MSP430 FAQ

1. 我无法在器件数据表中找到与 MSP430 器件外设模块相关的信息，在哪里可找到这些信息？

基本上，MSP430 器件有 3 个主要文档：

- 器件数据表：包含器件专用信息，诸如器件上可用的外设列表、存储器组织结构、电气特性等。
- 系列用户指南文档：包含与 MSP430 器件系列（例如 1xx，2xx，4xx，5xx/6xx）的内部模块（CPU，外设等）相关的通用信息。
- 勘误表：包含错误说明列表，以及封装标记。 这基本上是主要参考文档，其中包括了不同版本产品的差异性。

2. 如何从封装标记中读取芯片的修订版本？

可在器件专用勘误表中找到从封装标记中读取芯片修订版本的相关信息（连同诸如年月日代码、批次追踪代码和组装地点代码等其他信息）。

3. 在 MSP430 器件上有任何诸如器件 ID 的信息吗？

请参考《JTAG 编程用户指南》 tidoc:slau320，有一个标题为“整个器件系列的 JTAG 特性”的表格。 这个表格包含所有 MSP430 器件的器件 ID。 之前 1xx/2xx/4xx 器件上的器件 ID 并不是每个器件所特有，而是专门针对每个子系列产品（例如，所有 MSP430F13x，MSP430F14x 和 MSP430F14x1 具有一样的器件 ID 0xF1 和 0x49）。

在 MSP430F5xx/6xx/FRxx 器件中，可通过使用“器件描述符表”中的“芯片记录”字段来创建一个唯一的器件 ID，此表格通常位于器件专用数据表的末尾。 以下链接显示了器件描述符表 MSP430：

Table 65. Device Descriptor Table MSP430FR59xx[1]

	Description	MSP430FR59xx	
		Address	Value
Info Block	Info length	01A00h	06h
	CRC length	01A01h	06h
	CRC value	01A02h	per unit
		01A03h	per unit
	Device ID	01A04h	see Table 64
		01A05h	
	Hardware revision	01A06h	per unit
	Firmware revision	01A07h	per unit
Die Record	Die Record Tag	01A08h	08h
	Die Record length	01A09h	0Ah
	Lot/Wafer ID	01A0Ah	per unit
		01A0Bh	per unit
		01A0Ch	per unit
		01A0Dh	per unit
	Die X position	01A0Eh	per unit
		01A0Fh	per unit
	Die Y position	01A10h	per unit
		01A11h	per unit
	Test results	01A12h	per unit
		01A13h	per unit
ADC12 Calibration	ADC12 Calibration Tag	01A14h	11h
	ADC12 Calibration length	01A15h	10h
	ADC Gain Factor[2]	01A16h	per unit
		01A17h	per unit

4. 我在哪里可以找到 MSP430 应用说明列表？

请参考这一 链接。

5. 有 MSP430 在线培训吗？

有的，请查阅 此处。

6. 用哪个算法计算 5xx/6x 器件上的 TLV 校验和？

使用的算法是具有以下参数的 CRC_CCITT：

- 初始值（种子值）： 0xFFFF
- 多项式： 0x1021
- 间接： 假
- 反向数据： 假
- 最终 XOR 之前的反向 CRC： 假
- 最终 XOR 值： 0x0

CRC 的地址范围为 0x1A04 – 0x1AFF。

7. MSP430F471xx INFOA 存储器上提供校准数据吗？

不提供。数据表并未明确指出这一点，但是生产后未在 MSP430F471x 的 INFOA 内传送校准数据。

8. 有任何与 MSP430 器件可靠性相关的信息吗？

请参考 TI 可靠性估算器。

9. 如何在具有 USB 接口的 MSP430 器件上分配 USB VID（供应商 ID）和 PID（产品 ID）。

基本上，具有 USB 接口的 MSP430 器件的 VID 和 PID 号不是“固化”在硬件中，而是由免费且开源的 USB 堆栈软件指定。 请参考 MSP430USBDEVPACK 来下载 USB 软件堆栈和 USB 描述符工具，此工具被用来生成包含 USB 描述符在内的 USB 堆栈所需要的配置信息（其中包括 VID 和 PID）的头文件。

TI 为客户提供使用 TI USB VID（供应商 ID）配合客户的独特 PID 的可能性。 在以下链接中发送“VID 分配计划”请求： http://software-dl-1.ti.com/dsps/forms/vidtracker.html。

10. 如何获得与 MSP430 全新器件路线图相关的信息？

与全新器件路线图相关的信息并未公开发布。 请联系 TI 当地销售办公室 或 TI 授权分销商 来获得这些信息。

11. 迁移指南

下面是 MSP430 系列器件间的迁移指南列表

迁移为	原先为	链接	注释
MSP430FR58xx， MSP430FR59xx	MSP430F2xx， MSP430G2xx	tidoc:slaa559	
MSP430FR58xx， MSP430FR59xx	MSP430F5xx， MSP430F6xx	tidoc:slaa555	
MSP430FR57xx	MSP430F2xx	tidoc:slaa499	
MSP430F5xx	MSP430F2xx， MSP430F4xx	tidoc:slaa396	
MSP430F541xA/F543xA	MSP430F541x/F543x	tidoc:slaa419	
MSP430F21x2	MSP430F12x(2)	tidoc:slaa421	
MSP430F13x/14x	MSP430F23x/24x	tidoc:slaa381	
MSP430F16x	MSP430F261x	tidoc:slaa380	
MSP430F42x	MSP430F42xA		器件是硬件（引脚到引脚）和软件兼容的（可通过比较 CCS/IAR 头文件查看；也许只需在 IDE 项目中更改器件类型并重新编译）。 只是硬件参数有所不同（请参考器件数据表 tidoc:slas241）

MSP430F11x1	MSP430F11x1A		器件是硬件（引脚到引脚）和软件兼容的（可通过比较 CCS/IAR 头文件查看；也许只需在 IDE 项目中更改器件类型并重新编译）。 只是硬件参数有所不同（请参考器件数据表 & 勘误表 tidoc:slas587， tidoc:421）

工具和编程

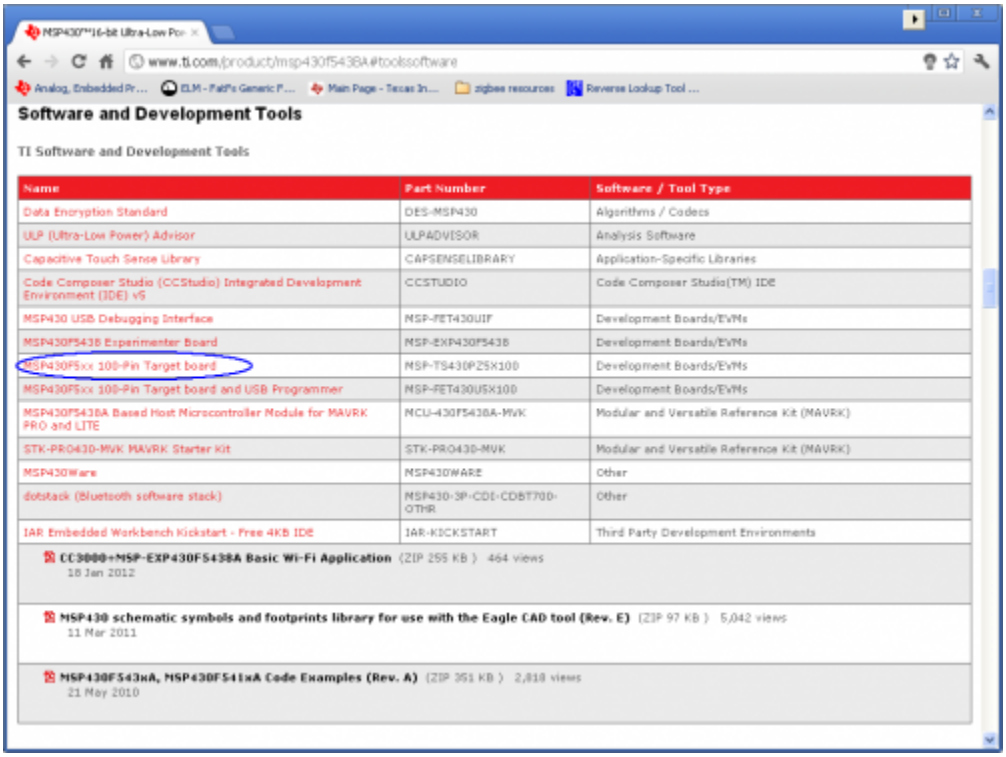
12. 我是 MSP430 的初学者，我如何用更加高效和快速的方法来开发我的应用？

如果你正在使用 C 语言进行编程（现在很常见），在开始使用全新微控制器平台时最困难的是了解外设。CPU 本身不是问题，这是因为代码由 C 语言编写。因此，研究 TI 提供的可能性，使你在使用这里的 MSP430 外设时更加轻松： MSP430 软件，其中包括：

- 示例代码： TI 提供很多针对每个 MSP430 器件的示例代码
- GRACE：用来设置/初始化 MSP430 外设的图形用户界面
- MSP430ware：所有示例代码的扩展集、驱动程序库（用于 5xx，6xx 和 FRAM 器件）、针对所有 MSP430 器件的用户指南。

13. TI 是否提供针对我的 MSP430 的开发套件/电路板？

TI 提供针对所有 MSP430 器件的开发套件，但是 **并不在** 所有封装中都提供。 通常情况下，可以在如下显示的器件产品网页上的“软件和开发工具”部分内找到开发套件。



14. 哪个 MSP430 目标器件为我的 FET（闪存仿真工具）提供支持？

这些信息可在《MSP430 硬件工具用户指南》中找到 (tidoc:slau278)，如下所示：

Table 1-1. Flash Emulation Tool (FET) Features and Device Compatibility[1]

	eZ430-F2013	eZ430-RF2500	eZ430-RF2480	eZ430-RF2560	MSP-WDSxx Metawatch	eZ430-Chronos	MSP-FET430PIF	MSP-FET430UIF	LaunchPad (MSP-EXP430G2)	MSP-EXP430FR5739	MSP-EXP430F5529
Supports all programmable MSP430 and CC430 devices (F1xx, F2xx, F4xx, F5xx, F6xx, G2xx, L092, FR57xx, FR59xx, MSP430TCH5E)							x	x			
Supports only F20xx, G2x01, G2x11, G2x21, G2x31	x										
Supports MSP430F20xx, F21x2, F22xx, G2x01, G2x11, G2x21, G2x31, G2x53									x		
Supports MSP430F20xx, F21x2, F22xx, G2x01, G2x11, G2x21, G2x31		x	x								
Supports F5438, F5438A				x							
Supports BT5190, F5438A					x						
Supports only F552x											x
Supports FR57xx, F5638, F6638										x	
Supports only CC430F613x						x					
Allows fuse blow								x			
Adjustable target supply voltage								x			
Fixed 2.8-V target supply voltage							x				
Fixed 3.6-V target supply voltage	x	x	x	x	x	x			x	x	x
4-wire JTAG							x	x			
2-wire JTAG[2]	x	x	x	x	x	x		x	x	x	x
Application UART		x	x	x	x	x			x	x	x
Supported by CCS for Windows	x	x	x	x	x	x	x	x	x	x	x
Supported by CCS for Linux								x			
Supported by IAR	x	x	x	x	x	x	x	x	x	x	x

(1) The MSP-FET430PIF is for legacy device support only. This emulation tool will not support any new devices released after 2011.
(2) The 2-wire JTAG debug interface is also referred to as Spy-Bi-Wire (SBW) interface.

基本上，上面的这个列表显示 FET 工具和目标器件（由 TI 提供品质保证和支持）间的相互关系。 这意味着，基本上可以使用一个 FET 工具来编辑上面列表中未列出的其他器件，但是在这个情况下，万一此工具不起作用时，TI 将不提供支持或排错。

15. 我如何用 CCSTUDIO 或 IAR 生成 TI TXT 输出文件？

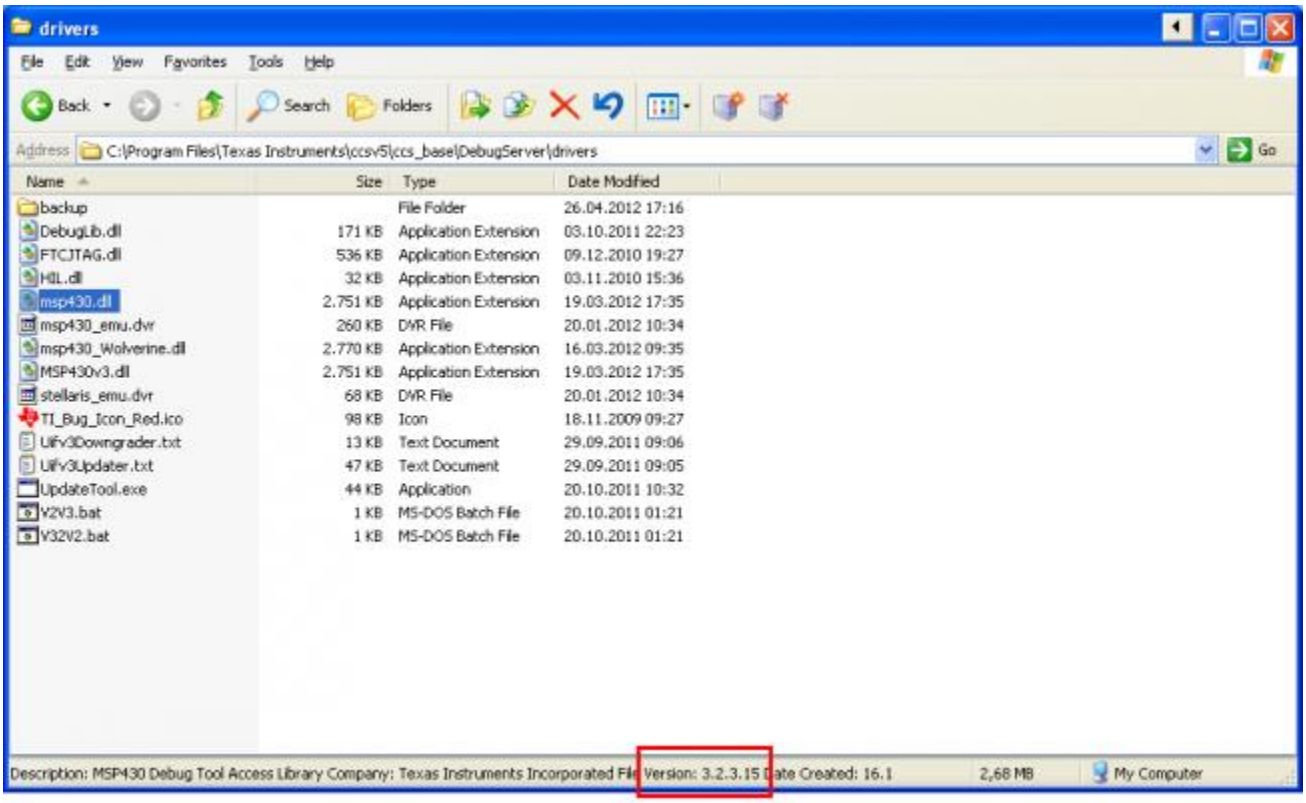
请参考以下维基网页： 生成_并_加载_MSP430_二进制_文件

16. 哪个软件可被用来将二进制（例如 TI TXT）文件下载/迅速存储到我的 MSP430 目标器件中？

使用 MSP430 Flasher 或者 Elprotronic FET Pro-430 Lite 软件 来下载/快速存储/编辑 MSP430 目标器件。这两款软件都是免费的。

17. 我的 IDE (IAR/CCS) 中使用哪个版本的 MSP430.DLL？

使用视窗浏览器，打开这里提及的包含 DLL 文件的缺省文件夹： MSP430/HIL DLL 缺省文件夹，并单击 DLL 文件。 此信息应该被如下显示在视窗浏览器的底部：



18. MSP430 器件的推荐 JTAG 引脚分配是什么样的？

请参考以下 维基网页。

19. 我在哪里能够找到具有 JTAG 类型（4 线制或 2 线制）的 MSP430 器件的列表？

这些信息可经由 JTAG 用户指南在 MSP430 编程中找到： tidoc:slau320，表 1-14“整个器件系列的 JTAG 特性”。

20. TI 是否为批量生产提供 MSP430 工具编辑器？

是的，请参见 MSP-GANG。

21. MSP430 JTAG 与 IEEE 1149.1 间的兼容性如何？

MSP430 JTAG 接口执行由 IEEE 标准 1149.1 规定的测试访问端口状态机（TAP 控制器）。 然而，有一些对于 MSP430 JTAG 的限制（不符合 IEEE 标准 1149.1）：

- MSP430 必须是 JTAG 链中的第一个器件（这是因为通过 TDI 和 JTAG 熔丝检查序列计时）。
- 没有 MSP430 器件具有边界扫描单元
- 只支持 BYPASS 指令。 不支持 SAMPLE，PRELOAD，或 EXTEST 指令。
- JTAG 引脚与特定器件上的端口功能共用；由 TEST 引脚控制 JTAG 功能。

22. 我在哪里能找到针对 MSP 器件的 BSDL（边界扫描描述语言）？

由于 MSP430 JTAG 与 IEEE 1149.1 不是 100% 兼容，所以它不支持边界扫描。 请参见 #MSP430 JTAG 与 IEEE 1149.1 间的兼容性如何？。

23. MSP-GANG430 使用哪个校验和算法来验证存储器内容？

MSP-GANG430 使用下面显示的 PSA（伪签名分析）：

```
for (PSA = StartAddr - 2, i = 0; i < Length; i++)
{
  if (PSA & 0x8000)
    PSA = ((PSA ^ 0x0805) << 1) | 1;
  else
    PSA <<= 1;

  PSA ^= Data[i];
}
```

24. 我如何编译 BSL 脚本解释器和 SLAU319 中的 BSLDEMO2 源代码？

从 SLAU319 的版本 E 开始 (tidoc:SLAU319)，源代码与 Microsoft Visual Studio 项目文件一同交付。

25. MSP430F54xx（非 A）器件有 SYS4 错误，但是我仍然可以擦除且重新编辑 BSL。 这怎么可能？

擦除或写覆盖 MSP430F54xx（非 A）器件的 BSL 在技术方面都是可能的，但是不建议这么做，这是因为有些错误会使得 F5438 非主存储器闪存中的代码执行不可靠。 非常详细的工作区曾经被用于 F5438 BSL 执行。 在大多数时间里，不可能从 F5438 中的非主闪存中成功执行代码。

26. 如何在 CCSTUDIO 中找到 MSP430 应用的存储器大小？

缺省情况下，当 CCSTUDIO 已经成功编译代码时，它将生成一个 MAP 文件（缺省情况下，在“调试”文件夹下，名称为 <PROJECT_NAME>.map）。 在 MAP 文件内，有一个存储器段列表，连同与已使用和未使用存储器大小相关的信息。 这些存储器段主要源自链接器命令文件 (lnk_msp430xxxxx.cmd)。 计算存储器大小并未考虑从堆存储器中动态分配的存储器（例如，使用 malloc() 函数）。

以下示例取自针对 MSP430G2553 的简单闪烁 LED 的 MAP 文件：

存储器配置

名称	源	长度	已使用	未使用	属性	填充
SFR	00000000	00000010	00000000	00000010	RWIX	
PERIPHERALS_8BIT	00000010	000000f0	00000000	000000f0	RWIX	
PERIPHERALS_16BIT	00000100	00000100	00000000	00000100	RWIX	
RAM	00000200	00000200	00000050	000001b0	RWIX	
INFOD	00001000	00000040	00000000	00000040	RWIX	
INFOC	00001040	00000040	00000000	00000040	RWIX	
INFOB	00001080	00000040	00000000	00000040	RWIX	
INFOA	000010c0	00000040	00000000	00000040	RWIX	
FLASH	0000c000	00003fe0	000000b2	00003f2e	RWIX	
INT00	0000ffe0	00000002	00000000	00000002	RWIX	
INT01	0000ffe2	00000002	00000000	00000002	RWIX	
INT02	0000ffe4	00000002	00000000	00000002	RWIX	
INT03	0000ffe6	00000002	00000000	00000002	RWIX	
INT04	0000ffe8	00000002	00000000	00000002	RWIX	
INT05	0000ffea	00000002	00000000	00000002	RWIX	
INT06	0000ffec	00000002	00000000	00000002	RWIX	
INT07	0000ffee	00000002	00000000	00000002	RWIX	
INT08	0000fff0	00000002	00000000	00000002	RWIX	
INT09	0000fff2	00000002	00000000	00000002	RWIX	
INT10	0000fff4	00000002	00000000	00000002	RWIX	

```
INT11           0000fff6   00000002  00000000  00000002  RWIX
INT12           0000fff8   00000002  00000000  00000002  RWIX
INT13           0000fffa   00000002  00000000  00000002  RWIX
INT14           0000fffc   00000002  00000000  00000002  RWIX
RESET            0000fffe   00000002  00000002  00000000  RWIX
```

有一个从 CCS v5.x 中启动的基于 MAP 文件内容的存储器分配的图形化表示。 此图形工具的访问方法如下："View" -> "Other..." -> "Code Composer Studio" -> "Memory Allocation".

27. 如何调用 BSL？

基本上，有两个 BSL 类型： UART BSL 和 USB BSL。

- 可在复位期间运用一个特殊的 BSL 进入序列来调用 UART BSL，如下所示：

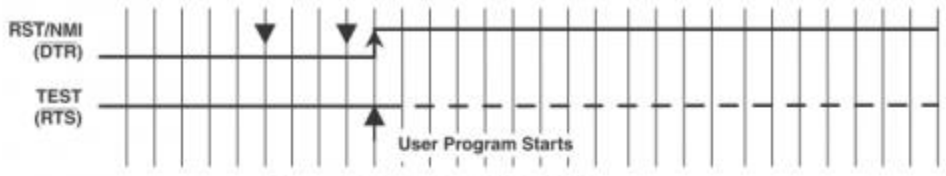


Figure 1-1. Standard RESET Sequence

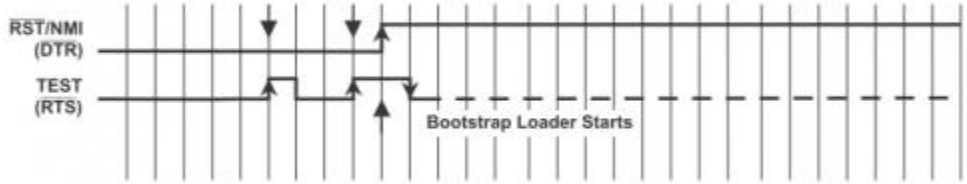


Figure 1-2. BSL Entry Sequence at Shared JTAG Pins

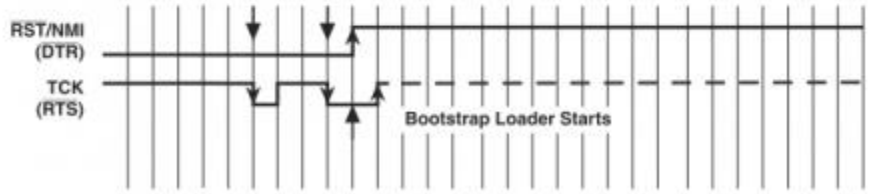


Figure 1-3. BSL Entry Sequence at Dedicated JTAG Pins

（具有共用 JTAG 引脚的器件有 TEST 引脚，而具有专用 JTAG 引脚的器件没有 TEST 引脚）

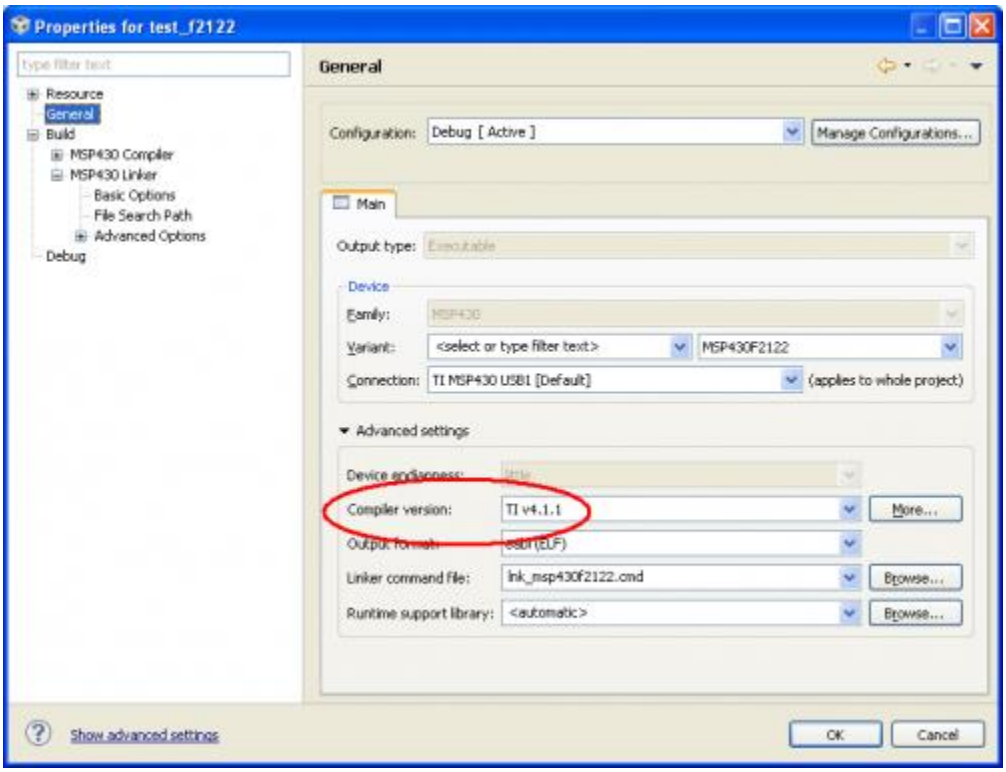
- 当器件由 VBUS 供电时，在满足以下两个条件中的任何一个时可调用 USB BSL：
  - 器件由 USB 供电且复位矢量为空。
  - 器件在 PUR 引脚被接至 VBUS 时加电。

28. BSL 脚本解释器使用哪个 RS232 引脚连接 RST 和 TEST/TCK 信号？

BSL 脚本解释器和 BSLDEMO 使用 DTR 引脚来控制 RST 信号，而使用 RTS 引脚来控制 MSP430 目标器件上的 TEST/TCK 信号。

29. 如何在 CCSTUDIO 中找到我正在使用的编译器版本？

前往 "Project" -> "Properties"，并选择 "General" 选项，你可以在 "Compiler Version" 下找到此信息：

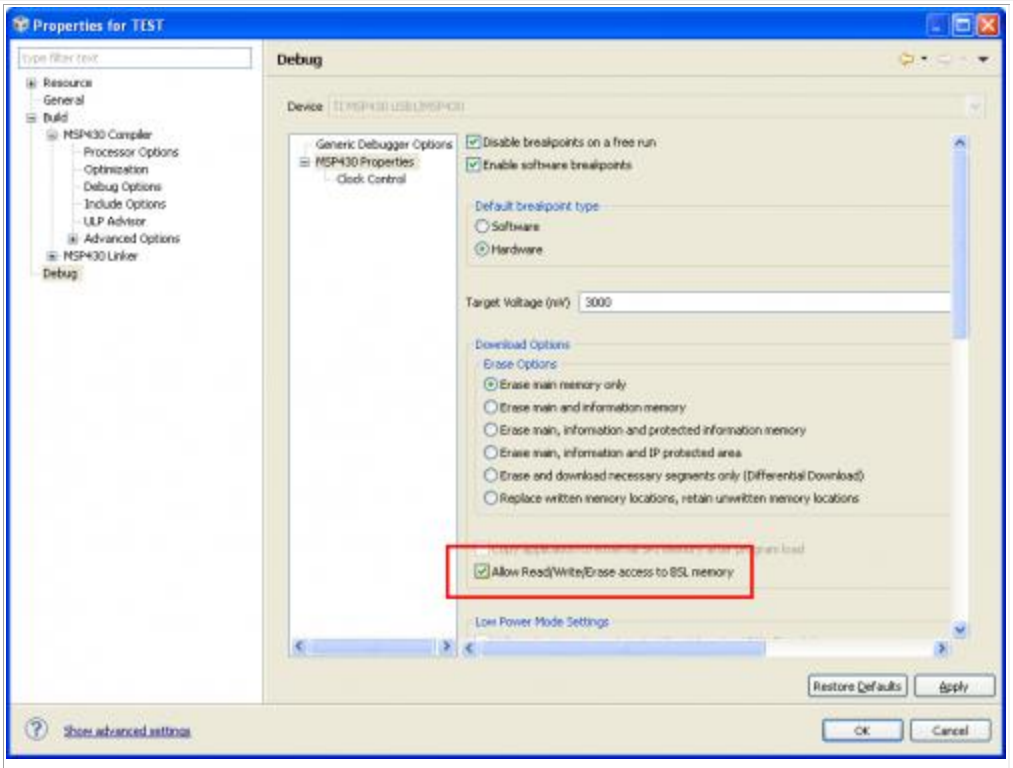


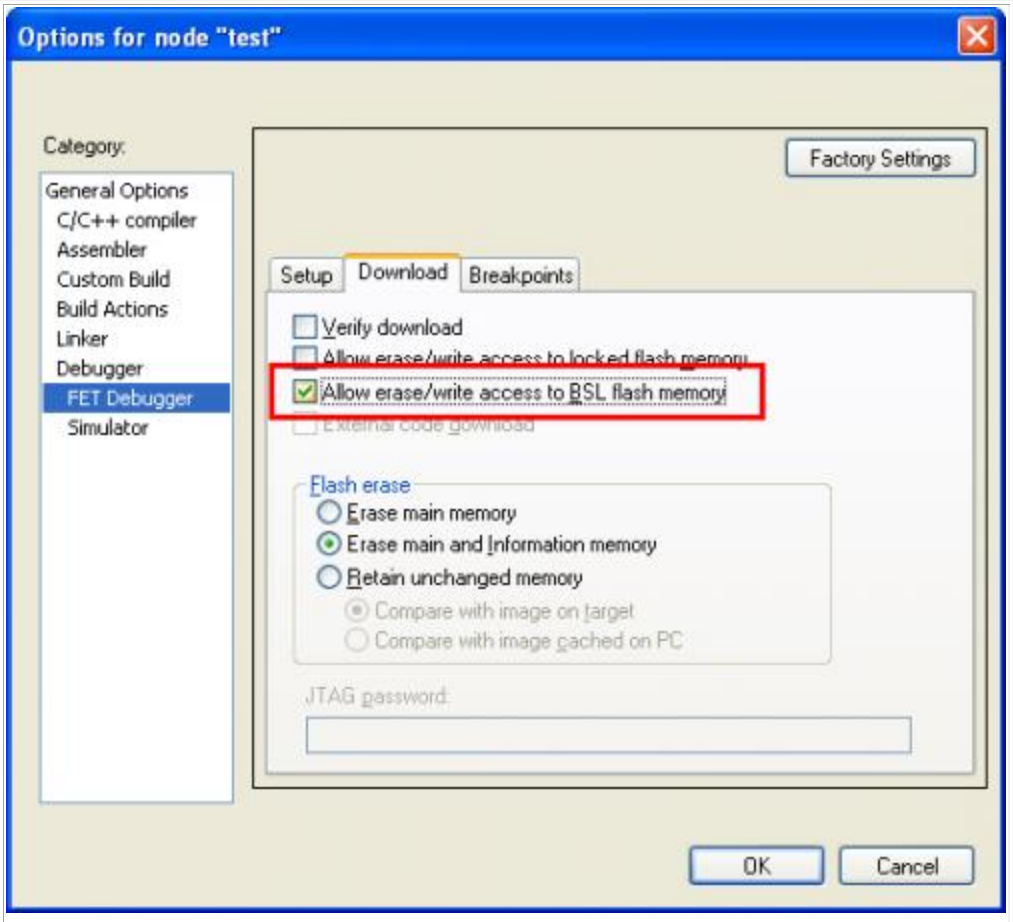
30. 如何启用到 MSP430F5xx/6xx 器件中 BSL 闪存存储器的访问？

根据缺省情况，到 MSP430F5xx/6xx 器件 BSL 闪存存储器的访问受到 SYSBSLC 寄存器内 SYSBSLPE 位的保护。 因此，为了能够获得访问权限，需要将此保护关闭。 通常情况下，调试器/程序设计器将具有一个额外选项：

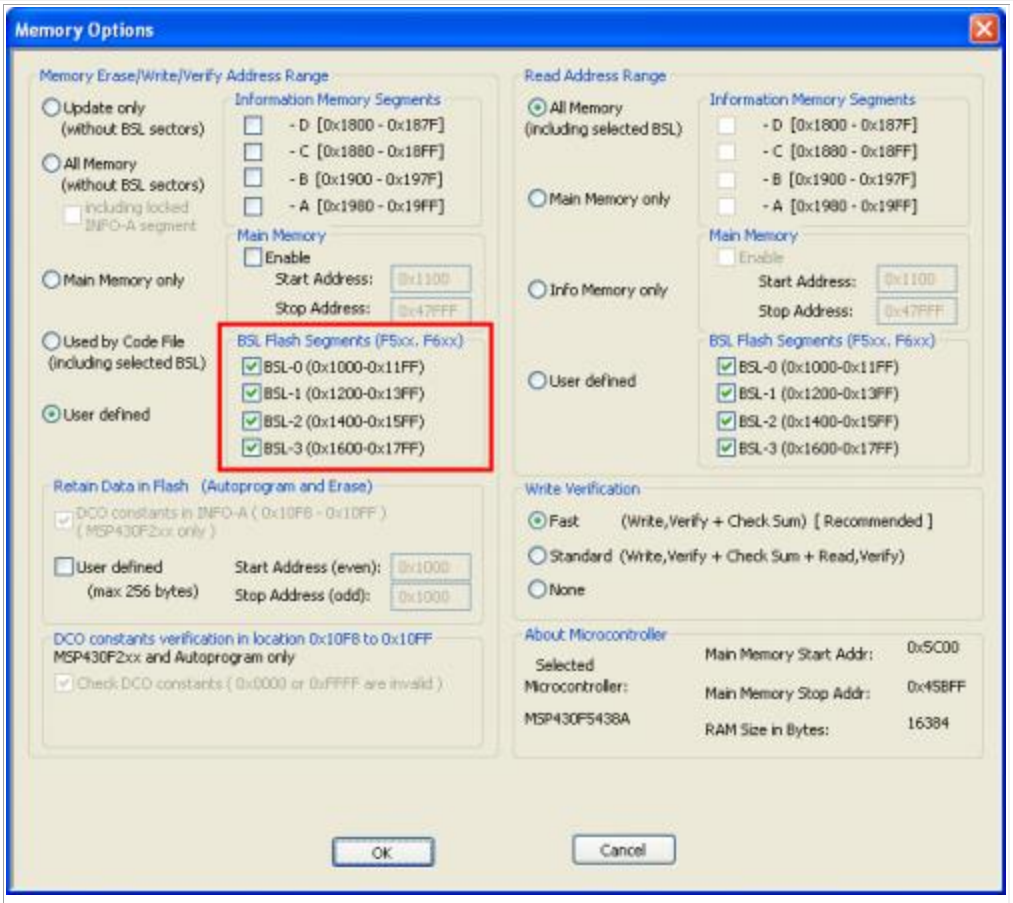
谨记： 某些已发布的 5xx/6xx BSL 的源代码和二进制文件可在 SLAA450 应用说明的相关/随附文件中找到。

调试器/程序设计器	BSL 访问选项

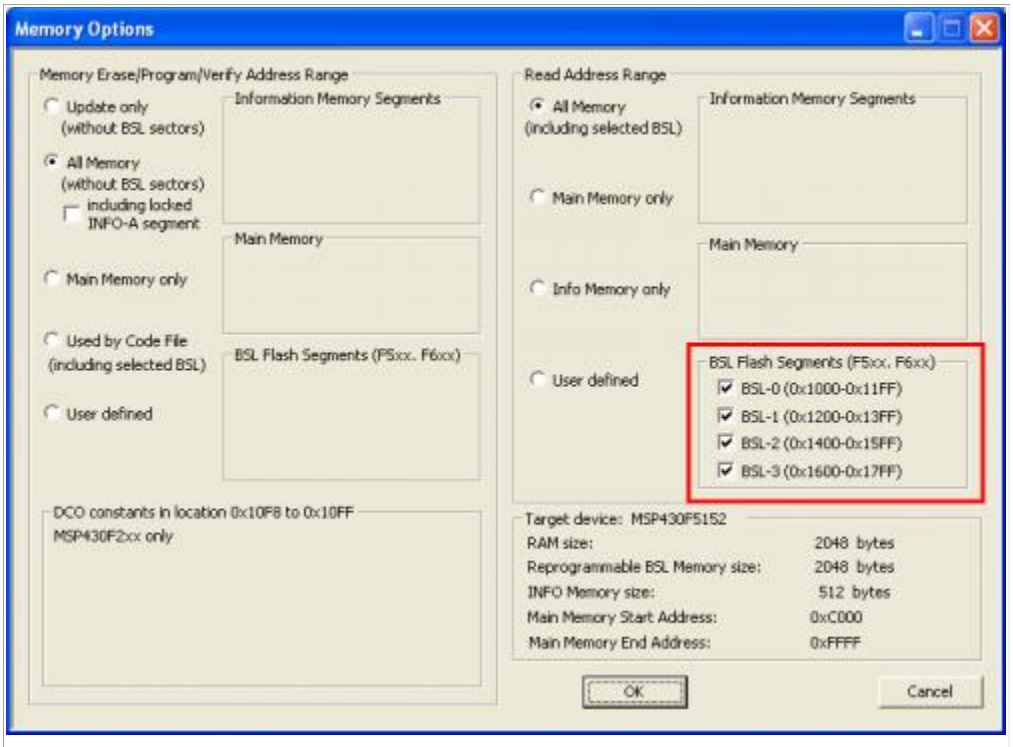
CCS (Code Composer Studio)	

IAR EWB	 Options for node "test" Category: General Options C/C++ compiler Assembler Custom Build Build Actions Linker Debugger FET Debugger Simulator Factory Settings Setup Download Breakpoints Verify download Allow erase/write access to locked flash memory Allow erase/write access to BSL flash memory External code download Flash erase Erase main memory Erase main and Information memory Retain unchanged memory Compare with image on target Compare with image cached on PC JTAG password: OK Cancel

Elprotronic FET-Pro430



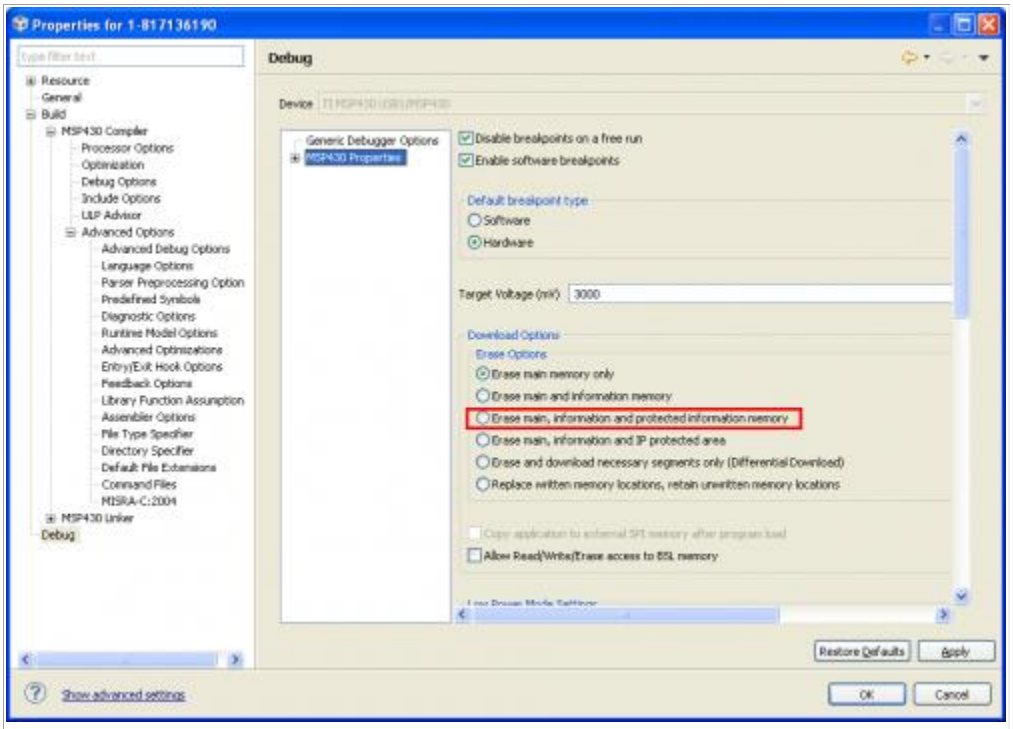
MSP-GANG



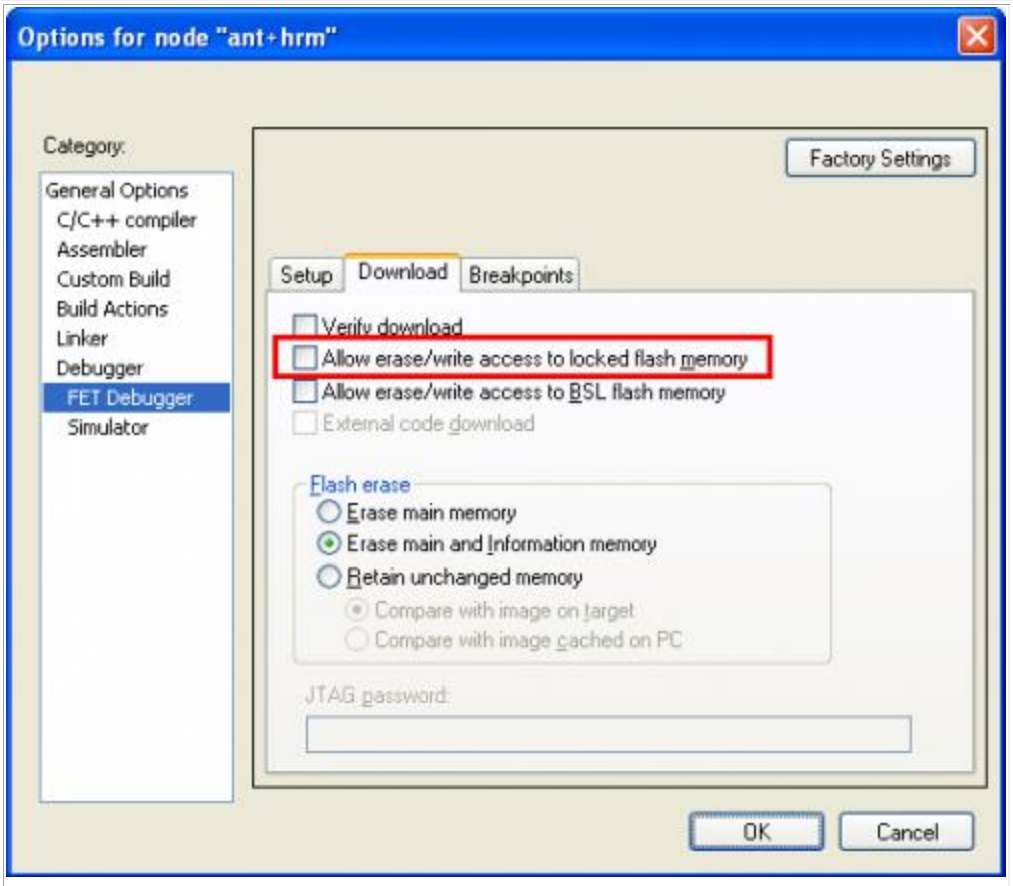
MSP430 Flasher	-b 参数

31. 编程期间如何保护 MSP430x2xx 器件上 INFOA 存储器中的校准数据？

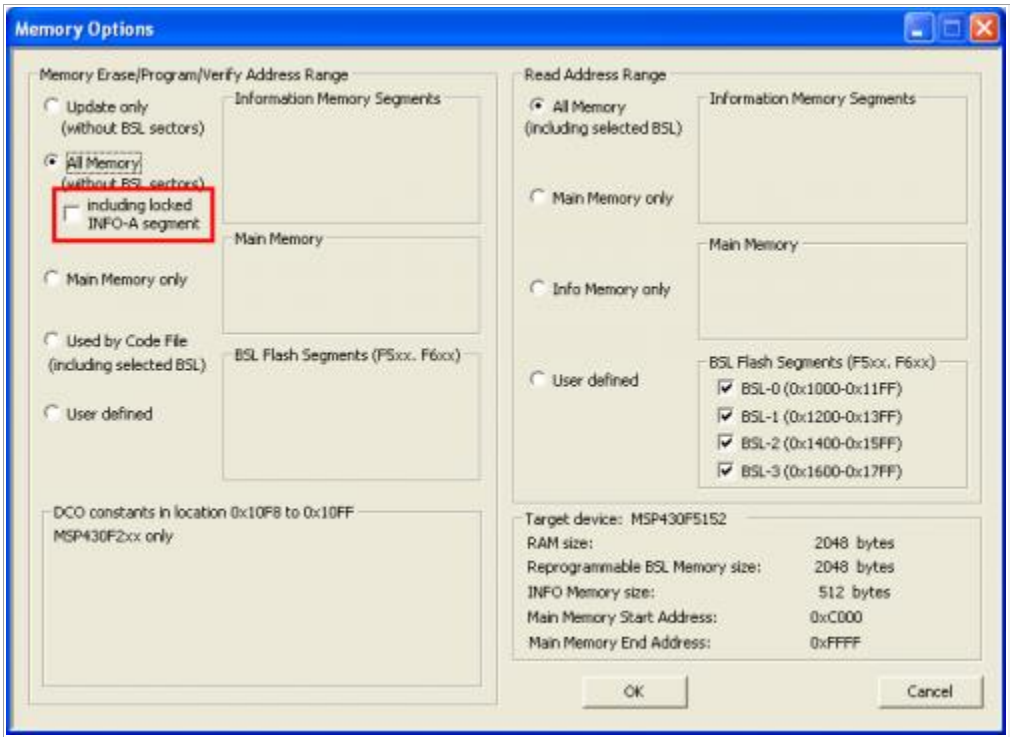
缺省情况下，对 MSP430F2xx/6xx 器件内包含校准数据的 INFOA 存储器的访问会受到 FCTL3 寄存器的 LOCKA 位的保护。 只要 LOCKA 位保持置位，任何批量擦除命令将不会擦除 infoA 存储器。 通常情况下，调试器/程序设计器有用来启用对 InfoA 存储器进行擦除操作的一个额外选项：

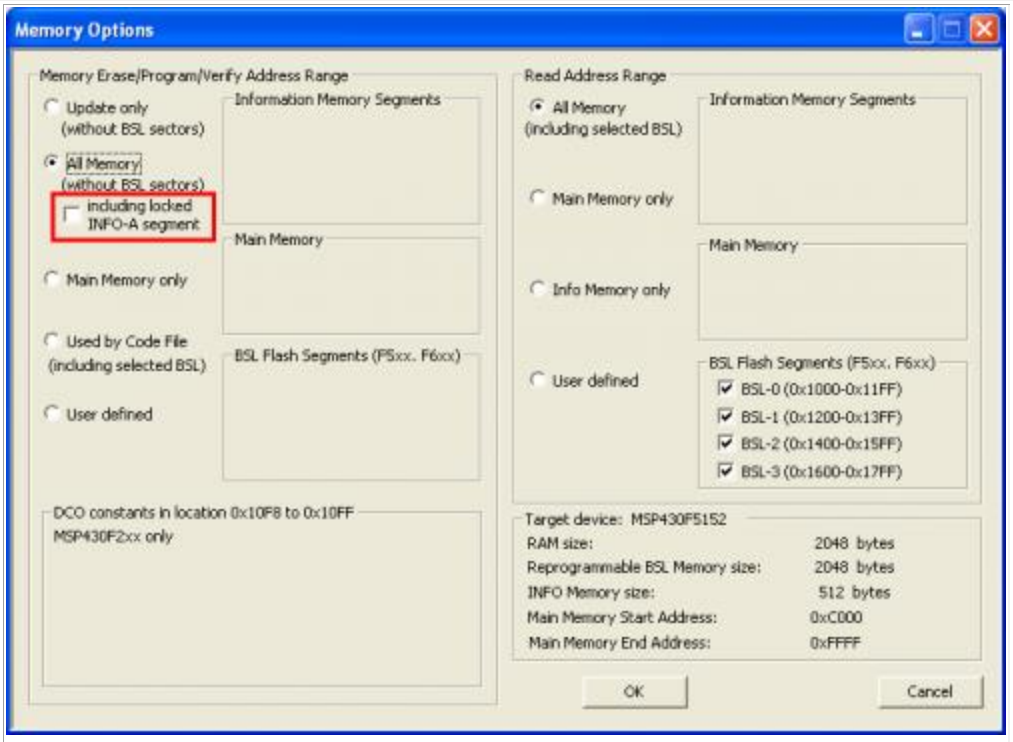
调试器/程序设计器	BSL 访问选项
CCS (Code Composer Studio)	

IAR EWB



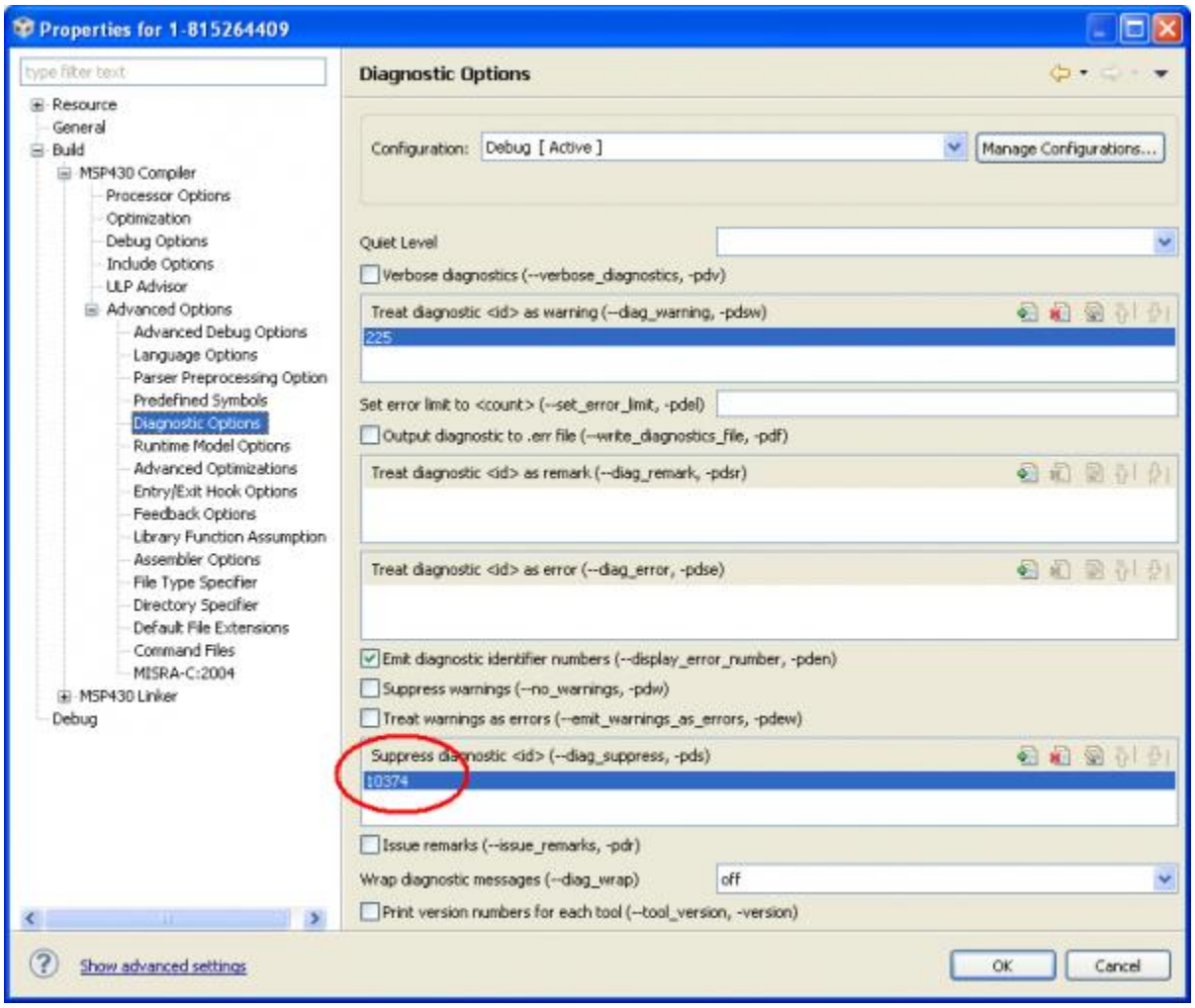
Elprotronic FET-Pro430



MSP-GANG	
MSP430 Flasher	-u 参数

32. 如何在 CCS 中阻止警告消息？

可按照以下的方法，通过使用 --diag_suppress 编译器选项来阻止 CCSTUDIO 中的警告消息：



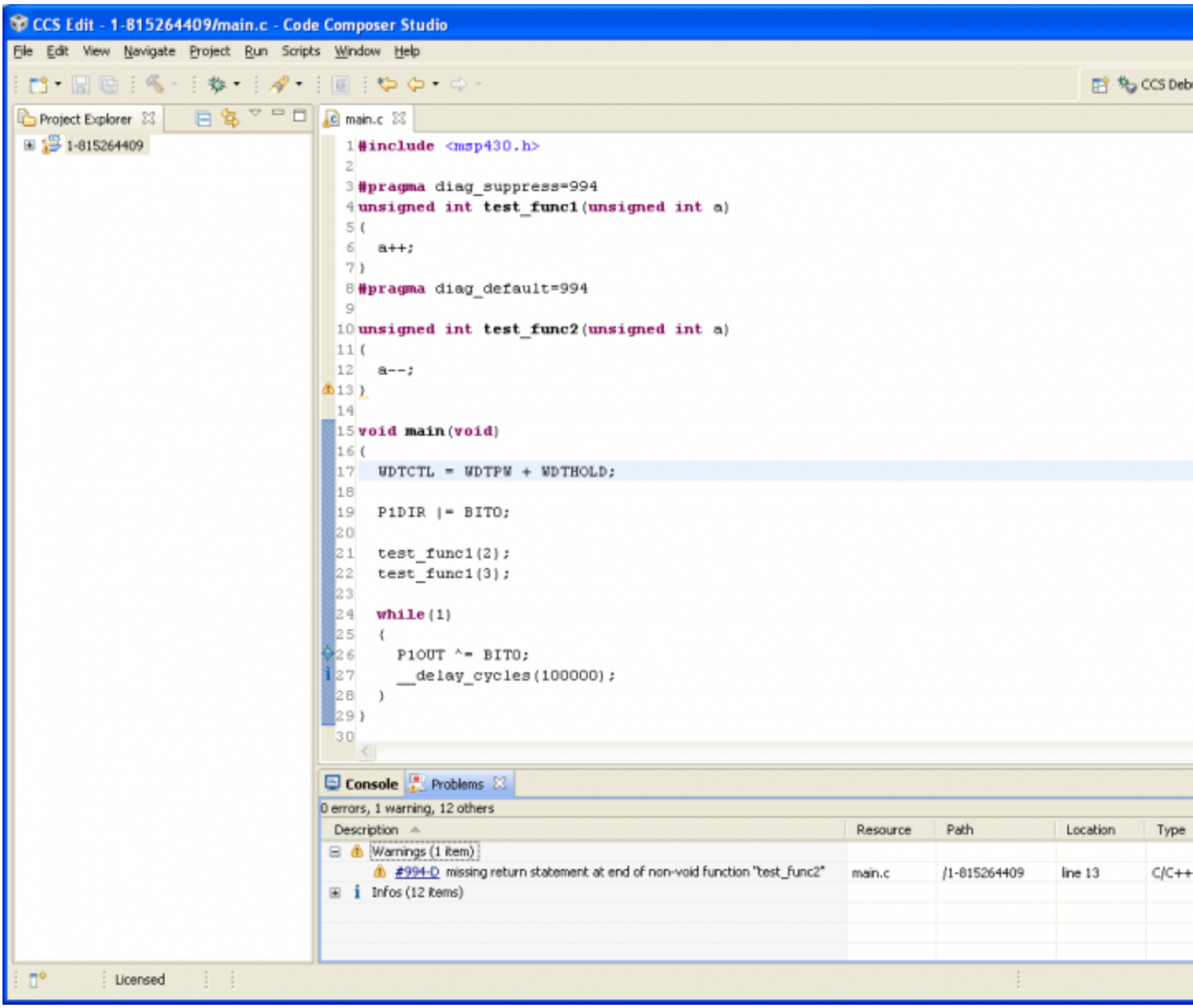
这将在 CCS 项目的整个源代码内全局阻止警告消息。 如果只应在特定代码部分中本地阻止警告消息，可使用 pragma **diag_suppress'** 和 **diag_default** ：

```
#pragma diag_suppress=WARNING_NUM

// all warning messages with WARNING_NUM in this section will be suppressed
.....

#pragma diag_default=WARNING_NUM
```

下面显示了一个示例：



如上所示，编译器基本上应该返回针对 **test_func1** 和 **test_func2** 的警告消息，这是因为两个函数基本上应该根据函数类型声明传递一个返回值。 然而，由于 **test_func1** 在 pragma **diag_suppress** 和 **diag_default** 的范围内，这样就禁用/阻止了警告消息，编译器只给出针对 **test_func2** 的警告消息。

33. 计算 MSP430 BSL 校验和

以下 javascript 可被用来计算 MSP430 BSL 校验和值：

文件：MSP430 BSL CHK Javascript.zip

免责声明：此脚本应该“按现状”使用，没有任何支持或担保

34. 可以调试一个正在运行的 MSP430 器件吗？

请参考以下维基网页： MSP430_-_连接_至_一个_正在运行的_目标

35. 我在哪里能找到 CCSTUDIO 和 IAR 固有函数和参数的列表？

这些固有函数在名为 "in430.h" 的头文件内声明，而参数（例如，针对 __bis_SR_register() 的 LPM0_bits）在器件专用头文件中定义（例如，对于 MSP430FR5969 为 "msp430f5r5969.h"）。 通常可在以下目录中找到~~的~~头文件：

- CCS v5： <CCS_BASE_DIRECTORY>\ccsv5\ccs_base\msp430\include
- IAR： C:\Program Files\IAR Systems\Embedded Workbench x.y_z\430\inc

提示和技巧

36. 有没有在 P1 和 P2 以外端口的引脚上获得中断的方法？

根据缺省设置，只有 P1 和 P2 可以获得 GPIO 输入中断。 然而，有一些小技巧或许可以模拟其他端口引脚上的中断： MSP430_-_其他_GPIO_中断。

37. 如何分配正确的 Timer_A 中断矢量？

基本上，每个 Timer_A 具有两个中断矢量：

- 一个用于 CCR0
- 另外一个用于 TAIFG 和剩余的 CCRx。

CCS 和 IAR 头文件中的中断矢量的格式为 TIMER(X)_A(Y)_VECTOR，其中：

- x 是模块号（例如，对于 MSP430G2553 来说，它具有两个 TImer_A 模块，TA0 和 TA1： 0=TA0，1=TA1)
- Y 是矢量号（0 = CCR0，1 = TAIFG & 其他 CCR）

38. 有可能生成软件复位吗？

生成软件复位的最简单方法是使用下面的看门狗定时器：

```
#define SW_RESET()      WDTCTL = WDT_MRST_0_064; while(1);  // watchdog reset
```

在具有 PMM（电源管理模块）的 5xx/6xx 和 CC430 器件上，可用如下方式生成软件 BOR 和软件 POR：

```
#define SW_RESET()      PMMCTL0 = PMMPW + PMMSWBOR + (PMMCTL0 & 0x0003);  // software BOR reset
#define SW_RESET()      PMMCTL0 = PMMPW + PMMSWPOR + (PMMCTL0 & 0x0003);  // software POR reset
```

39. 可以检索引起复位的原因吗？

在 5xx/6xx 器件上，可以通过校验 SYSRSTIV 寄存器来检查最近一次复位的原因。

Reset Interrupt Vector Register (SYSRSTIV)

15	14	13	12	11	10	9	8
7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
0	**0**	**SYSRSTVEC**					**0**
r0	r0	r-0	r-0	r-0	r-0	r-1	r0

SYSRSTIV Bits 15-0 Reset interrupt vector. Generates a value that can be used as address offset for fast interrupt service routine handling to identify the last cause of a reset (BOR, POR, PUC) . Writing to this register clears all pending reset source flags.

Value	Interrupt Type
0000h	No interrupt pending
0002h	Brownout (BOR) (highest priority)
0004h	RST/NMI (BOR)
0006h	PMMSWBOR (BOR)
0008h	Wakeup from LPMx.5 (BOR)
000Ah	Security violation (BOR)
000Ch	SVSL (POR)
000Eh	SVSH (POR)
0010h	SVML_OVP (POR)
0012h	SVMH_OVP (POR)
0014h	PMMSWPOR (POR)
0016h	WDT time out (PUC)
0018h	WDT password violation (PUC)
001Ah	Flash password violation (PUC)
001Ch	PLL unlock (PUC)
001Eh	PERF peripheral/configuration area fetch (PUC)
0020h	PMM password violation (PUC)
0022h-003Eh	Reserved for future extensions

为了能够正确地对其进行调试，启动时，需要使用如下的低级别 C 语言初始化函数来保存 SYSRSTIV 寄存器值：

对于 CCS 编译器：

```
// global variable for storing the reset cause
#pragma NOINIT (SysRstIv);
unsigned int SysRstIv;

int _system_pre_init(void)
{
  // stop WDT
  WDTCTL = WDTPW + WDTHOLD;
```

```
  // save reset information
  SysRstIv = SYSRSTIV;

  // Perform C/C++ global data initialization
  return 1;
}
```

对于 IAR 编译器：

```
// global variable for storing the reset cause
__no_init unsigned int SysRstIv;

int __low_level_init(void)
{
  // stop WDT
  WDTCTL = WDTPW + WDTHOLD;

  // save reset information
  SysRstIv = SYSRSTIV;

  // Perform data segment initialization
  return 1;
}
```

40. 我如何禁用闪存存储器访问来保护我的 IP？

访问闪存存储器主要有三种方法：

- 经由 JTAG 访问
- 经由 BSL（引导加载程序）
- 经由应用程序中执行的定制访问（可选）

为了禁用经由 JTAG 的闪存访问，JTAG 熔丝应该被烧断。 在之前的 1xx，2xx，4xx 器件上，JTAG 熔丝的实现方式为物理熔丝，而在 5xx/6xx 器件上为电子熔丝。

可通过擦除 BSL 存储器闪存来禁用 5xx/6xx 器件上的 BSL，或者在某些之前的 2xx 器件上，通过禁用 JTAG（通过常见于中断矢量表下的特殊寄存器的设置）来达到此目的。

41. 调试器/程序设计器工具意外地报告 JTAG 熔丝被烧断，我能对此做些什么吗？

与之前使用物理 JTAG 熔丝的 1xx，2xx，4xx 器件不同，5xx/6xx/FRAM 器件主要采用电子 JTAG 熔丝（例如，请参考《5xx/6xx 用户指南》文档 - 1.11.2 章“经由电子熔丝的 JTAG 锁定机制”）。 只要 BSL 未被禁用/擦除，就有可能经由 BSL 来检查 JTAG 的状态。 要获得与 MSP 相关的更多信息，请参考 MSP430 BSL 维基网页： BSL_(MSP430).

TODO：为其提供示例 BSL 脚本

42. 可以重新分配 MSP430 上的中断矢量吗？

在 5xx/6xx 系列器件上，可以通过设置 SYSCTL 寄存器的 SYSRIVECT 来将中断矢量重新分配至 RAM。通过将中断矢量重新分配至 RAM，BSL 代码能够使用中断，其中 BSL 的中断矢量将不会与用于应用的中断矢量相冲突。

以下的示例代码显示了如何在 MSP430F5438A 上实现此操作：

- CCS v5.x (compile option: --code_model==small):

文件：MSP430F5438A RAM INT VECT CCS.zip

- IAR：

文件：MSP430F5438A RAM INT VECT IAR.zip

43. 如何用 MSP430 器件生成随机数？

大多数 MSP430 器件在交付使用时具有片上低功率 VLO 时钟，此时钟的额定值通常为数据表中大范围时钟频率（通常从 6kHz 最小值到 14kHz 最大值）。

有一份 应用说明 描述了如何根据 2xx 器件的 VLO 来生成随机数。 此份应用说明的基本概念是将 VLO 用作源 ACLK，而将 ACLK 也用作输入，此输入作为捕捉一个自由运行定时器的 Timer_A 捕捉事件。 因此，有必要首先检查器件专用数据表来分配正确的 CCR（捕捉比较寄存器），此 CCR 能够将 ACLK 用作输入事件（输入信号被内部连接至 ACLK）。 以下 Timer_A 信号连接表引用自 MSP430F51xx 器件数据表：

Table 12. TA0 Signal Connections

INPUT PIN NUMBER		DEVICE INPUT SIGNAL	MODULE INPUT SIGNAL	MODULE BLOCK	MODULE OUTPUT SIGNAL	DEVICE OUTPUT SIGNAL	OUTPUT PIN NUMBER	
RSB (40-PIN QFN)	DA (38-PIN TSSOP)						RSB (40-PIN QFN)	DA (38-PIN TSSOP)
P3.3 - 30	P3.3 - 34	TA0CLK	TACLK	Timer	NA	NA	-	-
ACLK (internal)	ACLK	ACLK	ACLK				-	-
SMCLK (internal)	SMCLK	SMCLK	SMCLK				-	-
P3.3 - 30	P3.3 - 34	TA0CLK	$\overline{\text{TACLK}}$				-	-
P3.7 - 36	-	TA0.0	CCI0A	CCR0	TA0	TA0.0	P3.7 - 36	-
-	-	CBOUT	CCI0B				-	-
-	-	V_{SS}	GND				-	-
-	-	V_{CC}	V_{CC}				-	-
P3.6 - 35	-	TA0.1	CCI1A	CCR1	TA1	TA0.1	P3.6 - 35	P3.6 - 38
-	-	ACLK	CCI1B				ADC10_A(1) (internal) ADC10SHSx = 001b	ADC10_A(1) (internal) ADC10SHSx = 001b
-	-	V_{SS}	GND				-	-
-	-	V_{CC}	V_{CC}				-	-
P3.5 - 34	P3.5 - 37	TA0.2	CCI2A	CCR2	TA2	TA0.2	P3.5 - 34	P3.5 - 37
-	-	V_{SS}	CCI2B				-	-
-	-	V_{SS}	GND				-	-
-	-	V_{CC}	V_{CC}				-	-

如上所示，对于 MSP430F51xx 器件，CCR1 寄存器被用来生成随机数。 此应用说明也描述了某些将随机性添加到代码中的技巧。 下面是生成 MSP430F51xx 器件随机数的示例代码：

```
int TI_getRandomIntegerFromVLO(void)
{
  unsigned int i;
  int result = 0;

  // setup Timer_A
  TA0CCTL1 = CM_1 + CCIS_1 + CAP;
  TA0CTL |= TASSEL__SMCLK + MC__CONTINOUS;
```

```
  for(i=0 ; i<16 ; i++)
  {
    // shift left result
    result <<= 1;

    // wait until Capture flag is set
    while(!(TA0CCTL1 & CCIFG));

    // clear flag
    TA0CCTL1 &= ~CCIFG;

    // check LSB
    if(TA0CCR1 & 0x01)
    {
      result |= 0x01;
    }

    // change the divison of timer input clock
    TA0CTL = (TA0CTL & 0xFCFF) | ((TA0CCR1 & 0x03) << 8);
  }

  return result;
}
```

CCS 编译器警告 #10374-D

在 CCS v5.x 的全新 CCS 编译器中（代码生成工具/CGT），编译器将给出这样的警告：

#10374-D 中断矢量 "USI" 没有一个中断句柄例程。

如果它发现有任何中断矢量未被初始化时就会给出此警告。 原因在于，这基本上是一个初始化所有中断矢量的好做法，所以在假中断意外出现时，MSP430 将会因为未将中断矢量初始化而无法获得复位。

以下示例显示了如何创建 "Traplsr" 来删除 CCS 编译器给出的关于这个问题的警告：

```
#include <msp430g2231.h>

int main(void)
{
  WDTCTL = WDTPW + WDTHOLD;                 // Stop watchdog timer
  P1DIR |= BIT0;                                        // Set P1.0 to output direction
```

```
  for(;;)
  {
    P1OUT ^= BIT0;                                  // Toggle P1.0 using exclusive-OR
    __delay_cycles(1000000);
  }
}

// trap isr assignation - put all unused ISR vector here
#pragma vector = ADC10_VECTOR, NMI_VECTOR, PORT1_VECTOR, PORT2_VECTOR, \
                 TIMER0_A0_VECTOR, TIMER0_A1_VECTOR, USI_VECTOR, WDT_VECTOR
__interrupt void TrapIsr(void)
{
  // this is a trap ISR - check for the interrupt cause here by
  // checking the interrupt flags, if necessary also clear the interrupt
  // flag
}
```

也请参见 #如何在 CCS 中阻止一条警告消息？ 。

44. 如何将一个变量放置在特定存储器位置内？

请参考以下维基网页： 将_变量_放置在_特定_存储器_位置内_-_MSP430

45. 如何将生成 MSP430x5xx/6xx JTAG 锁定放置在代码中？

请参考以下维基网页： 将_变量_放置在_特定_存储器_位置内_-_MSP430#Generating_JTAG_Lock

46. 4 引脚 SPI 在 USCI 模块上的工作模式是怎样的？

在 USCI 模块上，UCxSTE 被用作处于主控和受控模式中的 USCI 模块的 4 引脚 SPI 模式下（UCMODEx = 01 或 10）的激活 **输入引脚**，如以下表格所示：

UCMODEx	UCxSTE Active State	UCxSTE	Slave	Master
01	High	0	Inactive	Active
		1	Active	Inactive
10	Low	0	Active	Inactive
		1	Inactive	Active

4 引脚 USCI SPI 主控模式

在 4 引脚主控模式下，UCxSTE 被用来避免与其他主控的冲突。 当 UCxSTE 处于主控未激活模式中时：

- UCxSIMO 和 UCxCLK 被设置为输入并不再驱动总线。
- 出错位 UCFE 置位表明出现一个将由用户处理的通信错误。
- 内部状态机被复位并且移位操作取消。

4 引脚 USCI SPI 受控模式

在 4 引脚受控模式中，UCxSTE 被受控用于使能发送和接收操作并由 SPI 主机提供。 当 UCxSTE 处于受控未激活状态时：

- UCxSIMO 上任何正在进行中的接收操作被暂停。
- UCxSOMI 被设置为输入方向。
- 移位操作被暂停直到 UCxSTE 线路转换进入受控传输激活状态。

47. 如何使用 USCI 模块在传输完成时生成中断？

在 USCI 模块中，发射中断生成在 UCxTXBUF 数据字节被复制/被移入 ~~TI~~TX 移位寄存器时。 然而，有时需要在整个数据字节已经从 ~~TI~~TX 移位寄存器中移出时检测/生成中断。有几种可能的方法来实现此操作：

- 在发送中断发生时运行一个定时器，以便根据计算出的、发送整个数据字节所需的时间来生成定时器中断。
- 为了将已发送数据字节回送至接收器，将 UCSxSTAY 寄存器内的 UCLISTEN 位置位。 通过激活接收中断，可使用接收中断来仿真中断。 更多详细信息可在以下 E2E 讨论 中找到。

48. 有没有推荐用于 MSP430 器件的晶体振荡器列表？

TI 并未给出推荐在 MSP430 器件中使用的晶体振荡器列表。 通常情况下，建议用户根据晶振制造商的额定值，通过使用内部 XCAP 值或外部负载电容器来设置晶体振荡器的负载电容器。 我们还建议客户按照应用说明 tidoc:slaa322 章节 4 来全面测试晶振。

代码参考

- FatFS: MSP-EXP430F5529 用户体验软件，基于 ELM-Chan 开源 FatFs 嵌入式文件系统模块。
- 加密算法： tidoc:slaa547, tidoc:397 (AES128)

49. 用 MSP430 设计定制板的检查清单

- 请确保引脚分配连接符合 TI 提供的开发套件的要求。 TI 几乎为每一个 MSP430 器件提供开发套件（但是并未为每个封装类型提供开发套件）。 MSP430 器件的电路原理图可在 tidoc:slau278 中找到。
- 请确保 JTAG 引脚分配符合建议的引脚分配： JTAG_(MSP430).
- 其他建议的 MSP430 微控制器硬件设计指南参考
  - MSP430 系统级静电放电 (ESD) 注意事项： tidoc:slaa530
  - MSP430 外部振荡器： tidoc:slaa322
  - MSP430 电容式触摸硬件设计指南： tidoc:slaa576
  - MSP430 USB 设计指南： tidoc:slaa457
- 请按照 这里描述的那样考虑电源与 CPU 频率之间的关系。

50. 从 SRAM 运行代码

为了降低功耗，有时从 SRAM 中执行 codex 会有一定作用。 对于这一点，有几个示例： 针对 MSP430F543x 的闪存写入示例 或者参考这份 短篇指南。

51. 设置 USCI 模块 UART 模式波特率

使用 USCI_UART_Baud_Rate_Gen_Mode_Selection#USCI_UART_Calculator USCI UART 波特率计算器来计算或者使用 这个链接中的 usci_settings.h 来计算。

52. 将 MSP430F5xx/6xx BSL 闪存存储器区域用于应用数据/代码

在 MSP430F5xx/6xx 器件上，如果 BSL 不被应用使用的话，可使用 BSL 闪存存储器区域。 针对这一用途，请参考以下 指南。

技术 FAQ

53. 如何将一个 I/O 引脚设置为外设引脚？

在每个器件专用数据表中，有一个描述 I/O 引脚电路原理图，以及将一个引脚设定为正常 GPIO 引脚还是外设引脚的寄存器设置的特殊章节（此章节通常在文档末尾电气特征参数之后，名为“应用信息”或“输入/输出电路原理图”）。

54. 最大 GPIO 源电流/灌电流

MSP430 数据表通常不指定一个 GPIO 引脚能够吸收/灌入的最大电流。 然而，应该考虑两个限制因素：

(1) 由于 MSP430 具有 CMOS GPIO，高电平输出电压将 VOH 将在引脚吸收电流时减少，而低电平输出电压 VOL 在引脚灌入电流时增加。 比如说，以下值是取自 MSP430F22x2 和 MSP430F22x4 数据表中的值：

Outputs (Ports P1, P2, P3, and P4)

over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted)

PARAMETER		TEST CONDITIONS	V_{CC}	MIN	MAX	UNIT
V_{OH}	High-level output voltage	$I_{OH(max)}$ = -1.5 mA (1)	2.2 V	V_{CC} - 0.25	V_{CC}	V
		$I_{OH(max)}$ = -6 mA (2)		V_{CC} - 0.6	V_{CC}	
		$I_{OH(max)}$ = -1.5 mA(1)	3 V	V_{CC} - 0.25	V_{CC}	
		$I_{OH(max)}$ = -6 mA(2)		V_{CC} - 0.6	V_{CC}	
V_{OL}	Low-level output voltage	$I_{OL(max)}$ = 1.5 mA(1)	2.2 V	V_{SS}	V_{SS} + 0.25	V
		$I_{OL(max)}$ = 6 mA(2)		V_{SS}	V_{SS} + 0.6	
		$I_{OL(max)}$ = 1.5 mA(1)	3 V	V_{SS}	V_{SS} + 0.25	
		$I_{OL(max)}$ = 6 mA(2)		V_{SS}	V_{SS} + 0.6	

(1) The maximum total current, $I_{OH(max)}$ and $I_{OL(max)}$, for all outputs combined, should not exceed ±12 mA to hold the maximum voltage drop specified.
(2) The maximum total current, $I_{OH(max)}$ and $I_{OL(max)}$, for all outputs combined, should not exceed ±48 mA to hold the maximum voltage drop specified.

如上所见，举例来说，如果一个输出试图在 Vcc = 2.2V 或 3V 时吸收 1.5mA 的电流，它应该等到 VOH 下降到 Vcc - 0.25V 的最小值。当在同一 Vcc 电平上吸收 6mA 电流时，此输入甚至可以下降至 Vcc - 0.6V 的最小值。当输出端口试图灌入电流时，它应该等待上面指定的低电平输出电压 VOL 的一个增加值，例如，当在 2.2V 或 3V 电压上灌入 1.5mA 电流时，最大值 Vss + 0.25V。 如脚注中所描述的那样，还需注意的一点是，上面的 VOL 和 VOH 额定值只在最大总源/灌电流未超过脚注中的额定值时才有效。

(2) 当试图灌入大量电流时，增加的功率耗散带来的主要影响是结温的增加。 例如，通过使用 ROT 公式计算： 温度（结温）= theta (j-a) * P + 温度（环境温度）。 例如，如果输出引脚在 50 摄氏度时有 5mA 灌电流：

功率耗散 P = 压降 * I = 3V * 35mA = 105mW

对于封装网站中的 MSP430F1232（28 DW 封装），它指定 Theta (j-a) = ~50 C/W。 使用一个 200mW 的最大功率耗散（来自 CPU + 模块 + GPIO），我们得到；

温度（结温）= 50 * 0.2 W + 50C = ~60C.

这仍然在器件的最大绝对额定值，即 85C，范围内。 谨记，这些计算只是用于理解灌入过多电流所造成的影响的一般性指导原则。 用户有责任检查应用硬件，并防止在应用硬件上出现短路。

55. 器件处于复位状态时，GPIO 处于什么状态？

在器件被保持在复位中时，GPIO 处于其缺省状态，也就是说，输入/高阻抗和上拉电阻器悬空也未在这个状态中被启用。

56. 最大 CPU 频率

最大 CPU 频率通常在器件专用数据表中“建议运行条件”下被定义，但是总的来说，对于 1xx 器件为 8MHz，2xx 器件为 16MHz，5xx 器件为 8/16MHz，而 5xx/6xx 器件为 25MHz。

这一主题下，另外一个常被问到的问题就是为什么数据表将 DCO /外部时钟 (XT1/XT2) 频率指定为高于最大 CPU 频率。 时钟系统也许可以接受来自 XT1/XT2 的较高外部时钟信号（例如，对于 MSP430F5438A，最大 XT1/XT2 额定值为 32MHz），甚至可以将 DCO 频率运行为更高的频率（例如，对于 MSP430F5438A，高达 135MHz），这些都是事实，但是在提供 ACLK，MCLK 或 SMCLK 时钟信号之前，应该使用 DIVA，DIVM 或 DIVS 预分频器将时钟频率按比例减小。

57. 为什么 5xx/6xx UCS DCO 时钟运行频率可高于 25MHz？

简言之，主要是为了减少由 FLL 完成的 DCO 调制所导致的抖动。 UCS 模块的 FLL 通过在两个频率：fDCO 和 fDCO+1 之间切换来稳定 DCO 输出。 这引起了一个抖动，例如，对于 MSP430F5438A 来说，这个值在 2% - 12%（请参见以下取自数据表中的技术规格）。

DCO Frequency

over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
$f_{DCO(0,0)}$	DCO frequency (0, 0)	DCORSELx = 0, DCOx = 0, MODx = 0	0.07		0.20	MHz
$f_{DCO(0,31)}$	DCO frequency (0, 31)	DCORSELx = 0, DCOx = 31, MODx = 0	0.70		1.70	MHz
$f_{DCO(1,0)}$	DCO frequency (1, 0)	DCORSELx = 1, DCOx = 0, MODx = 0	0.15		0.36	MHz
$f_{DCO(1,31)}$	DCO frequency (1, 31)	DCORSELx = 1, DCOx = 31, MODx = 0	1.47		3.45	MHz
$f_{DCO(2,0)}$	DCO frequency (2, 0)	DCORSELx = 2, DCOx = 0, MODx = 0	0.32		0.75	MHz
$f_{DCO(2,31)}$	DCO frequency (2, 31)	DCORSELx = 2, DCOx = 31, MODx = 0	3.17		7.38	MHz
$f_{DCO(3,0)}$	DCO frequency (3, 0)	DCORSELx = 3, DCOx = 0, MODx = 0	0.64		1.51	MHz
$f_{DCO(3,31)}$	DCO frequency (3, 31)	DCORSELx = 3, DCOx = 31, MODx = 0	6.07		14.0	MHz
$f_{DCO(4,0)}$	DCO frequency (4, 0)	DCORSELx = 4, DCOx = 0, MODx = 0	1.3		3.2	MHz
$f_{DCO(4,31)}$	DCO frequency (4, 31)	DCORSELx = 4, DCOx = 31, MODx = 0	12.3		28.2	MHz
$f_{DCO(5,0)}$	DCO frequency (5, 0)	DCORSELx = 5, DCOx = 0, MODx = 0	2.5		6.0	MHz
$f_{DCO(5,31)}$	DCO frequency (5, 31)	DCORSELx = 5, DCOx = 31, MODx = 0	23.7		54.1	MHz
$f_{DCO(6,0)}$	DCO frequency (6, 0)	DCORSELx = 6, DCOx = 0, MODx = 0	4.6		10.7	MHz
$f_{DCO(6,31)}$	DCO frequency (6, 31)	DCORSELx = 6, DCOx = 31, MODx = 0	39.0		88.0	MHz
$f_{DCO(7,0)}$	DCO frequency (7, 0)	DCORSELx = 7, DCOx = 0, MODx = 0	8.5		19.6	MHz
$f_{DCO(7,31)}$	DCO frequency (7, 31)	DCORSELx = 7, DCOx = 31, MODx = 0	60		135	MHz
$S_{DCORSEL}$	Frequency step between range DCORSEL and DCORSEL + 1	$S_{RSEL} = f_{DCO(DCORSEL+1,DCO)}/f_{DCO(DCORSEL,DCO)}$	1.2		2.3	ratio
S_{DCO}	Frequency step between tap DCO and DCO + 1	$S_{DCO} = f_{DCO(DCORSEL,DCO+1)}/f_{DCO(DCORSEL,DCO)}$	1.02		1.12	ratio
Duty cycle		Measured at SMCLK	40	50	60	%
df_{DCO}/dT	DCO frequency temperature drift	f_{DCO} = 1 MHz,		0.1		%/°C
df_{DCO}/dV_{CC}	DCO frequency voltage drift	f_{DCO} = 1 MHz		1.9		%/V

通过使 DCO 产生较高频率，然后将此频率分频为所需的输出频率提供给 ACLK/MCLK/SMCLK，将最大限度地减少抖动影响。 例如，当你试图在 8MHz 频率（最大频率的 12%）上运行器件时，将 DCO 设定为频率 64MHz 的时钟源，并进行 8 分频，这将使输出时钟为最大值的 1.5% (12%/8)。

58. 我的 MSP430 在启动时好像没有运行，这是怎么了？

有几个常见问题会导致 MSP430 器件在启动时出现故障（器件看起来根本就没有工作）：

59. 在没有足够供电的情况下高频运行 CPU

如果你在较高的频率下运行 CPU，最常见的问题是在达到最低电源电压前，将 CPU 设定为较高的运行频率。 如果 Vcc 的斜升速度相对慢于设定 CPU 频率的缺省代码，这个问题就会出现。 这些信息通常可以在数据表的“建议运行条件”章节中找到。 例如，以下图表显示了 CPU / 系统频率 (MCLK) 与 MSP430F44x 的电源电压 Vcc 之间的关系图表。

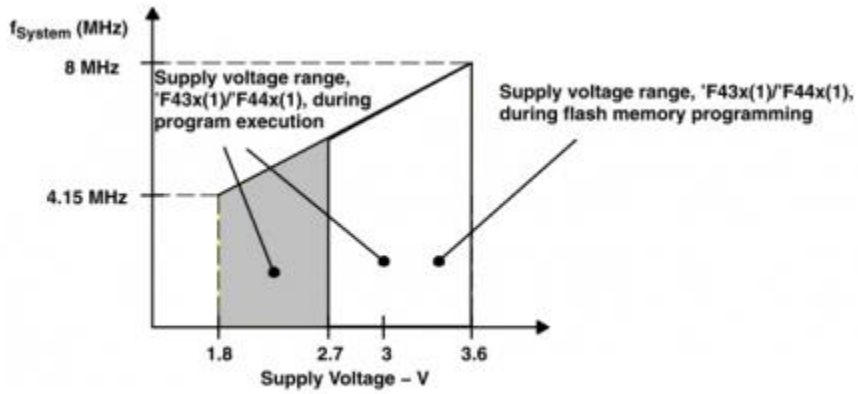
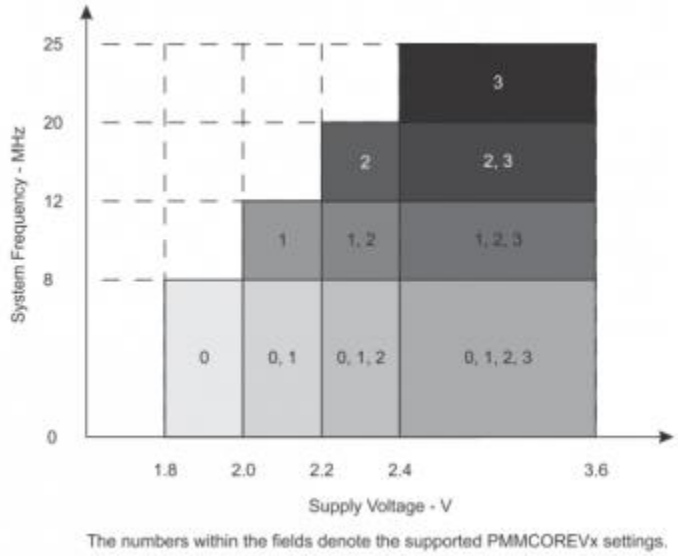


Figure 1. Frequency vs Supply Voltage, MSP430F43x(1) or MSP430F44x(1)

在这个情况下，在将 CPU 设置为较高频率运行前，工作区将产生一个较小的启动延迟，或者使用内部 ADC 来测量 Vcc，以确保 CPU 以较高频率运行前已经达到适当的 Vcc 电平。

在 5xx/6xx 器件上，与 CPU 高频运行相关的不是电源电压, 而是 PMM（电源管理模块）的 VCORE 电平。要设定 VCORE 电平，建议使用 MSP430Ware 的 driverlib。 下方显示的是 CPU / 系统频率 (MCLK) 与 MSP430F543xA 的 VCORE 电平之间的关系图。



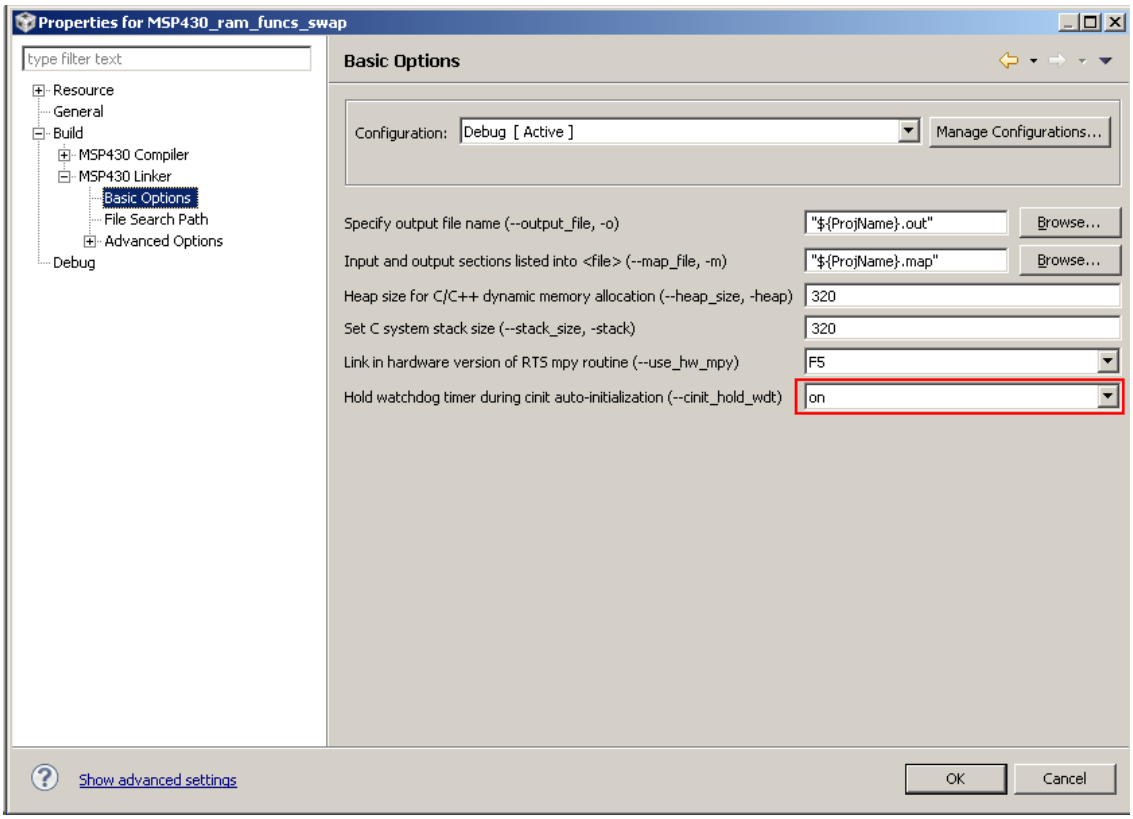
如果 CPU 以较高频率运行，并且电源电压在运行期间下降的话，会产生另外一个问题。 在这个情况下，只要电源电压下降到低于 CPU 高频运行所需的最小足够电压以下，就将器件保持在复位状态。 如果没有这么做的话，就不能保证器件工作正常，甚至有可能导致严重损坏，诸如闪存存储器损坏（请参考： MSP430_闪存_最佳_做法）。

在 1xx，2xx，4xx 器件上，某些器件具有可用于此用途的 SYS（电源电压监控器）片上模块。 对于那些不具有 SVS 模块的器件，建议使用 外部电压监控器。 在所有 5xx/6xx 和 FRAM 器件上，PMM（电源管理模块）可被用来在电源电压下降到低于特定阀值时生成复位。

C 启动代码期间的 WDT 触发

如果应用程序是用 C 语言编写的，那么另外一个常见问题就是启动期间的看门狗超时。 在缺省情况下，所有 MSP430 上的看门狗定时器被设定为启动后激活。 因此，如果在应用程序刚开始时不需要 WDT，那么有必要将其关闭。 如果应用代码正在使用需要在启动期间初始化的大型变量，这会导致看门狗定时器在启动期间已经运行，而代码将永远不会运行。

对于 CCS MSP430 编译器 4.2.0 和之后的编译器版本，此解决方案是将连接器 --cinit_hold_wdt 选项打开，在 C 语言自动初始化期间保持看门狗：



将连接器 --cinit_hold_wdt 选项打开意味着看门狗只在 C 语言自动初始化期间被保持为未激活，这样的话，看门狗将在进入主程序时被激活。

对于其他编译器，针对这一问题的解决方案是使用编译器的低级 C 语言初始化函数，此函数的调用甚至在 C 语言变量初始化之前。 在 CCS 编译器中，它被称为 "int_system_pre_init(void)"，而在 IAR 中，此函数被称为 ini_low_level_init(void)。 返回值被用来确定是否执行 C/C++ 全局数据初始化（0 返回值绕开 C/C++ 自动初始化）。 与这个问题相关的更多详细信息请参考 《MSP430 软件编码技术应用报告》，3.6 章“使用低级初始化函数”。 在低级 C 语言初始化函数中将看门狗保持在未激活状态意味着看门狗将在进入主程序时保持在未激活状态。

对于 CCS 编译器：

```
int _system_pre_init(void)
{
  // stop WDT
  WDTCTL = WDTPW + WDTHOLD;

  // Perform C/C++ global data initialization
  return 1;
}
```

对于 IAR 编译器：

```
int __low_level_init(void)
{
  // stop WDT
  WDTCTL = WDTPW + WDTHOLD;

  // Perform data segment initialization
  return 1;
}
```

中断矢量的错误分配

中断矢量会在某些情况下被不恰当地/错误地分配。 例如，最常见的困惑在于设置 Timer_A 矢量（请参见 MSP430_FAQ#如何去分配正确的 Timer A 中断矢量.3F。 如果中断矢量未被正确分配，器件将在中断发生时复位。

60. 我的应用中的闪存存储器好像被损坏了，问题有可能出在哪里呢？

基本上，MSP430 器件上的闪存存储器损坏的最常见原因是在电源处于欠压状态中时高频运行 CPU。 要获得更多详细信息，请参考 MSP430_闪存_最佳_做法 维基网页。

61. 当我改变新器件修订版本时，应用程序停止工作，这是由什么原因造成的呢？

在大多数情况下，当使用新的/不同的器件修订版本时，应用程序停止工作的主要原因是应用程序的写入方式违反了数据表中的技术规范。 数据表中定义的额定值/参数是为了确保所有修订版本中的全部器件可在这些范围内工作，但是它并未排除器件在额定参数值以外运行时，特定修订版本不工作的可能性。 当之前的器件出现这一问题时（被写入的应用程序在额定参数以外工作），这也许是应用程序不与更新的/不同的修订版本一同工作的真正原因。

62. MSP430 如何在 LPMx.5 中保持 I/O 端口和 RTC 配置？

基本上，进入 LPM3.5 和 LPM4.5 中将在唤醒时复位所有外设寄存器配置。 然而，为了保持 I/O 端口和 RTC 配置寄存器，寄存器值将由相应的模块内部锁存。 这些内部设置在进入 LPMx.5 时被锁定，其原因是 PM5CTL0 寄存器中的 LOCKLPM5 位和 BACKCTL 寄存器的 LOCKBAK 位也被置位。 当从 LPMx.5 中唤醒时，在将 LOCKLPM5 和 LOCKBAK 位清零前，应该首先配置 I/O 和 RTC 寄存器，以避免一个中间状态，在此状态中，复位后的缺省寄存器设置被复制到内部寄存器模块设置中。

63. 如何实现数据表中的额定功耗值？

测试参数

数据表中的所有额定值在特定条件下测得，这些条件会影响测量结果，诸如：

- 环境温度
- 电源电压
- 对于激活模式：时钟源频率和代码位置（闪存 / RAM）
- 无模拟外设激活（通常情况下，模拟外设消耗大量电流，而数字外设流耗很低）
- 等等

避免 I/O 引脚悬空

缺省情况下，所有 I/O 引脚在复位后被设置为输入。 为了避免悬空输入引脚消耗更多电流，所有未使用的 I/O 应该被设置为输出低电平，除非引脚被连接至外部上拉电阻器，这一连接意味着引脚应该被设置为输出高电平。

未使用引脚的连接

请确保所有其他未使用的非 I/O 引脚遵守建议的连接方式，此连接方式可在《系列用户指南》中找到（通常在讨论系统复位、中断等内容的章节中，名称为“未使用引脚的连接”）。

64. 复位状态中的功耗

在数据表中并没有功耗/流耗的技术规格，并且不建议将器件保持在最低功耗水平上。 这一建议的原因是，在复位状态中，GPIO 引脚处于缺省高阻抗悬空状态，而此状态会导致额外的功耗。 而推荐的方法是使用一个 GPIO 中断，此中断将在 LPM4 中（比如在输入的下降边沿上）将器件置位，并重新将中断配置为在信号的上升边沿唤醒器件。