0x2000

借用硬件参考设计中 16KB 的数据存储器 SRAM,建议下载到 SPI-Flash 中的代码借用空间不要超过数据存储器 SRAM 的一半。

IROM1(MCU 指令存储器)配置

- Start: 0x0
- Size: 0x8000

IRAM1(MCU 数据存储器)配置

- Start: 0x2000000
- Size: 0x2000

图 3-2 ram_func.c 示例

原 16KB 数据存储器 SRAM 空间,借出 8KB 用来执行内嵌 SPI-Flash 里的代码。

3.3.2 SRAM 代码归纳

将所有需要在数据存储器 SRAM 中运行的代码,集中放置一个文件中,例如 ram_func.c,如图 3-2 所示。

<u>File Edit View Project Flash Debug Peripherals Tools SVCS Window Help</u> 🖳 🍂 🖉 - 🖕 🖉 🐛 🔄 -🧇 🔛 🕮 🧼 🕶 🧱 🔤 retarget 🖂 🔊 📥 🕾 🔶 🕎 幽 **д** 🗙 • × am_func.c 🖮 🦢 STARTUP -1/-startup_gw1ns4c.s B 🗁 SYSTEM 3 system_gw1ns4c.c 5 E PERIPHERAL 6 7 gw1ns4c_gpio.c 8 9 gw1ns4c_i2c.c 9 ↓ 10 /* Includes ------11 #include <stdio.h> 12 13 /* Functions ------14 void ram_func(void) 15 ⊟{ 16 for(int i = 0;i < 17 ⊟ { 18 printf("Hello W 19 } 21 ↓ 21 gw1ns4c_misc.c gw1ns4c_rtc.c gw1ns4c_spi.c gw1ns4c_syscon.c gw1ns4c_timer.c for(int i = 0;i < 10;i++)
{</pre> gw1ns4c_uart.c gw1ns4c_wdog.c printf("Hello World!\r\n"); 🗉 🧀 USER main.c retarget.c gw1ns4c_it.c spi_nor_flash.c a ram func.c E Project Books | {} Functions | 0, Templates Build Output **д X** compiling spi nor flash.c... Compliing spi_nor_liss.... linking.. Program Size: Code=1532 RO-data=240 RW-data=28 ZI-data=1636 After Build - User command fil D:\Keil_v5\ARM\ARMCC\bin\fromelf.exe --bin -o retarget .\Objects\retarget.axf ".\Objects\retarget.axf" - 0 Error(s), 0 Warning(s). Build Time Elapsed: 00:00:07 J-LINK / J-TRACE Cortex L:1 C:1

3.3.3 SRAM 代码配置

右键 ram_func.c 文件,选择 "Options for File 'ram_func.c'…",如图 3-3 所示。



在"Options for File 'ram_func.c'... > Memory Assignment > Code/Const"选项中,选择"ROM1 [0x40002400-0x400043FF]",如图 3-4 所示。

图 3-3 选择 "Options for File 'ram_func.c'..."

W Options for File 'ram_func.c'	>
Properties C/C++	
Path:\USER\ram_func.c	
File Type: C Source file	✓ Include in Target Build
Size: 716 Bytes	Always Build
last change: Tue Oct 19 08:16:49 2021	$\overline{\lor}$ Generate Assembler <u>S</u> RC File
	Assemble S <u>R</u> C File
Stop on Exit Code: Not specified	🔽 Image File Compression
Custom Arguments:	
Memory Assignment:	
Code / Const: ROM1 [0x40002400-0x400043FF]	
Zero Initialized Data: 	-
Other Data: <pre><default></default></pre>	
OK Cancel D	efaults Help

图 3-4 配置 "Memory Assignment > Code/Const"

3.3.4 SCT 文件配置

配置 "Options for Target > Linker",取消勾选 "Use Memory Layout from Target Dialog" 选项,编辑 "Scatter File" 选项,手动选择 SCT 文件,如图 3-5 所示。

🔣 Options for Target 'retarget'	×
Device Target Output Listing User C/C++ Asm Linker Debug Utilities Use Memory Layout from Target Dialog X/O Base:	
Scatter File	L
Misc controls -diag_suppress=L6329 Linker control string -cpu Cortex-M3 *.o -strict -scatter ".\Objects\retarget.sct"	< >
OK Cancel Defaults	Help

图 3-5 配置 "Options for Target > Linker"

手动打开编辑 SCT 文件,修改 "LR_ROM1 > ER_ROM1",修改方法 如图 3-6 所示。

图 3-6 修改 SCT 文件

```
2 ; *** Scatter-Loading Description File generated by uVision ***
4
  LR_IROM1 0x0000000 0x00008000 { ; load region size_region
5
    ER IROM1 0x00000000 0x00008000 { ; load address = execution address
 6
     *.o (RESET, +First)
7
8
     *(InRoot$$Sections)
9
     .ANY (+RO)
     .ANY (+XO)
10
11
    ł
    RW IRAM1 0x20000000 0x00002000 { ; RW data
12
13
     .ANY (+RW +ZI)
14
     }
15 }
16
  LR ROM1 0x40002400 0x00002000 {
17
    ER ROM1 0x20002000 0x00002000 { ; ; load address = execution address
18
19
      ram_func.o (+RO)
20
     .ANY (+RO)
21
22 }
    ł
23
```

3.3.5 主程序代码配置

下载到内嵌 SPI-Flash 的代码,在数据存储器 SRAM 中执行前,需要将这部分代码搬运到数据存储器 SRAM 中。

编辑 main.c 主程序,调用 SPI-Flash 的驱动函数 "Read",用于搬运 代码到数据存储器 SRAM 中。

3.4 下载方法

软件设计编译后,程序自动产生两个分离的文件 ER_IROM1(主程序 代码,运行于指令存储器 FLASH)和 ER_ROM1(下载到内嵌 SPI-Flash 的代码,即 ram_func.c 的代码)。

手动修改这两个文件的后缀为.bin,即 ER_IROM1.bin 和 ER_ROM1.bin。

3.4.1 提取 ER_ROM1.bin 数据

执行软件开发工具包中的软件工具

- "solution\running_in_sram_from_emb_spiflash\tool\bin2hex\bin2hex.exe
- ",提取 ER_ROM1 数据。

```
例如,执行命令 "bin2hex.exe ER_ROM1.bin",产生
ER_ROM1.bin.txt 文件,提取 ER_ROM1 数据。
```

3.4.2 下载内嵌 SPI-Flash

使用如下硬件设计和软件设计加载提取的 ER_ROM1 数据,以及下载 到内嵌 SPI-Flash:

硬件设计

solution\running_in_sram_from_emb_spiflash\tool\program\gowin_em pu_spinorflash

软件设计

solution\running_in_sram_from_emb_spiflash\tool\program\spi_nor_fl ash

1. 加载 ER_ROM1 数据

修改软件设计 spi_nor_flash 的 main.c 主程序,将提取的 ER_ROM1 数据填入数组,调用 SPI-Flash 的驱动函数 "Write",写入内嵌 SPI-Flash。

请根据实际 ER_ROM1 数据,修改 spi_nor_flash 工程代码。

2. 下载 ER_ROM1 数据

使用云源软件的 Programmer 下载软件,下载上述的 ER_ROM1 的硬

件设计和软件设计。

打开下载软件 Programmer, 单击菜单栏 "Edit > Configure Device" 或工具栏 "Configure Device" (*■*), 打开 Device configuration, 如图 3-7 所示。

- Access Mode 下拉列表,选择"MCU Mode"选项。
- Operation 下拉列表,选择 "Firmware Erase, Program"选项。
- Programming Options > File Name,选择硬件设计码流文件。
- FW/MCU/Binary Input Options > Firmware/Binary File,选择软件编程 设计 Binary 文件。

图 3-7 Programmer 配置

🙀 Device configuration		?	\times
Device Operation			
Access Mode:	MCV Mode		•
Operation:	Firmware Erase, Program		•
Firmware Erase, Progr	am		
Programming Options File name: win_empu_s User Flash Initiali	vinorflash/impl/pnr/gowin_empu_spinorflas zation	h.fs .	
FW/MCU/Binary Input Options Firmware/Binary File: ktop/spi_nor_flash/PROJECT/spi_nor_flash.bin			
	Save	Canc	el

完成配置后,单击 "Program/Configure" (事),下载硬件设计码流文件和软件编程设计 Binary 文件,即下载 ER_ROM1 数据到内嵌 SPI-Flash。

3.4.3 下载主程序

使用下载软件 Programmer,下载主程序的硬件参考设计码流文件和软件参考设计的 ER_IROM1.bin 文件,到指令存储器 FLASH。

4 SRAM 中运行内嵌 User-Flash 代码

4.1 关于本方案

如果程序中有对运行速度有较高要求的代码,可以将这部分代码放到数据存储器 SRAM 中运行,以提高代码的运行速度。

4.2 参考设计

点击如下链接获取 Running in SRAM from Embedded User-Flash 的硬件参考设计和软件参考设计:

cdn.gowinsemi.com.cn/Gowin EMPU(GW1NS-4C) V1.2.zip

4.2.1 硬件参考设计

Gowin_EMPU(GW1NS-4C)支持云源软件(V1.9.8.10 及以上版本)的 硬件参考设计:

solution\running_in_sram_from_emb_userflash\ref_design\FPGA_Ref Design\DK_START_GW1NSR4C_QN48G_V1.1\gowin_empu

4.2.2 软件参考设计

Gowin_EMPU(GW1NS-4C)支持 ARM Keil MDK(V5.26 及以上版本) 软件的软件编程参考设计:

solution\running_in_sram_from_emb_userflash\ref_design\MCU_Ref Design\retarget

4.3 软件配置

4.3.1 SRAM 代码归纳

将所有需要在数据存储器 SRAM 中运行的代码,集中放置一个文件中,例如 ram_func.c,如图 4-1 所示。

<u>File Edit View Project Flash Debug Periphe</u>	rals <u>T</u> ools <u>S</u> VCS <u>W</u> indow <u>H</u> elp
🗎 😂 🛃 🗸 🐁 🛍 🥙 🗠 🔶	隆 🏡 🛝 🙀 律 //: //長 🖄 0x00007FC 🛛 😡 🖓 🔍 🗸 🔷 🔿 🔗 🇶 💼 🗸
🧼 🎬 🕮 🧼 🕶 🎆 🙀 retarget	V 🔊 🛔 🗟 🔶 🐡 🏨
Project 📮 🗵	with ram_func.c ▼ X
PERIPHERAL gwins4c.gpio.c gwins4c.grisc.c gwins4c.risc.c gwins4c.grisc.c gwins4c.spic gwins4c.spic gwins4c.gpic gwins4c.wrenc.c gwins4c.wrenc.c	<pre> 1 /* 2 * @file ram_func.c 3 * @file ram_func.c 4 * @author GowinSemiconductor 5 * @device Gowin_EMPU(GWINS-4C) 6 * @brief Running codes in sram. 7 */ 1 fincludes</pre>
■ Project ③ Books {} Functions 0, Templates	< >>
Build Output	4 🗹
5	^
	J-LINK / J-TRACE Cortex L:1 C:1 .::

图 4-1 ram_func.c 示例

4.3.2 SRAM 代码配置

右键 ram_func.c 文件,选择 "Options for File 'ram_func.c'…",如图 4-2 所示。

图 4-2 选择 "Options for File 'ram_func.c'..."



在 "Options for File 'ram_func.c'... > Memory Assignment >

Code/Const"选项中,选择"IRAM1 [0x2000000-0x20003FFF]",如图 4-3 所示。

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help			
□ 📸 🗟 🗿 👗 🖻 🖏 り で ← → や 森 森 森 森 澤 津 川川 川長 🏙 0x00007FC 🔤 🗟 🌾 🔍 🔍 • ○ 🔗 🍖 🖬 🖬 • 🥄			
🖗 🕮 🕮 + 🔤 🙀 petaroet 🔍 🔊 🛔 🖶 🚸 🧇 🏟	1		
Project a 🖬 Options for File 'ram_func.c' X			
gw1ns4c_gpio.c Properties C/C++ gw1ns4c_it.c. Path: MUSERvam_func.c gw1ns4c_rit.c. Path: MUSERvam_func.c gw1ns4c_rit.c. Path: MUSERvam_func.c gw1ns4c_rit.c. Path: MUSERvam_func.c gw1ns4c_rit.c. File Type: C Source file If gw1ns4c_rit.c. Size: 531 Bytes If gw1ns4c_vart.c. Size: Size: File Type: gw1ns4c_wdog. Stop on Ext Code: Not specified If Wemory Assignment: Custom Arguments: Image File Compression Memory Assignment: Code / Const: IRAM1 [0x2000000-0x20003FFF] Zero Intialized Data:			
Other Data: Cdefault> Impr Build Output OK Cancel Defaults	n 2		
	~		
<	>		
J-LINK / J-TRACE Cortex			

完成代码编写以及配置后,编译产生软件编程设计 Binary 文件,使用 云源软件的 Programmer 下载软件,下载硬件设计码流文件和软件编程设计 Binary 文件。

图 4-3 配置 "Memory Assignment > Code/Const"

