

Application Note

Hercules 器件上的 FreeRTOS



Veena Kamath

摘要

FreeRTOS 是嵌入式系统中常用的开源实时操作系统。它提供了任务、信号量、互斥量、计时器等 API。

HALCoGen 是面向 Hercules 系列器件的、基于 GUI 的驱动程序生成工具。HALCoGen 还为 Hercules 系列中的各种器件提供 FreeRTOS 支持。它让用户能够生成 FreeRTOS 代码以及其他驱动程序。

本应用报告概述了 FreeRTOS。本文档不是 FreeRTOS 用户指南，仅介绍如何在 Hercules 器件上结合 HALCoGen 使用 FreeRTOS。

内容

1 FreeRTOS 源文件	2
2 HALCoGen 中的 FreeRTOS 支持	2
3 作为安全使用场景附加项的 MPU 设置	4
3.1 MPU 包装器函数	4
3.2 任务的 MPU 区域	4
4 将 FreeRTOS 移植到不同器件型号	6
5 修订历史记录	11

插图清单

图 2-1. 选择器件	2
图 2-2. OS 选项卡	3
图 4-1. 创建 HALCoGen 工程	6
图 4-2. 配置 PLL	7
图 4-3. 新建工程	8
图 4-4. VIM 配置	9
图 4-5. VIM ISR 分配	9
图 4-6. 使能 PortSWI	10

表格清单

表 3-1. 器件系列	4
-------------	---

商标

Arm® and Cortex® are registered trademarks of Arm Limited.

所有商标均为其各自所有者的财产。

1 FreeRTOS 源文件

FreeRTOS 源文件可分为平台相关文件和平台无关文件。它支持各种系列，包括 TI 的 Hercules 微控制器。

FreeRTOS\Source 文件夹包含各子模块（如任务、信号量、队列等）的驱动程序。这些驱动程序与平台无关。

FreeRTOS\Source\include 文件夹包含头文件，其中包括名为 mpu_wrappers.h 的头文件。对于支持 MPU 的平台，这些头文件将内核函数重定义为相应的 MPU 包装器函数。后续章节将更详细地介绍 MPU 包装器函数。

FreeRTOS\Source\portable\XXX 文件夹包含平台相关代码。对于每个平台，FreeRTOS 提供了用于移植函数的 .c 和汇编文件，以及用于平台相关宏的 portmacro.h 头文件。

除了这些源文件外，FreeRTOS 工程还需要以下文件：

- FreeRTOSConfig.h — 包含用于自定义 FreeRTOS 驱动程序的宏。示例文件作为演示的一部分提供。
- mpu_wrappers.c — 包含内核函数的 MPU 包装器函数。对于 FreeRTOS v9 及更高版本，FreeRTOS\Source\portable\Common 文件夹中提供了示例文件。

2 HALCoGen 中的 FreeRTOS 支持

要生成 FreeRTOS 代码和 HALCoGen 驱动程序，请在创建 HALCoGen 工程时选择 DeviceName_FREERTOS 作为器件，如图 2-1 所示。并非器件系列中的每个器件型号都支持 FreeRTOS 驱动程序。本文档介绍了针对不同器件型号自定义驱动程序的步骤。

FreeRTOS 源文件生成时带有前缀 “os_”。

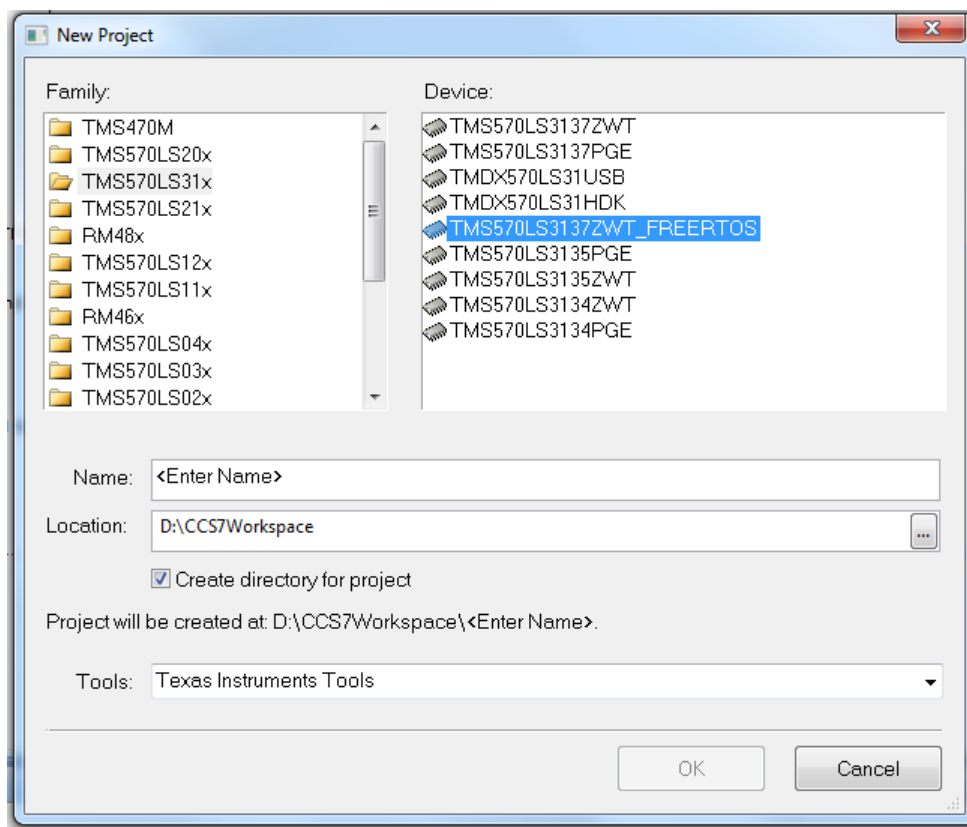


图 2-1. 选择器件

导航到 OS 选项卡以配置 OS 文件的各种选项，如图 2-2 所示。

Start Page TMS570LS3137ZWT_FREERTOS OS PINMUX GIO SCI LIN SCI2 MIBSPI1

General

Configuration

Configuration options will set macros in FreeRTOSConfig.h

☒ Use Task Preemption	☐ Use Mutexes	☒ Use Verbose Stack Checking
☐ Use Idle Hook	☐ Use Recursive Mutexes	☐ Use Timers
☐ Use Tick Hook	☐ Use Counting Semaphores	☐ Generate Runtime Statistics
☐ Use Co-Routines	☒ Idle Task Should Yield	☐ Use Malloc Failed Hook
☐ Use Trace Facility	☐ Use Stack Overflow Hook	

Task Configuration

RTI Clock (Hz):	90000000	Tick Rate (Hz):	1000
Max Priorities:	5	Total Heap Size:	8192
Task Name Length:	16	Min Stack Size:	128

Coroutine Configuration

Coroutine Priorities:

Timers Configuration

Timer Task Priority:	0	Queue Length:	0	Stack Size:	0
----------------------	--------------------------------	---------------	--------------------------------	-------------	--------------------------------

图 2-2. OS 选项卡

这些配置经过转换以创建 FreeRTOSConfig.h，并作为所生成驱动程序的一部分提供。

3 作为安全使用场景附加项的 MPU 设置

对于具有内存保护单元 (MPU) 的器件，FreeRTOS 提供了用于 MPU 配置的钩子函数。MPU 允许用户将器件内存划分为多个区域，并为每个区域配置访问权限。TI 的 Hercules 微控制器基于 Arm®Cortex®-R 并支持 MPU。

3.1 MPU 包装器函数

CPU 有两种模式：特权模式和用户模式（名称可能因器件而异）。用户模式属于受限模式，通常用于执行用户任务。特权模式用于执行内核代码和系统初始化。内核代码和数据通常置于特权内存段中，因此用户任务不能直接访问它们。MPU 包装器函数为内核函数提供包装器，以便在特权模式下执行这些函数。

MPU 包装器函数示例：

```
void MPU_vTaskDelete ( TaskHandle_t xTaskToDelete )
{
    BaseType_t xRunningPrivileged = prvRaisePrivilege();
    vTaskDelete ( xTaskToDelete );
    portRESET_PRIVILEGE( xRunningPrivileged );
}
```

3.2 任务的 MPU 区域

FreeRTOS 任务驱动程序支持为每个任务分配 n 个内存区域。内存区域及其访问权限可能因任务而异。

在 FreeRTOS 环境中，MPU 区域可分为任务特定区域和公共区域。任务特定区域在每次任务切换时配置。公共区域在调度器初始化时配置。

前四个区域和最后一个区域（区域 0-3 和 n-1）在调度器启动时配置，为所有任务均为公共区域。这包括闪存、RAM 和外设内存。其他区域特定于任务，可在任务创建期间配置。任务栈有专用的 MPU 区域。

表 3-1 列出了器件系列及其 MPU 区域数量，以及它们如何划分为公共区域和任务特定区域。

表 3-1. 器件系列

器件系列	MPU 区域数	公共区域	任务特定区域
TMS570LS31x RM48x TMS570LS12x RM46x	12	区域 0：非特权闪存区域 区域 1：特权闪存区域 区域 2：特权 RAM 区域 区域 3：外设区域 区域 11：特权系统区域	区域 4：任务栈 区域 5-10：用户定义
TMS570LS04x RM42x	8	区域 0：非特权闪存区域 区域 1：特权闪存区域 区域 2：特权 RAM 区域 区域 3：外设区域 区域 7：特权系统区域	区域 4：任务栈 区域 5、6：用户定义
TMS570LC43x RM57x	16	区域 0：非特权闪存区域 区域 1：特权闪存区域 区域 2：特权 RAM 区域 区域 3：外设区域 区域 15：特权系统区域	可在 HALCoGen GUI 中更改任务可配置区域的数量。默认值 = 3 区域 11：任务栈 区域 12-14：用户定义 其余区域可在 HALCoGen GUI 中配置，对所有任务均为公共区域。

备注

特权闪存和 RAM 区域包括内核使用的特权代码和数据段。

通过 HALCoGen GUI 完成的所有 MPU 配置会在调度器初始化时复位。对于 TMS570LC43x 和 RM57x 器件，可选择一个选项来设置 FreeRTOS 使用的区域数量。FreeRTOS 未使用的区域可通过 HALCoGen GUI 配置。这些区域不会被 FreeRTOS 代码复位。

要修改 FreeRTOS 设置的默认 MPU 区域，请修改 os_port.c 文件中的 prvSetupDefaultMPU 函数。

可使用两个 FreeRTOS API 创建任务：

- **xTaskCreate** — 禁用所有用户定义的任务特定区域，并允许访问整个 RAM 区域，将其作为任务栈区域的一部分。使用此 API 创建的任务称为非受限任务，因为它可以访问整个 RAM 区域。
- **xTaskCreateRestricted** — 允许用户定义其需要访问的区域。任务栈区域按实际栈大小设置。使用此 API 创建的任务称为受限任务，因为它除自己的栈之外不能访问任何其他 RAM 区域（除非明确提及）。

有关如何创建受限和非受限任务的更多详细信息，请参阅 HALCoGen 示例。

4 将 FreeRTOS 移植到不同器件型号

HALCoGen 仅支持为少数器件生成 FreeRTOS 驱动程序。本节介绍如何为任何器件型号生成 FreeRTOS 驱动程序。本示例使用 TMS570LS1227PGE 器件。对于 TMS570LS12x 系列，HALCoGen 为 TMS570LS1227ZWT 型号提供了 FreeRTOS 移植功能。

1. 使用 TMS570LS1227ZWT_FREERTOS 创建 HALCoGen 工程，如图 4-1 所示。

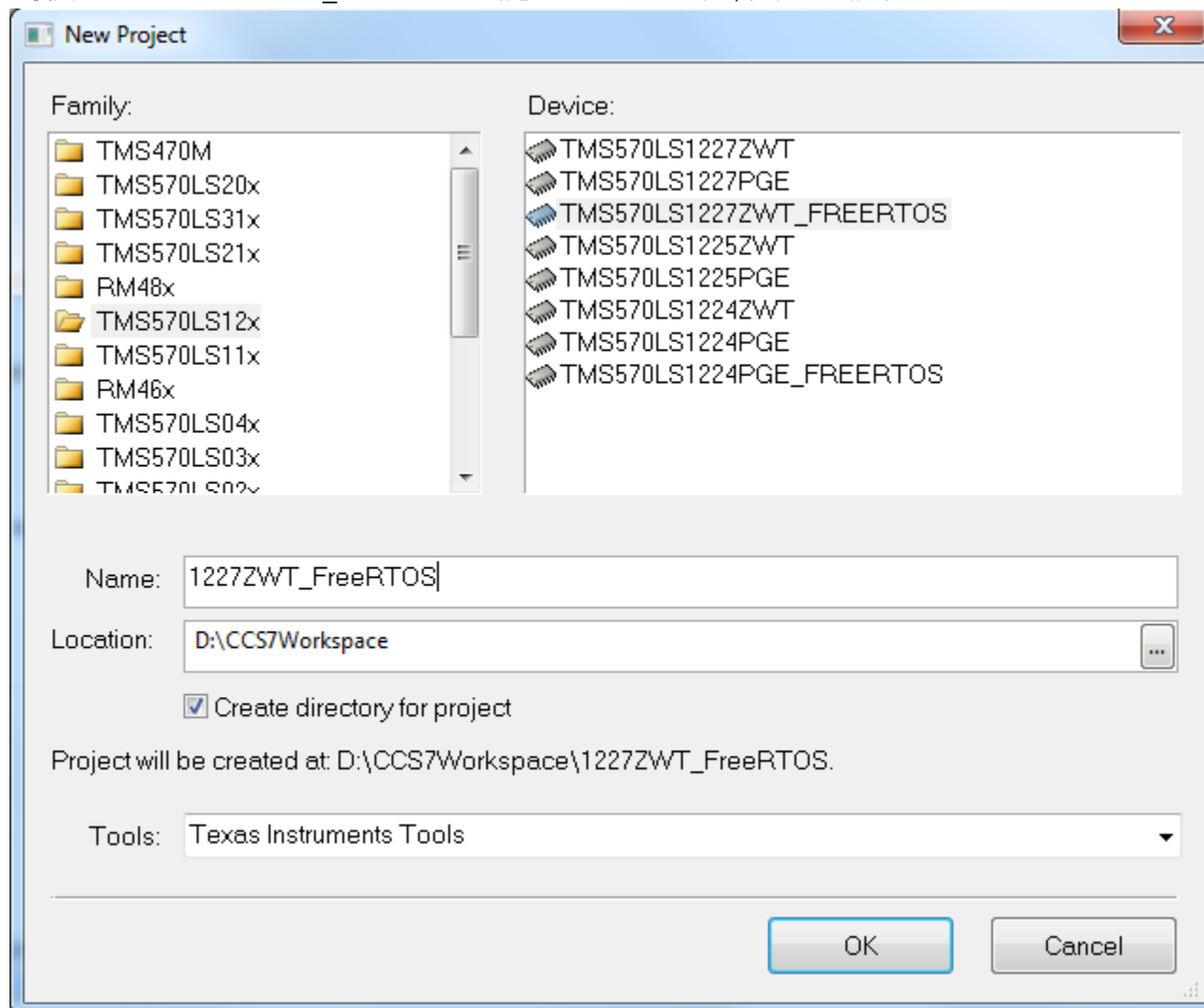


图 4-1. 创建 HALCoGen 工程

- 根据实际器件要求配置 PLL 和其他时钟设置。在图 4-2 所示的示例中，TMS570LS1227PGE 的最大 CPU 时钟为 160MHz，而 TMS570LS1227ZWT 为 180MHz。

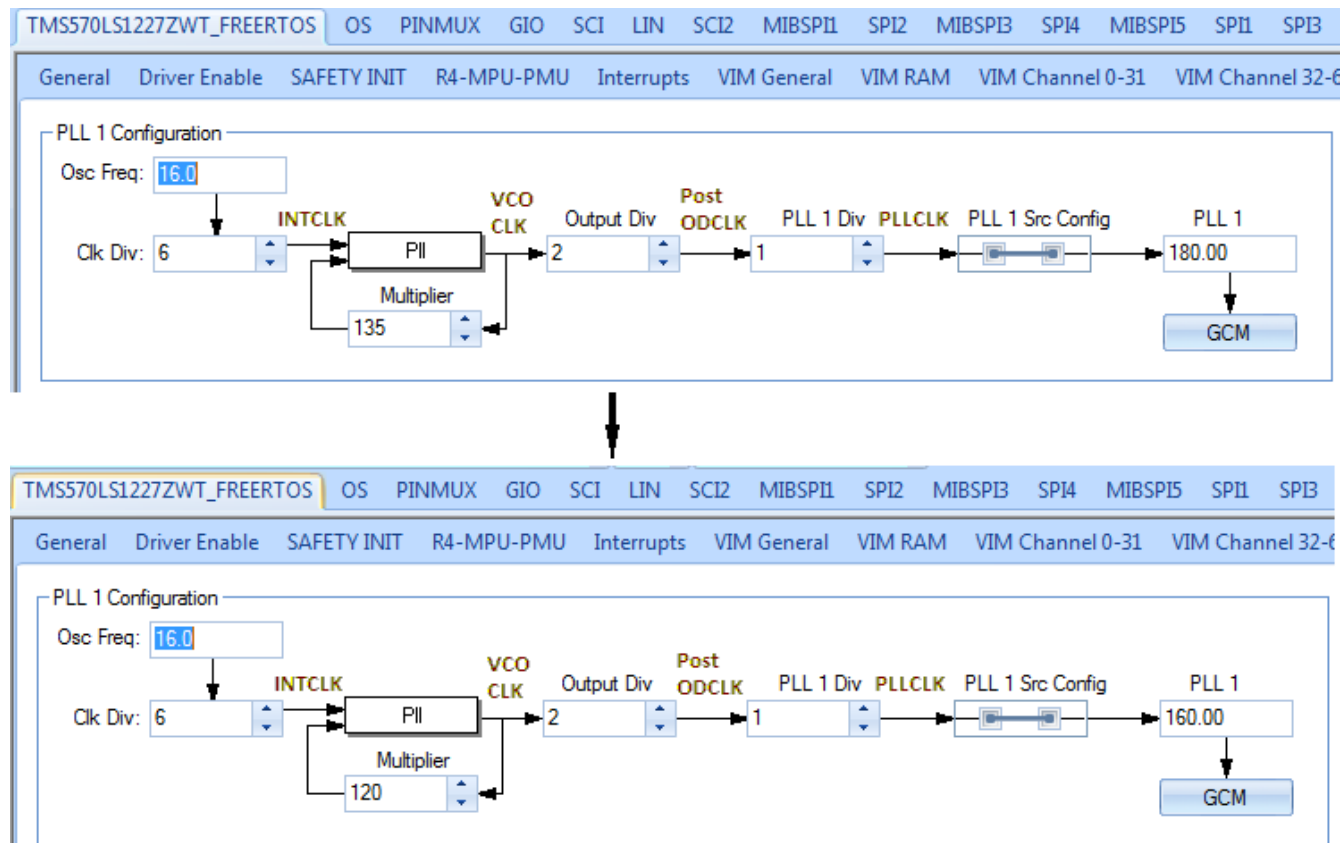


图 4-2. 配置 PLL

- 在 OS 选项卡中按需配置 FreeRTOS 移植。保存并生成代码。

4. 使用 TMS570LS1227PGE 创建 HALCoGen 工程，如图 4-3 所示。

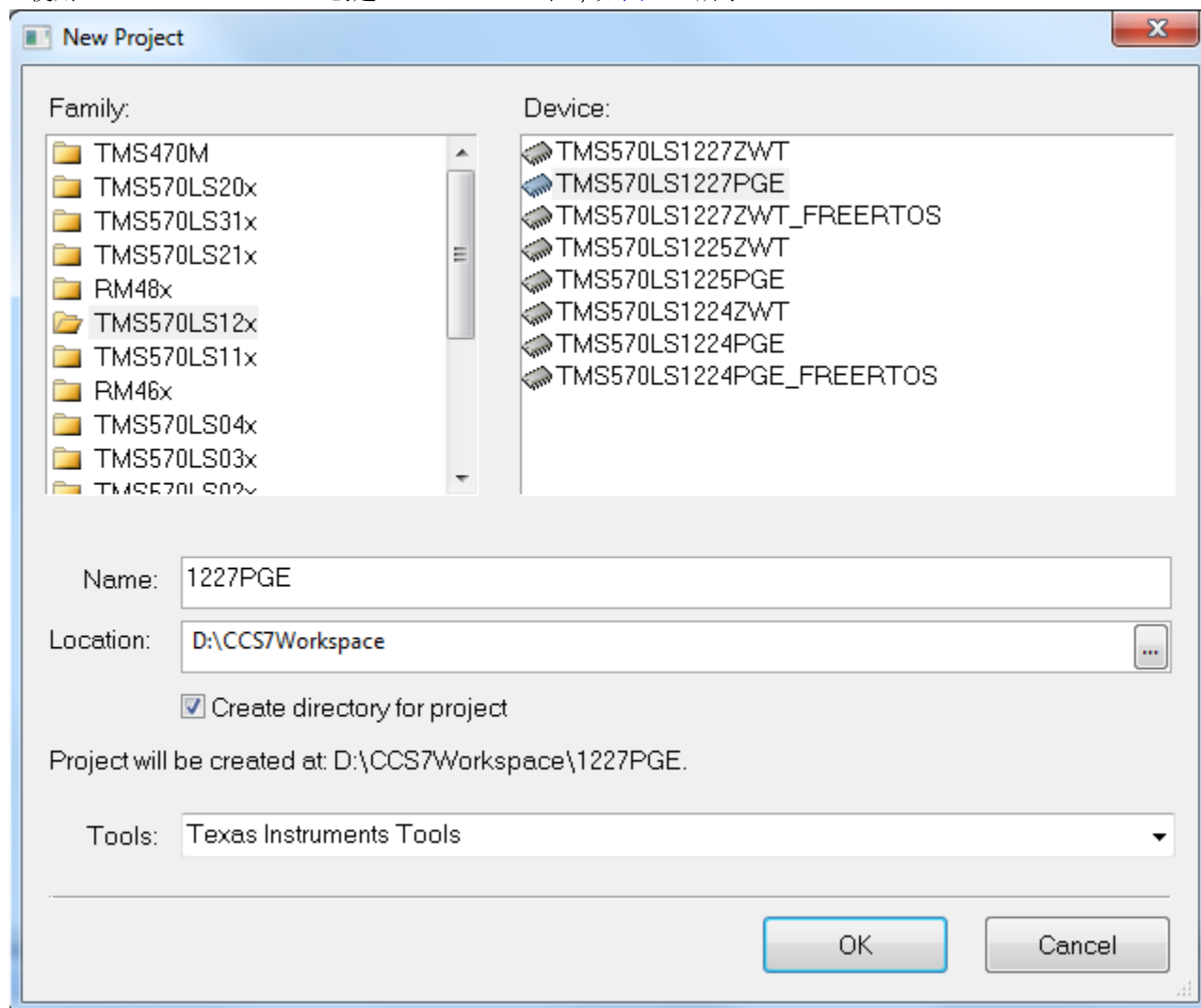


图 4-3. 新建工程

5. 按图 4-4 所示设置 VIM 配置。FreeRTOS 使用 2 个中断：RTI Compare 0 和系统软件中断 (SSI) (见图 4-5)。

使能中断通道：2 - RTI Compare 0。将 ISR 名称设置为 vPortPreemptiveTick

使能中断通道：21 - SSI。将 ISR 名称设置为 vPortYieldWithinAPI。

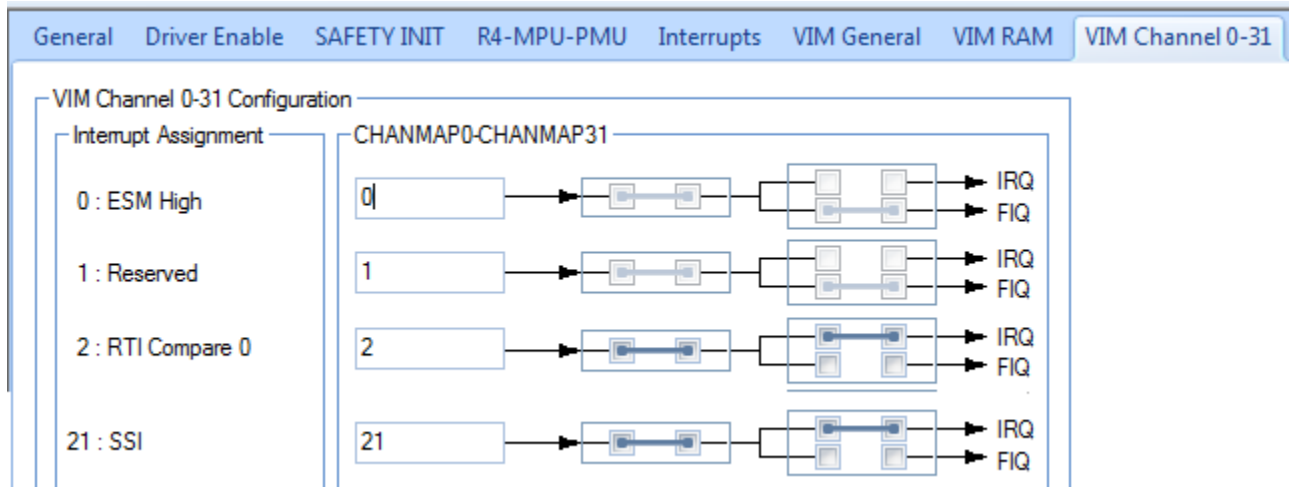


图 4-4. VIM 配置

TMS570LS1227PGE PINMUX RTI GIO SCI LIN SCI2 MIBSPI1 MIBSPI3 SPI4 MI				
General Driver Enable SAFETY INIT R4-MPU-PMU Interrupts VIM General VIM RAM				
VIM RAM : ISR Assignments				
Base Address: 0xFF82000		☒ VIM RAM Parity enable		
0x00000000:PH	phantomInterrupt	0x00000040:15	adc1Group1Interrupt	0x00000080:31
0x00000004:00	esmHighInterrupt	0x00000044:16	can1HighLevelInterrupt	0x00000084:32
0x00000008:01	phantomInterrupt	0x00000048:17	phantomInterrupt	0x00000088:33
0x0000000C:02	vPortPreemptiveTick	0x0000004C:18	phantomInterrupt	0x0000008C:34
0x00000010:03	rtiCompare1Interrupt	0x00000050:19	crcInterrupt	0x00000090:35
0x00000014:04	rtiCompare2Interrupt	0x00000054:20	esmLowInterrupt	0x00000094:36
0x00000018:05	rtiCompare3Interrupt	0x00000058:21	vPortYieldWithinAPI	0x00000098:37
0x0000001C:06	rtiOverflow0Interrupt	0x0000005C:22	phantomInterrupt	0x0000009C:38

图 4-5. VIM ISR 分配

6. FreeRTOS 还使用 SVC 异常。使能异常处理程序 vPortSWI，如图 4-6 所示。

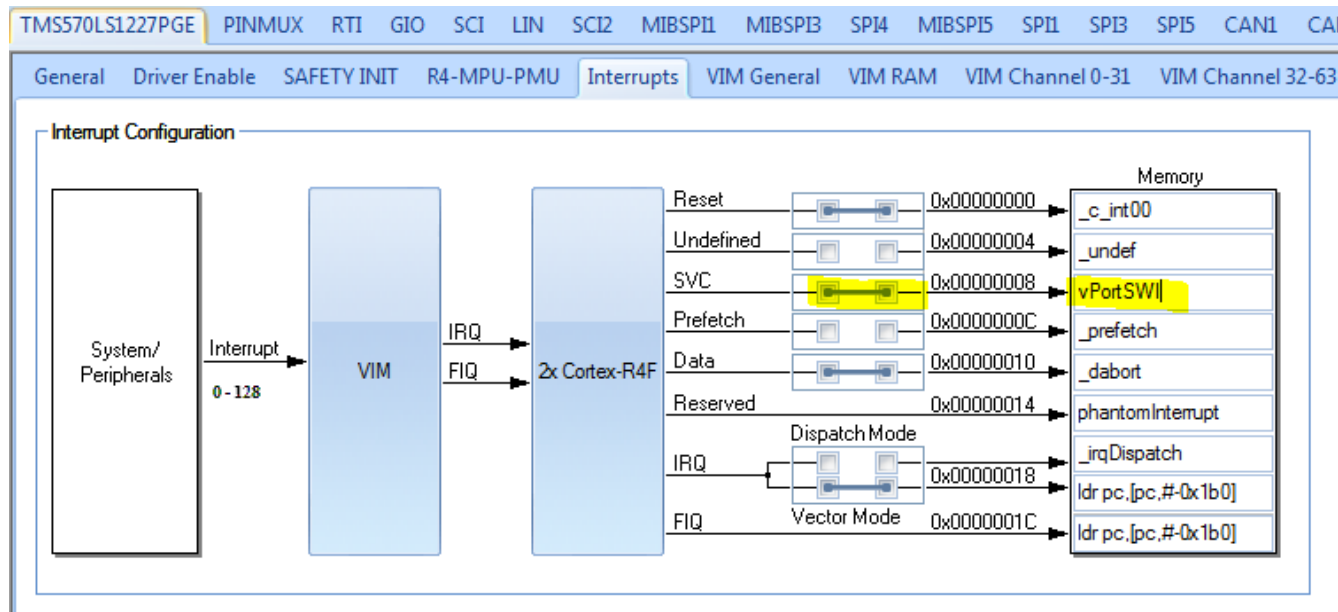


图 4-6. 使能 PortSWI

- 禁用 RTI 驱动程序并根据需要配置其他外设，然后生成代码。
- 将 OS 文件从 1227ZWT 工程复制到 1227PGE 工程。这包括以 “os_” 前缀开头的文件、FreeRTOS.h 和 FreeRTOSConfig.h。
- 在 1227PGE 工程中的链接器命令文件中进行以下更改：

```
MEMORY
{
    VECTORS (X) : origin=0x00000000 length=0x00000020
    KERNEL (RX) : origin=0x00000020 length=0x00000000
    FLASH0 (RX) : origin=0x00000020 length=0x0013FFE0 - 0x8000
    STACKS (RW) : origin=0x08000000 length=0x00001500
    KRAM (RW) : origin=0x08001500 length=0x00000800
    RAM (RW) : origin=0x08001500 + 0x800 length=0x0002EB00 - 0x800

    /* USER CODE BEGIN (2) */
    /* USER CODE END */
}

/* USER CODE BEGIN (3) */
/* USER CODE END */

/*-----
/* Section Configuration

SECTIONS
{
    .intvecs : {} > VECTORS
    /* FreeRTOS Kernel in protected region of Flash */
    .kernelTEXT : {} > KERNEL
    .cinit : {} > KERNEL
    .pinit : {} > KERNEL
    /* Rest of code to user mode flash region */
    .text : {} > FLASH0
    .const : {} > FLASH0
    /* FreeRTOS Kernel data in protected region of RAM */
    .kernelBSS : {} > KRAM
    .kernelHEAP : {} > RAM
    .bss : {} > RAM
    .data : {} > RAM

    /* USER CODE BEGIN (4) */
    /* USER CODE END */
}
```

- 现在 FreeRTOS 移植功能已可用于 TMS570LS1227PGE 器件。

5 修订历史记录

注：以前版本的页码可能与当前版本的页码不同

Changes from Revision * (April 2018) to Revision A (October 2026)	Page
• 更新了整个文档中的表格、图和交叉参考的编号格式.....	1

重要通知和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做任何明示或暗示的担保，包括但不限于对适销性、与某特定用途的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他安全、安保法规或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的相关应用。严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。对于因您对这些资源的使用而对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，您将全额赔偿，TI 对此概不负责。

TI 提供的产品受 [TI 销售条款](#)、[TI 通用质量指南](#) 或 [ti.com](#) 上其他适用条款或 TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。除非德州仪器 (TI) 明确将某产品指定为定制产品或客户特定产品，否则其产品均为按确定价格收入目录的标准通用器件。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

版权所有 © 2026，德州仪器 (TI) 公司

最后更新日期：2025 年 10 月