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

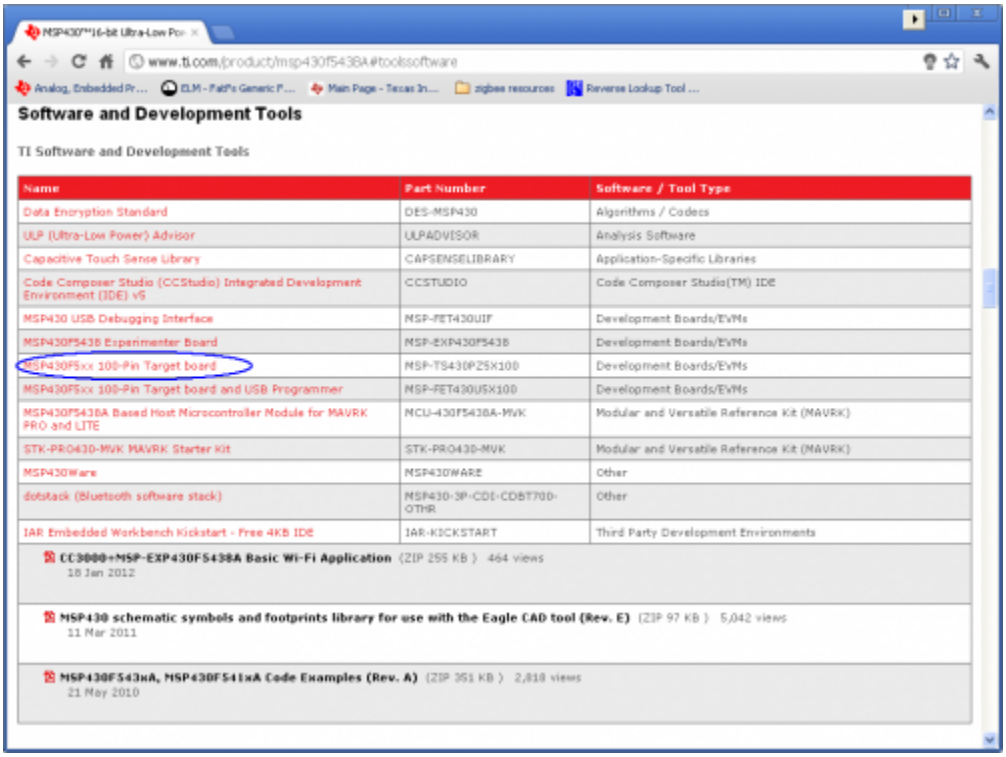
12. 我是 MSP430 的初学者，我如何用更加高效和快速的方法来开发我的应用？

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？

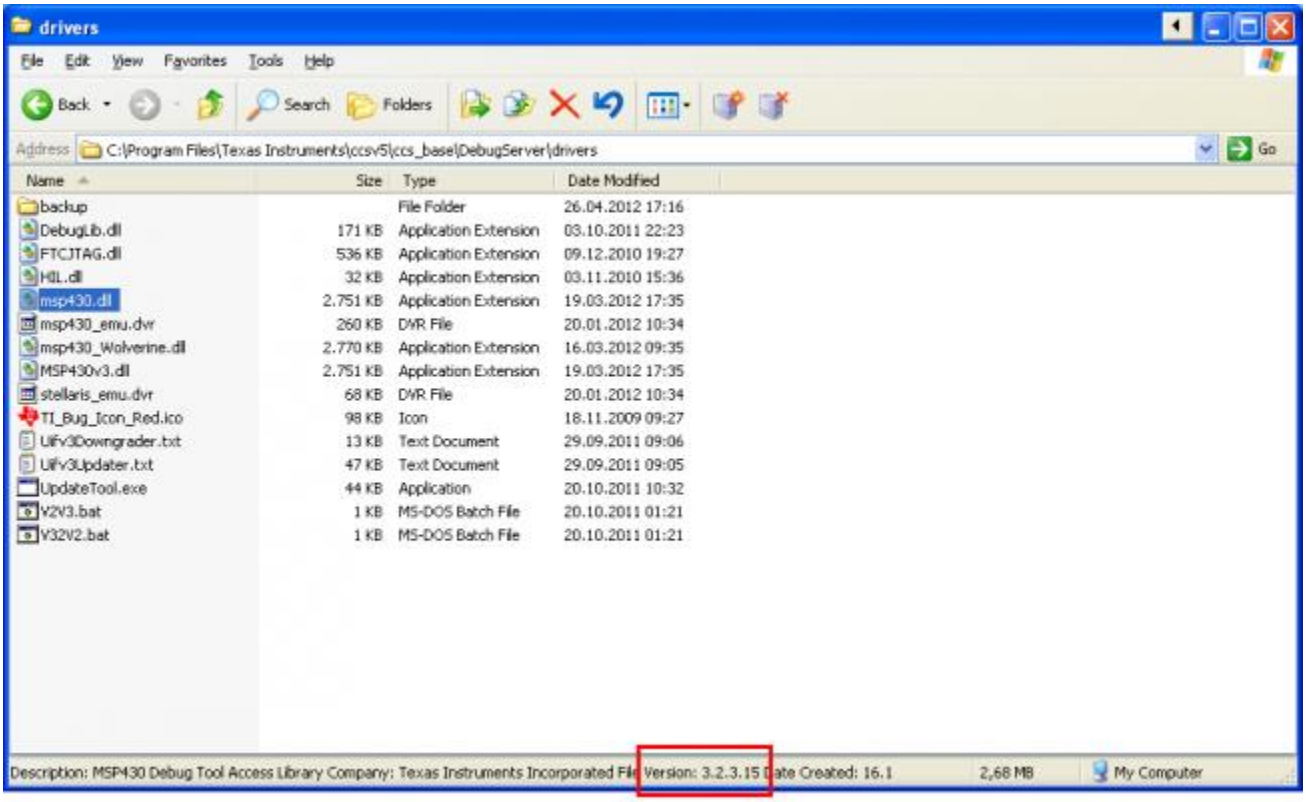
请参考以下维基网页： 生成_并_加载_MSP430_二进制_文件

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：

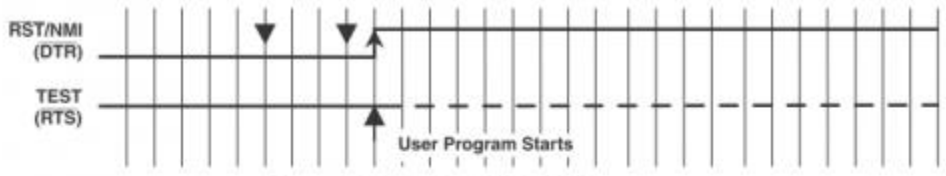


Figure 1-1. Standard RESET Sequence

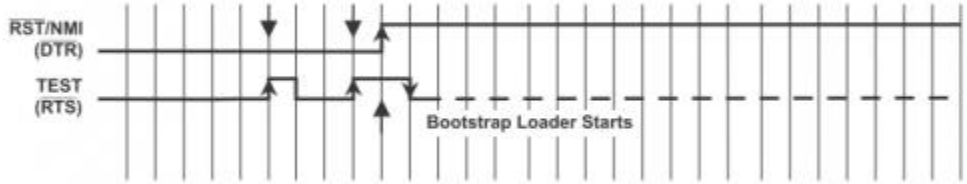


Figure 1-2. BSL Entry Sequence at Shared JTAG Pins

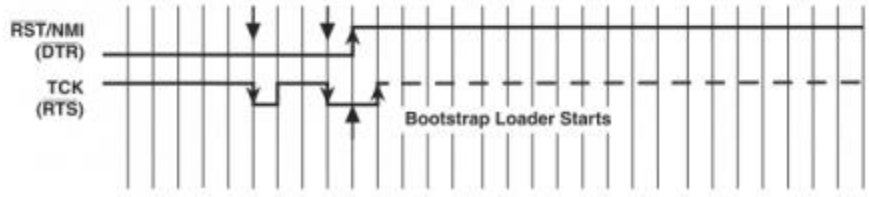


Figure 1-3. BSL Entry Sequence at Dedicated JTAG Pins

（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）

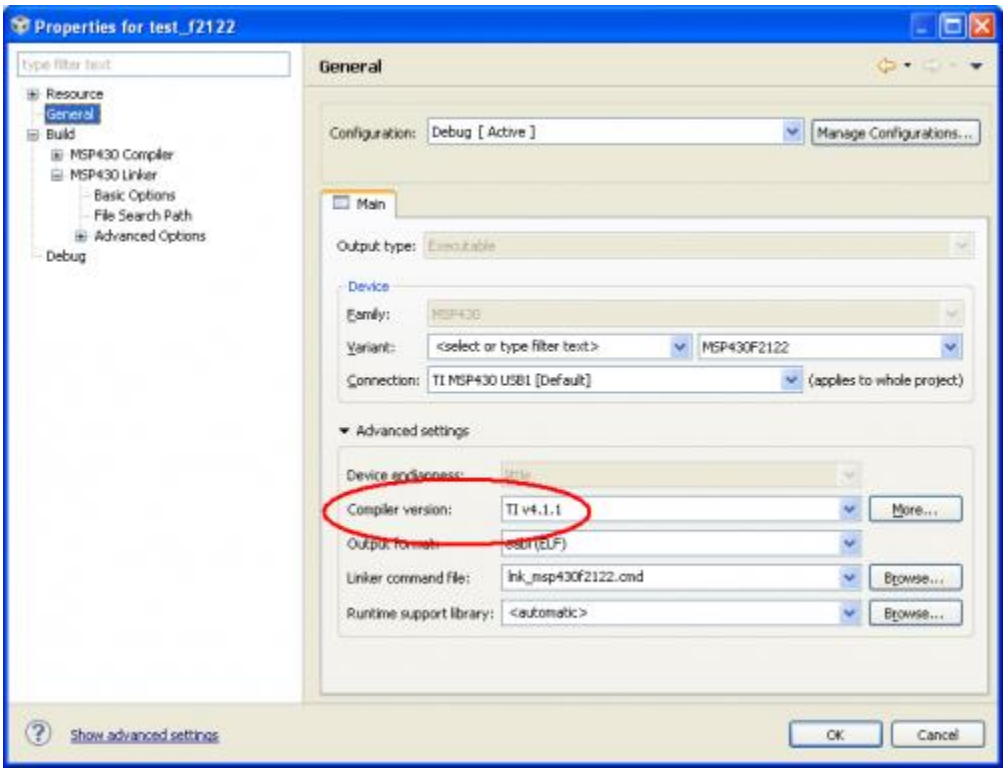
- 当器件由 VBUS 供电时，在满足以下两个条件中的任何一个时可调用 USB BSL：
  - 器件由 USB 供电且复位矢量为空。
  - 器件在 PUR 引脚被接至 VBUS 时加电。

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？

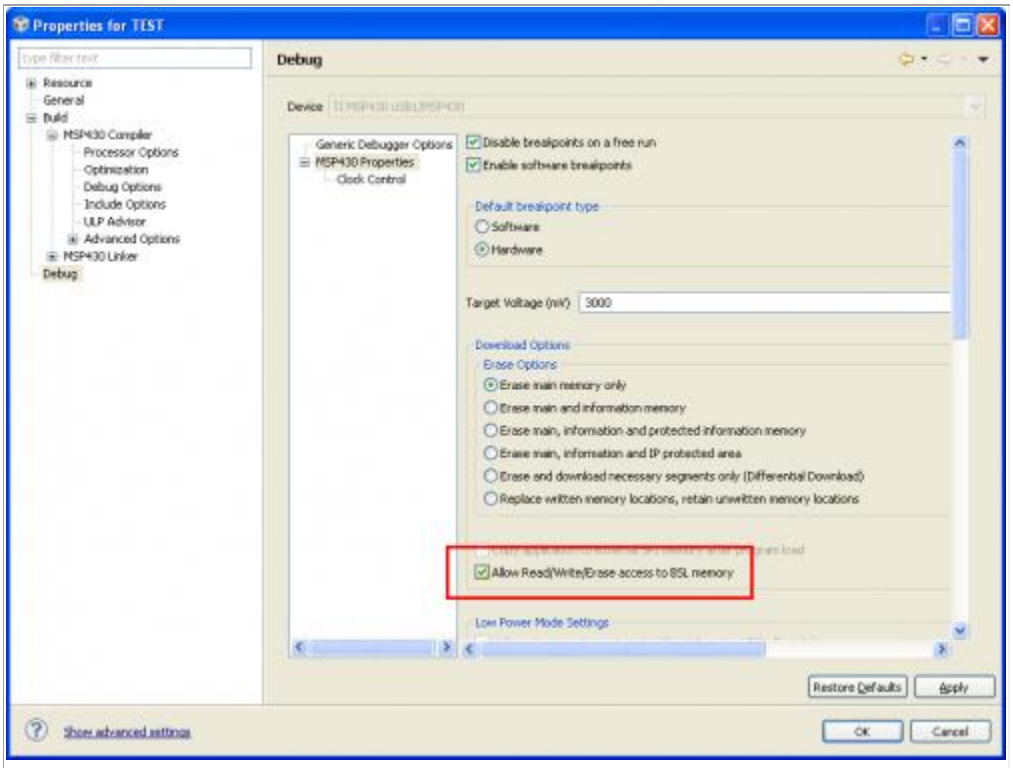
前往 "Project" -> "Properties"，并选择 "General" 选项，你可以在 "Compiler Version" 下找到此信息：

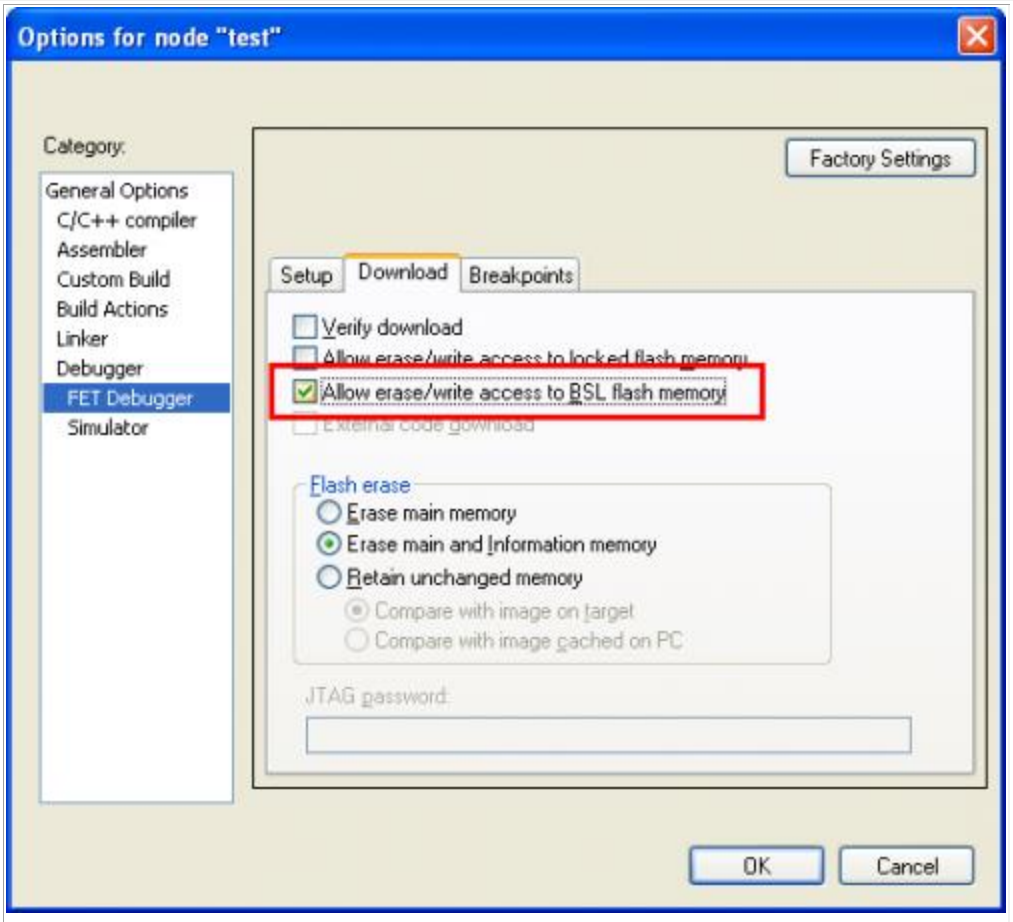


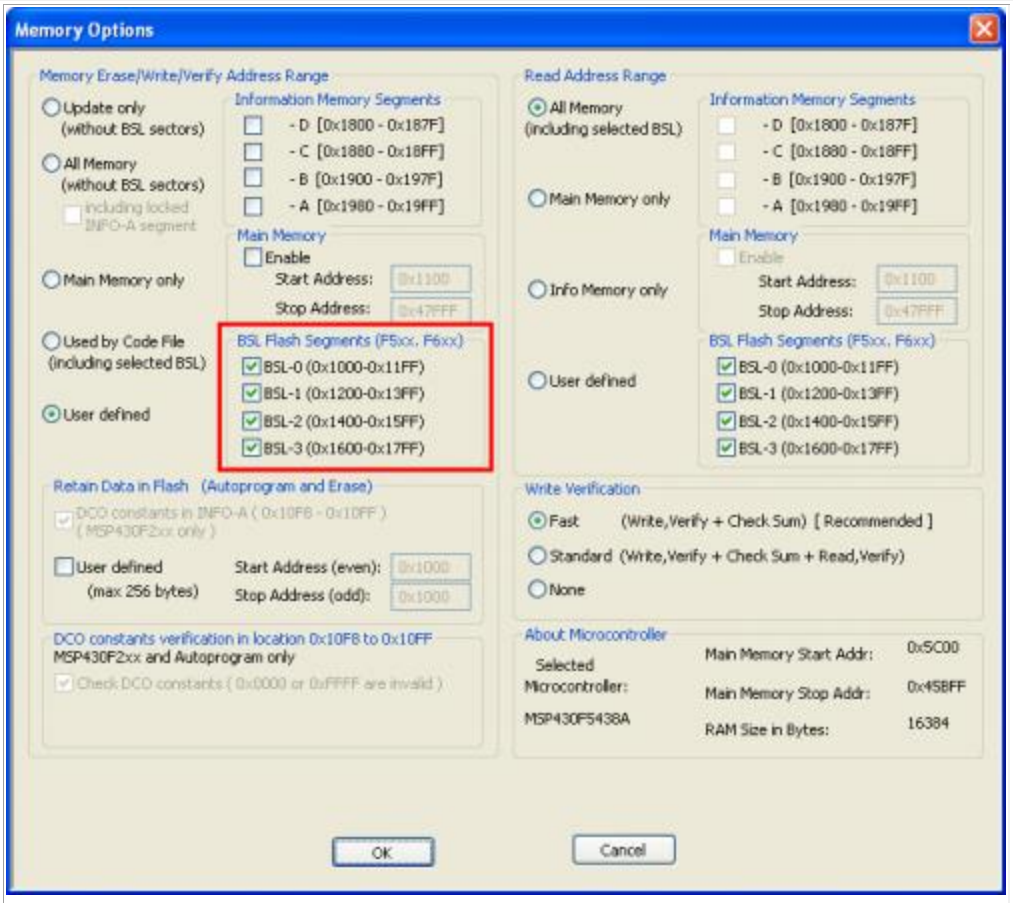
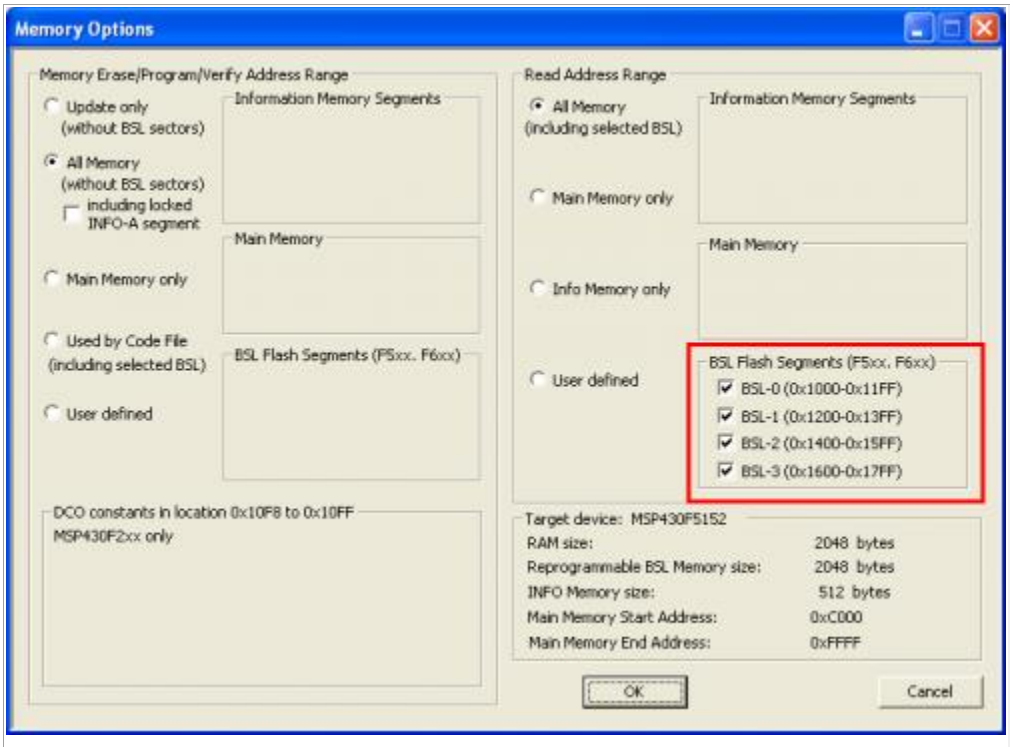
30. 如何启用到 MSP430F5xx/6xx 器件中 BSL 闪存存储器的访问？

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)	

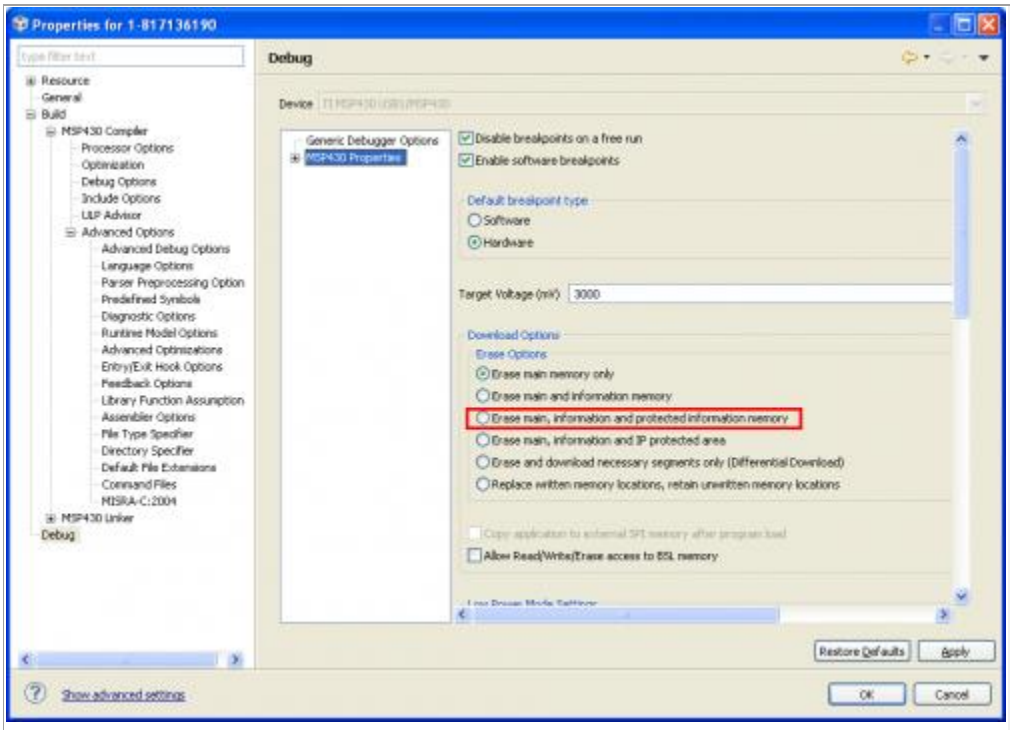
IAR EWB	 Options for node "test" Category: General Options C/C++ compiler Assembler Custom Build Build Actions Linker Debugger FET Debugger Simulator Factory Settings Setup Download Breakpoints Verify download Allow erase/write access to locked flash memory Allow erase/write access to BSL flash memory 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	

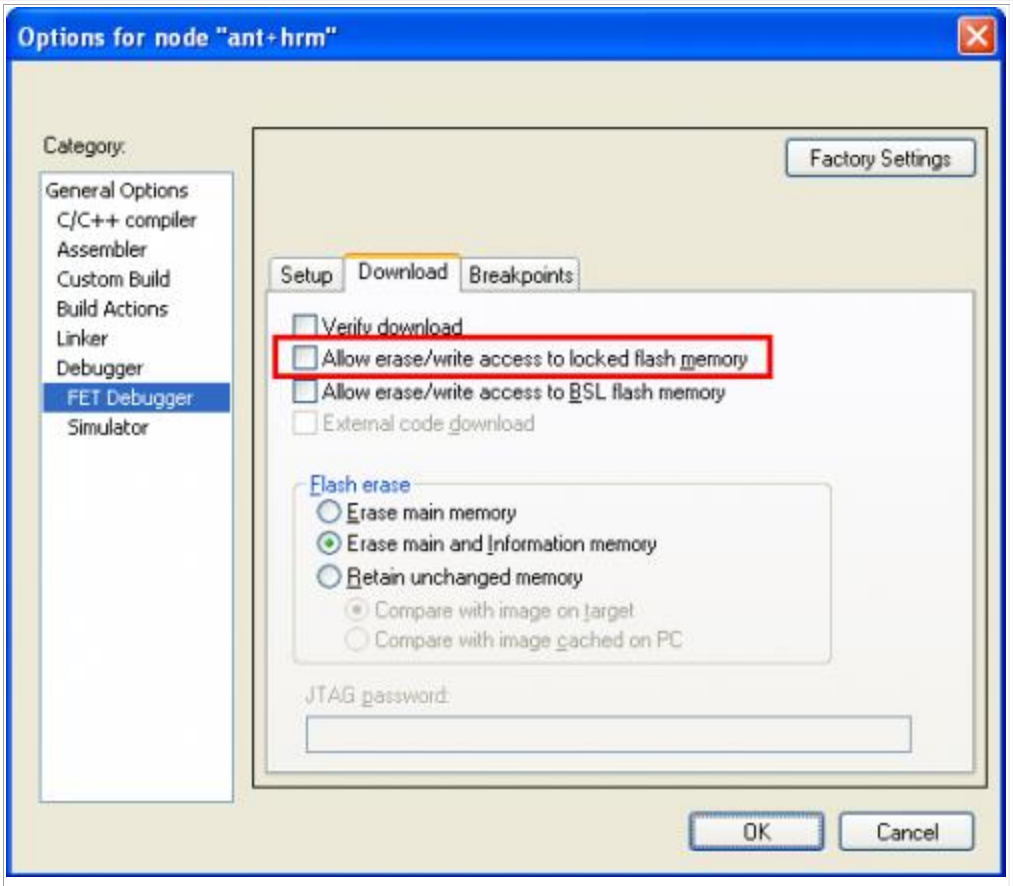
MSP430 Flasher	-b 参数

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

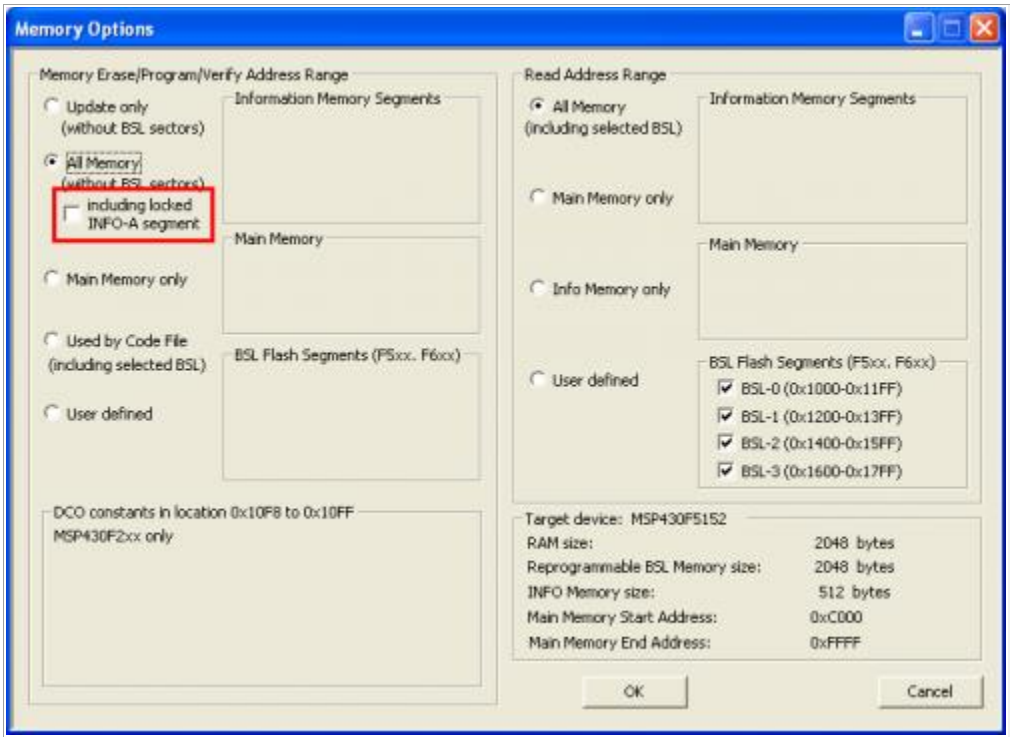
缺省情况下，对 MSP430F2xx/6xx 器件内包含校准数据的 INFOA 存储器的访问会受到 FCTL3 寄存器的 LOCKA 位的保护。 只要 LOCKA 位保持置位，任何批量擦除命令将不会擦除 infoA 存储器。 通常情况下，调试器/程序设计器有用来启用对 InfoA 存储器进行擦除操作的一个额外选项：

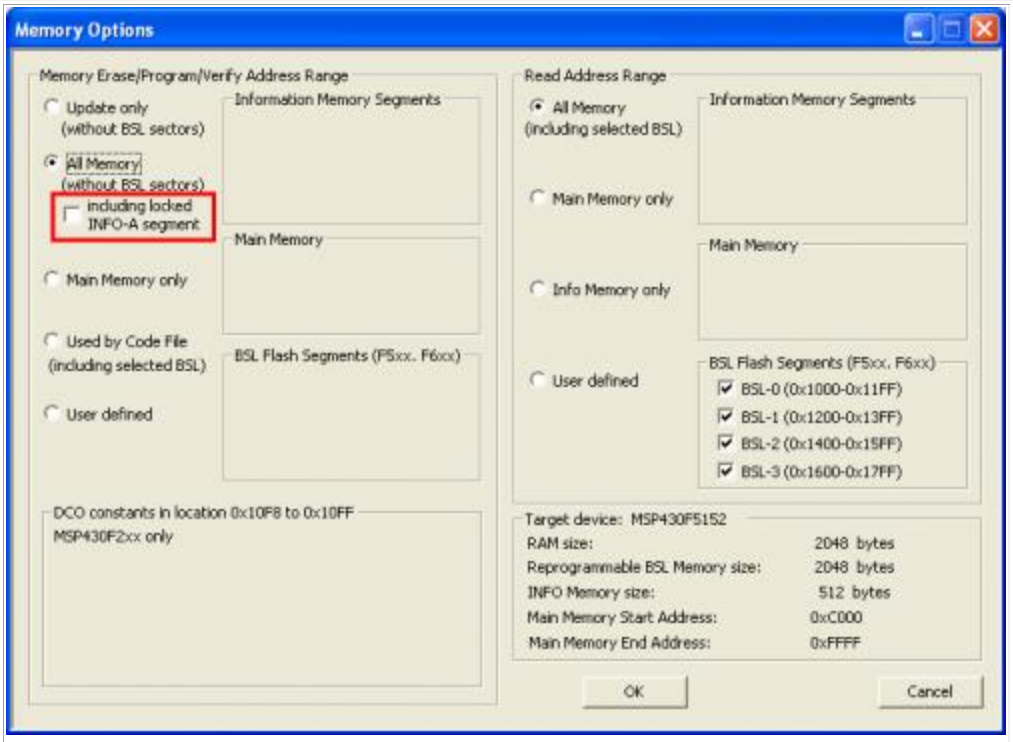
调试器/程序设计器	BSL 访问选项
CCS (Code Composer Studio)	

IAR EWB



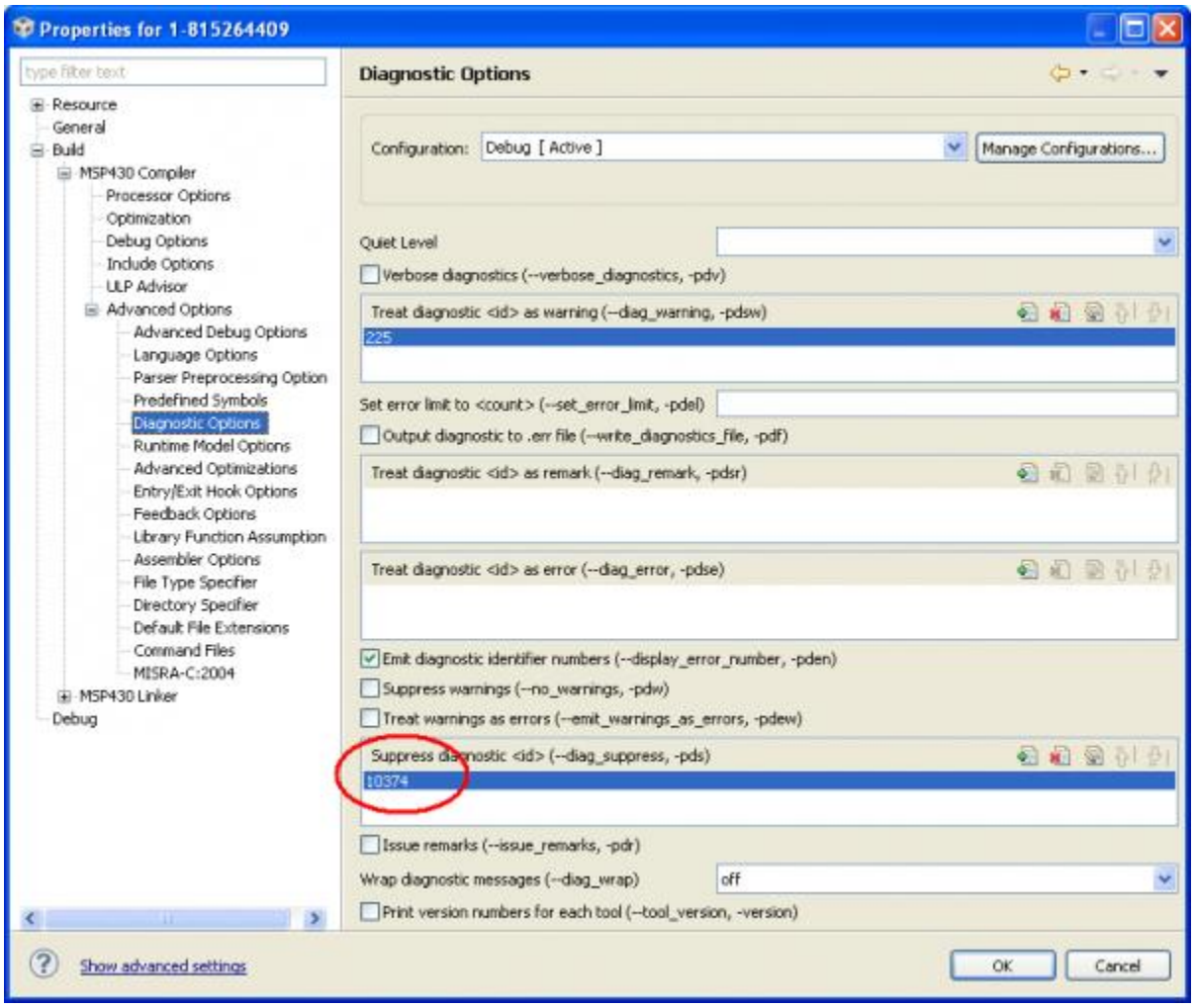
Elprotronic FET-Pro430



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：



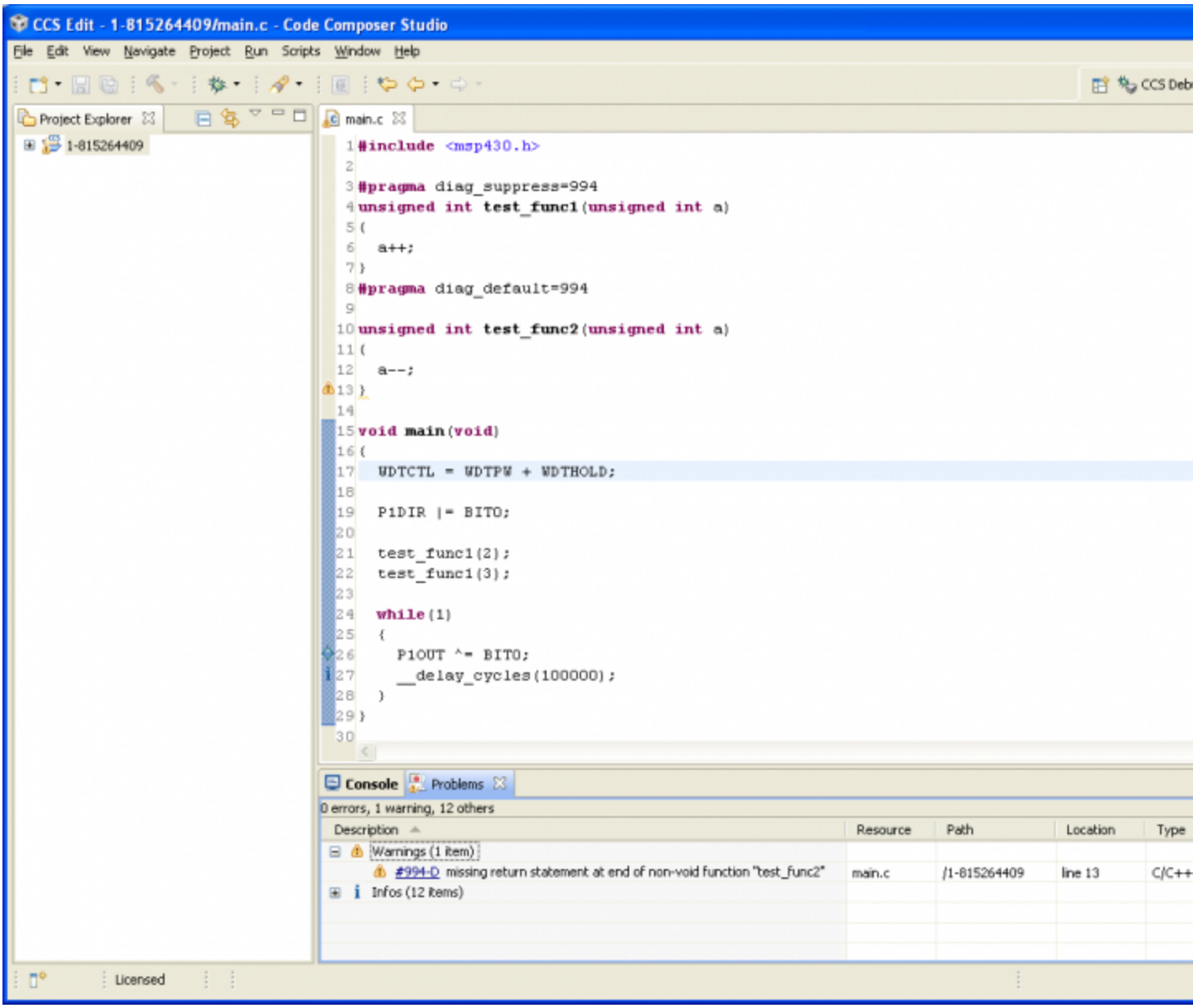
这将在 CCS 项目的整个源代码内全局阻止警告消息。 如果只应在特定代码部分中本地阻止警告消息，可使用 pragma **diag_suppress'** 和 **diag_default** ：

```
#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。

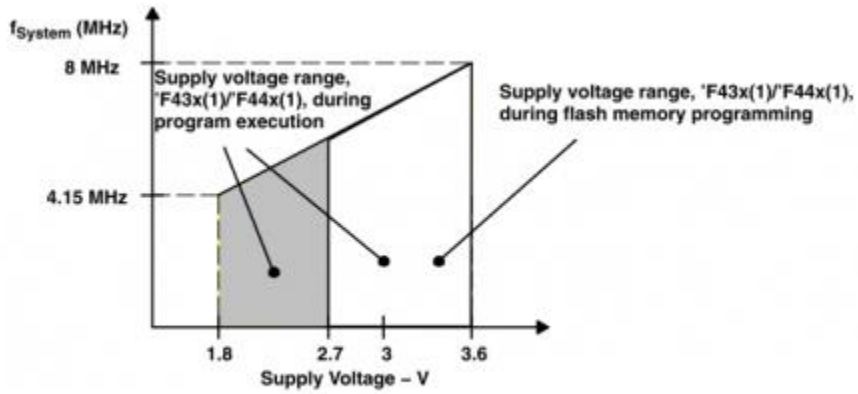
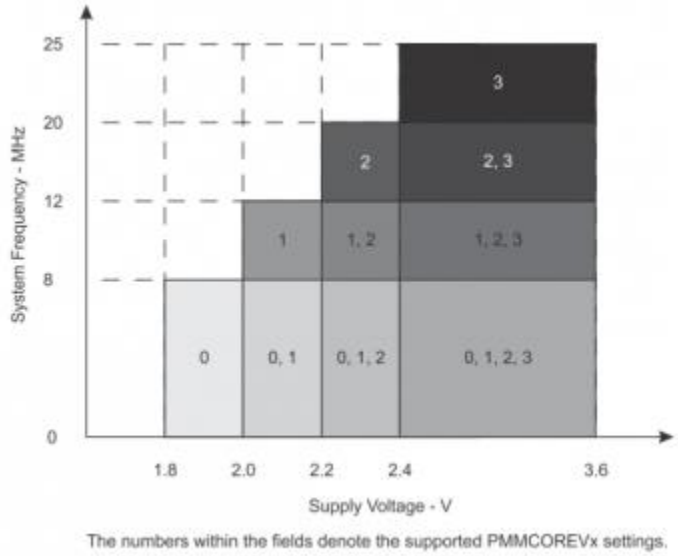


Figure 1. Frequency vs Supply Voltage, MSP430F43x(1) or MSP430F44x(1)

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。



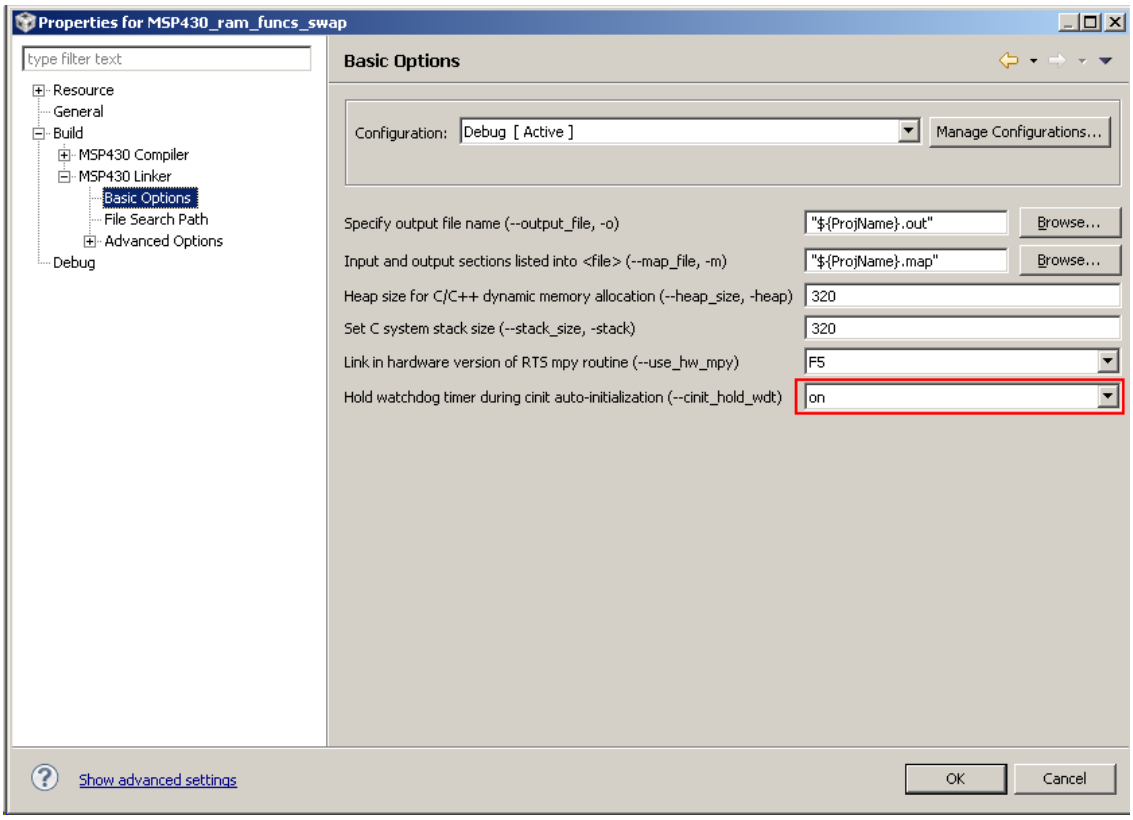
如果 CPU 以较高频率运行，并且电源电压在运行期间下降的话，会产生另外一个问题。 在这个情况下，只要电源电压下降到低于 CPU 高频运行所需的最小足够电压以下，就将器件保持在复位状态。 如果没有这么做的话，就不能保证器件工作正常，甚至有可能导致严重损坏，诸如闪存存储器损坏（请参考： MSP430_闪存_最佳_做法）。

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。

65. MSP430 内部 FLASH 的最小编程电压是多少？

MSP430 的工作电压范围为 1.8—3.6V，但必须在不低于 2.2V 的情况下工作才能满足内部 FLASH 的最小编程电压。

66. MSP430 如何校准 DCO 振荡频率？

MSP430 的 FLASH 是不能以位为单位进行编程的！MSP430 的 FLASH 的擦除通常是以段为单位的，而写入往往以字节或字为单位进行。但是 FRAM 型的 MSP430 却不同：FRAM 可以以位为单位，进行编程、读取或擦除操作。

67. MSP430 的波特率如何计算？

直接上网：
http://processors.wiki.ti.com/index.php/USCI_UART_Baud_Rate_Gen_Mode_Selection#USCI_UART_Calculator USCI UART baud rate calculator

68. MSP430 Timer_A 的中断矢量如何分配？

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 的头文件中有中断矢量的格式为 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)？是矢量号（0 = CCR0，1 = TAIFG & 其他 CCR）

69. MSP430 的 GPIO 口在复位状态中的功耗？

在数据表中并没有功耗/流耗的技术规格，只是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。

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

进入 LPM3.5 和 LPM4.5 模式，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 模式中唤醒时，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。

71. MSP430 如何在 USCI 模块传输完成时生成中断？

在 USCI 模块中，在 UCxTXBUF 数据字节被复制/被移入 TX 移位寄存器时，将产生发送中断。然而，有时需要在整个数据字节已经从 TX 移位寄存器中移出/检测时生成中断。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。

72. MSP430 如何实现软件复位？

- **看门狗：**
  #define SW_RESET() WDTCTL = WDT_MRST_0_064; while(1);
- **具有 PMM 功能的器件上使用 BOR 和 POR：**
  #define SW_RESET() PMMCTL0 = PMMPW + PMMSWBOR + (PMMCTL0 & 0x0003); #define SW_RESET() PMMCTL0 = PMMPW + PMMSWPOR + (PMMCTL0 & 0x0003);

73. MSP430 捕获功能和定时器的暂停应注意什么顺序？

当定时器暂停时，顺序应是先停止捕获功能，再停止定时器计数。捕获功能重新开始时，顺序是先开启捕获功能，再开始定时器计数。

74. MSP430 如何改变看门狗的定时时间？

MSP430 改变看门狗定时时间而不同时清除 WDTCNT 将导致不可预料的系统立即复位或中断。定时时间改变应伴随计数器清除，并在一条指令中完成。例如“MOV #05A0Ah,&WDTCTL”，如果先后分别进行清除和定时时间选择，则可能立即引起不可预料的系统复位或中断。（在正常工作时，改变时钟源可能导致 WDTCNT 额外的计数时钟）

75. MSP430 中断标志 P1IFG.0—P1IFG.7 和 P2IFG.0—P2IFG.7 在中断被接受时，是否会自动复位？

MSP430 中每一组中断标志 P1IFG.0—P1IFG.7 和 P2IFG.0—P2IFG.7 只用一个中断向量，它们都是多源中断向量。当中断被接受时，这些标志位不会自动复位。由中断服务程序确定服务的事件，并将相应的标志复位。任何外部中断事件必须等于或大于 1.5 倍 MCLK 的时间，以保证该中断请求被接受并使相应的中断标志置位。

76. MSP430 如何从外部晶振获取 MCLK？

在 PUC 复位信号之后，基本时钟模块使用 DCOCLK 作为 MCLK。如果需要更高的频率可以选用 LFXT1 时钟或 XT2 时钟（注：MSP430G22x0 中不含有 LFXT1 系列振荡器，XT2 时钟部分型号没有，具体详见所用单片机的数据手册）。从 DCO 切换到外部晶振的步骤如下：

（1）使能对应的振荡器并选择合适的工作频段；
（2）将状态寄存器 SR 中的 OFIFG 位清零；
（3）延时至少 50us；
（4）检测 OFIFG 标志位是否清零。如果清零，则可以选择对应的振荡器作为 MCLK；
否则重复步骤（2）—步骤（4）。

77. MSP430 的复位信号在哪些情况下会发生？

MSP430 的复位信号有两种，分别是上电复位信号 POR 和上电清除信号 PUC。对于上电复位信号 POR，它只在以下 3 种情况下发生：

- 在芯片上时
- RST/NMI 设置成复位模式，在 RST/NMI 引脚上出现低电平复位信号
- 电源电压监测片上外设模块 SVS 的 POR 使能位 PORON=1 时，并监测到低电压状态发生。

对于上电清除信号 PUC，能触发的 PUC 的事件为：

- 发生上电复位信号
- 看门狗定时时间到
- 看门狗定时器的配置寄存器写入错误的安全密码
- FLASH 存储器的寄存器写入错误的安全密码
- CPU 从外设地址范围 0H--01FFH 取数据。

78. MSP430 内部 FLASH 支持几种编程方式？

MSP430 内部 FLASH 同时具备位寻址、字节寻址和字寻址，对应这 3 种寻址方式，具备位编程、字节编程和字编程 3 种编程方式。

79. MSP430 片上集成的低频振荡器 VLOCLK 输出的振荡频率稳定性怎样？

VLOCLK 是片上集成的低功耗低频振荡器，典型的振荡频率为 12KHz。VLOCLK 受工作电压和温度的影响较大，振荡电源电压漂移约为 4%/V，频率温度漂移为 5%/℃。

80. MSP430 调试接口 JTAG 和 SBW 如何与仿真器连接？

对于带有 JTAG 口的芯片，JTAG 引脚如下定义：
TCK——测试时钟输入；
TDI——测试数据输入；
TDO——测试数据输出；
TMS——测试模式选择（用来设置 JTAG 口处于某种特定的测试模式）
TRST——测试复位，输入引脚，低电平有效（此引脚可选）
因此，仿真器与芯片引脚连接方法：仿真器上的 TDO，TDI，TMS，TCK，GND，RESET 分别连接单片机上的 TDO/TDI，TDI/TCLK，TMS，TCK，GND，RST/NMI。对于引脚数较少的单片机（比如 MSP430G2553），可用两线制下载的解决方案 SBW，连接方法为：将单片机的 SBWTDIO 和 SBWTCK 分别与仿真器上的 TDO 和 TCK 引脚连接起来即可。（如果单片机靠仿真器供电的话，则需要将单片机的 VCC 接仿真器的第二引脚 VCC 上）

81. 如何利用 MSP430 实现电容触摸按键？

电容式触摸感应的“按键”实际只是 PCB 上的一小块“覆铜焊盘”，当手指触摸到 PCB“覆铜焊盘”部分时,手指将会影响电容的电场,相当于在两个电容极板间增加了一部分介质,使电容值增大,电容式触摸感应按键原理即通过检测这个电容值的变化达到识别有无手指按下的目的。第一种用于测量电容触摸传感器的方法就是使用振荡器。从根本上说，通过将 MSP430 的片上比较器和电容传感器用作调优元件，可以构建简单的弛张振荡器。传感器的任何电容变化都有对应的变化频率，通过使用 MSP430 的内部 Timer_A 硬件可以测量该变化。第一法是测量电容触摸传感器的方法就是使用振荡器。从根本上说，通过将 MSP430 的片上

比较器和电容传感器用作调优元件，可以构建简单的弛张振荡器。传感器的任何电容变化都有对应的变化频率，通过使用 MSP430 的内部 Timer_A 硬件可以测量该变化。

82. MSP430 不同的睡眠模式有什么差异？

首先不同的模式外设关闭的程度不同，唤醒所需的时间也不同。不同的 MSP430 系列有不同的数据和低功耗模式，这个需要参考相应型号的 datasheet。

83. MSP430 系统时钟 ACLK、MCLK、SMCLK 有什么区别？

MSP430 基础时钟模块包含以下多种时钟输入源。

- LFXT1CLK: 外部晶振或时钟 1 低频时钟源 低频模式：32768Hz 高频模式：(400KHz-16MHz)
- XT2CLK: 外部晶振或时钟 2 高频时钟源(400KHz-16MHz)
- DCOCLK: 内部数字 RC 振荡器,复位值 1.1MHz
- VLOCLK: 内部低功耗振荡器 12KHz
- 注：MSP430x20xx: LFXT1 不支持 HF 模式, XT2 不支持, ROSC 不支持.

(1)LFXT1CLK 低频时钟源：由 LFXT1 振荡器产生(如图 2 所示)。通过软件将状态寄存器中 OSCOff 复位后，LFXT1 开始工作，即系统采用低频工作。如果 LFXT1CLK 没有用作 SMCLK 或 MCLK 信号，则可以用软件将 OSCOff 置位，禁止 LFXT1 工作。

(2)XT2CLK 高频时钟源：由 XT2 振荡器产生。它产生时钟信号 XT2CLK，其工作特性与 LFXT1 振荡器工作在高频模式时类似。可简单地通过软件设置 XT2 振荡器是否工作，当 XT2CLK 没有用作 SMCLK 或 MCLK 信号时，关闭 XT2，选择其他时钟源。

(3)DCOCLK 数字控制 RC 振荡器。由集成在时钟模块中的 DCO 振荡器产生。DCO 振荡器是一个 RC 振荡器，频率可以通过软件调节，其控制逻辑如图 3 所示。当振荡器 LFXT1、XT2 被禁止或失效时，DCO 振荡器被自动选作 MCLK 的时钟源。因此由振荡器失效引起的系统中断请求可以得到响应，甚至在 CPU 关闭的情况下也能得到处理。

由基础时钟模块可以提供系统所需的 3 种时钟信号，即：ACLK、MCLK、SMCLK。其中辅助时钟 ACLK 是 LFXT1CLK 信号经 1、2、4、8 分频后得到的。ACLK 可由软件选作各个外围模块的时钟信号，一般用于低速外设;系统主时钟 MCLK 可由软件选择来自 LFXT1CLK、XT2CLK、DCOCLK 三者之一，然后经 1、2、4、8 分频得到。MCLK 主要用于 CPU 和系统。子系统时钟 SMCLK 可由软件选择来自 LFXT1CLK 和 DCOCLK，或者 XT2CLK 和 DCOCLK，然后经 1、2、4、8 分频得到，主要用于高速外设模块。

84. IAR 编译环境下 MSP430 如何简单的实现精准的延时？

IAR for MSP430 编译器提供了一个编译器内联的精确延时函数(并非真正的函数)以提供用户精确延时使用,该函数原型是:__intrinsic void __delay_cycles(unsigned long __cycles);该内部函数实现__cycles 个 CPU 周期的延时，但对于该参数的设置,我要陈述一下:

- __cycles 需要我们传递的是 CPU 运行的周期个数。
- __delay_cycles 并不是真正的函数, 只是提供编译器内联展开,该函数并不支持变量参数, 其参数只能是常数。所以如果需要传递变量的话，需要将这个“函数”进行封装，但是这样势必造成延时精度的改变。

例如：

```
void delay_ms(unsigned int delay)
{    while (delay--)
 {
    __delay_cycles(PUT_CPU_CLOCK_SPEED_IN_HZ_DIVIDED_BY_1000_HERE);
```

```
                }
            }
```

如果需要实现精准的延时，可以考虑使用定时器。这样的话，你可以在等待的时候进入休眠模式，进一步降低系统的功耗

85. MSP430 单片机的干扰问题？

- 干扰源。指产生干扰的元件、设备或信号，用数学语言描述如下：du/dt,di/dt 大的地方就是干扰源，如雷电，继电器，可控硅，电机，高频时钟等都可能成为干扰源。
- 传播路径。指干扰从干扰源传播到敏感器件的通路或媒介。典型的干扰传播路径是通过导线的传导和空间的辐射。
- 敏感器件。指容易被干扰的对象。如：A/D、D/A 变换器，单片机，数字 IC，弱信号放大器等。干扰的分类有好多种，通常可以按照噪声产生的原因，传导方式，波形特性等进行不同的分类。按产生的原因：可分为放电噪声，高频振荡噪声，浪涌噪声。按传导方式：可分为共模噪声和串模噪声。按波形：可分为持续正弦波、脉冲电压、脉冲序列等。干扰源产生的干扰信号是通过一定的耦合通道才对测控系统产生作用的。因此，有必要看看干扰源和被干扰对象之间的传递方式。

干扰的耦合方式，无非是通过导线，空间、公共线等方式，细分下来有以下几种：

- 直接耦合。这是最直接的方式，也是系统中存在最普遍的一种方式。比如干扰信号通过电源线侵入系统。对于这种常见的形式，最有效的方法就是去耦电路。
- 公共阻抗耦合。这也是常见的耦合方式，这种形式常常发生在两个电路电流有共同通路的情况。为了防止这种耦合，通常在电路设计上就要考虑，使干扰源和被干扰对象没有公共阻抗。
- 电容耦合。又长称电场耦合或静电耦合，是由于分布电容的存在而产生的耦合。
- 电磁感应耦合。又称磁场耦合，是由于分布电磁感应而产生的耦合。
- 漏电耦合。这种耦合是纯电阻性的，在绝缘不好时就会发生。

86. MSP430FRAM 的内部时钟能否在电容下供电下继续工作呢？

可以继续工作，根据手册时钟可以在电容供电情况下继续工作。

87. 当 MSP430F5969 在功耗模式下，能否有足够的能力驱动外设？

在低功耗模式下 MSP430F5969 的串口功能将会关闭，但不影响其内部的正常工作，因此还可以驱动外设，具有一定的驱动能力。

88. 我到哪里可以申请到免费的 MCU 评估板？

可以关注 EEWORLD 等网站的 TI 活动，积极参加，即有机会获取。

89. Launch pad 的板载串口转 USB，能否单独工作？

能够作为串口单独使用，具体接法可以参考 Launch pad 原理图。

90. Launchpad 开发板能否用于其他 MSP430 的调试以及下载？

支持双线下载的 TI-MCU 均可以用 Launch pad 开发板下载以及调试。

91. MSP430 低功耗设备功耗不够低的原因？

- 作为输出口上拉电阻是不是太小造成的
- 不用的 IO 口最好设置为输出
- 进入的低功耗模式选择错误；

92. MSP430 上电不工作是什么原因？

如果晶振起振，电源也没完问题，检查下是否是复位电路错误，和 51 内核单片机不同，MSP430 是低电平复位。

93. TI MSP430 FLASH 能存储用户数据吗

MSP430 的单片机内部专门留有一段 Flash 区域(information memory)，用于存放掉电后需要永久保存的数据。利用 430 内部的 Flash 控制器，可以完成较大容量的数据记录、用户设置参数在掉电后的保存等功能。

94. 如何设计 MSP430 ADC 的驱动电路？

MSP430 内部的 adc 是伪差分 SAR ADC，输入阻抗较低（数欧姆以内）时，无需设计缓冲器，而当被测信号内阻较高（数千欧姆以上）时将引起较大误差，此时应使用低失调电压、轨到轨输入/输出的运放做为缓冲器，以提高测量应用的准确性，降低非线性度。

95. MSP430G2 系类单片机编写字符显示函数时，显示堆栈太小错误怎么办？

将字符大数组前面加上 const 关键字，让其变成常量，既可以防止被更改，也可以减小堆栈的使用。

96. MSP430G2 要在一个端口号中使用两个 AD 采样，怎么做？

MSP430G2 中多个端口中可用 AD 采用，但是却只有一个寄存器储存收集到的值，所以如果一定要这样的话，那就只能在采集完一个 AD 值后重新，初始化 AD 寄存器，然后再采集另一个端口的 AD 值，不过还有一种方法，那就是 430 可以使用轮流采集 AD 值方法，使用的好，也不错。

97. CCS 中出现#10099-D program will not fit into 怎么回事？

最有可能程序中堆栈占用过大，可能定义有大型全局数组，比如字符的显示，解决办法：减小数组大小，或者增加堆栈，可在 CCS 软件中设置。

98. 用 MSP430 做的最小系统版为啥不能工作？

MSP430 中的复位脚与测试脚没有连接好，悬空是不能工作的。

99. MSP430 的串口发送数据发现会多次发送重复数据，例如发送 1 和 2 结果显示 1111122222，为什么？

原因是当数据正在发送中，UTXIFG0=1，此时不能再发送数据，必须等当前数据发送完毕（UTXIFG0=0）才能进行发送。
正确的程序如下：

```
void senddata(uchar data_buf){while (!(IFG2&UCA0TXIFG)); UCA0TXBUF = data_buf;}
```

100. 可以用 G2 系列的 LaunchPad 调试其他系列的 MSP430 芯片么？

可以，G2 系列 Launchpad 板载 SBW 调试电路，可以对支持 SBW 接口的 msp430 进行程序下载和调试仿真。将 launchpad 上的 G2 系列 430IC 拔掉（或者你把 launchpad 上的 5 个跳线帽断开也行，当然这样 launchpad 上的 RESET 键就不能用了，推荐拔芯片的方案），将 launchpad 与其他支持 SBW 的 MSP430 单片机（如 MSP430F5438A）的 VCC、GND、TEST、RESET 脚用杜邦线连起来即可。注意，因为 1.4 版的 G2LaunchaPad 和 1.5 版的复位电路电容值有所不同，以上方法只适用于 1.5 版的 G2LaunchPad。

101. time A 定时器输出模块中 EQUx 和 EQU0 有什么区别？

捕获/比较器在比较模式时设置 EQUX 信号有差别：当 TAR 的值大于或等于 CCR0 的中的数字时，EQU0=1 当 TAR 的值等于相应的 CCR1 或 CCR2 的值时，EQU1=1 或 EQU2=1EQUx 和 EQU0 它们是用来控制输出单元的，软件中可以不用设置，由硬件自动触发。综上所述，EQUX 可以理解为一个信号，是为了描述方便加上的一个名字

102. 怎么利用 F155 实现计数脉冲功能 ？

- 用定时器做的话，也可以将定时器设置在捕获状态下，如上升沿捕获，当定时器捕获到上升沿时会产生一次中断，此时定时器会记录当前计数器的值到 CCRX，您可以把这个值放到指定的变量里，两次中断的记数差值就是你实际计数个数，这样你可以根据你计数个数调整增益，另外如果要计算时间的话只要将个数乘以定时器时钟就可以。
- 用捕获的方式就不用设置 1mS~1S 啦。你可以把 8MHZ 当 TIMERA 时钟，最小可以到 1/8uS，另外由于 1MHZ 捕获信号与 8MHZ 比较接近，如果采用两次捕获计算一个脉冲宽度精度不高，可以多采几次，如 100 次求得平均，这样精度会高很多。

103. MSP430 有 eeprom 么，要保存信息怎么办？

MSP430 基本型号没有 eeprom，但是都有内部的 flash 区域，可以擦写，记录一些必要的参数，防止掉电遗失。

104. MSP430 内部 dco 的频率稳定么？偏差多少？

常温下大概+-3.5%的飘逸，做低码率的通讯或解码没有问题。

105. cc1110 设计不能下载程序，如何解决？

我经历的需要注意几点：

- 检测程序下载口连线排序正确性
- RESET_N 下拉 10k
- 芯片地是否焊接好
- 管脚是否焊接好

106. 使用 cc1110 开发套件中，433M 天线使用的是柱状天线怎么修改？

在实际使用中需要将天线微型化，可以使用 pcb 天线，在设计板子时，将天线沿 pcb 外部边框进行设计。长度为波长的 1/4 即可。

107. MSP430F149 之看门狗（WDT）的模块问题？

1.看门狗定时器寄存器计数单元 WDTCNT：
WDTCNT 是 16 位增计数器，由 MSP430 系列单片机选定的时钟电路所产生的固定周期脉冲信号对计数器进行加计数。如果计数器事先被预置的初始状态不同，那么从开始计数到计数溢出为止所用的时间就不同。WDTCNT 不能直接通过软件存取，必须通过看门狗定时器的控制寄存器 WDTCTL 来控制。（2）控制寄存器 WDTCTL　　WDTCTL 由两部分组成：高八位被用作口令，低八位是对 WDT 操作的控制指令。要领如操作 WDT 的控制指令，出于安全原因必须先正确写入高字节看门狗指令，口令问 5AH。如果口令写错了，将导致系统复位。读 WDTCTL 时，不需要口令，可直接读取地址 120H 中的内容，独处数据低字节为 WDTCTL 的值，高字节始终未 69H.。WDTCTL 除了看门狗定时器的控制委外，还有灵位用于设置 NMI 引脚的功能。
2.看门狗定时器的操作：
用户可以通过 WDTCTL 寄存器中的 TMSEL 和 HOLD 控制位设定 WDT 工作在看门狗模式、定时器模式和低功耗模式。看门狗模式　　在上电复位或系统复位时，WDTCNT 和 WDTCTL 两个寄存器内容被全部清除（晶振为 32 768 HZ，SMCLK=1 HZ）。这些情况将导致看门狗定时器的自动运行并进入看门狗模式。因此，用户软件一班都要进行如下操作：　　进行看门狗定时器的初始化，设置合适的时间（通过 SSEL、ISO、IS1 位来确定）。　　周期性的对 WDTCNT 清零，防止看门狗定时器溢出，保证看门狗定时器的正常使用。　　在看门狗模式下，如果计时器超过了定时时间，就会产生服 Wie 和激活系统上电清楚信号，系统从上电复位的地址重启动。　　如果系统不用看门狗功能，ing 改在系统看是处禁止看门狗功能。定时器模式　　WDTCTL 的 TMSEL 位置可以选择定时器模式。这一模式产生选定时间的周期性中断。定时时间可以通过 WDTCTL 的 CNCTL 位置位来开始。　　改变定时时间而不同时清除 WDTCNT 将导致不可预料的系统立即复位或者中断。定时时间改变应伴随计数器清楚，并在一条指令中完成。　　如果先后分别进行清楚和定时时间选择，则不能立即引起不可预料的系统复位或者中断。　　在正常工作时该变时钟源可能导致 WDTCNT 额外的计数时钟。低功耗模式　　当系统不需要 WDT 做看门狗和定时器时，可关闭 WDDT 以减小功耗。控制位 HOLD=1 时关闭 WDT，这时看门狗停止工作。
3.看门狗定时器的中断控制功能：
看门狗定时器用到 SFR 地址的两位：　　中断标志位 WDTIFG 位于 IFG1.0。初始状态为复位。　　中断允许位 WDTIE 位于 IE1.0，初始状态为复位。　　WDTCTL 的控制位 NMI 和 NMIES 与中断功能相关，NMIES 位于 IE1.4，MNIIFG 位于 IFG1.4。前者的优先级低于后者，另外，两者的中断向量地址不同，使用时请参见相关芯片手册。

108. TI Launchpad 的 SBW 接口适合仿真哪些 MSP430 芯片？

TI Launchpad 的 SBW 调试器支持所有带 SBW 接口的 MSP430 器件。TI 的 MSP430 LaunchPad (MSP-EXP430G2) 的确是一款性价比极高的开发工具，仅售 4.3 美元，还免运费，非常适合学生朋友们。

109. MSP430 SPI 或 UART 的速度？

在 SPI 主模式下,通信速率可以达到 4Mbps,而在 UART 模式下,速率也可达到 2Mbps。USART 可进行配置,以便同时支持同步 (SPI) 与异步 (UART) 操作,并且可从几个内部及外部时钟源(与 CPU 时钟无关)中进行选择。在 SPI 主模式下,USART 的运行速率可达到应用时钟的 1/2。例如,如果使用 8MHz 时钟,则 SPI 主模式的传输速率可达到 4Mbps。在 UART 模式下,实现可靠通信至少要求每位 3 或 4 个时钟。例如,8MHz 时钟除以 4 可以支持高达 2Mbps 的速率。MSP430xxxx 用户指南中提供了有关 USART 功能的完整说明,其网址是:http:///msp430 获得。此外,还可访问 MSP430 网站,以查找可提供现成引导加载程序工具或解决方案的第三方公司。

110. 在何处可以找到 BSDL 文件来构建 JTAG 链?

所有 MSP430 均具有仅用于程序开发与快闪编程的 JTAG 接口。但这个 JTAG 接口并不 完全与 IEEE 1149.1 兼容。例如,任何 MSP430 均没有边界扫描单元 (Boundary Scan Cell)。我们仅支持所需的命令 BYPASS,但不支持其它所需的命令:EXTEST 与 SAMPLE/PRELOAD。

111. 在 MSP430 引导加载程序 (BSL)通信中使用十六进制 80？

十六进制 80 在每次传输之前均作为同步字符进行发送。该器件通过十六进制 90 进行确认。然后再发送数据帧。每个帧均以报头字节 = 十六进制 80 开头。其它字节的帧则紧跟在十六进制 80 报头的后面。MSP430 website 上的“MSP430 引导加载程序的功能”应用手册 SLAA089 中定义了 BSL 数据帧的正确格式。对随“引导加载程序在 MSP430 w/Flash 中的应用-硬件与软件建议”应用手册 SLAA096 一起提供的代码进行仔细检查后会发现,这是最佳的技术。该应用手册中的软件与硬件均经过测试,证明其可以正常工作。

112. MSP430 ADC12 模块的速度是多少？

ADC12 的转换速率是转换所需的 ADC12CLK 以及时钟的一项功能。ADC12CLK 的近似最小值与最大值分别为 500kHz 及 6.5MHZ。速度最快的整个转换过程可以在 17 个周期内完成(13 个周期进行转换,4 个周期进行采样及保持)。6.5MHz/17 = 382ksps。ADC12 的运行速率不能低于最小值的 ADC12CLK,但在软件的控制下,采样门可以无限制保持打开状态。如欲了解有关采样与转换时间规范的更多详情,敬请参阅数据表。

113. MSP430 I/O 引脚的汲极电流与源极电流的问题？

MSP430 未指定来自 I/O 引脚的最大绝对电流。如欲了解 Voh 与 Vol 的规范,敬请参阅数据表。其中显示了每个 I/O 引脚均可提供几毫安的电流,但输出电压将随着电流的增大而发生变化。这些规格的附注通常提供了要维持特定电压,所有组合的输出提供的最大总电流。MSP430 I/O 不适于驱动高电流的 20mA LED。

114. MSP430 I/O 引脚的电流

MSP430 没有明确规定 I/O 引脚的最大输入输出电流。其实每个 I/O 引脚均可提供几毫安的电流，但输出电压将随着电流的增大而发生变化。这些规格的附注通常提供了要维持特定电压，所有组合的输出提供的最大总电流。除非特殊说明，MSP430 I/O 不适于驱动高电流的 20mA LED。

115. JTAG 与 I/O 功能之间的 MSP430 引脚复用？

四个引脚 P1.7 - P1.4 在 20 与 28 引脚 MSP430F1xx 器件上均同时具有 I/O 与 JTAG 功能。这些引脚的默认功能是,当器件通电时具有 I/O 功能。当测试引脚拉高时,则将这些引脚选为 JTAG。当使用交互式系统内调试程序时,这些器件的 FET 会将这些引脚处于 JTAG 模式下。如欲了解有关在使用调试程序时从 JTAG 模式发布引脚的信息,敬请参阅《FET 工具用户指南》。

116. MSP430 单片机需要加密吗？

430 单片机需要加密吗？MSP430 的保密是通过加密熔丝实现的，在下载完程序后可以用 JTAG 烧断熔丝，熔丝一旦被烧断，JTAG 接口绝大部分功能失效，就再也不能通过它进行编程了。此时要想读出，烧写程序，只能通过 BSL。通过 BSL 擦除所有 Flash 信息时不需要验证密码，但是要进一步操作，就得输入 32 字节密码进行验证。BSL 的协议规定这 32 字节密码为芯片 FLASH 区域的最高 32 字节，也就是程序的 16 个中断向量，如果您拥有这段程序的最后 32 字节，就能通过 BSL 将芯片内部所有代码读取出来。但是 msp430 的 16 个中断向量未必每一个都用到了，为了更好的加密性能，建议将所有未用到的中断向量全部填充为随机数据，这样可实现高级加密。但是，切记 任何加密手段都不是万能的，没有破解不了的 MCU。对于 MSP430F5438A 而言，熔丝不是硬件熔丝，‘烧断’之后，还是可以通过 BSL 恢复的，但是一定要记得您的‘密码’啊!

117. MSP430 在应用中如何降低功耗？

- 硬件方面:
  [1]尽可能采用低功耗的器件或电路设计。比如，低功耗场合能不用 LED 尽可能不要用，声响的电路也是一样。
  [2]尽可能选择带有关闭功能的器件，比较运放、R232 电路、逻辑电路等等...在不必要的时候使其关闭。
  [3]在显示方便也要选择低功耗的显示方式，比如采用 LCD 片，而不要用 LCD 模块。或采用 LCD 模块时将背光关掉。
  [4]一些常用开关晶体管由三极管改为 MOSFET 管。
  [5]有可能的话，不要选择小阻值分压；这样同样可减少功耗。按键上接电阻同样可以选择大点。对于模拟前端部分可能不态适合，因为当用高精度 ADC 时，电阻值越大热噪声就会越大。所以这做法不适宜用在高精度 ADC 前端。
  [6]关于 MSP430 的 IO 处理，我个人的理解是可以空着，并设置为输入。因为设置为输入时 IO 处理高阻态，IO 的漏电流只有 50nA。
  [7]能不用 LDO 尽可能不要用 LDO，因为线性电源器件会带功耗上的增加。确实没办法了可以选择 CMOS 型的 LDO 器件。或采用高效的 DC/DC 电源管理电路，以提高效能利用。关于以上几项，MC430F44 开发板在设计都是基于这些原则上设计的，同时兼容了通用器件的使用。也就是说两种类型的器件都可以用。
- 软件方面：
  [1]你要了解 MSP430 的 4 种不同模式下的时钟与模块使用情况，这样你才控制好整个设计的功耗管理。

[2]若不是很需要很高精度的时钟的话尽可能不要外部晶振，尽可能使用内部的 DCO 作为 MCLK。当程序中需要在串口时，这时可以开启所需的时钟源以得到精度的波特率，不用时则要关闭掉时钟和串口模块。如果不是高速响应处理任务的话尽可能不要用选择外部晶体时钟作为 MCLK。
[3]在进入低功耗模式前，尽可能将 MCLK 改为 DCO 模式。因为 DCO 模式在进入功耗模式后，在得到中断唤醒时是最速度启动工作的时钟源。这样可以大大减小在唤醒时节省能源。如果唤醒后确实需要高速度时钟源，此时可以再转换到高速度的时钟源上使用。
测量验正： 采用精度高的电流表去测量电流值；在实际中，有些质量不太好的表会误导。采用串联高精度的电阻，直接测量电压值。这样电压法测量有时也很有用。

118. MSP430F5438 中断函数编写方法？

以 USCI0 为例，说明该两种不同方法。
1.switch-case 方法[cpp] view plaincopy

```
#pragma vector=USCI_A0_VECTOR
__interrupt void USCI_A0_ISR(void)
{
       switch (__even_in_range(UCA0IV,4))
       {
              case 0: break;
              case 2:       // 接收中断  // do something here
                     break;
              case 4:       // 发送中断  // do something here
                     break;
              default: break;
       }
}
```

2.查询标志为方法[cpp] view plaincopy

```
#pragma vector=USCI_A0_VECTOR
__interrupt void USCI_A0_ISR(void)
{    // 接收中断
       if( ( UCA0IFG & UCRXIFG ) != 0 )    {          }    // 发送中断
       if( ( UCA0IFG & UCTXIFG ) != 0 )    {          }
}
```

TI 的官方例程都是用 switch-case 方法，而本人则更喜欢第二种——查询标志位。总之两种方法的最终效果都一样。

119. MSP430F149 的 A/D 转换过程是怎样的？

关于 MSP430F149 的 A/D 转换：（都必须经过以下阶段）

- 设置通道
- 打开 ADC,设置采样时间
- 使用采用定时器
- 设置参考电压
- 使能开始
- 采样开始
- 等待转换完成
- 把转换的值存入变量

120. MSP430F149 I/O 口如何控制？

所谓 IO 口控制就是控制单片机的端口输出 0 或 1，或者读出端口的状态，也就是输入和输出。先说输出。想让 MSP430 单片机的端口输出 0 或 1 必须做的一步就是设置对应端口的方向寄存器，就是你必须得告诉单片机你想让那个端口作为输出端口。比如你想让 P2 端口的第 2 位（P2.2）作为输出端口就得这样设置：P2DIR |= BIT2; P2DIR 就是 P2 口的方向寄存器的地址（可以在头文件里面查到），DIR 就是 direction（方向）。这条语句其实就是把 P2DIR 这个寄存器的第 2 个 Bit 位置 1。当然你完全可以这样写：P2DIR |= 0x04；之所以用“|=”而不直接用“=”是只操作第二个 Bit 位而不影响其他 Bit 位。聪明的你应该已经学会设置了吧。方向设置为输出后就可以让这个端口输出 0 或 1 了。比如你想让 P2.2 输出 0 可以这么写：P2OUT &= ~BIT2; 输出 1 可以这么写 P2OUT |= BIT2; 当让也可以直接这么写： P2OUT &= ~（0x04）; P2OUT |= （0x04）;再说输入。输入和输出差不多，首先也是得设置对应端口的方向寄存器，就是你必须得告诉单片机你想让那个端口作为输出端入。比如你想让 P3.1 作为输入端口，那你就可以这么设置：P3DIR &= ~BIT1; 设置完端口方向寄存器就可以读这个端口的状态了，不如我们将 P3.1 端口的状态付给变量啊就可以这么写：a = P3IN&BIT1;好了，现在就可以基本运用 MSP430 单片机端口的输入输出功能了吧！点亮一个 LED 灯应该不成问题了，灯点亮了那么你也就对 MSP430 单片机入门了。另外说明一下端口操作的一些写法。P2OUT &= ~BIT2 其实就是将 P2OUT 寄存器的第二个 Bit 位清零而不影响其他 Bit 位，P2OUT |= BIT2 其实就是将 P2OUT 寄存器的第二个 Bit 位置 1 而不影响其他的 Bit 位。这都是 C 语言运算的基础，相信大家都是搞程序的，仔细想想都会明白的。这样写的关键是只操作对应的 Bit 位而不影响其他 Bit 位，如果直接操作 8 个 Bit 位了那也就不必这么麻烦了，可以直接这样写：P2OUT = 0xf0。

121. MSP430 的存储器读写模式是什么？

首先要明白有两种存储器读写模式，分别是小端模式和大端模式。

- 小端模式（Little-Endian）：数据的低字节存放在内存低地址中，高字节存放在高地址中 。
- 大端模式（Big-Endian）：数据的低字节存放在内存高地址中，高字节存放在低地址中。

MSP430 采用小端模式读写存储器，也就是说 MSP430 的存储器读写模式是小端模式。

122. MSP430F149 如何选择时钟源？

MSP430 的基本时钟源有 3 个：

- LFXT1CLK
- XT2CLK
- DCOCLK

其中： LFXT1CLK:可以用低频钟表晶体、标准晶体、陶瓷谐振器或外接时钟源工作。
XT2CLK:可以用标准晶体、陶瓷谐振器或外接 450khz~8mhz 的时钟源工作。
DCOCLK:它是内部数字控制 RC 振荡器，可以调节。

MSP430 的 3 种时钟信号是：ACLK,MCLK,SMCLK;其中：
ACLK(辅助系统时钟):可选时钟源 LFXT1CLK(只能是外部时钟源)，且一般为 32768hz 手表晶体)。
MCLK(主时钟):可选 LFXT1CLK,XT2CLK, DCOCLK 三种时钟源。用于 CPU 和系统。
SMCLK(子时钟):可选 LFXT1CLK,XT2CLK, DCOCLK 三种时钟源。用于外围器件.

ACLK 和 MCLK 的区别：ACLK 一般用于低速外设

SMCLK 主要用于高速外围模块，上电默认是内部 800K 的 RC 振荡器，下面给出了切换 LFXT 和 XT2 作为系统时钟的例子：

```
切换为 LFXT:
do  {  IFG1 &= ~OFIFG;
       for (i = 0xFF; i > 0; i--);
} while ((IFG1 & OFIFG));
BCSCTL2 |= SELM_3;     //选择钟表时钟切换为 XT2:
```

```
BCSCTL1&=~XT2OFF;      //启动 XT2 时钟
do  {  IFG1 &= ~OFIFG;
       for (i = 0xFF; i > 0; i--);
} while ((IFG1 & OFIFG));
BCSCTL2 |= SELM_2;
BCSCTL2 |= SELS;//选择 XT2 时钟
```

123. MSP430 是否支持位变量？

位操作指令常见于 CISC （复杂指令集）型处理器（例如大家比较熟悉的 8051）上 ，目的是为了提高 CISC 型处理器的执行效率。与之相对的是 RISC（精简指令集） 型处理器（MSP430 当然也名列其中），几乎所有的 RISC 型处理器都取消了位操作指令。MSP430 的 C 语言中是不支持位变量的，MSP430 的位操作往往由变量与掩模位之间的逻辑操作来实现。

124. MSP430 的 FLASH 是否能以位为单位进行编程？

很明显，MSP430 的 FLASH 是不能以位为单位进行编程的！MSP430 的 FLASH 的擦除通常是以段为单位的，而写入往往以字节或字为单位进行。但是 FRAM 型的 MSP430 却不同：FRAM 可以以位为单位，进行编程、读取或擦除操作。

125. MSP430 有哪些非屏蔽中断？

非屏蔽中断不受 GIE 的控制，具备独立的中断使能。MSP430 的非屏蔽中断主要有 3 个：

- 外部引脚 NMI 的触发
- FLASH 非法访问
- 振荡器错误

126. msp430 的低功耗模式有几种？分别是什么？

MSP430 的低功耗模式有 5 种，分别是 LPM0，LPM1，LPM2，LPM3，LPM4，这五种低功耗各种解释如下 ：

- LPM0：CPU 停止工作，MCLK 时钟停止，SMCLK、ACLK 时钟还在工作。
- LPM1：CPU 停止工作，MCLK 时钟停止，在活动模式如果 DCO 没有作为 MCLK 和 SMCLK 时钟时，则直流发生器被禁止，否则就保持活动状态，SMCLK、ACLK 时钟依然还在工作。
- LPM2：CPU 停止工作，MCLK、SMCLK 时钟停止工作，如果 DCO 没有作为 MCLK、SMCLK，自动被禁止直流发生器保持有效，ACLK 还处于工作中。
- LPM3：CPU 停止工作，MCLK、SMCLK 时钟停止工作，DCO 时钟也停止工作，仅 ACLK 时钟还处于工作状态。
- LPM4：CPU 停止工作，MCLK、SMCLK 时钟停止工作，DCO 时钟也停止工作，ACLK 也停止工作。此时功耗最低。

127. MSP430 单片机怎么查看时钟的校准信息？

MSP430 出厂时在 Info Flash 保存了时钟的调整参数，可以通过下面的方法很容易查看，以防误写入后恢复。首先用 ccs 往 430 里烧写任一程序，下载，进入仿真界面，按下图操作，其中的 Value 就是 Flash 中的值，记录下备份即可。

128. MSP430 的复位信号有哪几种？

两种：上电复位信号（POR）、上电清除信号（PUC）。能够触发 POR 和 PUC 的信号：5 种来自看门狗，1 种来自复位管脚，1 种来自写 FLASH 键值出现错误所产生的信号。
POR 信号只在 2 种情况下发生：

- 微处理上电；
- RST/NMI 管脚上产生低电平时系统复位。

PUC 信号产生的条件：

- POR 信号产生
- 看门狗有效时，看门狗定时器溢出
- 写看门狗定时器全键值出现错误
- 写 FLASH 存储器安全键值出现错误。

129. MSP430 单片机中断是怎么响应的？

(1) 如果 CPU 处于活动状态，则完成当前指令
(2) 若 CPU 处于低功耗状态，则退出低功耗状态
(3) 将下一条指令的 PC 值压入堆栈
(4) 将状态寄存器 SR 压入堆栈
(5) 若有多个中断请求,响应最高优先级中断
(6) 单中断源的中断请求标志位自动复位,多中断源的标志位不变,等待软件复位
(7) 总中断允许位 SR.GIE 复位。SR 状态寄存器中的 CPUOFF、OSCOFF、SCG1、V、N、Z、C 位复位
(8) 相应的中断向量值装入 PC 寄存器,程序从此地址开始执行。

130. MSP430 MCU 不使用的 I/O 口如何处理？

（1）将未使用的 I/O 切换到输出模式
（2）将未使用的输入连接到 VCC 或 VSS
（3）通过电阻器将未使用的输入连接到 VCC 或 VSS

131. MSP430 复位后引脚是什么状态？

默认状态下，所有 I/O 引脚在复位后均为输入状态

132. MSP430 和 51 有什么区别？

- MSP430 是 16 位单片机，51 是 8 位单片机
- MSP430 采用 RISC 精简指令集，单个时钟周期就可以执行一条指令，相同晶振，速度较 51 快 12 倍
- 其它片上资源也是 MSP 较丰富。

总体而言，MSP430 功能强大，速度快，相比 51 而言，这些是明显的优势。

133. 430 的片内 DCO 受环境因素抖动？

供电电压 环境温度等因素都会影响 DCO 的输出，造成输出不稳定，影响精度。

134. 430 的 I/O 无保护吗？

如果输入信号过压过流会立即击穿，但是 I/O 的阻抗和灵敏度很高，捕获功能也很强大。

135. MSP430 的 DCO 频率稳定性如何？

DCO 模块混有两个 DCO 频率，fDCO 和 fDCO+1，用以产生介于 fDCO 和 fDCO+1 之间的频率。这样就可得到带有所需的平均频率的调制时钟。调制的影响表现形式就是频率的抖动。本质上来说，这种调制将时钟能量扩散到一个宽带中，减小了电磁干扰(EMI)。DCO 频率会随着温度和电压的变化而有所波动。请参阅器件数据手册关于 DCO 的具体说明。

136. 如何使 MSP430 的基本时钟模块中的 DCO 保持稳定

通过补偿电压、温度方面的变化以及部件之间的差异，可以对 DCO 频率进行校准并将其设定为指定的频率。通常，低速晶振或外部信号可以通过比较一个低速参考频率周期内出现高速 DCO 时钟周期的次数来实现这一点。借助软件，可以调整基本时钟控制寄存器，以便将 DCO 的频率设置为较慢的晶振或信号的所需倍数。该器件的数据表详细介绍了 DCO 的工作范围。《MSP430x1xx 用户指南》提供了有关基本时钟的详细信息。通过 MSP430 网站可以获得证明 DCO 设置的范例代码及应用报告。

137. 怎样降低 MSP430 的功耗？

降低功耗的最重要的途径是使用 MSP430 的时钟系统来最大限度地提高 MSP430 处于低功耗模式的时间。以下是其他的一些减小功耗的原则：

- 使用中断来唤醒处理器，控制程序流向。
- 外围模块仅当在需要时将其打开。
- 使用低功耗的集成外围模块来取代软件驱动。例如 Timer_A 和 Timer_B 可以自动产生 PWM 波、捕获外部定时而不占用 CPU 资源。
- 使用计算分支和快速查找表来取代标记的设置和大量的软件计算。
- 避免频繁的子程序和函数调用以降低软件开销。
- 对于较长的软件程序，最好用单周期 CPU 寄存器。
- 确保所有未使用的端口引脚是开路的，并且设置成输出。

138. MSP430 存储器的存储空间不够使用，如何进行扩展？

任何 MSP430 器件都没有外部数据和地址线。然而，扩展外部数据存储器可以使用 I/O。或者外部的 I2C 或串行存储器 EEPROM 可用于数据存储器扩展。如果您需要扩展外部程序存储器，这是不允许的，建议采用更大 ROM 的 msp430 器件。

139. “金刚狼”平台的功率与能源消耗有何提升？

运行模式功耗低至 100μA/MHz 待机流耗低于 400 nA（RTC 和欠压保护模式）FRAM 每位能耗下降了 250 倍可在不到 7μs 的时间里从待机模式唤醒至运行模式。

140. 为什么 MSP430 单片机我实测电流比官方公布的功耗要大很多？

- CMOS 电路的电流消耗主要发生在 CMOS 管状态翻转的时刻。
- 设置成输入后，处于高阻状态的输入开关会发生未知的状态翻转，从而消耗电流。
- 设置成输出后，无论是上拉还是下拉，都会消耗电流。
- 建议普通 I/O 口设置成输出，并且悬空；或者设置成输入，并且上拉或下拉。

141. 上电复位和硬件看门狗复位有什么区别吗？在程序里将两者分开，请问有办法将两者分开吗？

上电复位时，内存被清零或为任意值，看门狗清零时并没有断电，内存里的原有信息被保留，同时上电复位无法通过标志位来判别，看门狗复位才可以通过 WDTIFG 来判别。同时注意 RESET 之后：1、判断有无复位标志，若有，则为 WDT 复位；若无，则为上电复位，并且设置复位标志。手动按键复位同此理。2、保证复位标志在复位程序中不被清除。汇编好办，那是自己在控制 RAM 清除程序，C 呢，就要注意了。3、要注意快速断电/上电的问题。处理不好的话，不但 RAM 中原先的内容有可能还存在，而且 MCU 复位很可能会失败

142. 430 如何将程序成功烧入？烧片子的具体顺序是怎样的？

首先 option 里得选择正确的芯片型号，还要在 Debugger 选项卡里的 driver 选择 FET_Debugger ， 在按工具栏内的下载按钮（快捷键 ctrl+D）

143. 请问 MSP430 仿真器和编程器有什么区别啊？

一般来讲，仿真器是在先期调试程序时使用的，他不会烧断单片机熔丝，能把程序下载到单片机中，能够单步，跟踪，快速调试。编程器就没有这些调试功能，就是单纯把你做好的程序的编译后文件写到单片机中去，就和 51 的编程器一样，有加密熔丝烧断等功能，是在你产品成型后，生产时使用的 MSP430 的仿真器是使用 JT AG 接口的，分别有四线制的 JT AG、带 TEST 脚的四线 JT AG 和两线制的 SBWJT AG 三种接口， UIF 上三种都支持， 并支持烧熔丝， UIF 就是 USB 接口的仿真； PIF 不能支持 SBWJT AG 接口，不能烧熔丝， PIF 是并口的仿真器。 任何一种 JT AG 接口的仿真器在烧断熔丝后都不能仿真和写入，而 BSL 可以通过密码访问 FLASH 空间，读出写入均可，BSL 是串口实现的，但 BSL 不能仿真，注意部分器件不支持 BSL，如 F20XX 系列就不能用 BSL，烧掉了熔丝就变板砖。MSP430 任何系列的仿真器只要接口方式一致都是兼容的， 比如 FG461X， 可以使用标准的带 TEST 的四线 JT AG，而 F22X4 可以使用带 TEST 的四线 JT AG，当然 F22X4 还可以选择使用 SBWJT AG，它支持两种 JT AG 接口。如果不是 TI 标准的 430 系列用 JT AG 仿真器那就不行了，应该是不兼容的，不是什么“很多仿真器和编程器都不支持”，而是专用。

144. 学习 MSP430 用汇编语言还是用 C 语言？

严格来讲 430 的 C 是 ANSI C 的一个子集，与汇编的差别主要有：

- C 有 if、（do）while、switch 等流程控制语句
- C 有有限的数据格式，如 char、int、float、dou××e 等
- 对 430 最有特色的 R0--R15 的使用，C 不如汇编
- 430 的 C 不易进行 RAM 管理
- 430 各版本的 C，互相之间存在差异，好象 C 在 430 上还不成熟
- C 的优点是在 PC 或 PDA 上，也就是在有操作系统的平台上，C 的优点才会表现出来，但那已经不是 430 的 C 了，而是 C++，它有丰富的数据类型，如结构、对象等

- 汇编的缺点，基本上就是 C 的优点，而汇编的优点基本上都是 C 的缺点

145. 如何快速上手 MSP430？

任何一款 MCU，其基本原理和功能都是大同小异，所不同的只是其外围功能模块的配置及数量、指令系统等。对于指令系统，虽然形式上看似千差万别，但实际上只是符号的不同，其所代表的含义、所要完成的功能和寻址方式基本上是类似的。因此，对于任何一款 MCU，主要应从如下的几个方面来理解和掌握：

- MCU 的特点：要了解一款 MCU，首先需要知道就是其 ROM 空间、RAM 空间、IO 口数量、定时器数量和定时方式、所提供的外围功能模块（Peripheral Circuit）、中断源、工作电压及功耗等等。
- 了解这些 MCU Features 后，接下来第一步就是将所选 MCU 的功能与实际项目开发的要求的功能进行对比，明确那些资源是目前所需要的，那些是本项目所用不到的。对于项目中需要用到的而所选 MCU 不提供的功能，则需要认真理解 MCU 的相关资料，以求用间接的方法来实现，例如，所开发的项目需要与 PC 机 COM 口进行通讯，而所选的 MCU 不提供 UART 口，则可以考虑用外部中断的方式来实现；
- 对于项目开发需要用到的资源，则需要对其 Manua*进行认真的理解和阅读，而对于不需要的功能模块则可以忽略或浏览即可。对于 MCU 学习来讲，应用才是关键，也是最主要的目的。
- 明确了 MCU 的相关功能后，接下来就可以开始编程了。对于初学者或初次使用此款 MCU 的设计者来说，可能会遇到很多对 MCU 的功能描述不明确的地方，对于此类问题，可以通过两种方法来解决，一种是编写特别的验证程序来理解资料所述的功能；另一种则可以暂时忽略，程序设计中则按照自己目前的理解来编写，留到调试时去修改和完善。前一种方法适用于时间较宽松的项目和初学者，而后一种方法则适合于具有一定 MCU 开发经验的人或项目进度较紧迫的情况；
- 指令系统千万不要特别花时间去理解。指令系统只是一种逻辑描述的符号，只有在编程时根据自己的逻辑和程序的逻辑要求来查看相关的指令即可，而且随着编程的进行，对指令系统也会越来越熟练，甚至可以不自觉地记忆下来；
- MCU 的基本功能：对于绝大多数 MCU，下列功能是最普遍也是最基本的，针对不同的 MCU，其描述的方式可能会有区别，但本质上是基本相同的：
- Timer（定时器）：Timer 的种类虽然比较多，但可归纳为两大类：一类是固定时间间隔的 Timer，即其定时的时间是由系统设定的，用户程序不可控制，系统只提供几种固定的时间间隔给用户程序进行选择，如 32Hz，16Hz，8Hz 等，此类 Timer 在 4 位 MCU 中比较常见，因此可以用来实现时钟、计时等相关的功能；另一类则是 Programmable Timer（可编程定时器），顾名思义，该类 Timer 的定时时间是可以由用户的程序来控制的，控制的方式包括：时钟源的选择、分频数（Prescale）选择及预制数的设定等，有的 MCU 三者都同时具备，而有的则可能是其中的一种或两种。此类 Timer 应用非常灵活，实际的使用也千变万化，其中最常见的一种应用就是用其实现 PWM 输出（具体的应用，后续会有特别的介绍）。由于时钟源可以自由选择，因此，此类 Timer 一般均与 EventCounter（事件计数器）合在一起；
- IO 口：任何 MCU 都具有一定数量的 IO 口，没有 IO 口，MCU 就失去了与外部沟通的渠道。根据 IO 口的可配置情况，可以分为如下几种类型：
  (1) 纯输入或纯输出口：此类 IO 口有 MCU 硬件设计决定，只能是输入或输出，不可用软件来进行实时的设定；
  (2) 直接读写 IO 口：如 MCS-51 的 IO 口就属于此类 IO 口。当执行读 IO 口指令时，就是输入口；当执行写 IO 口指令则自动为输出口；
  (3) 程序编程设定输入输出方向的：此类 IO 口的输入或输出由程序根据实际的需要来进行设定，应用比较灵活，可以实现一些总线级的应用，如 I2C 总线，各种 LCD、LED Driver 的控制总线等；
  (4) 对于 IO 口的使用，重要的一点必须牢记的是：对于输入口，必须有明确的电平信号，确保不能浮空（可以通过增加上拉或下拉电阻来实现）；而对于输出口，其输出的状态电平必须考虑其外部的连接情况，应保证在 Standby 或静态状态下不存在拉电流或灌电流。

- 外部中断：外部中断也是绝大多数 MCU 所具有的基本功能，一般用于信号的实时触发，数据采样和状态的检测，中断的方式由上升沿、下降沿触发和电平触发几种。外部中断一般通过输入口来实现，若为 IO 口，则只有设为输入时其中断功能才会开启；若为输出口，则外部中断功能将自动关闭（ATMEL 的 ATiny 系列存在一些例外，输出口时也能触发中断功能）。外部中断的应用如下：
  （1） 外部触发信号的检测：一种是基于实时性的要求，比如可控硅的控制，突发性信号的检测等；而另一种情况则是省电的需要；
  （2） 信号频率的测量：为了保证信号不被遗漏，外部中断是最理想的选择；
  （3） 数据的解码：在遥控应用领域，为了降低设计的成本，经常需要采用软件的方式来对各种编码数据进行解码，如 Manchester 和 PWM 编码的解码；
  （4） 按键的检测和系统的唤醒：对于进入 Sleep 状态的 MCU，一般需要通过外部中断来进行唤醒，最基本的形式则是按键，通过按键的动作来产生电平的变化；
- 通讯接口：MCU 所提供的通讯接口一般包括 SPI 接口，UART，I2C 接口等，其分别描述如下：
  （1） SPI 接口：此类接口是绝大多数 MCU 都提供的一种最基本通讯方式，其数据传输采用同步时钟来控制，信号包括：SDI（串行数据输入）、SDO（串行数据输出）、SCLK（串行时钟）及 Ready 信号；有些情况下则可能没有 Ready 信号；此类接口可以工作在 Master 方式或 Slave 方式下，通俗说法就是看谁提供时钟信号，提供时钟的一方为 Master，相反的一方则为 Slaver；
  （2） UART（Universal Asynchronous ReceiveTransmit）：属于最基本的一种异步传输接口，其信号线只有 Rx 和 Tx 两条，基本的数据格式为：Start Bit + DataBit(7-bits/8-bits) + arity Bit(Even, Odd or None) + Stop Bit(1~2Bit)。一位数据所占的时间称为 Baud Rate（波特率）。对于大多数的 MCU 来讲，数据为的长度、数据校验方式（奇校验、偶校验或无校验）、停止位（StopBit）的长度及 Baud Rate 是可以通过程序编程进行灵活设定。此类接口最常用的方式就是与 PC 机的串口进行数据通讯。
  （3） I2C 接口：I2C 是由 Philips 开发的一种数据传输协议，同样采用 2 根信号来实现：SDAT（串行数据输入输出）和 SCLK（串行时钟）。其最大的好处是可以在此总线上挂接多个设备，通过地址来进行识别和访问；I2C 总线的一个最大的好处就是非常方便用软件通过 IO 口来实现，其传输的数据速率完全由 SCLK 来控制，可快可慢，不像 UART 接口，有严格的速率要求。
- Watchdog（看门狗定时器）：Watchdog 也是绝大多数 MCU 的一种基本配置（一些 4 位 MCU 可能没有此功能），大多数的 MCU 的 Watchdog 只能允许程序对其进行复位而不能对其关闭（有的是在程序烧入时来设定的，如 MicrochipIC 系列 MCU），而有的 MCU 则是通过特定的方式来决定其是否打开，如 Samsung 的 KS57 系列，只要程序访问了 Watchdog 寄存器，就自动开启且不能再被关闭。一般而言 watchdog 的复位时间是可以程序来设定的。Watchdog 的最基本的应用是为 MCU 因为意外的故障而导致死机提供了一种自我恢复的能力。
- MCU 程序的编写：MCU 的程序的编写与 PC 下的程序的编写存在很大的区别，虽然现在基于 C 的 MCU 开发工具越来越流行，但对于一个高效的程序代码和喜欢使用汇编的设计者来讲，汇编语言仍然是最简洁、最有效的编程语言。对于 MCU 的程序编写，其基本的框架可以说是大体一致的，一般分为初始化部分（这是 MCU 程序设计与 PC 最大的不同），主程序循环体和中断处理程序三大部分（见图 1 a 和 b），其分别说明如下：
  （1） 初始化：对于所有的 MCU 程序的设计来讲，出世化是最基本也是最重要的一步，一般包括如 下内容：
  （2） 屏蔽所有中断并初始化堆栈指针：初始化部分一般不希望有任何中断发生； （3） 清除系统的 RAM 区域和显示 Memory：虽然有时可能没有完全的必要，但从可靠性及一致性的角度出发，特别是对于防止意外的错误，还是建议养成良好的编程习惯；
  （4） IO 口的初始化：根据项目的应用的要求，设定相关 IO 口的输入输出方式，对与输入口，需要设定其上拉或下拉电阻；对于输出口，则必须设定其出世的电平输出，以防出现不必要的错误；
  （5） 中断的设置：对于所有项目需要用到的中断源，应该给予开启并设定中断的触发条件，而对于不使用的多余的中断，则必须给予关闭；

（6） 其他功能模块的初始化：对于所有需要用到的 MCU 的外围功能模块，必须按项目的应用的要求进行相应的设置，如 UART 的通讯，需要设定 Baud Rate，数据长度，校验方式和 Stop Bit 的长度等，而对于 Programmer Timer，则必须设置其时钟源，分频数及 Reload Data 等；

（7） 参数的出世化：完成了 MCU 的硬件和资源的出世化后，接下来就是对程序中使用到的一些变量和数据的初始化设置，这一部分的初始化需要根据具体的项目及程序的总体安排来设计。对于一些用 EEPROM 来保存项目预制数的应用来讲，建议在初始化时将相关的数据拷贝到 MCU 的 RAM，以提高程序对数据的访问速度，同时降低系统的功耗（原则上，访问外部 EEPROM 都会增加电源的功耗）。

- 主程序循环体：大多数 MCU 是属于长时间不间断运行的，因此其主程序体基本上都是以循环的方式来设计，对于存在多种工作模式的应用来讲，则可能存在多个循环体，相互之间通过状态标志来进行转换。对于主程序体，一般情况下主要安排如下的模块：

（1） 计算程序：计算程序一般比较耗时，因此坚决反对放在任何中断中处理，特别是乘除法运算；

（2） 实时性要求不高或没有实时性要求的处理程序；

（3） 显示传输程序：主要针对存在外部 LED、LCD Driver 的应用；

- 中断处理程序：中断程序主要用于处理实时性要求较高的任务和事件，如，外部突发性信号的检测，按键的检测和处理，定时计数，LED 显示扫描等。一般情况下，中断程序应尽可能保证代码的简洁和短小，对于不需要实时去处理的功能，可以在中断中设置触发的标志，然后由主程序来执行具体的事务——这一点非常重要，特别是对于低功耗、低速的 MCU 来讲，必须保证所有中断的及时响应。
- 对于不同任务体的安排，不同的 MCU 其处理的方法也有所不同。例如，对于低速、低功耗的 MCU（Fosc＝32768Hz）应用，考虑到此类项目均为手持式设备和采用普通的 LCD 显示，对按键的反应和显示的反应要求实时性较高，应此一般采用定时中断的方式来处理按键的动作和数据的显示；而对于高速的 MCU，如 Fosc>1MHz 的应用，由于此时 MCU 有足够的时间来执行主程序循环体，因此可以只在相应的中断中设置各种触发标志，并将所有的任务放在主程序体中来执行；
- 在 MCU 的程序设计中，还需要特别注意的一点就是：要防止在中断和主程序体中同时访问或设置同一个变量或数据的情况。有效的预防方法是，将此类数据的处理安排在一个模块中，通过判断触发标志来决定是否执行该数据的相关操作；而在其他的程序体中（主要是中断），对需要进行该数据的处理的地方只设置触发的标志。——这可以保证数据的执行是可预知和唯一的。

总之，对于 MCU 开发来讲，必须记住一点：“条条大路通罗马”，没有做不到的事，关键是看方法是否正确！再就是多做多动手和多想。

146. 为何 MSP430fr5969 无法仿真及下载程序？

请安装最新的 IAR 或者 CCS 使用环境可以用 IAR EW430 6.10 以上版本或者 CCSV6.0 以上版本

147. MSP430fr5969 如何使用 32K 外部晶振定时唤醒 LPM3？

```
#include int main(void)
{       WDTCTL = WDTPW | WDTHOLD;           // Stop WDT  // XT1 Setup
        CSCTL0_H = 0xA5;
        CSCTL2 = SELA_0 + SELS_3 + SELM_3;  // set ACLK = XT1;MCLK = DCO
        CSCTL3 = DIVA_0 + DIVS_0 + DIVM_0;  // set all dividers
        CSCTL4 |= LFXTDRIVE_0;
        CSCTL4 &= ~LFXTOFF;
        do  {  CSCTL5 &= ~LFXTOFFG;         // Clear XT1 fault flag
               SFRIFG1 &= ~OFIFG;
        }while (SFRIFG1&OFIFG);             // Test oscillator fault flag
```

```
        TB0CCTL0 = CCIE;                        // TBCCR0 interrupt enabled  TB0CCR0 = 50000;
        TB0CTL = TBSSEL_1 + MC_1;               // ACLK, continuous mode
        __bis_SR_register(LPM3_bits + GIE); // Enter LPM3 w/ interrupt
}



// Timer B1 interrupt service routine
#pragma vector = TIMER0_B0_VECTOR__interrupt
void Timer0_B0_ISR(void)
{
        P1OUT ^= BIT0;
}
```

148. MSP430F2XX 系列带有 TEST 引脚的单片机在设计 JTAG 接口时需要注意些什么？

还是 MSP430 系列单片机的问题，2XX(msp430G2553)系列有 TEST 引脚，以前用的 1XX 系列没有，而这个 TEST 引脚要接入 JTAG 口的第 8 脚，否则将不能下载仿真。此外，MSP430 系列管脚比较少的片子（一般 28 脚以及以下）管脚复用的比较多，因此为了节约管脚，TI 推出了 SBW 两线制仿真，而编译 430 系列单片机常用的开发环境 IAR 其默认的方式即为 SBW 方式，因此需要在选项中将 SBW 方式改为 JTAG 方式，方能正常使用。亦可在设计板子的时候设计为 SBW 方式，这样选择 SBW 方式即可下载仿真。在 Debug protocol 中选择相应的下载方式，默认为 SBW 方式。

149. MSP430fr5969 金刚狼系列功耗如何？

TI 公司最新一代 MSP430FRXX 系列 MCU 采用了 FRAM 作为代码和数据存储器，替代传统 MCU FLASH+SRAM 的结构，并且其 FRAM 带有分区管理和 ECC 校验功能，增强存储器可靠性，FRAM 运行时的低功耗特性，将 MCU 的功耗降低至 100uA/MHz。除了 FRAM 外与 SCI/IIC/SPI/GPIO/ADC/CMP/TIMER 等普通外设外，其还增加了 AES 硬件加解密模块，32 位硬件乘法器等，其余性能指标可详见官方。

150. MSP430 如何启用触摸功能？

MSP430FR5969 等 CPU 所有 IO 均支持触摸功能只需将 IO 置为输入，并且取消上拉或下拉即可。

151. MSP430FR5969 金钢狼开发环境都有什么需求？

MSP430FR5969 金钢狼可以用 IAR 或者 CCSIAR EW430 6.10 以上版本 CCSV6.0 以上版本

152. 为什么在调试 MSP430LaunchPad 时，程序在运行，可是板子却没有反应？

默认建立工程时，已经设置了调试模式是软件仿真。如果想要程序在板上调试运行，需要设置工程的选项，在调试选项中，选择 FET 调试器，再次运行程序调试，即可在板子上运行了。

153. MSP430FR5969 管脚功能选择的对应值，到哪里能找到？（UserGuide 里面没有具体说明）

以下两个 URL 分别是硬件手册和用户手册。问题相关的设定值，可以在硬件手册的 Input/Output Schematics 部分里查到。处理器的寄存器设定，大部分都能在用户手册中找到。
http://www.ti.com.cn/cn/lit/ds/symlink/msp430fr5969.pdfhttp://www.ti.com.cn/cn/lit/ug/slau367e/slau367e.pdf

154. 使用 MSP430 与设备进行 SPI 通信，通信线上有信号，速率也在接受范围，为什么设备不能正常工作？

SPI 通信有几种模式，在 MSP430 中，通常用 CKPH 和 CKPL 来设置。CKPL 的设置值表示时钟线在空闲时是低电平(CKPL=0)还是高电平(CKPL=1)；CKPH 的值表示发送数据是利用时钟开始沿(CKPH=0)还是结束沿(CKPH=1)，另一个沿用于接收数据。请根据设备的要求来设定这两个值。

155. 我按照 MSP430 头文件的内容设置看门狗为 WDT_MRST_32，为什么不到 32ms 就 Reset 了？

看门狗所使用的时钟的频率也会影响其计数的速度，头文件中的 WDT_MRST_32 是指使用 1MHz 的 fSMCLK，如果你设定了 fSMCLK，那么看门狗的时间也会受到影响。

156. MSP430 单片机设置 PWM，可是管脚上没有输出？

管脚的功能是需要选择的，请检查是否没有将相关管脚的功能选择成 PWM 输出：选择寄存器通常被命名为 PxSEL 或者 PxSELx

157. 自己写的 MSP430 的工程，代码明明没有问题，为什么程序一运行到某处就执行不下去了？

没有问题的程序，一般是不会出现执行不下去的情况的。一旦发现这样的情况，就优先检查是否看门狗设置不正确。默认 MSP430 的看门狗是启动状态，请检查是否没有喂狗，或者喂狗间隔时间太长。

158. MSP430G2 系列单片机 RAM 较小，在做一些显示开发的时候会出现内存不够用的情况，有什么办法可解决？

这一类显示开发，可能会涉及到许多固定不用的数据，比如图片、字体数据等，可将这些数据定义到代码段，即给变量加上 const 关键字，可将数据存放在 flash 中了。

159. MSP430G2 系列单片机 IO 有上下拉电阻吗，大概是多大？

IO 口有上下拉电阻，在数据手册里有说明，此阻值是 33K 欧左右。

160. MSP430G2 系列单片机 SPI 的速度最高可到达多少？

SPI 通信属于同步通信方式，原理上是同步时钟越高速度就能达到越快，但是限于单片机的处理速度，G2 系列 SPI 速度最高达 4Mbps。

161. 使用 TI 官方最新的 ccs6.0 开发，新建工程时为何找不到创建 Grace 工程选项？

CCS6.0 中加入了 APP Center 应用中心，需要在这里面添加 Grace 工具组件后才能使用。

162. MSP430F147 的 AVss 和 DVss 引脚直连注意事项有哪些？

公司的一款仪表使用的是 MSP430F147，对于 AVss 和 DVss 引脚一直都是直连（原理图见下图左边），其中 PCB 文件如下图中间图，并且 MCU 板是外发贴片。某个批次贴片完组装发现，仪表显示会一闪一闪，在排除了驱动问题后，通过放大镜检查发现，MCU 的 AVss 和 DVss 引脚之间被割掉了，从而造成 DVss 引脚悬空。究其原因，原来是 PCB 贴片厂家检查的时候以为 AVss 和 DVss 引脚 之间是连焊，从而割开造成的。通过此事，公司所有的旧的 PCB 都进行升级，特别是对于管脚多且小的器件的 PCB，如果有相邻两个引脚直连，尽量拉出线来互联，从而可以避免次此问题发生。

163. 在 MSP430FR5969 中看到在设置系统的时钟的时候，需要开启 CS 控制寄存器，配置好了还需要重新锁上这时为什么？以往的 430 怎么不需要呢？

在设置系统的时钟的时候需要用：CSCTL0_H = CSKEY >> 8;这句来解锁 CS 控制寄存器，在配置完后需要用:CSCTL0_H = 0;这句重新锁上，设置为了防止这些寄存器被误修改，在 MSP430FR5969 中系统中使用的存储器是 FRAM,这样做能增加系统的安全性。

164. PJ 端口的 BIT4 和 BIT5 可以接外部的低频晶振，这两个管脚可以作为普通的 GPIO 使用吗？

可以的，但是一旦设置了 PJSEL0 |= BIT4 | BIT5；说明这两个管脚已经作为外部低频晶振的功能了，这时就不能作为普通的 GPIO，如果没有设置的话就是可以的。

165. PM5CTL0 &= ~LOCKLPM5；这句在以往的 430 的开发中都没有遇到过，在 MSP430RF5969 中确有这一条，这条语句有什么作用？

这句的作用是：关闭上电端口默认输出高阻抗的功能，使能上电保持起始设置。在 MSP430FR5969 中默认上电的情况下，端口是输出高阻抗的。

166. 默认的情况下 MSP430 的主时钟是多少？

在没有进行设置的情况下，MCLK 的时钟来源是芯片内部的 DCO，而且 MCLK=1mhz。

167. __delay_cycles 是在哪里定义的？？

该本征函数在 intrinsics.H 里：意思是延时 __cycles 个机器周期如果 MCLK 为 1MHZ 那么，__delay_cycles(100000); 这句延时为 0.1s。

168. MSP430F5438 和 MSP430F5438A 有什么区别？

MSP5438 是刚推出时的芯片，MSP5438A 是 TI 后面优化后推出的芯片，两者除供电电压，主频有差别之外没有什么其余不同，后者完全兼容前者。

169. MSP 430 在下载程序时能不能不擦除 Flash 中 A/B 段 256B 数据？

可以，在 IAR 的下载界面，点击下载选项可以选择不擦除这两段 Flash 内容。

170. MSP430 中 ADC 的采样率和官方提供的 200Ksps 一致么？

在 MSP430 单片机中经常看到 ADC 的一共指标是 200Ksps，注意这个指标在官方手册中 Conversion 转化速率，ADC 除了转化之外还要有采样保存时间，这样其实时间的 ADC 采样率就小于 200Ksps,曾经测试过的 G2553 芯片的 ADC 最大为 136Ksps。

171. MSP430 一共有几种方式下载程序？

一共有三种：

- JATG 下载
- SBW 两线下载
- BSL 下载

注意第三种方式只能下载不能在线调试。

172. MSP430 可以通过外部矩形信号的上升或下降沿来触发 ADC 么？

MSP430 的 ADC 触发有三种：

- 软件置位 Start
- TimeA
- TimeB

如果想通过上升沿触发 ADC 可以打开 Time 的捕获功能，利用捕获来触发 ADC，从而控制采样率触发 ADC 采样。

173. 要实现 MSP430 编程，应如何连接 JTAG？

MSP430 器件可以通过 JTAG 来编程，可以使用串行编程 PRGS430、组合编程器 GANG430 （仅适用于 Flash 器件），或者基于 JTAG 的复制器（仅适用于 Flash 器件）。要查阅使用以上所述的任何一种工具为 MSP430 编程的 JTAG 连接方法，可以从他们相应的工具技术手册文件或 MSP430 JTAG 应用笔记中找到答案。

174. MSP430 允许的输入输出电流有多大?管脚是否能驱动 LED？

MSP430 并没有指定 I/O 引脚的绝对最大电流。对于 Voh 和 Vol，请参阅数据手册所提供的参数。
每个 I/O 引脚可以提供几毫安的电流，但是随着电流的增大，输出电压将会改变。对这些参数说明的脚注给出了总电流的最大值（包括所有的输出脚），这样就允许指定的电压值得以维持。MSP430 的 I/O 并不能驱动 20mA 的 LED 电流。

175. MSP430 I2C 硬件模块的速度有多快？

MSP430 I2C 模块同时支持高达 100kbps 的标准模式和高达 400kbps 通信的快速模式。

176. 为什么 MSP430 的 USART 无法达到 USART 控制寄存器所设置的预期的效果？

MSP430 的 USART 模块是一个状态机，每次 USART 的配置被重新设置后，必须复位才有效。这可以通过设置 UCTL 寄存器中的 SWRST 位的设置/复位顺序来实现。在上电复位时，SWRST 位是默认置位的。如果上电复位后第一次通过配置控制寄存器来定义 USART 模块参数，那么，必须最后配置 UCTL 寄存器，这样，SWRST 就被复位从而使用定义的设置来启动状态机。这可以通过汇编语言 MOV.B #000X**0B,&UCTL 和 C 语言 UCTL = 0b000X **0 来实现。请参阅器件用户指南和代码示例。
如果 USART 模块在固件中被重配置，这样一来，SWRST 位的置位/复位顺序必须在重配置之后进行，以便用新的配置来重启 USART 的状态机。

177. MSP430Flash 数据保持率是多少？

MSP430Flash 数据保持率至少是 100 年。 在数据手册的 JTAG、程序存储器和熔丝特性部分都可以查找到这个数据。

178. MSP430 USART 硬件外围操作有多快？

在 SPI 主控模式下，通信速度可以达到 4Mbps。在 UART 模式下速度高达 2Mbps。
USART 可以配置成同步（SPI）或异步（UART）操作模式，可以选择几种内部和外部的时钟源（这些时钟源是独立于 CPU 时钟源的）。在 SPI 主控模式中，USART 的运行速度可以达到应用时钟的 1/2。例如，如果使用 8MHz 的时钟源，4Mbps 的传输速度对于 SPI 主控模式来说是可能的。在 UART 模式下，对于可靠的通信，每比特 3 或 4 个时钟周期是必须的。例如，使用 4 分频的 8MHz 的时钟将支持 2Mbps 的速度。在网站中的 MSP430**x 用户指南中提供了 USART 性能的完整的描述。

179. MSP430 Flash 的写入/擦除周期数的最大值可达到多少？

MSP430Flash 器件正常的写入/擦除周期是 100,000 次。可以在器件的数据手册的 JATG、程序存储器和熔丝特性部分找到这个数据的说明。

180. MSP430 ADC12 外围模块中有多少个转换通道？

在 MSP430 的 ADC12 中总共有 12 个转换通道。8 个通道（A0 - A7）是专门面向外部信号的，2 个通道（A8 和 A9）用来转换所使用的外部参考电压，A10 用于转换内置的温度传感器的数据，A11 用于转换

Avcc 引脚的电压。注意如果不使用外部参考电压，A8 和 A9 是可以被用户使用的，这样就又 10 个通道可以用于外部信号。

181. 在 MSP430 中，哪些端口引脚具有中断能力？

MSP430 器件具有高达 6 个数字 I/O 端口，P1-P6。每个端口有 8 个 I/O 引脚。每一 I/O 引脚可以独立地被配置成输入或输出，并且每个 I/O 可以独立读写。
P1 和 P2 具有中断能力。P1 和 P2 的每个中断可以独立地被使能且配置成上升沿或下降沿中断。所有的 P1 口中断源共享一个中断向量，所有的 P2 口中断源共享一个不同于 P1 口的中断向量。

182. 如何得到 MSP430 操作码的列表（助记符）？

我们并不提供所有操作码的列表，因为有很多可用的寻址模式。然而，关于组成多样的操作码的独立的位流的描述还是有资料可参考的。（根据指令和寻址模式）
MSP430**x 用户指南的`RISC 16-Bit CPU`这个章节提供了可用的指令的信息。
`Addressing Modes`（寻址模式）这一部分解释了`As` 和 `Ad`位。在`Instruction Set（指令集）这个部分你可以看到怎样从这些位流产生十六进制的指令形式：
操作码
S-Reg (0b0000 = R0, 0b0001 = R1 ... 0b1111 = R15)
D-Reg (0b0000 = R0, 0b0001 = R1 ... 0b1111 = R15)
Ad
As
B/W
`Instruction Set Description` 这一部分包含了核心的指令表。
`Instruction Cycles and Lengths` 这一部分概括了这些指令所需的时钟周期数。

183. MSP430 工具支持哪些操作系统？

MSP-FET430X110、MSP-FET430P120、MSP-FET430P140、MSP-FET430P410 和 MSP-FET430P440 Kickstart IAR Embedded Workbench 软件，MSP-PRGS430 和 MSP-GANG430 编程软件都被 Windows 95, 98, ME, NT 4.0, 2000 和 XP 操作系统所支持。
MSP-STK430X320, MSP-EVK430X320, MSP-EVK430X330, MSP-EVK430X110,MSP-PRG430 和 ADT430 软件仅被 Windows 95/98 所支持，同时建议不要用于新设计。

184. MSP430 DCO 的频率会有抖动吗？

DCO 模块混有两个 DCO 频率，fDCO 和 fDCO+1，用以产生介于 fDCO 和 fDCO+1 之间的频率。这样就可得到带有所需的平均频率的调制时钟。调制的影响表现形式就是频率的抖动。本质上来说，这种调制将时钟能量扩散到一个宽带中，减小了电磁干扰(EMI)。
DCO 频率会随着温度和电压的变化而有所波动。请参阅器件数据手册关于 DCO 的具体说明。注意任何供电电源的不稳定也会造成 DCO 频率的抖动。

185. 在什么情况下可以使用 MSP430Flash 的 BLKWRT 模式？

块写入 BLKWRT 模式是以 64 字节的块大小来对 Flash 编程。选择这个选项加快了向 Flash 写入字节或字的速度。仅当执行 BLKWRT 操作的代码处于 RAM 的外部时，这个操作才可以被用于系统中。被写入 Flash 的数据也必须位于 RAM 区域中。在 BLKWRT 阶段，用户代码决不能访问 Flash，否则将发生访问冲突，同时 ACCVIFG 将被置 1。

186. 为识别一个有效的外部中断，MSP430 中断输入所需的最小脉冲宽度是多少？

最小的中断脉冲宽度必须大于 1.5 个主时钟长度(MCLKs) 以保证一个有效的中断。

187. MSP430 的端口引脚中断时边沿有效还是电平有效？

端口引脚中断是边沿有效并且可以单独设置。用户可以为每一个引脚选择上升沿或下降沿中断。注意在 MSP430x3xx 器件中仅仅是有专门中断向量的 P0.0 和 P0.1 的中断标志会被自动清零。在其他具有中断能力的端口引脚上，中断标志不会自动清零，用户必须软件清零。对于任何一个需要服务的中断，除了那些独立的中断使能位，在状态寄存器中的全局中断使能位(GIE)也必须被置位。

188. 除了 32.768kHz 的晶振频率，MSP430 还可以与多大频率的晶振协同工作？

MSP430x3xx 器件被设计成专门使用 32KHz 的晶振，然后再从一个独立的内部数字控制振荡器（DCO）中产生一个内部的主时钟（MCLK）。MSP430x3xx 器件使用 FLL 电路使 MCLK 稳定到用户所需的值。
MSP430x1xx 和 MSP430x4xx 器件具有一个支持 32KHz 或者更高速度的晶振。 有些 MSP430x1xx 和 MSP430x4xx 还有另一个晶振，这个晶振仅支持高速的晶振。这样，就允许在同一时刻有一个或两个晶振同时连接于器件上且在需要时可以只使用其中一个。
MSP430x1xx 和 MSP430x4xx 器件还具有可编程的内部 DCO ，DCO 可以在独立于任何晶振的情况下产生高速时钟。类似于 MSP430x3xx， MSP430x4xx 器件也使用 FLL 来使 DCO 稳定多种不同于外部的 32KHz 的时钟。

189. MSP430 的静电效应值是多大？

MSP430 符合 TI 标准静电效应规格，并且达到静电效应测试的标准，包括外围模块和端口引脚。TI 使用标准的静电效应测试了 MSP430 器件(Human Body Model=1.5KV,Charged Device Model=500V and Machine Model=200V)。

190. MSP430 在上电清除和上电复位后的初始状态？

POR 是指设备复位。仅在下面两个事件下设备才会发生复位：

- 设备上电
- 当设备设定为复位模式时，RST/NMI 引脚为低电平

上电复位一定能引起上电清除，但是上电清除却不一定能引起上电复位。下面的事件可引发一次上电清除：

- 一个上电复位信号
- 当仅在看门狗模式时，看门狗定时器溢出
- 看门狗定时器安全键冲突
- Flash 存储器安全键冲突

请参阅用户使用手册中的系统复位，中断和操作模式章节来了解更多的信息。
在 MSP430 的用户使用手册中，每一个寄存器都有一个关键字标志来代表寄存器中的每个位，以及它们的初始状态。请参阅用户使用手册中的寄存器位的规定。
初始条件的关键字是：
上电清除后是-0、-1
上电复位后是-（0）、-（1）
有括号的符号仅在上电复位后受影响。没带括号的符号在上电复位和上电清除时都受影响。
有了这些信息，如果你查看每个章节最后的设备寄存器，你将会看到每个位对应的关键字，以及这些位在上电复位和上电清除后的状态。

191. 在 MSP430 上有没有静电保护二极管？

在每个引脚端都有静电保护二极管，可以被认为是连接到电源电压的钳位二极管。静电保护的等效电路可以认为是两个二极管共同连接到输入信号，而二极管的另一端一个连接到 Vcc，另一个连接到 Vss。二极管的最大绝对额定电流范围是+ / - 2mA。

192. 在哪我能找到一个 BSDL 文件来建立 JTAG 链？

MSP430：我需要一个 BSDL 文件为我的 MSP430 建立一个与其他 JTAG 兼容设备的 JTAG 链。
解答：所有 MSP430 有一个用于项目的开发和 Flash 编程的 JTAG 接口。然而，这个 JTAG 接口不是与 IEEE 1149.1 100％的兼容。例如没有一款 MSP430 有边界扫描单元。
我们只支持所需的 BYPASS 命令，但是不支持其他所需的命令：外测试和取样/预加载。
后果：
对 MSP430 设备，没有 BSDL 文件。
你不能通过一个 JTAG 链把 MSP430 与其他设备连接在一起。

193. 编译完 MSP430 的一个工程，IAR 嵌入式工作台的连接器提示这个错误：无法打开文件`cl430`MSP430 的 USART 可以同时工作在 UART 和 SPI 模式吗？

一个 USART 模块不能同时工作在 UART 和 SPI 模式。但 MSP430 可以通过软件在两个模式这间切换。如果 MSP430 单片机有两个的 USART 模块：USART0 和 USART1，则它们可以同时工作在 SPI 或 UART 的任何组合模式。
If switching between modes is done, please make sure the SWRST bit is set and cleared to reset the USART state machine between the change of modes by software. Please refer the device family user`s guide for more details.
如果模式之间进行切换，请在通过软件切换模式的过程中确保 SWRST 位的设置和清除来复位 USART 状态机。

194. MSP430 的 Flash 存储器能否修改，或作为 EEPROM 的使用，同时程序代码依然从 Flash 存储器中中读取？

在 MSP430 能够编辑系统内 Flash 存储器的任何位置的任何一个位，字节或字。在系统中，Flash 可以被编程，即使程序正在 Flash 中执行，甚至程序正在从一段正在被编辑的程序段中执行代码。一个代码段不必擦除来修改，但 1`S 只能被修改为 0`S。擦除是对所有的代码段进行的，并且擦除代码段上的所有位为 1`S。在系统中，Flash 的修改或擦除依然可以进行，即便是正从 Flash 中执行代码，程序计数器在设备数

据表指定的时间内自动停止。作为一个选择，在系统 Flash 存储器修改或擦除期间，程序计数器可以移动到 RAM 去执行一个应用程序—这样代码继续在 RAM 中全速执行。无论是信息存储器和主存储器 Flash 可用于任何数据或代码或两者兼而有之。唯一不同的是，信息存储器由规模较小的 128 字节段组成，而主存器由 512 字节段组成。编辑 Flash 不需要更高的电压。

195. MSP430 能在一组端口上同时处理多个中断，例如在 P1.0 到 P1.7 或者 P2.0 到 P2.7，而不漏一个中断吗？对这些端口似乎只有一个总中断标志能被 CPU 所识别。

是的，只要所要求的最低中断事件脉冲宽度得到保证，MSP430 决不会错过任何一个中断。这些多源中断标志依然会置位，即使中端请求已经接收和提供服务，所以已经服务的中断标志在他相应的中断服务程序中必须复位。这样在这期间依然保持着的中断就会被 CPU 所识别。

196. MSP430F11x1 和 MSP430F11x1A 有什么区别？

MSP430F11x1：

- BSL 版本号为 1.10（勘误：BSL2 和 BSL3）
- 烧熔丝功能不对应用开放（勘误：FUSE2）
- 为操作安全，需要在 Test/Vpp 引脚加一个外部的下拉电阻（勘误：TEST1）

MSP430F11x1A：

- BSL 版本号为 1.30（勘误：BSL2 和 BSL3 修订，查阅 "Features of the MSP430 Bootstrap Loader"）
- 烧熔丝功能对应用开放（勘误：FUSE2 修订）
- 不需要在 Test/Vpp 引脚加外部的下拉电阻（勘误：TEST1 修订）

提示：
如果不使用 Test/Vpp 引脚，最好将其接至 VSS。如果 Test/Vpp 信号接在 JTAG 口上进行在线调试和下载，可以接一个下拉电阻以提高 EMI 和 ESD 性能。

197. MSP430x4xx 系列单片机在上电时 32K 的晶振不能可靠起振，怎么办？

MSP430x4xx 系列单片机集成了软件可编程的输入电容 C（XIN）和输出电容 C（XOUT），以提供石英晶体振荡器振荡所需的负载电容。在上电时，默认的负载电容为 0pF（即没有负载电容）。
为了使 32KHz 晶体正常启动，必须用软件配置好所需的负载电容。
实例：对于手表晶振，需要 10 到 12pf 的负载电容，所以在初始化代码中需要配置 XCAPxPF 为 18pf。切换负载电容时，等待晶振起振和 FLL 稳定需要确保有足够的时间。有一个监测振荡器的正常运作与否的方法就是检查振荡器故障位。

198. 使用 MSP430 串口编程工具 MSP-PRGS430 的问题？

对于使用 MSP-PRGS430 问题解决方案的完整列表，请参阅最新版本的`MSP430 系列串行编程编程器手册`，TI 文献编号 SLAU048。同时请安装最新版本的 PC 用户界面软件。软件可以在 MSP430 下的 tool updates 下载。这一工具已经经过测试，支持所有的 MSP430 OTP 和 FLASH 器件。

199. 80 h 仅用于同步还是必须跟每个 bootstrap loader 命令一起发送？

在每次传输中 80h 用于同步。芯片返回 90h 作为响应，然后发送一个数据帧。每一帧以一个 80h 的字节开头，帧其它字节紧跟在 80h 之后。BSL 数据帧的格式在“Features of the MSP430 Bootstrap Loader”（文献编号 SLAA089）有定义。，可以在 MSP430 网站上下载。在应用笔记“Application of Bootstrap Loader in MSP430 w/Flash -Hardware and Software Proposal”（文献编号 SLAA096）中，有详细的代码测试证明此技术的正确性。在应用笔记中提到的软件和硬件已经经过测试并且可以工作。

200. 拥有 BasicClock 模块，而没有 FLL 的 MSP430x1xx 系列芯片中，可不可以利用内部高速的 DCO 得到稳定的频率？

DCO 具有电压、温度补偿及器件差异，可以进行校准并设置为一个特定频率。校准时一般会用到一个低速晶振或外部信号，通过比较低速参考频率一周期内高速 DCO 时钟周期的计数值完成。通过软件调整 BasicClock 控制寄存器，设置 DCO 到希望的频率，一般为低速晶振或外部信号的整数倍。芯片的数据手册上有更为详细的 DCO 频率范围。
MSP430x1xx 用户手册提供了 BasicClock 的详细使用方法。可以在 MSP430 官方网站上下载到如何进行 DCO 设置的代码范例和应用报告。

201. 在一些 MSP430x1xx 系列的芯片上如何实现 JTAG 接口和 I/O 引脚的复用？

20 脚和 28 脚封装的 MSP430F1xx 芯片，其引脚 P1.7 - P1.4 具有 I/O 和 JTAG 接口两个功能。当芯片上电时，这些引脚的默认功能为 I/O 口。当 Test 引脚被拉高的时候，这些引脚就会被选择为 JTAG 接口。针对这些芯片的 FET 工具在使用在线调试的时候就会将这些引脚置为 JTAG 接口功能。请参阅 FET 工具用户手册查看在调试时如何释放这些管脚的 JTAG 功能。
注意：如果外部电路连接到复用引脚上时，一定要考虑这些引脚上的 JTAG 信号。通过 JTAG 接口对芯片进行在线调试，需要考虑电路的影响。如果电路将复用引脚拉低或者偏置改变，就会影响到 JTAG 通信。管脚更多的芯片则设计了专门用于调试和下载的 JTAG 引脚。

202. MSP430 ADC12 的最快和最慢速度是多少？

ADC12CLK 的功能是为 ADC12 提供转换时钟。ADC12CLK 的最小和最大时钟分别近似为 500kHz 和 6.5MHz。最快的完全转换周期为 17 个时钟周期（转换为 13 个时钟周期，采样保持为 4 个时钟周期）。6.5MHz/17 = 382ksps.ADC12 的时钟周期不能小于 ADC12CLK 的最小周期，但在软件控制下可以使采样一直处于开启状态。有关采样和转换时间更详细的说明请参考数据手册。

203. 如果 ADC12 的 VREF+，VeREF+和 VREF-/VeREF-引脚没有用到，它们该如何连接？

请将 ADC12 没有用到的参考引脚连接如下：
VREF+ =开路
VeREF+ = DVss
VREF-/VeREF- = DVss

204. ADC12 模块的 VeREF+, VREF+ 或 VREF-/VeREF-引脚需要接外部电容吗？

如果使用外部基准或者某个内部基准作为正极参考基准电压源，需要在 VeREF+ 或 VREF+引脚外部接一个 5-10uF 的电容。如果使用外部基准作为负极参考基准电压源，需要在 VREF-/VeREF-引脚外部接一个 5-10uF 的电容。但是如果使用 AVCC 作为正极参考基准电压源，那 VeREF+ 或 VREF+外部就不需要接电容，如果用 AVSS 作为负极参考基准电压源，那 VREF-/VeREF-引脚外部就不需要接电容。当使用 AVcc 和 AVss 作为参考时，电源退耦电容提供必要的低阻抗路径。

205. 28035 CCS 哪里配置错了，出现这个警告？

warning: entry-point symbol other than "_c_int00" specified: "code_start"
Description Resource Path Location Type
entry-point symbol other than "_c_int00" specified: "code_start" Example_2803xClaAdcFir
C/C++ Problem

在设置中 Properties->Build->C2000 linker->Advanced Options->Symbol Management 中，下方的 Specify program entry point for the output module 处，将 code_start 替换成 _c_int00，或者不填，重新编译就可以消除这个警告了

206. 在看 FR5969 的例程时，其中看到这样定义了一个语句：

```
#define WriteCmdData(ucCmdData)
        do
        {
            while (!(UCB0IFG & UCTXIFG));
            UCB0TXBUF = ucCmdData;
        }
        while(0)
```

很明显，这个语句只执行一次。那么，它为什么一定要用 DO -- WHILE 呢？

在宏定义里面经常看到 do{...}while(0)这样的语句。
确实很让人疑惑，do while(0)就是让代码只执行一次，何必这样呢，为什么需要用 do while 呢？
其实，宏定义就是一个代码替换的过程。
#define CODE_SEG(a) printf("sample macro:%s\n",a)
那么在出现了 CODE_SEG(some_str)的地方，它都会在编译时被替换成 printf("sample macro:%s\n",some_str)

那么，我们写代码的习惯都是

```
...
char*s="hello world";
CODE_SEG(s);   //我们会习惯性地加上分号
...
```

这时，可以编译通过，因为 CODE_SEG(s); 被替换成了 printf("sample macro:%s\n",s);
但是，如果我们的宏里面有不止一条语句，比如
#define CODE_SEG(a) {printf("sample macro:%s\n",a);printf("done\n");}
这样就行不通了，因为 CODE_SEG(s); 会被替换成
{printf("sample macro:%s\n",a);printf("done\n");}; 最后面这个分号回导致编译不通过

因此，通常用 do while(0)来包裹，这样就可以避免这个分号的问题
#define CODE_SEG(a) do{printf("sample macro:%s\n",a);printf("done\n");}while(0)
CODE_SEG(s); 会被替换成
do{printf("sample macro:%s\n",a);printf("done\n");}while(0);
最后的这个分号此时就是恰到好处了，编译通过。

207. 怎么计算两个不同时间中的分钟之间的差值？

比如 A 时间为 12:11:13,（时，分，秒）B 时间为 12:16:15
当 A 和 B 时间的分钟都没有越过 59 的时候，相减一下就可以了。
那如果 A 时间的分钟刚刚越过 59，B 没有越过 59，我可以借助小时进位，A 小时比 B 小时多了 1，然后加上 60 分钟去减去 B 的分钟。
那如果 A 时间是 23:59:59
B 时间是 23:55:55
当 A 的分钟越过 59 的时候，A 的小时变为 0，我又没有设置天，那该怎么计算？
有没有简单一点的方法，不用考虑这么多的？

设 A、B 为两个时间的分钟值，若 A>=B，二者之差为 A-B。若 A< B，二者的差为 A+60-B。

208. msp430 同时执行多少任务最稳定？

我做的 msp430F149 单片机,同时执行以下任务：
1.TXD0 和 RXD0 连接 GPRS 模块进行指令发送。
2.TXD1 和 RXD1 连接 485 和其他模块进行通信。
3.A/D 测量模拟量。
4.同时液晶显示。
这个任务量多不多啊？现在感觉程序很不稳定啊，时不时就不按照预定的效果进行，有时候中断都进不去。
while(1);//这就是阻塞语句
while(flag);//这还是阻塞语句

```
int main()
{
while(1)
{
   if(flag==1){do_something_1();flag=2;}
   if(flag==2){do_something_2();flag=4;}
   if(flag==3){do_something_3();flag=5;}
   if(flag==4){do_something_4();flag=0;}
   if(flag==5){do_something_5();flag=6;}
   if(flag==6){do_something_6();flag=0;}
}
}
```

类似上面这样，在中断里 i 就设置 flag 的值就好了，设置完就退出中断，由于 flag 初始为 0，上面的所有语句都不执行，知道 flag=1，才开始依次执行 1-2-4，然后 flag 回到 0。
还有，某些时候我们看到 delay 延时函数是用 for(;i>0;i--);这样的方式实现的。其实是让单片机空跑 N 个循环，虽然单片机在空跑，但是 CPU 资源依旧被占用着不能执行其他的任务。如果把这部分时间拿出来好好利用好，也可以提高多任务执行的效果。
举例说一下：

还是上面的状态机，我们看到 flag 的执行顺序是 1-2-4-0，假定只有 1-4 两个任务，中间是需要延时 1 秒。那么我们可以这么做：
1.
设置一个定时器中断，中断周期 1ms，中断服务里写 if(delay>0)delay--;这里的 delay 是一个全局无符号整形变量。
2.
```
if(flag==1){do_something_1();delay=1000;flag=2;}
if(flag==2){do_something_2();}
do_something_2()
{
if(!delay)flag=4;
}
```
delay 会从 1000 往下减，每 1ms 减 1，知道 0 的时候符合 flag==2 且!delay=true 条件，flag=4 跳转到下面的语句。在这之前 flag=2，且 if(flag==2)的语句很快就执行完，不会耽误 CPU 执行其他的任务（例如 if(flag2==1)之类的）
不执行任务最稳定，即使程序跑飞了你也不知道。
执行 N 个任务也能很稳定，只要单片机有足够的周期完成所有的任务。

给几个提醒：

- 中断要短
- 尽可能用中断唤起程序，不要把程序阻塞住
- 在主函数里执行需要的任务，必要的时候把任务分解成状态机的形式
- 各模块尽量用查询的方式，不要用 WHILE 等一些延时循环；

209. tms320F27028 如何选择外部晶振？

我最近在 CCSv5 上写 tms320F27028 的程序，看了 controlsuite 的例程。里面都是把系统时钟设置成内部 RC1.现在想换成外部晶振。不知道该怎么做。请大神指导一下。谢谢了！
1. 硬件上连接晶振
2. 软件上在 systemctrl.c 中，改成下面的

```
//---------------------------------------------------------------------------
// Example: XtalOscSel:
//---------------------------------------------------------------------------
// This function switches to External CRYSTAL oscillator and turns off all other clock
// sources to minimize power consumption. This option may not be available on all
// device packages

void XtalOscSel (void)  {
     EALLOW;
     SysCtrlRegs.CLKCTL.bit.XTALOSCOFF = 0;     // Turn on XTALOSC
     SysCtrlRegs.CLKCTL.bit.XCLKINOFF = 1;      // Turn off XCLKIN
     SysCtrlRegs.CLKCTL.bit.OSCCLKSRC2SEL = 0;  // Switch to external clock
     SysCtrlRegs.CLKCTL.bit.OSCCLKSRCSEL = 1;   // Switch from INTOSC1 to INTOSC2/ext clk
     SysCtrlRegs.CLKCTL.bit.WDCLKSRCSEL = 1;    // Switch Watchdog Clk Src to external clock
     SysCtrlRegs.CLKCTL.bit.INTOSC2OFF = 1;     // Turn off INTOSC2
     SysCtrlRegs.CLKCTL.bit.INTOSC1OFF = 1;     // Turn off INTOSC1
     EDIS;

}
```

210. MSP430 的 DTC 和 DMA 的区别？

The DTC allows ADC10 samples to be converted and stored anywhere in memory without CPU intervention.

DTC 用在 adc 中，是转换结果传输控制的，也就是转换结果不需要 CPU 干涉。

The DMA controller module transfers data from one address to another without CPU intervention.

DMA 用于数据结果的搬运

211. MSP430 的 JTAG 口的使用，请教！

1） 我要用 MSP430F123，JTAG 口是连 8 根线吗？

2） 用上述的 JTAG 口，和 280 元的适配器就可以调试和下载了吗？

3） MSP430 连接 JTAG 口的引脚还可以做其他用途吗？如何使用？

对于 FLASH 系列 的 MSP430F11X 和 12X 系列的 JTAG 口是需要 8 根线接口；MSP430 FLASH 系列 FET 是通用的；对于 F11X、F12X 的 JTAG 接口的 TDO、TDI、TMS、TCK 这几个管脚是有 P1.4 、P1.5、P1.6、P1.7 是复用的，只有当下载、调试完成后才可以作为通用 I/O 口用。

212. 430 浮点数传输，请问如何用 C 语言实现，一浮点数从串口传出，在上位机显示？

把浮点型数据强制转化成 4 个字节直接发送过去，接收到 4 个字节之后再强制类型转换回浮点数即可，至于到了上位机的显示，调用 MessageBox 或者 printf 函数即可。

当然，我的上位机用的是 VC 编写的程序，你要是用串口调试助手之类的工具可就没法直接观察了，呵呵

示例代码：

发送----

```
for ( i=0; i<4; i++ )
{
SendChar( *((unsigned char *)(&FloatVal)+i) );
}
```

接收----

```
FloatVal = *(float *)(&RevBuffer[0]);
```

213. MSP430 双电源设计，我做的双电源设计：

锂电池 3.6V---二极管-------MSP430 ××CC DVCC

纽扣电池 3V------二极管------MSP430 ××CC 用作外部电池断电以后 RAM 数据保持，不知是否可行，还有就是不知道二极管再导通后的功耗多大。

请教楼主，有没有成熟的双电源设计方法，怎样设计才能保证最低的功耗，在用外部电池供电时需不需要加电容滤波？

所描述的双电源实现起来不复杂，主要有两点要注意：

1、不要直接将锂电池与 MSP430 电源相接，应该经过 LDO。

2、锂电池供电线路与纽扣电池供电线路分别经过肖特基二极管向 430 供电，要选反向漏电流小的二极管。

最好用 430 的 SVS 进行供电电压监视，并有低压报警显示和（或）指示。为了省电，SVS 不要经常开启，适当拉大检测周期。

214. MSP430 launchpad 好像有两个 TimerA，这样是不是可以说 MSP430 launchpad 能输出两路独立 PWM 吗?

首先找选择合适的 PWM 输出引脚，比如第 1 路 PWM 想用 TA0 控制，第 2 路 PWM 用 TA1 控制查器件手册（是针对具体型号的，不是 UserGuide 哦）：
TA0 支持的引脚有 P1.1、P1.2、P1.5、P1.6、P2.6（P2.6 一般不用，因为要接晶振哦）
TA1 支持的引脚有 P2.0 --> P2.5。
选择方便的，假设我选择了 P1.6（TA0.1）和 P2.2（TA1.1），且第 1 路由 ACLK 驱动输出 4KHz 方波，第 2 路也有 ACLK 驱动但是输出 2KHz 占空比 25%的方波。
P1DIR |= BIT6; // 选择 P1.6 的第二功能，即 TA0.1 P1SEL |= BIT6; CCR0 = 7; // PWM 周期 = ACLK@32768/4KHz-1 = 8.192-1 = 7 CCTL1 = OUTMOD_6; // CCR1 reset/set CCR1 = 3; // CCR1 PWM 占空比 50% =
8.192*50% - 1 = 3
P2DIR |= BIT2; // 选择 P2.2 的第二功能，即 TA1.1 P2SEL |= BIT2; TA1CCR0 = 15; // PWM 周期= ACLK@32768/2KHz-1 = 16-1 = 15 TA1CCTL1 = OUTMOD_6; // CCR1 reset/set TA1CCR1 = 3; // CCR1 PWM
占空比 25% = 16*25% - 1 = 3
ACLK 最多支持 32768Hz 的 PWM，如果要更高频率，需要换时钟源为 SMCLK，即代码改为：
TACTL = TASSEL_2 + MC_1; // SMCLK, up mode
当然时钟源换了，控制频率的 CCR0 和 CCR1 的值要重新计算。
友情提示：如要进入 LPM3，则 PWM 的时钟只能为 ACLK，因为在 LPM3 下 DCO（一般作为 SMCLK 的源）会关闭。

215. 请教 MSP430 F5438 芯片后带 A 与否有啥区别？

MSP430F5438 和 MSP430F5438A 没有太大的差别，直接使用 MSP430F5438 的例程即可。“A”版本的 MSP430F5438 在晶圆上提供了更好的性能、功耗和 MSP430F5xx 家族所标称的所有功能。

216. 请教关于 MSP430 串口波特率计算的问题？

在用户手册上看到如果时钟为 32.768K，波特率设置为 9600，那么 uxbr0 = 0x03,uxmctl=0x4a.不太明白 uxmctl=0x4a 的由来。按照公式计算出来是 3.41，整数部分为 3，小数部分为 0.41，0.41*8 去整后得到为 3，怎么变换得到 0x4a 呢？
这个 3 表示在 UxMCTL 中的 8 位里要有 3 个 1，并且，UxMCTL 分为 First Stage Modulation 和 Second Stage Modulation，也就是前者为高 4 位，取值范围 0-F，后者为低 4 位，注意后 4 位最好选择偶数。把上步的到的小数部分取整后的数值分散到高位和低位，如 3，可以写为 0x16。
大意就是，你的出来的 3，要在 UxMCTL 的 8 位中占有 3 个 1.而且后 4 位中 1 的个数最好是偶数。所以说 0x4a= 01001010，这样，有 3 个 1，且后 4 位 1 的个数为偶数。这样的话 0x16 之类的也是符合的。

217. 430 能产生 1M 的方波吗？

不影响其他的工作，现需要一个 1M 的方波驱动抗混滤波器，能用 430F1611 产生一个 1M 的方波吗？
现在的情况是，如果用定时器 A 中断产生方波影响 CPU 的正常工作了。
可以用定时器的 PWM 功能，或者你把要产生方波的脚直接接到，SMCLK OR MCLK 上，然后接 XT2，分频。2M 或 4M 的都可以。

218. 有时 430 部分引脚损坏，如何有效又方便的检测出全部损坏的引脚呢？

每个 port 输出 0x55　间隔一低一高

219. MSP430 JTAG 下载问题 ？显示 cannot load program because flash emulation tool not found .检查线路没有问题！请教是什么问题！

需要改选项中的一些设置！在 project/options/c-spy/setup/driver 下，选中 flash emulation tool 就可以了

220. MSP430 烧熔丝有几种方法？要使用那些对应的工具？

方法都是一个，工具现在应该有 2 种可靠的 一个用 usb 口的带熔丝烧断功能的仿真器，另一个是编程器

221. 在熔丝不坏的情况下，有几种读 flash 的方法？BSL 功能怎么玩？

熔丝不坏用 jtag 和 bsl 都可读 bsl 需要密码 bsl 有专门的 bsl 编程器 专业编程器也支持 bsl

222. 现在 430 的破解有几种啊？听说有一种直接打磨 MCU 破解的，感到很神秘啊！

430 破解没啥好办法，只能用 bsl 去不断验证

223. MSP430 读写 8563 问题 ？

为什么我的 430 总是不能读到 8563 的应答低电平，SDA，SCL 都加了上拉，430 的输入口也正常，这几天一直被这个问题困饶，请大家帮我解决一下多谢！下面是我的程序，帮我看一下哪有问题？

P1.1 做 SDA,P1.2 做 SCL

```
void i2c_start (void)
{
P1DIR |= BIT1;
P1DIR |= BIT2;
P1OUT &=~BIT2;//sclk = 0;
P1OUT |= BIT1;//sda = 1;
P1OUT |= BIT2;//sclk = 1;
P1OUT &=~BIT1;//sda = 0;
}

void i2c_stop (void)
{
P1DIR |= BIT1;
```

```
P1DIR |= BIT2;
P1OUT &= ~BIT2;//sclk = 0;
P1OUT &= ~BIT1;//sda = 0;
P1OUT |= BIT2;//sclk = 1;
P1OUT |= BIT1;//sda = 1;
}

void i2c_send_data (unsigned char sd)
{
unsigned char i;
P1DIR |= BIT1;
P1OUT &= ~BIT2;//sclk = 0;
for (i = 0; i < 8; i++)
{

if(sd & 0x80) P1OUT |= BIT1;
else P1OUT &= ~BIT1;
P1OUT &= ~BIT2; //sclk = 0;
P1OUT |= BIT2;//sclk = 1;
sd <<= 1;
P1OUT &= ~BIT2;//sclk = 0;
}
P1OUT &= ~BIT2;//sclk = 0;
P1OUT |= BIT1;//sda = 1; //发送 8 个字节后 sda 产生高电平标志位
P1DIR &= ~BIT1;
P1OUT |= BIT2;//sclk = 1;
P1OUT &= ~BIT2;//sclk = 0;
do
{
i = (P1IN & BIT1);//i = sda;//?
}
while(i);
}

void i2c_receive_data (unsigned char *p_rd)
{
unsigned char rda = 0, i;
P1DIR |= BIT1;
P1OUT &= ~BIT2;//sclk = 0;
P1OUT |= BIT1;//sda = 1;
P1DIR &= ~BIT1;
for (i = 0; i < 8; i++)
{
P1OUT &= ~BIT2;//sclk = 0;
rda <<= 1;
```

```
P1OUT |= BIT2;//sclk = 1;
rda |= ((P1IN&BIT1)>>1);//SDA

}
P1OUT &= ~BIT2;//sclk = 0;
P1DIR |= BIT1;
P1OUT &= ~BIT1;//sda = 0;
P1OUT |= BIT2;//sclk = 1;
*p_rd = rda;
}
```

主频高了，降低时钟就好了

224. 50MHz 的 LM3S 读 SPI FLASH 稳定最快速度能到多少呀？

最快应该可以到 50M，稳定可以做到 8M。

225. msp430 的学习板用 BSL 和 JTAG 的区别 ？

BSL 只支持烧程序，JTAG 可以烧程序，也可以仿真调试，JTAG 速度比 BSL 快不少。

226. LM3S 用 IAR 还是 KEIL 开发比较好呢？

IAR 稳定些

227. flash 型 430 单片机它的 RAM 中保存的数据掉电后会不会丢失？

RAM 区掉电就丢了,你可以配合掉电检测,然后把要存的数存入 FLASH 或 EEPROM 中

228. MSP430 可以位操作吗？

位操作没有。
但是比如你要对 P1.2 置位：P1OUT |= BIT2;
对 P1.2 清零：P1OUT &= ~BIT2;
程序这样写可读性比较好。

229. 以 PWM 为控制来输出电压的，430 的 PWM 最大输出电压是多少啊？

这样实现的电压只是在很小的范围内变化，就是基波分量的比例的变化。
影响输出电压幅度的因素有 PWM（存储波形数据的傅立叶变换）的构成及外加滤波网络的性能。输出电压的最大值不会超过 VCC。

230. MSP430 是否可以同时产生 3 个 PWM 信号？

要看哪个型号的了，MSP430 是很大一个家族的，比如最新的 F5XXX 系列就肯定支持，很多低端的系列都没有 PWM。

231. MSP430 一条语句不太理解!

TATCL=TASSEL_1+MC_1;这条语句是不是相当于 TACTL=TASSEL0+MC0;?
因为头文件定义为：

```
#define TASSEL0             (0x0100u)  /* Timer A clock source select 1 */
#define MC0                 (0x0010u)  /* Timer A mode control 0 */

#define MC_1                (1*0x10u)  /* Timer A mode control: 1 - Up to CCR0 */
#define TASSEL_1            (1*0x100u) /* Timer A clock source select: 1 - ACLK  */
```

这是为什么呢，有大侠可以给我讲讲么，
另外#define TASSEL_3 (3*0x100u) /* Timer A clock source select: 3 - INCLK */
中 3*0x100u,怎么回事啊？

LZ，你好。两者是等效的。
是否可以这样理解，头文件中先把寄存器的每一位都表示出来，然后有些控制命令是由某些位组合控制的，比如说 TASSEL 时钟信号选择有两个位 TASSEL0 与 TASSEL1。这就能产生 4 种组合方式：

TASSEL1	TASSEL0
0	0
0	1
1	0
1	1

实际中为了方便，头文件把这 4 钟不同的方式也宏定义出来，也就是

```
TASSEL_0  = 0*TASSEL1+0*TASSEL0 =0*0x200u+0*0x100u = 0          =0*0x100u
TASSEL_1  = 0*TASSEL1+1*TASSEL0 =0*0x200u+1*0x100u = 0x100u  =1*0x100u
TASSEL_2  = 1*TASSEL1+0*TASSEL0 =1*0x200u+0*0x100u = 0x200u  =2*0x100u
TASSEL_3  = 1*TASSEL1+1*TASSEL0 =1*0x200u+1*0x100u = 0x300u  =3*0x100u
```

232. MSP430 普通 io 口设置,把某一引脚定义为输出，如果有信号输入，那我查看该脚的输入寄存器是不是就是这个输入信号呢？

因为用普通 io 口实现 iic 的通讯，要用到响应这一条，如果哪位大神有 iic 的示例代码万分感谢！！
用硬件 IIC,你读取 PxIN 寄存器的值即可得到输入的数据值。

233. 173. MSP430 的 A/D 参考电压问题,MSP430X149 系列的 A/D 参考电压 VREF-不能为负，但我们输入的信号为 0~3V 的正负方波，请达人指点，当输入为-0~-3V 时，MSP430 工作是否会正常，有没有什么影响？

VREF-接在 1.5 伏上，Veref+接在 3 伏上，然后将输入量的中性点提升到 1.5 伏，输入量用运放限制在 3 伏的幅值以内，将 A/D 转换结果在进行中性点偏移就行。

234. 如何用 MSP430F169 实现 PWM 波输出 ？

可以使用定时器来实现，配合定时器的多种模式和输出调制，可以实现带延迟时间或者死区控制的 PWM 波。

235. 当我需要对大量 MSP430 的 Flash 进行编程时我有什么选择?

当需要对大量 MSP430 进行编程时你有以下选择：
使用编程器（已可以使用, 不需要开发）
MSP-PRGS430
BSL 工具(e.g. from Gessler Elektronik, Softbaugh, Elprotronic)
Gang-Programmer MSP-GANG430
对于第三方工具，请查阅
http：//www.ti.com/sc/MSP430
>>> Third Party >>> Third Party tools
使用一个配合自有软件的编程器(需要一定的开发能力)：
使用 Windows DLL 带来的 MSP-PRGS430 和 MSP-GANG430 工具 。你可以使用 DLL 的函数，用你自己的开发软件对 MSP430 进行编程。DLL 的函数可以从工具用户指南 SLAU048 和 SLAU101 中获得 。
下面的事你可以独立完成：
JTAG-接口：
可以从网上获得 JTAG 接口相关的文档：
Programming a Flash-based MSP430 Using the JTAG Inte**ce (slaa149)
BSL-接口 (RS232)：
你可以从网上获得 BSL 接口相关的文档：
Features of the MSP430 Bootstrap Loader (slaa089a)
Appl. of Bootstrap Loader in MSP430 with Flash HW and SW Propo (slaa096b)

236. MSP430 最小系统需要加复位电路吗？

MSP430 一般是不要 RC 复位的，一般只要接个 100K 左右电阻就可以了，如果要加电容，它的大小要根据以下两个标准选择：

- 下载程序不会出现下载不了
- 程序上电会能稳定复位；

237. MSP430 的 I/O 的设置顺序在程序里的位置是有区别的

```
int i,a[10];int ADC10_Result;
ADC10CTL1 |= CONSEQ_2;        //单通道重复采样
ADC10CTL0|= SREF_1+REFON;     //选择并使能内部参考源 1.5V
ADC10CTL0 |= ADC10SHT_3+MSC;    //过采样率为 16 个采样周期，只用第一个 shi 信号触发采样转换
ADC10CTL1 |= ADC10SSEL_1+ADC10DIV_1+SHS_0;      //ACLK，2 分频，ADC10SC 触发采样。SHS：采样保持触发源选择
ADC10CTL0 |= ADC10ON;                           //开启 ADC10
ADC10AE0 = BIT1;                                //使能模拟输入外部通道 A1
P1DIR = BIT6;                 // 这里设置 P1 口是正确的。
/*********************************************************************************/
P1DIR = BIT6;
int i,a[10];int ADC10_Result;
ADC10CTL1 |= CONSEQ_2;                          //单通道重复采样
ADC10CTL0|= SREF_1+REFON;                       //选择并使能内部参考源 1.5V
ADC10CTL0 |= ADC10SHT_3+MSC;        //过采样率为 16 个采样周期，只用第一个 shi 信号触发采样转换
ADC10CTL1 |= ADC10SSEL_1+ADC10DIV_1+SHS_0;      //ACLK，2 分频，ADC10SC 触发采样。SHS：采样保持触发源选择
ADC10CTL0 |= ADC10ON;                           //开启 ADC10
ADC10AE0 = BIT1;           // 这样在开始时设置 P1 口是不正确的。
```

238. 怎样理解 MSP430 的 I2C 模块停止位的工作状态？

MSP430 的硬件 I2C 模块的停止位要求在最后一个字节的传送过程中发出 UCTXSTP 信号，一般可以通过判断 UCTXSTT，然后一送出下一个字节就置位 UCTXSTP。而控制字节是硬件自动发出的，一旦启动发送过程就开始向外传送。

239. MSP430 的 IO 口短路问题？

不加上拉电阻直接短路的话，IO 还是可以用，但是实际上内部电路可能已经损坏。用电流表测总电流，即使低功耗模式损坏 IO 口的单片机耗电量也比参考值大 10 倍左右。更换单片机后无问题，这个问题比较隐蔽。

240. MSP430 的 IO 口比较脆弱？

使用矩阵键盘时曾经发现 P1 口不接上拉电阻不能触发外部中断。

241. 作为 CC430, 使用方面值得注重的是什么方面?

首要特点是：宽电源电压范围，超低功耗，集成 RF 收发器内核，16 位 RISC CPU，16 位寄存器和可获得最大编码效率的常数发生器.主要涉及 RF 网络、能量采集、工业监控与篡改检测、个人无线网络以及自动抄表基础设施(AMI)等在内的应用。

242. MSP430 编程中 const 的用法是？

MSP430 IAR 编译器会把带 const 关键字变量放到 FLash 中，例如嵌入式常用的的表格军使用 const 关键字，const int Ta××e[64]...

243. MSP430 的开发环境的软件是否免费？

对于初级用户有体验版，足够使用。例如 IAR 提供 4K 的免费体验板。

244. MSP430F4152IPMR 型单片机的应用特点？

这款单片机的应用特点为：电池供电要求低功耗，且应用大部分时间工作在休眠模式，一段时间才唤醒工作一次，功能外设需要段式 LCD，模拟比较器，AD，数据通讯需要为 SPI/IIC/UART，程序对 RAM 需求不大，但是需要存储较多数据的场合；

245. 如何快速获取 MSP430G 系列器件 Datasheet？

1.打开网页浏览器，输入网址 http://datasheet.eeworld.com.cn/
2.在器件查询输入需要的器件型号，点击查询
3.找到自己需要的器件完整型号，注意封装大小，点击下载
4.打开后可以在线浏览，也可以下载到本机进行阅读

246. 为了增大 CC2530 的传输距离，我们一般会采取哪些方法？

首先，作为一款无线 MCU，其传输距离受天线设计，气候条件影响很大。除去不可控因素外，可以针对天线进行优化，另外一种解决方法是与 CC2590 组合使用，这样的电路设计的优点是增大了输出功率，并且提高了接收的灵敏度。

247. MSP430 如何通过 BSL 下载程序？

MSP430 可以通过 JTAG、SBW 及 BSL 下载程序(部分型号不支持 SBW)BSL 电路就是一个 USB 转串口的电路，把 232 电平转换成 TTL 电平，然后通过 PC 的上位机软件来模拟写入 430 程序的时序来完成。

248. TI MCU430 可以兼容或者做为 arduino 开发板吗？开发环境是什么？

可以的，TI MCU430 可以使用开源 arduino IDE：Energia，开发流程与 arduino 开发一致。详情请访问 Energia 官方网站：http://energia.nu/

249. MSP430 的串口通信问题？

注意一点： lauchpad 的 J4 跳线帽地方的 RX 和 TX 需要竖着插。横着插不能通信。
贡献自己写的 MSP430G2553 的串口通信代码：使用内部 DCO 到 16M,选择串口通信时钟来源为 SMCLK=8M（系统时钟 2 分频)。

1.设置波特率为9600。计算方法为：Fclk=SMCLK=8M.　N=Fclk/9600=833.333；则UCxBRx=INT(N)=833(N取整）。又UCxBRx=UCBR0+(UCBR1*256)，所以UCxBR1=3;UCxBR0=65 ；　UCBRSx=(N-INT(N))*8=2　(四舍五入)

下面是源代码：

```
void UartRegCfg()
{
UCA0CTL1 |=UCSWRST;      //reset UART module,as well as enable UART module
UCA0CTL1 |=UCSSEL_2;     //UART clock is SMCLK
UCA0BR0  |=65;           //Baud N=BCLK/rate,rate=9600,BCLK=SMCLK=8M
UCA0BR1  |=3;
    UCA0MCTL  = UCBRS1;      //UCBRSx=2
UCA0CTL1 &=~UCSWRST;     //UART reset end
}
void UartGpioCfg()
{
P1DIR  |= BIT2;           //P1.2  UART_TX
P1DIR  &=~BIT1;           //P1.2  UART_RX
P1SEL  |= BIT1+BIT2;      //select P1.1 and P1.2 as UART port
P1SEL2 |= BIT1+BIT2;
}
void UartInit()
{
UartRegCfg();
UartGpioCfg();
}
/*****************************************************************************
* Function Name : UARTPutChar
* Create Date : 2012/07/27
* Author  :
*
* Description :send a character
*
* Param : cTX is willing to send character
*****************************************************************************/
void UARTPutChar(unsigned char cTX)
{
UCA0TXBUF=cTX;
while (!(IFG2&UCA0TXIFG));  //waiting UCA0TXBUF is empty
    IFG2&=~UCA0TXIFG;           //clear TX interrupt flag
}
/*****************************************************************************
* Function Name : UARTGetChar
* Create Date : 2012/07/27
* Author  :
*
* Description :get a character
*
* Param : cRX is willing to get character
*****************************************************************************/
```

```
int UARTGetChar(void)
{
int GetChar=0;
while (!(IFG2&UCA0RXIFG));  //UCA1RXBUF has received a complete character
IFG2&=~UCA0RXIFG;           //clear RX interrupt flag
UCA0TXBUF=UCA0RXBUF;        //back to display
GetChar  =UCA0RXBUF;
while (!(IFG2&UCA0TXIFG));  //waiting UCA0TXBUF is empty
IFG2&=~UCA0TXIFG;           //clear TX interrupt flag
return GetChar;
}
/**************************************************************************
* Function Name : UARTPutstring
* Create Date : 2012/07/27
* Author  :
*
* Description :output string
*
* Param : char *str point send string
* return: the length of string
**************************************************************************/
int UARTPutstring( char *str)
{
   unsigned int uCount=0;
   do
   {
    uCount++;
    UARTPutChar(*str);
   }
   while(*++str!='\0');
   UARTPutChar('\n');
   return uCount;
}
void SysCtlClockInit()
{
    DCOCTL=0;
    BCSCTL1=CALBC1_16MHZ;
    DCOCTL =CALDCO_16MHZ;
    BCSCTL1|=DIVA_1;    //ACLK =MCLK/2=8M
    BCSCTL2|=DIVS_1;    //SMCLK=MCLK/2=8M
}
```

250. 定时器 TA1 中设置中断 TAIE 后中断函数为什么进不去？

TAIV 由定时器的多个中断矢量共用，在中断函数中若不加以区分中断源则会按照默认的优先级顺序进行响应。由于 TAIE 中断优先级在后，所以会默认响应 CCIE 的中断而不响应 TAIE 的中断。

251. MSP430 支持并口下载吗？

支持；缺少下载工具一直是困扰初学者的问题，朋友们完全可以自制基于电脑并口的下载线，配合 IAR 软件，完全可以满足初学者的要求；用到的元件只有 HC244,三极管 9013，以及电阻等；

252. MSP430 最小系统需要晶振吗？

不需要；MSP430 是典型的低功耗芯片，内部自带晶振；不需要外接晶振，也可以 DIY 一个最小系统。

253. MSP430 ADC12 模块的速度 ？

ADC12 的转换速率是转换所需的 ADC12CLK 以及时钟的一项功能。ADC12CLK 的近似最小值与最大值分别为 500kHz 及 6.5MHZ。速度最快的整个转换过程可以在 17 个周期内完成（13 个周期进行转换，4 个周期进行采样及保持）。6.5MHz/17 = 382ksps。ADC12 的运行速率不能低于最小值的 ADC12CLK，但在软件的控制下，采样门可以无限制保持打开状态。如欲了解有关采样与转换时间规范的更多详情，敬请参阅数据表。

254. Spy-Bi-Wire JTAG 能调试所有的 MSP430 单片机吗？

Spy-Bi-Wire JTAG 必须有 TEST 引脚，它有 5 根线组成，TEST、RST、RXD、TXD、VCC。并不是所有的 MSP430 都有，例如常见的 MSP430F149 就没有 TEST 引脚，所以必须使用 14 针的 JTAG 调试。

255. MSP430F 系列 TimerA 和 TimerB 的时钟基可以任意配置吗？

TimerA 和 TimerB 可以选用外部的时钟信号作为定时器的基准时钟！可以从 TACLK 和 TBCLK 输入，这样就摆脱了送 8MHz 进行分频了。

256. MSP430 晶振有哪些注意事项？

注意查看手册晶振选型指南，焊接谨慎选用助焊剂，部分助焊剂会导致晶振不能起振。

257. TI 的最新低功耗 CPU msp430fr5969 的示例代码在哪里可以找到？

有关此款 MCU 的示例代码在这个链接可以下载 http://www.ti.com/lit/zip/slac645
需要注意的是，所使用的编译器版本为 CCS v5.5 or later ，或者 IAR EW430 v5.60 or later。

258. 关于 MSP430 的仿真问题？

仿真器的型号一般有 UIF（USB 接口，支持 JTAG、SBW）、PIF（并口，只支持 JTAG）、EZ430（USB 接口的，只支持 SBW 模式）专业编程器有 GANG430（串口、一拖 8 个，支持 JTAG、SBW，不支持 BSL）；多功能编程器（JTAG、SBW、BSL）。这些编程器都可以做离线烧写，即脱离计算机来对目标板烧写。也可以用仿真器配专业的软件来做编程器，这类软件有 MSPFET、FET-PRO430 等。

259. FRAM 与 E2PROM 的区别及优势？

FRAM 可以作为 E2PROM 的第二种选择。它除了 E2PROM 的性能外，访问速度要快得多。但是决定使用 FRAM 之前，必须确定系统中一旦超出对 FRAM 的 100 亿次访问之后绝对不会有危险。

260. FRAM 与 Flash 的区别及优势？

现在最常用的程序存储器是 Flash，它使用十分方便而且越来越便宜，程序存储器必须是非易失性的，并且要相对低廉，还要比较容易改写。而使用 FRAM 会受访问次数的限制。

261. FRAM 与 DRAM 的区别及优势？

DRAM 适用于那些密度和价格比速度更重要的场合。例如 DRAM 是图形显示存储器的最佳选择，有大量的像素需要存储，而恢复时间并不是很重要。如果不需要下次开机时保存上次内容，使用易失性的 DRAM 存储器就可以。DRAM 的作用与成本是 FRAM 无法比拟的。事实证明，DRAM 不是 FRAM 所能取代的。

262. FRAM 与 SRAM 的区别及优势？

从速度、价格及使用方便来看，SRAM 优于 FRAM；但是从整个设计来看，FRAM 还有一定的优势。假设设计中需要大约 3KB 的 SRAM，还要几百个字节用来保存启动代码的 E2PROM 配置。非易失性的 FRAM 可以保存启动程序和配置信息。如果应用中所有存储器的最大访问速度是 70ns，那么可以使用 1 片 FRAM 完成这个系统，使系统结构更加简单。

263. MSP430 是否支持浮点运算？

一般来说,芯片并没有实现浮点运算单元.但这并不意味着无法方便地实现浮点运算，开发环境提供了软件实现方法,只是需要执行的指令较多,速度较慢。

264. MSP430G2 系列出厂频率怎么校正？

在出厂时，每一块单片机都校正了 4 个频率值（1/8/12/16MHZ），将这 4 个频率值的校验参数存在单片机片内 Flash 的 Info A 段中。每一块单片机的参数都不同。如图，最后一栏是基址和偏移地址。但是由于 Flash 储存器的数据是可以被改写的，所以在以后大家使用 Flash 的时候有可能将此块数据擦除，所以大家最好事先将此数据从编译器中储存下来，以备后用。

265. MSP430 的硬件 USART 配置问题？

MSP430 中的硬件 USART 模块是一种状态机制 (state machine)，每次定义新的 USART 配置时都必须将其状态复位。这可以通过固件，由 UCTL 寄存器中 SWRST 位的设置/复位序列来实现。默认情况下，SWRST 位是在上电复位 (POR) 后设置的。如果在 POR 之后通过配置控制寄存器第一次对 USART 模块参数进行定义，则配置 UCTL 寄存器应在序列中排在最后一位，这样就可以将 SWRST 复位，以启动带有预定设置的状态机制。这可以通过 MOV.B #000X XXX0B、汇编语言编写的 &UCTL 以及 C 语言编写的 UCTL = 0b000X XXX0 来实现。如欲了解更多详情，敬请参阅该器件的用户指南以及代码范例。如果在固件中重新配置了

USART 模块，则在重新配置后必须对 SWRST 位进行设置/复位序列操作，以重新启动带有新配置的 USART 状态机制。

266. 什么是 MSP430 的高级加密？

32 字节的密码看似几乎完全没可能使用穷举法来实现破解，但是各位别忘了，msp430 的 16 个中断向量未必每一个都用到了，没用到的中断向量为 0xffff，如果您的程序只用到了复位向量，那么破解者只需尝试最多 32768 次（中断向量为偶数，所以除以 2）就能将其破解，另外，如果芯片本身 Flash 容量较小，比如 4K 字节，那么破解者只需尝试最多 2K 次就能将其破解。这对自动操作的计算机来说几乎是一瞬间的事情。那么如果用到的中断向量越多，就越难破解，最好的办法就是将所有未用到的中断向量全部填充为随机数据，这就是“高级加密”。

267. 如何设置 MSP430 器件的低功耗模式？

通常，减少功耗的最重要的因素就是使所使用的 430 时钟系统能够在 LPM3 的模式下得到最优化的使用。在实时时钟功能和所有的终端都被激活的情况下，LPM3 模式下的典型功耗小于 2μA。32kHz 的晶振可用于 ACLK 的时钟信号源，CPU 时钟来自于 DCO，此时只需要 6us 的唤醒时间（对于 MSP430F2XXX）。

- 用终端去唤醒处理器和控制程序
- 外围设备在需要时被打开
- 用低功耗集成外设的模式去取代软件的功能。例如，在 CPU 不工作的情况下，Timer_A 和 Timer_B 能够自动的产生 PWM 波形，并捕获外部时钟
- 分支计算和快速查表应该替代查询标志位和长时间的软件计算
- 避免频繁的子程序和功能调用
- 对于较长的软件程序中，单周期的 CPU 寄存器应该被使用。

268. Msp430 系列有哪些双串口（UART）的控制器？

双串口的 MCU 仅有以下三个系列支持 MSP430F2x/4xMSP430F5x/6xMSP430FRxx FRAM 具体型号见下面的链接 http://www.ti.com/lsds/ti/microcontrollers_16-bit_32-bit/products.page#p2083=2&o7=

269. MSP30G2 系列单片机的 DCO 时钟有哪几个官方校准的频率可选？

MSP30G2 系列单片机中有经官方校准过的 1MHz、8MHz、12MHz、16MHz 可用

270. 怎样提升 MSP430 数学运算能力？

MSPMATHLIB 是一款浮点标量数学函数加速库，可在极短的时间内完成数学密集型运算，为开发人员提供高达 26 倍的性能。该高级数学函数库有助于 16 位 MCU 更快地执行代码，为任何数学密集型应用延长电池使用寿命。开发人员可轻松配置 MSPMATHLIB 软件，为所有 MSP430F5xx、MSP430F6xx 以及 MSP430FR5xx MCU 系列提高性能。IQmathLib 现已开始针对 MSP430 MCU 开发人员提供，可帮助优化 16 位及 32 位定点数学。该数学函数库不仅提供与 MSPMATHLIB 相同的优势，而且还可在只限制数据范围灵活性的同时，实现高达 100 倍的性能。IQmathLib 可在不影响功耗的情况下，提供提升性能所需的所有后端功能。IQmathLib 针对所有 MSP430 MCU 系列提供。使用上述两个库文件可以有效使用硬件设备辅助运算，可以大幅提升数学运算效率。

271. 我在哪里可以找到 MSP430 的软件例程？

CCS 软件的 View 菜单-TI Resource Explorer 下

重要声明

德州仪器(TI) 及其下属子公司有权根据 JESD46 最新标准, 对所提供的产品和服务进行更正、修改、增强、改进或其它更改， 并有权根据 JESD48 最新标准中止提供任何产品和服务。客户在下订单前应获取最新的相关信息, 并验证这些信息是否完整且是最新的。所有产品的销售都遵循在订单确认时所提供的TI 销售条款与条件。

TI 保证其所销售的组件的性能符合产品销售时 TI 半导体产品销售条件与条款的适用规范。仅在 TI 保证的范围内，且 TI 认为 有必要时才会使用测试或其它质量控制技术。除非适用法律做出了硬性规定，否则没有必要对每种组件的所有参数进行测试。

TI 对应用帮助或客户产品设计不承担任何义务。客户应对其使用 TI 组件的产品和应用自行负责。为尽量减小与客户产品和应 用相关的风险，客户应提供充分的设计与操作安全措施。

TI 不对任何 TI 专利权、版权、屏蔽作品权或其它与使用了 TI 组件或服务的组合设备、机器或流程相关的 TI 知识产权中授予 的直接或隐含权限作出任何保证或解释。TI 所发布的与第三方产品或服务有关的信息，不能构成从 TI 获得使用这些产品或服 务的许可、授权、或认可。使用此类信息可能需要获得第三方的专利权或其它知识产权方面的许可，或是 TI 的专利权或其它 知识产权方面的许可。

对于 TI 的产品手册或数据表中 TI 信息的重要部分，仅在没有对内容进行任何篡改且带有相关授权、条件、限制和声明的情况 下才允许进行复制。TI 对此类篡改过的文件不承担任何责任或义务。复制第三方的信息可能需要服从额外的限制条件。

在转售 TI 组件或服务时，如果对该组件或服务参数的陈述与 TI 标明的参数相比存在差异或虚假成分，则会失去相关 TI 组件 或服务的所有明示或暗示授权，且这是不正当的、欺诈性商业行为。TI 对任何此类虚假陈述均不承担任何责任或义务。

客户认可并同意，尽管任何应用相关信息或支持仍可能由 TI 提供，但他们将独力负责满足与其产品及在其应用中使用 TI 产品 相关的所有法律、法规和安全相关要求。客户声明并同意，他们具备制定与实施安全措施所需的全部专业技术和知识，可预见 故障的危险后果、监测故障及其后果、降低有可能造成人身伤害的故障的发生机率并采取适当的补救措施。客户将全额赔偿因 在此类安全关键应用中使用任何 TI 组件而对 TI 及其代理造成的任何损失。

在某些场合中，为了推进安全相关应用有可能对 TI 组件进行特别的促销。TI 的目标是利用此类组件帮助客户设计和创立其特 有的可满足适用的功能安全性标准和要求的终端产品解决方案。尽管如此，此类组件仍然服从这些条款。

TI 组件未获得用于 FDA Class III（或类似的生命攸关医疗设备）的授权许可，除非各方授权官员已经达成了专门管控此类使 用的特别协议。

只有那些 TI 特别注明属于军用等级或“增强型塑料”的 TI 组件才是设计或专门用于军事/航空应用或环境的。购买者认可并同 意，对并非指定面向军事或航空航天用途的 TI 组件进行军事或航空航天方面的应用，其风险由客户单独承担，并且由客户独 力负责满足与此类使用相关的所有法律和法规要求。

TI 已明确指定符合 ISO/TS16949 要求的产品，这些产品主要用于汽车。在任何情况下，因使用非指定产品而无法达到 ISO/TS16949 要求，TI不承担任何责任。

	产品		应用
数字音频	www.ti.com.cn/audio	通信与电信	www.ti.com.cn/telecom
放大器和线性器件	www.ti.com.cn/amplifiers	计算机及周边	www.ti.com.cn/computer
数据转换器	www.ti.com.cn/dataconverters	消费电子	www.ti.com/consumer-apps
DLP® 产品	www.dlp.com	能源	www.ti.com/energy
DSP - 数字信号处理器	www.ti.com.cn/dsp	工业应用	www.ti.com.cn/industrial
时钟和计时器	www.ti.com.cn/clockandtimers	医疗电子	www.ti.com.cn/medical
接口	www.ti.com.cn/interface	安防应用	www.ti.com.cn/security
逻辑	www.ti.com.cn/logic	汽车电子	www.ti.com.cn/automotive
电源管理	www.ti.com.cn/power	视频和影像	www.ti.com.cn/video
微控制器 (MCU)	www.ti.com.cn/microcontrollers		
RFID 系统	www.ti.com.cn/rfidsys		
OMAP应用处理器	www.ti.com/omap		
无线连通性	www.ti.com.cn/wirelessconnectivity	德州仪器在线技术支持社区	www.deyisupport.com

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265