应用报告
ZHCA684C–March 2015–Revised March 2017

移植到 SimpleLink™ MSP432™ 系列

Dung Dang — *MSP432 Marketing*
Evan Wakefield — *MSP432 Applications*
Priya Thanigai — *MSP432 Applications*

摘要

16 位 MSP430™和 32 位 SimpleLink™ MSP432™微控制器 (MCU) 系列以低功耗、高性能实现互补。本移植指南旨在帮助开发人员精确评估将现有应用从 16 位 MSP430 移植到 32 位 SimpleLink MSP432 ARM®的相关工作，最终通过完整的软硬件覆盖范围制订移植策略， 从而在避免因平台差异而引入漏洞的前提下正确移植现有应用， 同时充分利用 32 位器件引入的独特特性或性能改进措施。

如果开发人员充分了解硬件或软件组件的移植或保留需求，可将这些组件与其在各自系统中的特定用途重新关联，并精确评估整个应用所需的全部移植工作。

内容

附图目录

附表目录

SLAA656 — http://www-s.ti.com/sc/techlit/SLAA656

商标

1 SimpleLink MSP432 平台移植概述

MSP432 设备是 SimpleLink 微控制器 (MCU) 平台的组成部分，包括： Wi-Fi®， *Bluetooth*®低功耗、1GHz 以下器件和主机 MCU。它们都共用配有单核软件开发套件 (SDK) 和丰富工具集的通用、易用型开发环境。一次性集成 SimpleLink 平台后，用户可以将产品组合中器件的任何组合添加至您的设计中。SimpleLink 平台的最终目标是确保设计要求变更时，完全重复使用代码。

MSP432 器件将 32 位 ARM Cortex®-M4F 内核与 MSP 超低功耗 DNA 相结合，同时提供高度集成的外设。为了同时实现 Cortex-M 和 CMSIS 合规性以及从 16 位 MSP430 器件到 SimpleLink MSP432 器件的顺利移植，我们进行了慎重考量。更多关于 SimpleLink 平台的信息，请访问 www.ti.com/simplelink。

本移植指南分为以下几部分：

- 在应用报告第一部分中，从器件层级对两个 MSP 平台进行了对比，并强调了一些重要的硬件移植考注意事项，包括内核、系统级考注意事项、外设修改和附加内容。
- 应用报告的大部分篇幅侧重于软件移植，因为将现有应用移植到另一 MCU 平台时，软件移植的工作量通常是最大的。软件概述提供的高级指南用于根据选择的硬件和软件选项评估移植范围。以下各节根据内核系统、外设代码、编译器特定代码和内部特定代码、高级库和应用代码划分了移植步骤。

2 CPU 和内核

MSP430 CPU 采用 16 位 RISC 架构， 具有 专门针对最新高效编程而设计的特性，比如正交架构、单周期寄存器指令、无分页统一存储器映射以及存储器间直接传输等。MSP432 CPU 围绕着行业标准 ARM Cortex-M4F 32 位内核构建，并充分利用了与 ARM 处理器关联的完善的开发工具和软件解决方案生态系统的优势。虽然这两个处理器可能被视为完全不同的类型，但实际上，它们在架构方面有很多相似之处，如表 1所述。

表 1. MSP430 与 MSP432 内核比较

名称	MSP430™ 内核	MSP432™ 内核
数据宽度	16 位	32 位
程序总线宽度	16 位 (CPU) 或 20 位 (CPUX) 地址总线	32 位
总线类型	16位 MSP430 总线	AHB
架构	Von Neumann (Princeton)：数据运算和指令获取共享同一总线	Harvard：独立的数据总线和指令总线
指令集	RISC，MSP430 专利	RISC，thumb 和 thumb2
指令大小	16 位（每个操作数 16 位）	16 位和 32 位
指令周期（典型值）	1-4 个周期	1-2 个周期
管线	无	3 级管线
预取缓冲区	128 位	128 位
功耗模式	工作模式，LPM0-LPM4，LPMx.5	工作模式，低频，LPM0，LPM3，LPMx.5
调试接口	MSP430 4 线制 JTAG 和 2 线制 SBW	4 线制和 2 线制模式的 ARM JTAG
数学函数支持	硬件乘法器 (MPY)	硬件乘法器和除法器、DSP 扩展器以及集成式 FPU

SLAA656 — http://www-s.ti.com/sc/techlit/SLAA656

随着编译器的不断改进，CPU 和内核层级上的很多难点和差异将通过编译器和链接器工具进行管理。虽然此举减轻了 C 程序员的工作量，但了解每种架构的差异和优势对于优化软件性能、规模和实时特性很有必要。4 节还介绍了将软件从一种平台移植到另一种平台（从 16 位平台移植到 32 位平台，或相反）时的限制因素。

3 硬件 功能 和移植注意事项

3.1 系统 特性

3.1.1 系统级比较

表 2. MSP430 与 SimpleLink MSP432 MCU 整体系统级比较

参数	MSP430™ MCU	MSP432™ MCU
电源电压范围	1.8V 至 3.6V	1.62V 至 3.7V
模拟电源电压	1.8V 或 2.2V 至 3.6V	1.8V 至 3.7V
最大系统频率	8MHz、16MHz、20MHz 或 25MHz	48MHz
非易失性（闪存或 FRAM）存储器	512 字节至 512K 字节	最大 256KB
RAM 存储器	128 字节至 64K 字节	最大 64KB

3.1.2 复位

MSP432 MCU 的复位电路与 MSP430 MCU 的复位电路类似。复位引脚可配置为复位功能（默认），也可以配置为特殊功能寄存器 (SFR) 中的 NMI 功能，即 SFRRPCR。

在复位模式下，RST/NMI 引脚为低电平有效，对该引脚施加符合复位时序规范的脉冲会引起 BOR 型器件复位。将 SYSNMI 置 1 可将 RST/NMI 引脚配置为外部 NMI 源。外部 NMI 是边缘敏感的，其边缘是由 SYSNMIIES 选择的。设置 NMIIE 能使能外部 NMI 的中断。如果发生外部 NMI 事件，NMIIFG 会置 1。RST/NMI 引脚可连接上拉电阻或下拉电阻（使能或禁用）。SYSRSTUP 用于选择上拉电阻或下拉电阻，SYSRSTRE 用于使能（默认）或禁用上拉电阻（默认）或下拉电阻。

如果未使用 RST/NMI 引脚，需要选择并使能内部上拉电阻，或者将外部上拉电阻连接至 RST/NMI 引脚，并为 VSS 连接去耦电容。

3.1.3 电源

MSP432 系列器件新增了 包括 双稳压器（LDO 和 DC-DC）在内的一系列电源系统功能，可为内部逻辑和其他组件生成两个 VCORE 电平。

内核电压频率限制和运行时可选稳压器这类功能 可提高器件的灵活性，从而可 进一步优化电源系统以及系统功耗。更多信息，请参见*《最大限度提高 MSP432P4xx 稳压器的效率》*。

更多关于设计电源系统和制定应用功耗曲线的资源，请参见*《采用 SimpleLink™ MSP432™ 微控制器设计超低功耗 (ULP) 应用》*。

3.1.4 低功耗模式

MSP432 MCU 系列保留了 MSP430 MCU 的超低功耗架构，提供类似的电源和低功耗架构，开发人员可在此架构上开发高能效应用。除了工作模式之外，MSP430 和 MSP432 MCU 平台均可提供多种低功耗模式，各模式下会对不同时钟和外设的功率进行门控，从而可针对不同应用状态下的功耗灵活地进行优化。在 MSP430 平台上，这些低功耗模式的编号为 LPM0 到 LPM4，最近已扩展到 LPM3.5 和 LPM4.5。每个级别均可提供不同程度的时钟和外设可用性。MSP432 MCU 平台继续为低功耗模式使用类似的结构，保留了最有用的电源模式，并扩展为采用两种新模式实现慢速执行。表 3归纳了 MSP 平台上的电源模式及其彼此间的关联方式。

表 3. MSP430™ 和 MSP432™ MCU 的电源模式

MSP430™ MCU	MSP432™ MCU	行业 描述	注释
工作	工作	工作模式	CPU 和外设
	低频工作	低功耗运行	CPU 和外设 < 128kHz
LPM0	LPM0	ARM：休眠	外设开启，CPU 关闭
	低频 LPM0		休眠 + CLK < 128kHz
LPM1	不适用	MSP430 特有的模式	
LPM2	不适用	MSP430 特有的模式	
LPM3	LPM3	ARM：深度休眠 RAM 和 RTC 处于待机模式	A/BCLK，< 32kHz，一些外设可用
LPM4	不适用	RAM 处于待机模式	无时钟，一些外设可用
LPM3.5	LPM3.5	ARM：关断	RTC 不含 RAM
LPM4.5	LPM4.5	ARM：关断	关断

3.1.5 时钟

MSP 平台集成了稳健耐用的统一高度正交计时系统，该系统易于使用，并且在不同 MSP 系列和所有器件之间保持一致。时钟模块提供的统一时钟树最多具有四个时钟信号，可用于高速计时和低速计时，具有一定的灵活性，并能在性能和功耗之间达到平衡。可使用多个频率和精度不同的内部和外部时钟源来提供时钟信号。时钟信号也可用于为 CPU 提供时钟源 (MCLK) 以及为外设提供时钟。计时外设是通过时钟进行门控的，并且时钟选择可在外设层级进行配置（使用外设寄存器），但时钟信号可通过时钟模块或外设请求使能，无需使能外设时钟本身。

MSP432P4xx 系列保留了与 MSP430F5xx、MSP430FR5xx 和 MSP430FR6xx 相似的计时系统，结合了这些计时系统 提供的 最佳特性。需要使用高速、高精度内部时钟源 的应用可利用改进的 DCO 提供高达 48MHz 运算，该 DCO 高度可调，无需使用调制，抖动低，通过外部电阻提供精度较高的选项。MSP432 不仅扩展为支持更高速的运算，还在频率范围内增加了更多时钟源和信号，从而可提供更多的粒度和计时选项，更好地满足不同应用的需求。将应用从另一平台移植到 MSP432 平台时，可以考虑使用这些选项（参见表 4）来优化应用用例。

3.1.6 内核系统相关性及配置

表 4 列出了根据所选系统频率建议使用的系统配置。

表 4. 系统级配置

系统频率	VCORE	建议使用的稳压器	闪存等待状态
0MHz 至 12MHz	0	LDO	0
12MHz 至 16MHz	0	LDO	1
16MHz 至 24MHz	0	LDO	1
24MHz 至 32MHz	1	高有效占空比应用为 DC-DC，其他应用为 LDO	1
32MHz 至 48MHz	1	高有效占空比应用为 DC-DC，其他应用为 LDO	1

3.2 外设

最初的 MSP432 系列将之前在 MSP430 中引入的超低功耗外设以及一些新外设结合在一起，从而提高了性能。

共享的外设在 MSP430 和 MSP432 平台的运行方式相同，唯一例外的是它们的中断信号和处理程序不同，这是因为 MSP432 MCU 上新增了嵌套矢量化中断控制器 (NVIC) 模块。更多关于将 MSP430 移植到 NVIC 的详细信息，请参见4 节。不同系列器件头文件中的寄存器定义及其说明也是相同的，因此，无论是使用寄存器级访问还是 MSP Driver Library 等较高抽象层代码，均可实现代码重用。

MSP432 MCU 上的一些外设经稍加修改，性能也较其 MSP430 MCU 同类器件有所改进。举例来说，ADC14 就是在 ADC12_B 基础上改进的版本。两者的共享特性和 运算 可利用和重用，并且新增特性和改进特性 （比如分辨率从 12 位提升至 14 位、或采样速率从 200ksps 增至 1Msps） 则需要重新设计并进行软件方面的考量。

MSP 平台新增的 MSP432 MCU 上的最后一组外设来自或继承自 µDMA、Timer32 和 SysTick 等 ARM。这些模块为器件增加了新功能，需要对现有系统和代码进行进一步修改才能实现新功能。特别是 µDMA 模块，其中引入的很多新特性和新功能 可视为 MSP430 DMA 的 升级。要在 MSP432 MCU 应用中使用这些特性，应在系统设计层面进行深入 研究 。

4 软件移植

4.1 软件移植概述

将现有软件从一个器件移植到另一个器件时，最先要做的是评估并确定哪些现有代码组件可以重用，哪些组件需要修改或重新开发。预先进行的评估有助于估算出移植需要的工作量，从而可准确地规划资源并将软件分为两类：一类是可重用的代码， 一类是由于平台差异或设备新特性的原因而需要修改或新建的代码。为实现跨设备移植或跨平台移植，可将软件组件分为以下几类：与 CPU 和内核相关的代码、系统外设（例如电源、时钟和存储器）代码、外设代码（使用寄存器访问或外设驱动程序库）、与中断相关的代码、内部特定代码和编译器特定代码、最后是更高抽象层的软件库和顶层应用代码。根据待移植器件之间的差异大小，需要更新的软件组件数量会有所不同。

移植到 MSP432 系列器件时，需要浏览关于 RTOS 使用方法的内容。表 6中的许多分组可通过 RTOS（如 TI-RTOS）轻松管理。如表 5所示，SimpleLink SDK 具有很多 MSPWare 之前没有、但对于 SimpleLink SDK 来说是标配的 新特性， 包括预装 TI-RTOS 内核、符合 POSIX 规定等。SimpleLink SDK 还支持其他 RTOS 内核（比如 FreeRTOS），为开发人员提供了灵活性。每种内核均可提供实时多任务服务，如任务定时和任务调度。RTOS 内核为所有片上外设运行硬件抽象层以及一套功能驱动程序。例如，如果使用 RTOS，RTOS 可自动检测器件何时应进入低功耗模式并在空闲时间段内运行，并且 RTOS 会代开发人员管理此操作。

表 **5.** 软件组件

	MSPWare	SimpleLink™ SDK
示例和演示	是	是
TI 驱动程序	否	是
DriverLib 和 HAL	是	是
插件	否	是
TI-RTOS	TI-RTOS 链接	是
FreeRTOS	不可用	是
符合 POSIX 要求的 API	否	是
平台特定库	是	是
文档	是	是

表 6 分步介绍了移植到 MSP432 器件平台的过程。示例表中使用的外设集合基于 MSP432P4xx 系列器件上可用的配置。

表 **6.** 将软件组件从 **MSP430Ware** 移植到 **SimpleLink SDK**

组	组件	兼容等级	编译器支持	寄存器访问代码	DriverLib API
内核	CPU 和内核相关	低		新开发	新，智能 API
	数据类型	中		使用显式类型	使用显式类型
	中断	低	部分	新	可用作 API
	内联函数	部分	高		部分可用作 API
系统	系统模块：电源、时钟、存储器	低	不可用	新开发	新，建议使用易用型 DriverLib API
MSP 外设	共享 MSP 外设：Timer_A、eUSCI、REF_A、COMP_E、WDT_A、RTC_C、GPIO、AES256	高	不适用	兼容，检查本机数据类型和寄存器宽度（16 位）	兼容，检查本机数据类型和 BASE_ADDRESS 使用情况
	扩展或新修订的 MSP430 外设：ADC14、CRC32	中到高	不适用	部分兼容，查看不同的寄存器宽度（16 位到 32 位）、小位重定位，新标志清零机制	部分兼容，可视为新 API，以及现有 API 的参数检查数据类型和 BASE_ADDRESS 使用情况
ARM 外设	ARM 外设：DMA、SysTick、Timer32	无	不适用	新开发	新开发，建议使用 DriverLib API
软件库，顶层应用代码		高	不适用	兼容使用 grlib、iqmathlib 和其他库	

利用表 7的移植评分系统，开发人员可将其现有的应用分解为相似的组件和分组。由于各应用的抽象程度各不相同，因此分解可能会得出不同结果，但通常来讲，对于抽象层足够多且结构合理的软件来说，可以将低级系统和外设代码从驱动程序、库以及高级应用代码中分离。了解了每个分组和组件在应用代码中的分布后，开发人员不仅可利用表 7中的移植工作量确定其应用移植的工作量，还可以在更高层面上了解需要为每一组件执行哪些操作才能在新 MSP 平台上创建兼容、稳健的应用。表 7显示了软件组件在典型应用中的分布（假定该应用使用了所有组件和外设）。使用该图，在确定特定应用的分布后，开发人员可从更高的层面得出移植该应用需要的工作量。

SLAA656 — http://www-s.ti.com/sc/techlit/SLAA656

表 7. 移植到 **MSP432P4xx** 的软件组件的分布

组	组件	移植工作	在应用代码中所占的百分比
内核	CPU 和内核相关	低到中	
	数据类型	中	
	中断	中	
	内联函数	中	
系统	系统模块：电源、时钟、存储器	中	
MSP 外设	共享 **MSP** 外设：Timer_A、eUSCI、REF_A、COMP_E、WDT_A、RTC_C、GPIO、AES256	低	
	扩展或新修订的 **MSP430** 外设：ADC14、CRC32	低到中	
ARM 外设	ARM 外设：DMA、SysTick、Timer32	低到中	
软件库，顶层应用代码		低	

需要注意的是，每节中给出的移植工作量仅供参考。只有在特定应用环境中对这些组件进行仔细评估后，才能确定准确的分数。以下章节详细介绍了每个组件的移植步骤，并强调了每个组件移植过程中需要注意的细节。

4.2 CPU、内联函数和编译器支持

4.2.1 数据类型，16 位和 32 位

MSP430 与 MSP432 最显著的区别是两者分别采用 16 位和 32 位 CPU 架构，因此两者的本机数据类型和数据大小均不同。在 C 语言中，**int** 类型采用架构的本机大小；因此，对于 MSP430 MCU，int 表示 16 位整数，对于 MSP432 MCU，int 表示 32 位整数。如果变量依赖于其数据大小和类型，尤其是考虑的类型和运算可能涉及到符号和溢出的情况，数据类型不正确可能导致计算结果错误。如果变量用于检索，使用的数据类型不正确可能导致存储器访问无效或超出范围。从 MSP430 MCU 移植到 MSP432 MCU 时，因使用 **int** 而导致出错的可能性不太大，因为新变量额外有 16 位来缓存已有的 16 位数据。

要修改不正确或不明确的数据类型使用，开发人员可使用 C99 类型，该类型明确指出了数据大小和数据类型。无符号 **int** 或 **int** 以 **uint32_t** 或 **int16_t** 代替不仅可避免变量类型误用，还可帮助开发人员更清晰地在代码中标识出变量类型，并可指示变量用途。如果将为多个平台写入软件，为了确保使用的数据类型正确无误，同时确保使用的数据类型最适合其特定用途（检索变量、数据存储、临时计算等），并保证为特定 MSP 平台使用的数据类型是最佳数据类型，也可使用 C99 快速和类型和最小类型，如 **uint_fast32_t** 或 **uint_least16_t**。

快速整型采用平台上可用的较快速的整型，需要的位数最少。**uint_fast16_t** 会为 MSP430 MCU 使用 16 位无符号整型，但会为 MSP432 MCU 使用 32 位无符号整型，因为在 MSP432 MCU 上，本机 32 位运算始终是最快的。另一方面，leastN 整型表示特定架构上可用的整型，其宽度至少为 N 位。

不仅 CPU 或低级代码要使用正确、稳健的数据类型，还应在所有驱动程序层级或软件库中使用此类数据类型。

4.2.2 MSP430 内核和 Cortex-M4 内核使用和内联函数

由于 MSP430 MCU 和 MSP432 MCU 使用不同的内核，因此其关联器件会使用不同的内核函数集合。一些内联函数共用通用功能，并提供内联函数转换层，因此可兼容现有的 MSP430 MCU 代码。msp_compatibility.h 文件可帮助将多个现有的 MSP430 MCU 内联函数转换为 MSP432 MCU 等效函数。Example 1定义了一些常用内联函数。

Example 1. MSP432 上的 MSP 内联函数及定义 (msp_compatibility.h)

```
#define __sleep()                       __wfi()
#define __deep_sleep()                  { (*((volatile uint32_t *)(0xE000ED10))) |=
0x00000004; __wfi(); (*((volatile uint32_t *)(0xE000ED10))) and= ~0x00000004; }
#define __low_power_mode_off_on_exit()  { (*((volatile uint32_t *)(0xE000ED10))) &=
~0x00000002; }
#define __get_SP_register()             __get_MSP()
#define __set_SP_register(x)            __set_MSP(x)
#define __get_interrupt_state()         __get_PRIMASK()
#define __set_interrupt_state(x)        __set_PRIMASK(x)
#define __enable_interrupt()            __asm("  cpsie i")
#define __enable_interrupts()           __asm("  cpsie i")
#define __disable_interrupt()           __asm("  cpsid i")
#define __disable_interrupts()          __asm("  cpsid i")
#define __no_operation()                __asm("  nop")
```

CMSIS 还提供一套适用于 Cortex-M4 内核的指令和功能内联函数（请参见Example 2）。

Example 2. cmsis_ccs.h 中定义的 MSP432 内联函数

```
__attribute__( ( always_inline ) ) static inline void __nop(void)
{
     __asm("  nop");
}

// Wait For Interrupt
__attribute__( ( always_inline ) ) static inline void __wfi(void)
{
     __asm("  wfi");
}

// Wait For Event
__attribute__( ( always_inline ) ) static inline void __wfe(void)
{
     __asm("  wfe");
}

// Enable Interrupts
__attribute__( ( always_inline ) ) static inline void __enable_irq(void)
{
     __asm("  cpsie i");
}

// Disable Interrupts
__attribute__( ( always_inline ) ) static inline void __disable_irq(void)
{
     __asm("  cpsid i");
}

// Data Synchronization Barrier
__attribute__( ( always_inline ) ) static inline void __DSB(void)
{
     __asm("  dsb");
}
```

对于 DriverLib 用户，还会提供 DriverLib API 来运用内联函数实现的各种功能。还可使用其他 API 执行一系列指令，以提供更多信息。这些 DriverLib API 位于 CPU 模块 (cpu.h) 或电源模块 (pcm.h) 中。

```
PCM_setPowerState();
PCM_shutdownDevice();
PCM_gotoSleep();
PCM_gotoDeepSleep();
```

更多关于 DriverLib API 的信息，请访问 www.ti.com/tool/mspdriverlib 并查看《*MSP432 DriverLib 用户指南*》（随 DriverLib 提供）。

4.2.3 中断系统

MSP430 MCU 和 MSP432 MCU 使用两个完全不同的中断系统。如果要在这两个平台之间实现软件移植，还要特别考虑中断配置和处理。

MSP430 MCU 上的中断系统大部分集成到 MSP430 内核中。中断系统和外设中断源直接连回到受单个 GIE 位（SR 寄存器的一部分）控制的主中断系统。

另一方面，MSP432 MCU 采用 Cortex-M4 集成式嵌套矢量化中断处理器 (NVIC)，该控制器从 CPU 内核取得对中断管理的更多控制，同时提供更大的灵活性，并 具有 优先级可配置、尾部连接高效以及外设中断单独控制等特性。

4.2.3.1 使能 MSP432 MCU 上的中断

从移植过程的角度来讲，最后一个功能是最主要的，除了要使能外设的各个中断触发之外，还需要使用额外的软件寄存并使能 NVIC 系统上的外设中断源，如图 1中所示。

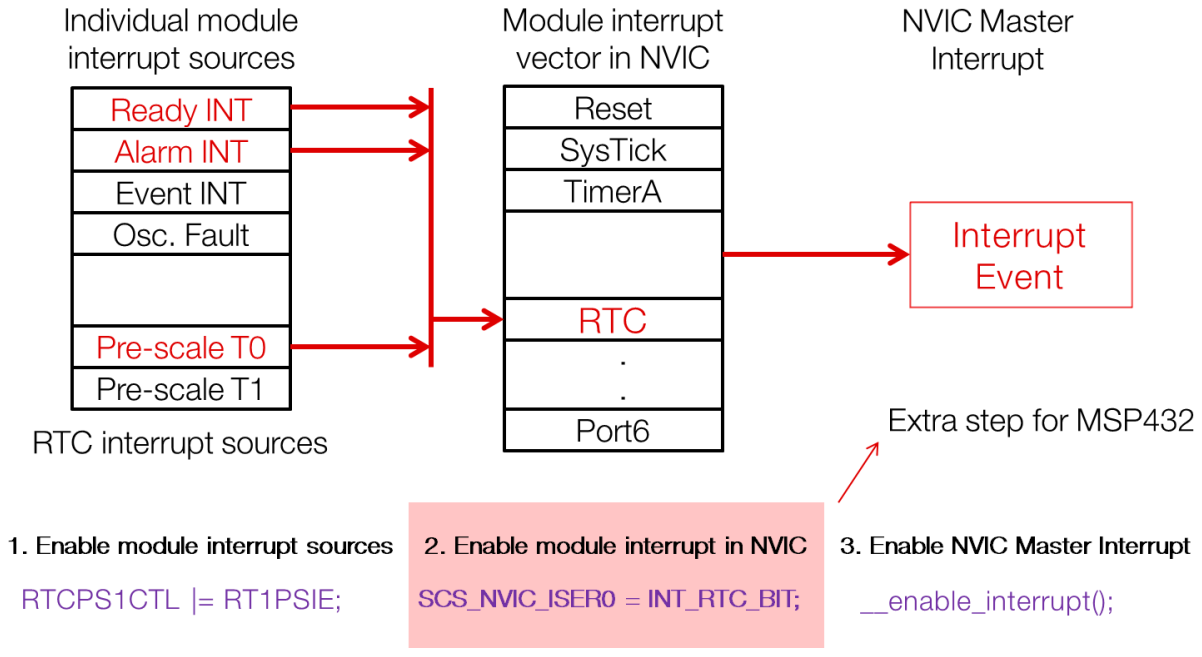


图 1. 使能 MSP432™ MCU 上的中断

请注意，这一附加步骤是在 NVIC 模块上完成的。也就是说，使用 DriverLib 时，用于该步的 NVIC 中断 API 为 `Interrupt_enableInterrupt(PERIPHERAL_INT);`。

4.2.3.2 寄存中断和中断矢量表

对于 MSP430 MCU，编译器通常通过检测前缀为关键字 **#pragma vector** 的特殊中断服务例程来管理中断矢量表的存储器空间分配、为未使用的中断矢量保留默认值以及寄存正在使用的中断矢量。ISR 函数也需要使用 *__interrupt* 关键字。

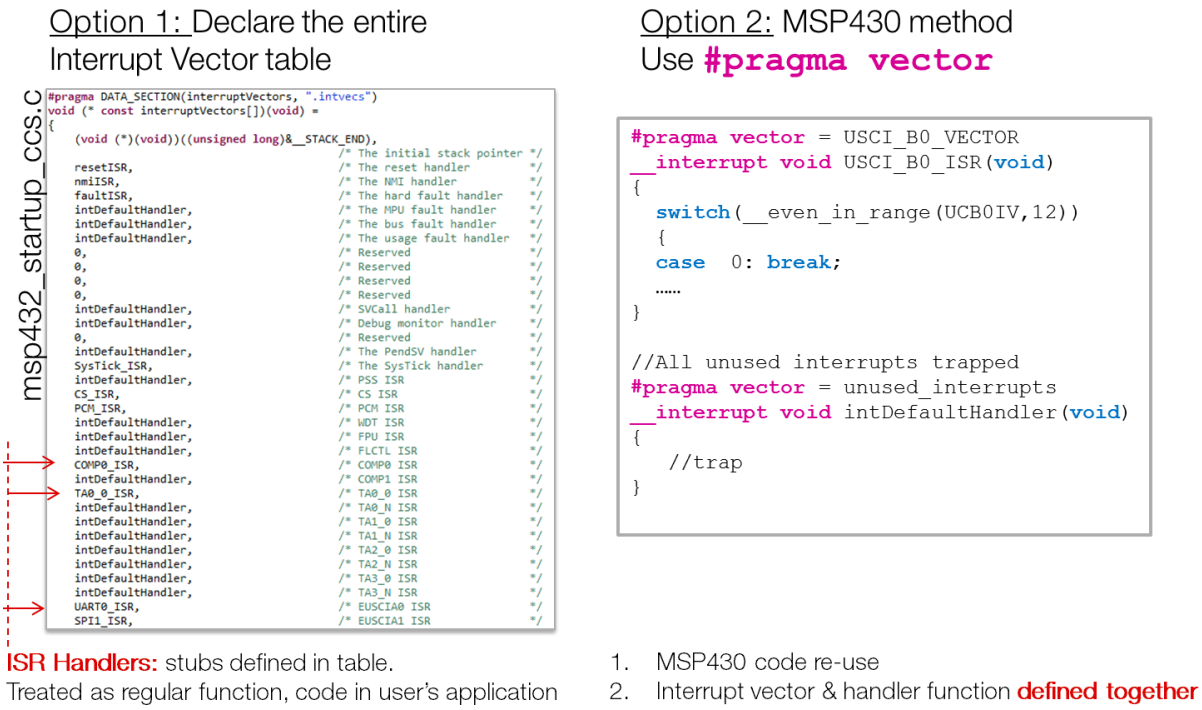
```
#pragma vector=ADC12_VECTOR
__interrupt void ADC12_ISR(void)
```

为 MSP432 MCU 定义中断矢量的默认方法是跨不同 Cortex-M 平台应用通用标准。整个中断矢量表定义为起始位置固定为 0x00000000 的数组。矢量表包含指向中断服务例程（定义为常规函数）地址的中断矢量地址。

注: 中增加了“应用功能”备注应用功能

目前正在开发 CCS 编译器，以便在 MSP432 上使能下文介绍的 MSP430 中断 #pragma 功能。该功能可用后，本文档将更新为包含支持此功能的第一个 CCS 和 TI 编译器版本。

为了保持 MSP 平台之间的兼容性，MSP432 MCU 还支持使用传统的 #pragma vector 方法来定义中断。为进行移植，仍可采用此方法重用 MSP430 MCU 中的现有中断代码。图 2显示了用于在 MSP432 MCU 上定义和分配中断矢量表的两个选项。



NOTE: 选项 2 当前不可用。更多相关信息，请参见“应用功能”备注（位于节 4.2.3.2中）。

图 2. 在 MSP432™ MCU 上定义中断服务例程 (ISR)

SLAA656 — http://www-s.ti.com/sc/techlit/SLAA656

4.2.4 中断和 LPM0 或 LPM3（休眠）模式

MSP430 CPU 与 MSP432 CPU 在架构上的区别之一是，MSP430 CPU 提供的原子指令允许用户使能或禁用全局中断并同时进入低功耗模式。在 MSP432 架构中这两个操作是独立的；因此，应用程序通常需要先使能或禁用主中断，然后再进入 LPM0（休眠）或 LPM3（深度休眠）模式。

在 MSP430 MCU 上，可使用以下原子指令使能中断并进入低功耗模式。

```
__bis_SR_register(LPM0_bits + GIE);
```

但在 MSP432 MCU 上，同样的过程需要两个单独指令来完成。因此该过程变为非原子过程。

```
__enable_irq();
__wfi();
```

上述 CMSIS 式内联函数会先通过禁用 PRIMASK 的方式使能中断，然后再使用等待中断指令进入 LPM0 模式。

请注意，MSP432 头文件还提供转换层来重新创建类似的内联函数，如 __enable_interrupts() 或 __sleep()，这些内联函数会模拟 MSP430 MCU 的行为，但其功能与 CMSIS 内联函数相同。

非原子操作给 MSP432 MCU 带来的最大差别是，如果在 enable_irq() 和 wfi() 质量之间发生待处理中断或异步中断，中断可能在器件进入 LPM0 或 LPM3 模式之前得到处理。如果该中断是唯一预期的中断，并且是器件从低功耗模式返回的唯一方法，应用程序需要确保器件不会永久停留在低功耗模式。在这种特有的情况中，可使用以下代码序列：

```
__disable_irq();
__wfi();
__enable_irq();
__ISB();
__disable_irq();
```

在这种情况中，如果因为使能了 PRIMASK 而导致主中断禁用，在存在待处理的中断时，__wfi() 命令将以 NOP 指令的形式执行（如果无效）。如果器件已执行了 __wfi() 命令，禁用了主中断的中断触发会使器件从 LPM0 或 LPM3 模式唤醒。最后三条指令基本上会立即使能中断模块，以确保中断得到处理。该序列的缺点是每次触发中断都会唤醒器件。开发人员将根据其应用需求、中断源以及低功耗模式配置确定出最优化、但安全有效的中断和低功耗模式序列。

MSP430 和 MSP432 MCU 的中断唤醒操作也稍有不同。在 MSP430 MCU 上，应用程序通常可使用内联函数 __bic_SR_Register_on_exit() 将 SR 寄存器中的相应位清零，指示 CPU 在中断服务例程 (ISR) 结束时唤醒。在 MSP432 Cortex-M CPU 上，应用程序可使用 SLEEPONEXIT 位将 CPU 配置为在任何中断服务例程结束时会唤醒还是返回到低功耗模式。特别是在 SLEEPONEXIT 置 1 的情况下，器件会在上一个待处理中断的 ISR 结束时自动重新进入低功耗状态。如果 SLEEPONEXIT 已清零，器件会在处理完上一待处理中断后唤醒并恢复 CPU 执行。

MSP432 DriverLib 结合了多个 API，可实现不同的中断和 LPM 操作，包括用于执行以上中断和 LPM 序列的 API。

4.2.5 存储器

虽然 MPS430 和 MSP432 MCU 均使用无分页统一存储器映射，但二者的存储器映射布局却有着显著区别。如果应用程序 有一些存储器组件或相关性（如数据记录或现场固件更新），则需要重建整体存储器布局。

图 3和图 4 中详细介绍了特定的组件布局。

Can generate NMI on read/write/fetch								
Generates PUC on fetch access								
Protectable for read/write accesses								
Always able to access PMM registers from(1); Mass erase by user possible								
Mass erase by user possible								
Bank erase by user possible								
Segment erase by user possible								
Address Range	**Name and Usage**	Properties						
00000h-00FFFh	Peripherals with gaps							
00000h-000FFh	Reserved for system extension							
00100h-00FEFh	Peripherals						x	
00FF0h-00FF3h	Descriptor type(2)						x	
00FF4h-00FF7h	Start address of descriptor structure						x	
01000h-011FFh	BSL 0	x				x		
01200h-013FFh	BSL 1	x				x		
01400h-015FFh	BSL 2	x				x		
01600h-017FFh	BSL 3	x			x	x		
017FCh-017FFh	BSL Signature Location							
01800h-0187Fh	Info D	x						
01880h-018FFh	Info C	x						
01900h-0197Fh	Info B	x						
01980h-019FFh	Info A	x						
01A00h-01A7Fh	Device Descriptor Table						x	
01C00h-05BFFh	RAM 16 KB							
05B80-05BFFh	Alternate Interrupt Vectors							
05C00h-0FFFFh	Program	x	x(1)	x				
0FF80h-0FFFFh	Interrupt Vectors							
10000h-45BFFh	Program	x	x	x				
45C00h-FFFFFh	Vacant							x(3)

图 3. MSP430™ MCU 存储器映射 (MSP430F548A)

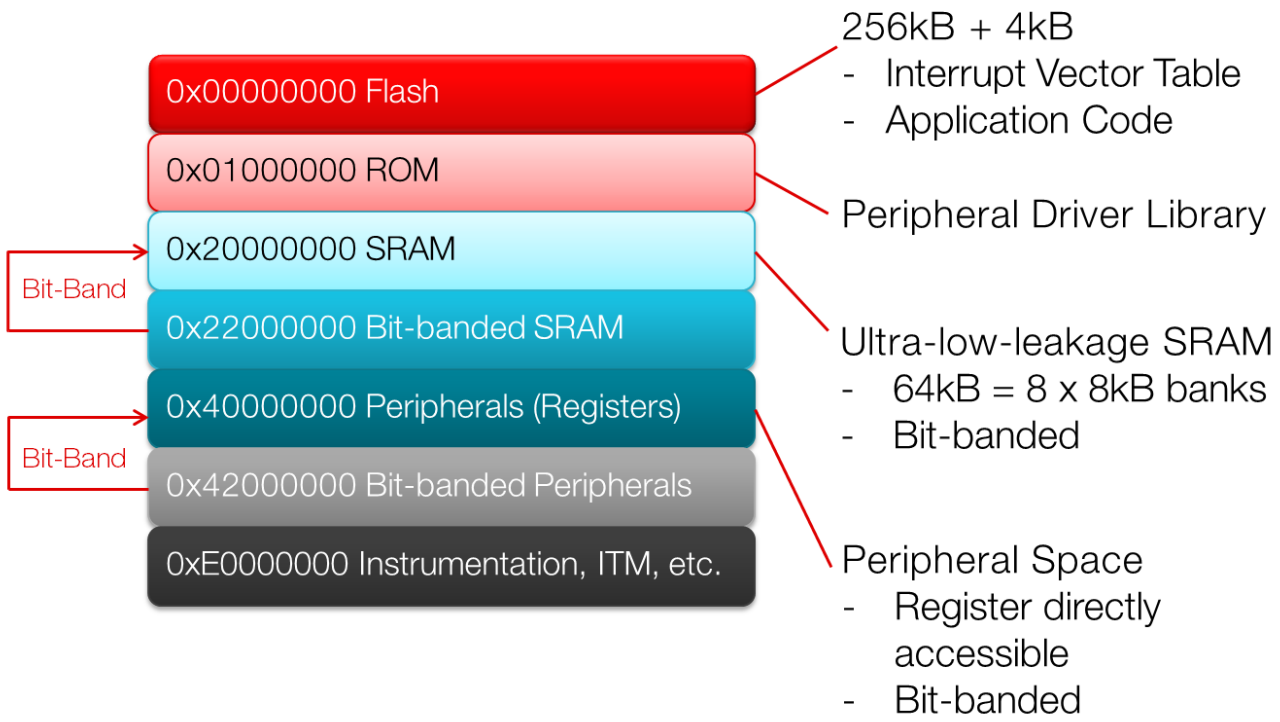


图 **4. MSP432™ MCU** 存储器映射 **(MSP432P401R)**

主要亮点包括：

- 中断矢量表
  - MSP430 MCU 起始地址为 0xFFFF（含复位矢量），并向下增长。
  - MSP432 MCU 起始地址为 0x0000（含复位矢量），并向上增大。
- 闪存、RAM 以及其他存储器布局：
  - MSP430 存储器组件共享 64KB 的空间（对于 16 位 CPU 器件）或 1MB 空间（对于支持 20 位运算的 CPUX），每个组件的地址有可能因器件而异。
  - MSP432 MCU 使用冗余 32 位地址空间为不同组件保留单独的存储器位置。
- MSP432 MCU 引入了位段功能，可补偿 Cortex-M 指令集的读-修改-写性质。
- 对 SRAM 或外设空间中各个位的操作，可利用位段功能将读-修改-写操作缩减为一条指令。DriverLib API 在对外设寄存器进行操作时使用位段功能。写入自己的寄存器访问代码时，应尽可能使用以下定义来利用位段。

Example 3. MSP432 位段定义

```
#define HWREGBIT8(x, b)
#define HWREGBIT16(x, b)
#define HWREGBIT32(x, b)
#define BITBAND_SRAM(x, b)
#define BITBAND_PERI(x, b)
```

4.3 核心系统软件

各器件系列的核心系统外设各不相同，可根据特定器件系列的用途提供最合适的功能，包括电源管理（Vcore、无 Vcore）、计时方案（DCO 或 FLL）或非易失性存储器类型（闪存、FRAM）。为此，对于从一个器件移植到另一系列或平台的移植代码，始终需要针对核心系统外设开发新代码。

表 8. MSP432 MCU 的新增核心系统外设

系统模块	兼容等级	寄存器访问代码	DriverLib API
电源、时钟、存储器	低	新开发	建议使用新的易用型 DriverLib API

可利用 DriverLib API 正确、高效地配置核心系统。这样可免去为核心模块重建和开发代码所占用的时间，但开发人员仍需要了解器件操作。深入理解配置核心系统（比如节 3.1.6中的核心系统）的外设间需求后，开发人员可以快速为电源、时钟和存储器模块设计出使用 DriverLib API 的配置例程，以满足核心配置的需求。

```
MAP_FlashCtl_setWaitState(FLASH_BANK0, 2);
MAP_FlashCtl_setWaitState(FLASH_BANK1, 2);
PCM_setPowerState(PCM_AM_DCDC_VCORE1);
MAP_CS_setDCOCenteredFrequency(CS_DCO_FREQUENCY_48);
```

此外，由于核心配置代码通常在应用程序开头的初始化代码中编写一次，与编写整个程序相比，需要的总软件数量相对较少，因此在需要编写新代码的情况下，如果使用 DriverLib API 和有用的代码示例，移植到新的核心外设系统所需的实际代码开发工作量是可管理的。其中主要的工作是了解新系统的运行方式并找出可以达到应用目标的最佳核心配置。更多关于如何利用核心系统新功能、 特别是 MSP432 MCU 新增的 独特电源系统的信息，请参见*《采用 SimpleLink™ MSP432™ 微控制器设计超低功耗 (ULP) 应用》*。

4.4 外设软件

4.4.1 MSP 共享外设

MSP 器件在设计过程中考虑了外设跨平台兼容的问题。MSP430 系列的很多高度集成、超低功耗外设均可在性能较高的 MSP432 系列中继续使用。从一个平台移植到另一平台时，这些外设保留其核心设计、功能以及对其进行访问和控制所采用的接口。从开发人员的角度来看，在各个器件系列和平台之间移植时，外设使用、寄存器和位定义以及头文件支持是完全相同的。从软件的角度来看，这意味着已为这些外设写入的代码仍可在新平台上运行并实现兼容，但需要注意节 4.2.3中介绍的每个外设的附加中断使能步骤。

表 **9.** 跨 **MSP** 平台的共享外设

共享 MSP 外设	兼容等级	寄存器访问代码	DriverLib API
Timer_A、eUSCI、REF_A、COMP_E、WDT_A、RTC_C、GPIO、AES256	高	兼容，检查本机数据类型和寄存器宽度（16 位）	兼容，检查本机数据类型和 BASE_ADDRESS 使用情况

Example 4和Example 5 分别举例介绍了 MSP430 和 MSP432 MCU 中针对 Timer_A 的高度兼容外设寄存器访问代码。

Example 4. MSP430 Timer_A 寄存器访问代码

```
P1DIR |= 0x01;                // P1.0 output
TA0CCTL0 = CCIE;             // CCR0 interrupt enabled
TA0CCR0 = 50000;
TA0CTL = TASSEL_2 | MC_1 | TACLR;      // SMCLK, upmode, clear TAR
```

Example 5. MSP432 Timer_A 寄存器访问代码

```
P1DIR |= 0x01;                // P1.0 output
TA0CCTL0 = CCIE;             // CCR0 interrupt enabled
TA0CCR0 = 50000;
TA0CTL = TASSEL_2 | MC_1 | TACLR;      // SMCLK, upmode, clear TAR

Add the line to enable nvic for timer  // Enable interrupts
```

Example 6和Example 7 分别演示了用于配置 MSP430 和 MSP432 MCU 上串行通信外设 USCI 所使用的高度兼容驱动程序库 API。请注意，二者之间唯一的区别是时钟模块 API。除了现有的 MSP430 基址定义（USCI_B0_BASE 和 USCI_B0）之外，MSP432 驱动程序库还引入了新的基址参数定义。每一种定义均可用于选择指定的外设实例。

SLAA656 — http://www-s.ti.com/sc/techlit/SLAA656

Example 6. MSP430 USCI DriverLib 代码

```
// Initialize Master
USCI_B_I2C_masterInit(USCI_B0_BASE, USCI_B_I2C_CLOCKSOURCE_SMCLK,
           UCS_getSMCLK(UCS_BASE), USCI_B_I2C_SET_DATA_RATE_400KBPS);

// Specify slave address
USCI_B_I2C_setSlaveAddress(USCI_B0_BASE, SLAVE_ADDRESS);

// Set Master in receive mode
USCI_B_I2C__setMode(USCI_B0_BASE, USCI_B_I2C_RECEIVE_MODE);

// Enable I2C Module to start operations
USCI_B_I2C_enable(USCI_B0_BASE);
```

Example 7. MSP432 eUSCI DriverLib 代码

```
// Initialize Master
USCI_B_I2C_masterInit(USCI_B0_BASE, USCI_B_I2C_CLOCKSOURCE_SMCLK,
           CS_getSMCLK(), USCI_B_I2C_SET_DATA_RATE_400KBPS);

// Specify slave address
USCI_B_I2C_setSlaveAddress(USCI_B0_BASE, SLAVE_ADDRESS);

// Set Master in receive mode
USCI_B_I2C__setMode(USCI_B0_BASE, USCI_B_I2C_RECEIVE_MODE);

// Enable I2C Module to start operations
USCI_B_I2C_enable(USCI_B0_BASE);
```

概括来讲，为 MSP430 和 MSP432 移植共享外设代码需要的工作量很小。但开发人员务必根据平台的中断系统考虑额外的中断代码，并应确保使用正确的数据类型，最好是使用显式类型。即使是在同系列的器件之间进行移植，也需要额外执行的移植步骤能够确保外设使用的端口引脚更新到新器件上。

4.4.2 扩展外设

增强型外设可使用修订字母（如 REF、REF_A、REF_B）进行标识。如果是模拟模块，可使用分辨率进行标识，比如 ADC 模块：ADC12、ADC12_A、ADC12_B、ADC14。MSP432P4xx 系列引入了 14 位 ADC14 模块作为 ADC12_B 的扩展增强版本，除了其他数字方面的增强之外，该模块还具有 14 位的分辨率和 1Msps 的采样速率。

表 10. 扩展外设的兼容等级

扩展外设和新修订外设	兼容等级	寄存器访问代码	DriverLib API
ADC14	中到高	部分兼容，查看不同的寄存器宽度（16 位到 32 位）、小位重定位，新标志清零机制。	部分兼容，视为新 API，以及现有 API 的参数。检查数据类型和 BASE_ADDRESS 使用情况。

这类逐渐改造的外设通常会保留之前进行过修改的共享功能，因此会在头文件中包含相同的寄存器和位定义。某些情况下，为了实现增强功能，可能会对少数寄存器位进行移位 或增加。因此，可按照针对共享外设的说明采用类似的方法重用一些现有的寄存器访问代码（参见节 4.4.1）。另一方面，新 功能 引入了新的寄存器和位定义。开发人员有两种选择：使用寄存器访问代码或使用新的 MSP432 DriverLib API。对于这两种方法，除了 利用 《MSP432P4xx SimpleLink™ 微控制器技术参考手册》中的规范和说明之外，开发人员还可以利用以寄存器访问格式和 DriverLib 格式编写的各种器件代码示例。

Example 8. MSP432 ADC14 DriverLib API

```
// ADC14 DriverLib
// APIs similar to MSP430 ADC12_A/B APIs but with new namespace or updated parameters
extern bool ADC14_initModule(uint32_t clockSource, uint32_t clockPredivider,
        uint32_t clockDivider, uint32_t internalChannelMask);
extern void ADC14_setResolution(uint32_t resolution);
extern bool ADC14_configureConversionMemory(uint32_t memorySelect,
        uint32_t refSelect, uint32_t channelSelect, bool differntialMode);
extern bool ADC14_enableComparatorWindow(uint32_t memorySelect,
        uint32_t windowSelect);
```

概括来讲，扩展外设通常会在功能和软件两方面保留一定程度的兼容性。TI 建议检查现有的代码，以确保功能得到保留。要完全实现新外设的 功能 ，可能还需要额外添加新的代码。此类移植和新代码开发工作可借助寄存器访问和 DriverLib API 中提供的器件代码示例完成。

4.4.3 新外设和 Cortex-M 外设

在 MSP432 平台上，最先引入到 MSP 配置文件的最后一类外设包括属于 Cortex-M4F 内核组成部分的新 Cortex-M 外设，或 Cortex-M 外设集合中具有集成和性能优势的外设。

表 11. Cortex-M 外设

Cortex-M 外设	兼容等级	寄存器访问代码	DriverLib API
SysTick，Timer32	无	新开发，简单	新，简单 API
DMA	无	可行，但非常复杂	强烈建议使用 API
FPU	无	简单，内置编译器支持	简单，内置编译器支持

由于这些外设可由 ARM 提供，因此用 CMSIS 定义编写的软件（除 MSP 寄存器访问和 DriverLib API 之外）也可使用。

FPU 这一模块的软件需求也比较低，因为最新的 ARM 编译器内置的支持功能可控制 FPU 执行代码中的所有浮点计算。

另一方面，如果需要创建新代码，µDMA 等模块的要求相对较高。将 MSP430 平台上现有的 DMA 代码移植到 MSP432 上的新 µDMA 模块时，TI 强烈建议开发人员考虑使用为模块编写的 DriverLib API 以及各种代码示例，以便快速了解如何正确配置和使用该模块。预期会进行大规模移植，因为需要通过系统级考量来重建 DMA 配置和使用。更多关于 DMA 模块的信息，请参见《MSP432P4xx SimpleLink™ 微控制器技术参考手册》和代码示例。

4.4.4 传统 C 代码和寄存器访问

利用位段实现单个寄存器位操作。检查应用程序中的 8 位、16 位和 32 位寄存器访问。

4.4.5 外设驱动程序库

MSP430 和 MSP432 DriverLib 为两个平台上可用的外设共享一些通用 API。但 MSP432 DriverLib 在函数调用、参数传递以及如何使用多个实例处理外设方面稍微做了一些改动。

4.4.6 其他外设操作软件

除了利用寄存器访问代码和 MSP DriverLib API 之外，开发人员还可以利用第三方开发者、社区发起的项目或 ARM 生态系统提供的其他外设驱动程序软件。移植该软件的策略仍类似于使用传统寄存器访问代码或 TI 提供的 MSP Driver Library 的情况。

CMSIS 是 ARM 提供的编码标准。除 CMSIS 外，各家芯片厂商还定义并开发出多种软件组件。除了常规的头文件和支持文件之外，MSP432 还提供 CMSIS 式器件头文件供 CMSIS 用户使用。该方法可利用 CMSIS 标准进行代码开发，但使用 CMSIS 从 MSP430 移植到 MSP432 的工作要求稍高一些。

4.4.7 应用代码和软件库

检查数据类型。

检查实时特性。

提取代码，将与核心相关的代码与更高的抽象层隔离。

5 工具与生态系统

MSP432 器件是 SimpleLink MCU 平台的组成部分，该平台包括 Wi-Fi、低功耗蓝牙、1GHz 以下器件和主机 MCU。它们都共用配有单核 SDK 和丰富工具集的通用、易用型开发环境。只需进行一次 SimpleLink 平台集成，便可将配置文件中的任意器件组合添加到设计中。SimpleLink 平台的最终目标是确保设计要求变更时，完全重复使用代码。更多详细信息，请访问 www.ti.com/simplelink。

MSP432 MCU 由具有 Cortex-M 功能的调试器提供支持，包括 TI 的 XDS200、XDS100v2/3、XDS110、IAR I-jet 和 SEGGER J-LINK。更多关于受支持调试器以及如何在调试 MSP432 MCU 时利用这些工具的信息，请参见*《MSP432™ SimpleLink™微控制器硬件工具用户指南》*。

MSP432 MCU 代码开发可在与 MSP430 MCU 相同的集成开发环境 (IDE) 中进行，包括 Code Composer Studio™IDE、 IAR Embedded Workbench®IDE 和 gcc。兼容工具链还提供兼容头文件、支持软件包、代码示例以及软件库，以最大限度地在 MSP430 MCU 和 MSP432 MCU 平台上提供相似的体验。

除了通用的 IDE 之外，还支持在 Keil® µVision®MDK 中使用 MSP432 MCU。从 MSP430 MCU 移植到 MSP432 MCU 涉及的工作量可能比较大，但开发人员可在其 MSP432 MCU 代码开发过程中利用 ARM 软件生态系统，包括 CMSIS 软件的各个层级。

6 移植示例和分析

以下步骤概括了将 MSP430F5529 MCU 上的现有应用移植到 MSP432P401R MCU 的过程。

1. 核心模块配置
   (a) 设置电源和 VCore
   (b) 根据系统频率设置闪存等待状态
   (c) 设置 DCO 频率
2. 外设代码：三类 **MSP432** 代码
   (a) 移植 Timer_A 模块的寄存器访问代码
   (b) 将 ADC12 模块的 DriverLib API 移植到 ADC14 模块
   (c) 使用 CMSIS 移植 RTC 模块模式
3. 与中断相关的配置代码
   (a) 将使用寄存器访问和 DriverLib API 执行的模块中断使能指令
   (b) 在要执行指令的 NVIC 上使能模块中断
   - 使用 DriverLib API：针对 ADC14 使用 Interrupt_enableInterrupt
   - 使用寄存器访问：针对 RTC 使用 SCS_NVIC_ISERx 寄存器

   (c) 使能 NVIC 主中断（等同于 MSP430 MCU 上的 GIE 位）
   - 使用 DriverLib API
   - 使用 MSP432 内联函数
4. 将 **ISR** 桩线添加到中断矢量表
5. 内核、**CPU** 和系统相关的函数
   (a) 如何进入 LPM0（休眠）模式
   (b) 如何进入 LPM3（深度休眠）模式
   (c) 控制通过 ISR 唤醒的时间和方式
   (d) 控制通过 ISR 返回休眠模式的时间和方式
6. 将所有片段联系在一起并进行调试

7 参考资料

1. *《采用 SimpleLink™ MSP432™ 微控制器设计超低功耗 (ULP) 应用》*
2. *《MSP432P4xx SimpleLink™ 微控制器技术参考手册》*
3. *《MSP432™ SimpleLink™ 微控制器硬件工具用户指南》*
4. *《MSP432P401R、MSP432P401M SimpleLink™ 混合信号微控制器》*

修订历史记录

注：之前版本的页码可能与当前版本有所不同。

Changes from March 16, 2016 to March 8, 2017 **Page**

- 更改了本文档的标题 1
- 添加了本文档的作者 1
- 更新了摘要 1
- 更改了1 节，*SimpleLink MSP432 平台移植概述*的标题并更新了目录 2
- 更改了节 3.1.6，*核心系统相关性及配置*中 32MHz 至 48MHz 对应的闪存等待状态列 5
- 删除了原有的表 5，*不同移植等级的软件组件* 5
- 新增了第二段和表 5，*软件组件*（位于4.1 节，*软件移植概述*中） 5
- 更改了表 6，*将软件组件从 MSP430Ware 移植到 SimpleLink SDK* 的标题，并更改了右下方表中的单元格内容 6
- 将节 4.2.2，*MSP430 内核和 Cortex-M4 内核使用和内联函数*中的 msp430dna.h 改为 sp_compatibility.h 7
- 更改了5 节，*工具与生态系统*中的第一段并添加了 XDS110 18
- 根据需要更新了参考文档的标题 19

有关 TI 设计信息和资源的重要通知

德州仪器 (TI) 公司提供的技术、应用或其他设计建议、服务或信息，包括但不限于与评估模块有关的参考设计和材料（总称“TI 资源”），旨在帮助设计人员开发整合了 TI 产品的 应用； 如果您（个人，或如果是代表贵公司，则为贵公司）以任何方式下载、访问或使用了任何特定的 TI 资源，即表示贵方同意仅为该等目标，按照本通知的条款进行使用。

TI 所提供的 TI 资源，并未扩大或以其他方式修改 TI 对 TI 产品的公开适用的质保及质保免责声明；也未导致 TI 承担任何额外的义务或责任。TI 有权对其 TI 资源进行纠正、增强、改进和其他修改。

您理解并同意，在设计应用时应自行实施独立的分析、评价和 判断， 且应全权负责并确保 应用的安全性， 以及您的 应用 （包括应用中使用的所有 TI 产品）） 应符合所有适用的法律法规及其他相关要求。你就您的 应用声明，您具备制订和实施下列保障措施所需的一切必要专业知识，能够 (1) 预见故障的危险后果，(2) 监视故障及其后果，以及 (3) 降低可能导致危险的故障几率并采取适当措施。您同意，在使用或分发包含 TI 产品的任何 应用前， 您将彻底测试该等 应用 和该等应用所用 TI 产品的 功能。除特定 TI 资源的公开文档中明确列出的测试外，TI 未进行任何其他测试。

您只有在为开发包含该等 TI 资源所列 TI 产品的 应用时， 才被授权使用、复制和修改任何相关单项 TI 资源。但并未依据禁止反言原则或其他法理授予您任何TI知识产权的任何其他明示或默示的许可，也未授予您 TI 或第三方的任何技术或知识产权的许可，该等产权包括但不限于任何专利权、版权、屏蔽作品权或与使用TI产品或服务的任何整合、机器制作、流程相关的其他知识产权。涉及或参考了第三方产品或服务的信息不构成使用此类产品或服务的许可或与其相关的保证或认可。使用 TI 资源可能需要您向第三方获得对该等第三方专利或其他知识产权的许可。

TI 资源系“按原样”提供。TI 兹免除对 TI 资源及其使用作出所有其他明确或默认的保证或陈述，包括但不限于对准确性或完整性、产权保证、无屡发故障保证，以及适销性、适合特定用途和不侵犯任何第三方知识产权的任何默认保证。

TI 不负责任何申索，包括但不限于因组合产品所致或与之有关的申索，也不为您辩护或赔偿，即使该等产品组合已列于 TI 资源或其他地方。对因 TI 资源或其使用引起或与之有关的任何实际的、直接的、特殊的、附带的、间接的、惩罚性的、偶发的、从属或惩戒性损害赔偿，不管 TI 是否获悉可能会产生上述损害赔偿，TI 概不负责。

您同意向 TI 及其代表全额赔偿因您不遵守本通知条款和条件而引起的任何损害、费用、损失和/或责任。

本通知适用于 TI 资源。另有其他条款适用于某些类型的材料、TI 产品和服务的使用和采购。这些条款包括但不限于适用于 TI 的半导体产品 (http://www.ti.com/sc/docs/stdterms.htm)、评估模块和样品 (http://www.ti.com/sc/docs/sampterms.htm) 的标准条款。

邮寄地址：上海市浦东新区世纪大道 1568 号中建大厦 32 楼，邮政编码：200122