《用于 MSP430™ 的 Code Composer Studio™ v5.1 用户指南》

User's Guide

Literature Number: ZHCU024T
May 2005–Revised September 2011

内容

图片列表

图表列表

请先阅读

关于本手册

这本手册说明了 德州仪器 (TI)™ Code Composer Studio™ v5.1 (CCSv5.1) 和 MSP430™ 超低功耗微控制器的使用方法。

如何使用本手册

阅读并按照 现在就开始!章节中的指令操作。 这一章提供了安装软件的指令，并对如何运行演示程序进行了说明。 在您发现开发工具是多么快速且易于使用之后， TI 建议您从头至尾阅读本手册。

这本手册只描述了软件开发环境的设置和基本操作，并没有对 MSP430 微控制器或者完整的开发软件和硬件系统进行完整说明。 要获得这些项的更多细节，请见1.3 节中列出的相关 TI 文档，*光盘和网络上的重要 MSP430 文档*。

这本手册应用于德州仪器 (TI) 的 MSP-FET430UIF，MSP-FET430PIF，和 eZ430 系列开发工具的使用。

这些工具包含封装时可以获得的最新材料。 要获得这些最新的材料（数据表、用户指南、软件、应用信息等），请访问 TI MSP430 网站www.ti.com/msp430或者与您当地的销售办事处联系。

注意事项和警告信息

本文档有可能包含注意事项和警告。

CAUTION

这是一个注意事项声明的例子。

注意事项声明描述了一种有可能损坏您的软件或者设备的情况。

WARNING

这是一个警告声明的例子。

一个警告声明描述了一种有可能对您造成伤害的情况。

注意事项或者警告中所提供的信息是为了保护您的安全。 请仔细阅读每一条注意事项和警告。

德州仪器 (TI) 提供的相关文档

CCSv5.1 文档

《MSP430™ 汇编语言工具用户指南》，文献号SLAU131

《MSP430™ 优化 C/C++ 编译器用户指南》，文献号SLAU132

MSP430 开发工具文档

《MSP430™ 硬件工具用户指南》，文献号SLAU278

《eZ430-F2013 开发工具用户指南》，文献号SLAU176

《eZ430-RF2480 用户指南》， 文献号 SWRA176

《eZ430-RF2500 开发工具用户指南》， 文献号SLAU227

《eZ430-RF2500-SEH 开发工具用户指南》，文献号SLAU273

《eZ430-Chronos™ 开发工具用户指南》， 文献号SLAU292

《MSP-EXP430G2 发射台实验板用户指南》，文献号SLAU318

MSP430xxxx 器件数据表

《MSP430x1xx 系列产品用户指南》，文献号SLAU049

《MSP430x2xx 系列产品用户指南》，文献号SLAU144

《MSP430x3xx 系列产品用户指南》，文献号 SLAU012

《MSP430x4xx 系列产品用户指南》，文献号SLAU056

《x5xx/MSP430x6xx 系列产品用户指南》，文献号SLAU208

《CC430 系列产品用户指南》，文献号 SLAU259

如果您需要协助

德州仪器 (TI) 产品信息中心 (PIC) 提供对 MSP430 微控制器和 FET 开发工具的技术支持。 PIC 的联系信息可从 TI 网站 www.ti.com/support 上获得。 提供一个专用 Code Composer Studio维基页面 (FAQ)，并且德州仪器 (TI)E2E 社区支持论坛还针对MSP430 和Code Composer Studio v5.1提供了与工程师、TI 工程师、和其他专家的开放式沟通交流的机会。 可在 MSP430 网站上找到附加的专用器件信息。

FCC 警告

本设备仅限于在实验室测试环境中使用。 它会生成、使用和发出射频能量，而且尚未依照 FCC 规则第 15 部分第 J 子部分中为提供合理的射频干扰保护而制定的计算设备限制执行符合性测试。 在其它环境中操作本设备可能会干扰无线电广播通信，在此情况下，将要求用户自行采取相应措施以消除这种干扰。

现在就开始!

这一章提供了安装软件的指令，并对如何运行演示程序进行了说明。

Topic **Page**

1.1 软件安装

为了安装 Code Composer Studio™ v5.1 (CCS)，从 DVD 中运行 setup_CCS_x.x.x.x.exe。 如果 CCS 包已经被下载，在运行 setup_CCS_x.x.x.x.exe 之前请确保从压缩文档中提取所有文件。 请按照屏幕显示的指令操作。 针对 USB JTAG 仿真器（MSP-FET430UIF 和 eZ430 系列产品）的硬件驱动器在安装 CCS 时自动安装。 针对并行端口 FET (MSP-FET430PIF) 的驱动器并不是默认安装，但是在安装过程中能够手工进行选择。

注: MSP-FET430PIF（并口仿真器）支持。

支持 MSP-FET430PIF 并行端口接口的驱动器和 IDE 组件不是默认安装。 在 CCSv5.1 安装过程中手工选择安装。

请在运行 setup_CCS_x.x.x.x.exe 之前从压缩文档 setup_CCS_x_x_x.zip 中提取全部文件。

表 **1-1.** 系统要求

	建议系统要求	最低系统要求
处理器	双核	1.5GHz
RAM	2GB	1GB
空闲硬盘空间	2GB	300MB（取决于安装期间所选择的特性）
操作系统	Microsoft® WIndows® 已安装 SP2 的 XP（32/64 位）或者 Windows Vista® 已安装 SP1（32/64 位）或者 Windows 7® （32/64 位）	Microsoft® Windows® 已安装 SP2 的 XP（32/64 位）或者 Windows Vista® （32/64 位）或者 Windows 7® (32/64 位)

1.2 LED 闪烁

在 FET 上演示的这部分内容相当于 C 语言中的“世界您好！”介绍性程序。 CCSv5.1 包括 C 和 ASM 代码文件以及完全预配置项目。 下面的部分描述了如何开发一个使 LED 闪烁的应用程序，此程序可下载至 FET 并运行。

1. 启动 Code Composer Studio Start → All Programs → Texas Instruments → Code Composer Studio → Code Composer Studio。
2. 通过选择 File → New → CCS Project 来创建一个新项目。
3. 输入一个项目名称并选择器件变量
4. 如果使用一个 USB 闪存仿真工具，例如 MSP-FET430UIF 或者 eZ430 开发工具，默认情况下，这些工具应该已被配置好。 万一您使用的是 MSP-FET430PIF LPT 接口，您需要选择 TI MSP430 LPTx（在这种情况下，在安装期间选择针对 MSP430 并行端口工具的支持）。
5. 在项目模板和示例部分选择“使 LED 闪烁”基本示例。
6. 单击完成。

注: 预先定义的示例与 MSP430 板一起工作。 特定的 MSP430x4xx 板使用端口 5.0 用于 LED 连接。 此外，MSP430L092 板需要一个不同的代码示例。 这些示例的代码是可以获得的。 详细信息请见表 1-2。

表 **1-2.** 代码示例

MSP430 器件	代码示例
MSP430x1xx 器件系列	<...>\msp430x1xx\C-Source\msp430x1xx.c
MSP430x2xx 器件系列	<...>\msp430x2xx\C-Source\msp430x2xx.c
MSP430x4xx 器件系列	<...>\msp430x4xx\C-Source\msp430x4xx.c
MSP430x5xx 器件系列	<...>\msp430x5xx\C-Source\msp430x5xx.c
MSP430x6xx 器件系列	<...>\msp430x6xx\C-Source\msp430x6xx.c

表 **1-2.** 代码示例 **(continued)**

MSP430 器件	代码示例
MSP430L092	<...>\msp430x5xx\C-Source\msp430l092.c

7. 为了编译此代码并将应用程序下载至目标方器件，点击 Run → Debug (F11)。
8. 可通过选择 Run → Resume (F8) 或者点击工具条上的播放按钮来启动应用程序。

如果 CCS 调试器不能与器件通信，请见 FAQ 调试中 #1。

恭喜，您刚刚建立并测试了一个 MSP430 应用程序！

位于 <Installation Root（安装根目录）>\ccsv5\ccs_base\msp430\examples\example projects 内的预先定义的项目可通过选择 Project → Import Existing CCS/CCE Eclipse Project 进行导入。

1.3 只读光盘和网络上重要的 MSP430™ 文档

MSP430 和 CCSv5.1 信息的主要来源是器件专用的数据表和用户指南。 在生产时可以获得的这些文档的最新版本已与这个工具一起存放在只读光盘上。 MSP430 网站www.ti.com/msp430内有这些文档的最新版本。

开发流程

在这一章中讨论了如何使用 Code Composer Studio (CCS) 来开发应用软件以及如何调试该软件。

Topic **Page**

2.1 使用 Code Composer Studio (CCS)

以下部分简要概括了如何使用 CCS。 要获得使用汇编语言或者 C 语言 CCS 进行软件开发流程的完整论述，请见《*MSP430 汇编语言工具用户指南*》（文献号：SLAU131）和《*MSP430 优化 C/C++ 编译器用户指南*》（文献号：SLAU132）。

2.1.1 从零开始创建一个项目

这一部分介绍了如何从零开始一步步地创建一个汇编语言或者 C 语言项目的指令以及在 MSP430 上下载和运行此应用（请见2.1.2 节，*项目设置*）。 此外，MSP430 Code Composer Studio 帮助对之一过程进行了更加综合全面的概括。

1. 启动 CCS (Start → All Programs → Texas Instruments → Code Composer Studio → Code Composer Studio)。
2. 创建新项目 (File → New → CCS Project)。 输入项目名称，点击下一步并为 MSP430 设定器件系列。
3. 选择适当的器件变量。 对于只使用汇编语言的项目，请选择“项目模板和示例”("Project template and examples") 中的“只使用汇编的空项目”("Empty Assembly-only Project")。
4. 如果使用一个 USB 闪存仿真工具，诸如 MSP-FET430UIF 或者 eZ430 开发工具，默认情况下，这些工具应该已经被配置好。 万一您使用的是 MSP-FET430PIF LPT 接口，您需要选择 TI MSP430 LPTx（在这种情况下，在安装期间选择针对 MSP430 并行端口工具的支持）。
5. 对于 C 语言项目，至此设置已经完成，main.c 将被显示且代码可以被输入。 万一是一个汇编语言项目，则必须创建一个全新的源文件 (File → New → Source File)。 输入文件名且不要忘记在文件名添加 .asm 后缀。 或者，如果您想为您的项目使用一个已经存在的源文件，点击 Project → Add Files... 并浏览所需的文件。 单击此文件并点击打开 (Open) 或者双击文件名来将此文件添加到项目文件夹。
6. 单击完成 (Finish)。
7. 将程序文本输入到文件。

注: 使用 .h 文件来简化代码开发。

CCS 被提供给每一个器件来定义器件寄存器和位的名称。 建议使用这些文件，因为这样能够大大简化程序开发的任务。 为了使目标方器件含有相应的 .h 文件，在汇编代码中添加 #include <msp430xyyy.h> for C and .cdecls C，LIST,"msp430xyyy"，其中 xyyy 为 MSP430 的部件号。

---

8. 建立项目 (Project → Build Project)。
9. 调试应用程序 (Run → Debug (F11))。 这样将启动调试器，从而获得对器件的控制，擦除目标方内存，使用应用程序编辑目标方内存，并将目标方复位。
   如果调试器不能与器件通信，请见常见问题和解答 (FAQ) 调试中 #1。
10. 点击运行 Run → Resume (F8) 来启动应用程序。
11. 点击 Run → Terminate 来停止应用程序并推出调试器。 CCS 将自动返回至 C/C++ 视图（代码编辑器）。
12. 点击 File → Exit 来退出 CCS。

2.1.2 项目设置

配置 CCS 所需的设置很多且十分详细。 大多数项目可以使用默认的出厂设置来进行编译和调试。 对于已激活的项目，通过点击 Project → Properties 可访问项目设置。 建议使用/必须使用以下项目设置：

- 为调试会话指定目标方器件 (Project → Properties → General → Device → Variant)。 相应的连接器命令文件和运行时间支持库会被自动选择。
- 为了使 C 语言项目的调试更加简便，禁用优化 (Project → Properties → Build → MSP430 Compiler → Optimization → Optimization level)。
- 为 C 语言预处理器指定查询路径 (Project → Properties → Build → MSP430 Compiler → Include Options)。
- 为任一使用的库指定查询路径 (Project → Properties → Build → MSP430 Compiler → File Search Path)。
- 指定调试器接口 (Project → Properties → General → Device → Connection)。 为并行 FET 接口选择 TI MSP430 LPTx 或者为 USB 接口选择 TI MSP430 USBx。
- 在项目代码下载之前启用主内存和信息内存擦除 (Project → Properties → Debug → MSP430 Properties → Download Options → Erase Main and Information Memory)。
- 为了保证正确的独立运行，禁用软件断点(Project → Properties → Debug → MSP430 Properties → Enable Software Breakpoints)。 如果软件断点被启用，确保目标方被连接时每个调试会话被适当终止；否则，由于器件上的应用程序仍将包含有软件断点指令，所以目标方也许不能够独立运行。

2.1.3 使用一个已有的 CCE v2，CCE v3，CCE v3.1 和CCS v4.x 项目

CCS v5.1 支持工作区和项目转换，这些项目由版本 CCE v2，v3，v3.1 和 CCSv4.x 至 CCSv5.1 格式创建 (File → Import → General → Existing Projects into Workspace → Next)。 浏览包含有将被导入项目的之前的 CCE 工作区。 导入向导列出了指定工作区内的所有项目。 然后可以选择并转换特定的项目。 CCEv2 和 CCEv3 项目在导入后也许需要对目标方配置文件 (*.ccxml) 进行手工操作。

根据之前 CGT 版本的不同，IDE 有可能返回一个警告，表示导入的项目是由另外一个代码生成工具 (CGT) 版本建立的。

然而对于汇编语言项目的支持并未改变，C 语言代码的头文件已经稍加改动以改进与 IAR 嵌入式工作平台 (Workbench) 之间的兼容性。® IDE（中断矢量定义）。 仍旧对 CCE 2.x 中使用的定义进行指定，但是已经在所有头文件中添加了对于这些定义的注释。 为了支持 CCE 2.x 代码，删除了#定义声明 (#define statements) 之前的 "//"，这一段代码位于每个 .h 文件中“中断矢量”("Interrupt Vectors") 部分的末尾。

2.1.4 堆栈管理

可以通过项目选项对话来配置保留堆栈尺寸(Project → Properties → Build → MSP430 Linker → Basic Options → Set C System Stack Size)。 堆栈的尺寸被定义为从 RAM 最后的位置扩展 50 至 80 字节（也就是说，堆栈从RAM 向下延伸 50 至 80 字节，具体扩展的字节数取决于所选器件的 RAM 大小）。

请注意堆栈会由于小尺寸或者应用程序错误而溢出。 跟踪堆栈大小的方法请见节 2.2.2.1。

2.1.5 如何生成二进制格式文件（TI-TXT 和 INTEL-HEX）。

CCS 安装包括 hex430.exe 转换工具。 通过配置，它能够生成 TI-TXT 格式的输出项目，此项目与 MSP-GANG430 和 MSP-PRGS430 程序编辑器一起使用，也能生成 INTEL-HEX 格式文件用于 TI 厂家器件编程。 此工具可在命令行中独立使用（位于 <Installation Root>\ccsv5\ccs_base\tools\compiler\msp430\bin 下）或者在 CCS 内直接使用。 在后一种情况下，通过在“应用预先定义的步骤”("Apply Predefined Step") 下拉菜单中选择诸如 TI-TXT 和 INTEL-HEX 等预先设定的格式，经配置，一个建立之后的步骤能够在每次建立之后自动生成文件 (Project → Properties → Build → Build Steps Tab → Post-Build Step → Apply Predefined Step)。 生成的文件存储在 <Workspace>\<Project>\Debug\directory 内。

2.1.6 示例程序和项目概述

针对 MSP430 的示例程序位于 <Installation Root>\ccsv5\ccs_base\msp430\examples 内。 在适当的子目录中提供汇编语言和 C 语言源。

为了使用此示例，创建一个新项目并通过点击 Project → Add Files... 来将示例源文件添加到项目中。 此外，与代码示例相对应的示例项目位于 <Installation Root>\ccsv5\ccs_base\msp430\examples\example projects 内。 通过点击 Project → Import Existing CCS/CCE Eclipse Project 来导入项目（更多信息请见1.2 节）。

2.2 使用集成调试器

对于 CCS 内特定 FET 的说明请见Appendix D。

2.2.1 断点类型

调试器断点机制使用有限数量的片载调试资源（特别是，N 断点寄存器，请见表 2-1）。 当 N 个或者更少断点被设定时，应用程序必须以器件的全速运行（或者“实时”）。 当设定的断点大于 N 且使用软件断点被启用时 (Project → Properties → Debug → MSP430 Properties → Enable Software Breakpoints)，可以设定的软件断点没有数量限制同时仍然满足实时条件。

注: 通过一个中断代码执行的调用，一个软件断点取代了断点地址上的指令。 因此，当设置一个软件断点时会有一个小延迟。 此外，软件断点的使用一个要求正确终止每个调试会话；否则，由于器件上的应用程序仍然包含有软件断点指令，所以应用程序也许不能够独立运行。

既支持地址（代码）断点也支持数据（值）断点。 数据断点和范围断点均要求两个 MSP430 硬件断点。

表 **2-1.** 器件架构、断点和其它仿真特性

器件	**MSP430** 架构	**4** 线制 **JTAG**	**2** 线制 **JTAG** (1)	断点 **(N)**	范围断点	时钟控制	状态程序设置	跟踪缓冲器
CC430F51xx	MSP430Xv2	X	X	3	X	X		
CC430F61xx	MSP430Xv2	X	X	3	X	X		
MSP430AFE2xx	MSP430	X	X	2		X		
MSP430BT5190	MSP430Xv2	X	X	8	X	X	X	X
MSP430F11x1	MSP430	X		2				
MSP430F11x2	MSP430	X		2				
MSP430F12x	MSP430	X		2				
MSP430F12x2	MSP430	X		2				
MSP430F13x	MSP430	X		3	X			
MSP430F14x	MSP430	X		3	X			
MSP430F15x	MSP430	X		8	X	X	X	X
MSP430F16x	MSP430	X		8	X	X	X	X
MSP430F161x	MSP430	X		8	X	X	X	X
MSP430F20xx	MSP430	X	X	2		X		
MSP430F21x1	MSP430	X		2		X		
MSP430F21x2	MSP430	X	X	2		X		
MSP430F22x2	MSP430	X	X	2		X		
MSP430F22x4	MSP430	X	X	2		X		
MSP430F23x	MSP430	X		3	X	X		
MSP430F23x0	MSP430	X		2		X		
MSP430F24x	MSP430	X		3	X	X		
MSP430F241x	MSP430X	X		8	X	X	X	X
MSP430F2410	MSP430	X		3	X	X		
MSP430F261x	MSP430X	X		8	X	X	X	X
MSP430G2xxx	MSP430	X	X	2		X		
MSP430F41x	MSP430	X		2		X		
MSP430F41x2	MSP430	X	X	2		X		
MSP430F42x	MSP430	X		2		X		
MSP430FE42x	MSP430	X		2		X		
MSP430FE42x2	MSP430	X		2		X		
MSP430FW42x	MSP430	X		2		X		
MSP430F42x0	MSP430	X		2		X		
MSP430FG42x0	MSP430	X		2		X		
MSP430F43x	MSP430	X		8	X	X	X	X
MSP430FG43x	MSP430	X		2		X		
MSP430F43x1	MSP430	X		2		X		
MSP430F44x	MSP430	X		8	X	X	X	X
MSP430F44x1	MSP430	X		8	X	X	X	X
MSP430F461x	MSP430X	X		8	X	X	X	X
MSP430FG461x	MSP430X	X		8	X	X	X	X
MSP430F461x1	MSP430X	X		8	X	X	X	X
MSP430F47x	MSP430	X		2		X		
MSP430FG47x	MSP430	X		2		X		
MSP430F47x3	MSP430	X		2		X		
MSP430F47x4	MSP430	X		2		X		

(1) 此 2 线制 JTAG 调试接口也被称为 Spy-Bi-Wire (SBW) 接口。 请注意只有 USB 仿真器（eZ430-xxxx 和 MSP-FET430UIF USB JTAG 仿真器）和 MSP-GANG430 s生产编程工具支持这个接口。 MSP-FET430PIF 并行端口 JTAG 仿真器并不支持 2 线制 JTAG 模式中的通信。

表 2-1. 器件架构、断点和其它仿真特性 (continued)

器件	MSP430 架构	4 线制 JTAG	2 线制 JTAG (1)	断点 (N)	范围断点	时钟控制	状态程序设置	跟踪缓冲器
MSP430F471xx	MSP430X	X		8	X	X	X	X
MSP430F51x1	MSP430Xv2	X	X	3	X	X		
MSP430F51x2	MSP430Xv2	X	X	3	X	X		
MSP430F52xx	MSP430Xv2	X	X	8	X	X	X	X
MSP430F530x	MSP430Xv2	X	X	3	X	X		
MSP430F5310	MSP430Xv2	X	X	3	X	X		
MSP430F532x	MSP430Xv2	X	X	8	X	X	X	X
MSP430F533x	MSP430Xv2	X	X	8	X	X	X	X
MSP430F534x	MSP430Xv2	X	X	8	X	X	X	X
MSP430F54xx	MSP430Xv2	X	X	8	X	X	X	X
MSP430F54xxA	MSP430Xv2	X	X	8	X	X	X	X
MSP430F550x	MSP430Xv2	X	X	3	X	X		
MSP430F5510	MSP430Xv2	X	X	3	X	X		
MSP430F552x	MSP430Xv2	X	X	8	X	X	X	X
MSP430F563x	MSP430Xv2	X	X	8	X	X	X	X
MSP430FR57xx	MSP430Xv2	X	X	3	X	X		
MSP430F643x	MSP430Xv2	X	X	8	X	X	X	X
MSP430F663x	MSP430Xv2	X	X	8	X	X	X	X
MSP430F67xx	MSP430Xv2	X	X	3	X	X		
MSP430L092	MSP430Xv2	X		2		X		

2.2.2 使用断点

如果在设定的断点数量大于 N 时调试器启动并且软件断点被禁用，会显示一个消息通知用户并不是所有的断点都可用。 请注意 CCS 允许设定任一数量的断点，而不用考虑 CCS 的使用软件断点设置。 如果软件断点被禁用，调试器可设定的断点数量 N 为最大值。

程序复位需要一个断点，此断点设定在 Project → Properties → Debug → Generic Debugger Options → Auto Run Options → Run to symbol 中定义的地址上。

运行至光标 (Run to Cursor) 操作临时需要一个断点。

控制台 I/O (CIO) 功能，正如 printf，需要使用一个断点。 如果这些功能已经被编译在内，但是您并不希望使用一个断点的话，通过改变 Project → Properties → Debug → Generic Debug Options → Enable CIO function use 中的选项来禁用 CIO 功能性。

2.2.2.1 CCSv5.1 中的断点

CCS 支持一定数量的预先定义的断点类型，这些类型可以通过打开断点窗口中断点 (Breakpoint) 图标旁的菜单进行选择 (Window → Show View → Breakpoints)。 除了传统断点，CCS 允许设置一个用于中断数据地址访问而非地址访问的观测点。 通过右键点击断点并选择属性 (Properties)，可在调试器中更改断点/观测点的属性。

- 程序地址之后的中断

  当程序尝试在一个特定地址之后执行代码时，停止代码执行。
- 程序地址之前中断

  当程序尝试在一个特定地址之前执行代码时，停止代码执行。
- 程序运行中中断

  当程序尝试在一个特范围内执行代码时，停止代码执行。
- **DMA** 转移中断
- **DMA** 转移范围内中断

  当一个特定地址范围内的 DMA 访问发生时，产生中断。
- 堆栈溢出中断

  有可能对引起堆栈溢出的应用程序进行调试。 在堆栈溢出时设置中断（在调试窗口中右键点击，然后在背景菜单中 (context menu) 选择“堆栈溢出中断”("Break on Stack Overflow")）。 程序执行在引起堆栈溢出的指令上停止。 可在 Project → Properties → C/C++ Build → MSP430 Linker → Basic Options 中调整堆栈的大小。
- 断点

  设定一个断点。
- 硬件断点

  如果软件断点未被禁用时，施加一个硬件断点。
- 监视数据地址范围

  当对一个特定范围内的地址进行数据访问发生时，停止代码执行。
- 监视

  如果执行一个到特定地址的特定数据访问，停止代码执行。
- 带有数据的观测点

  如果执行一个带有特定值的到特定地址的特定数据访问，停止代码执行。

  限制 1：观测点适用于全局变量并且没有寄存器本地变量。 在后一种情况下，在函数中设置一个断点 (BP) 来停止执行，在这里需要观测变量（在那里设置代码 BP）。 然后在函数中设置监视点并删除（或禁用）代码断点并运行/重新启动应用程序。

  限制 2：监视点适用于 8 位和 16 位宽的变量。

注: 并不是每个 MSP430 衍生器件都支持所有选项（请见表 2-1）。 因此，根据所选器件的不同，在断点菜单中预先定义的断点类型的数量会发生变化。

要获得与使用 CCS 进行高级调试有关的更多信息，请见应用报告*《使用 CCS 版本 4 增强型仿真模块 (EEM) 进行高级调试》*（文献号：SLAA393）。

常见问题和解答

这个附录介绍了硬件、程序开发和调试工具方面常见问题的解决方案。

Topic **Page**

A.1 硬件

要获得硬件相关 FAQ 的完整列表，请见《*MSP430 硬件工具用户指南*》SLAU278。

A.2 程序开发（汇编语言、C 语言编译器、连接器、IDE）

注: **考虑 CCS 版本注释**

对于意外操作的情况，请见 CCS 版本注释以了解已知缺陷和当前 CCS 版本的限制。 可通过菜单项 Start → All Programs → Texas Instruments → Code Composer Studio → Release Notes 来获得这些信息。

1. 一个常见的 **MSP430“错误”就是禁用安全装置失败**；安全装置默认被启用，如果没有被禁用或者应用程序管理不当的话，安全装置会将器件复位。 使用WDTCL=WDTPW+WDTHOLDM来明确禁用安全装置。 这个声明最好放置于在 main() 之前执行的 _system_pre_init() 函数内。 如果安全装置定时器未被禁用，并且在CSTARTUP期间安全装置启动并将器件复位，由于调试器不能定位用于 CSTARTUP 的源代码，**源屏幕变成空白**。 请注意，如果使用大量的初始化全局变量，CSTARTUP 的执行会花费大量的时间。

```
            int _system_pre_init(void)
            {
            /* Insert your low-level initializations here */
            WDTCTL = WDTPW + WDTHOLD; // Stop Watchdog timer
            /*===================================*/
            /* Choose if segment initialization */
            /* should be done or not. */
/* Return:   0 to omit initialization */
            /*      1 to run initialization */
            /*===================================*/
            return (1);
            }
```

2. 在 **C 语言库内部，在 GIE**（全局中断启用）之前（和恢复之后）**硬件乘法器被使用。**
3. 可以在 **CCS** 之内将汇编程序和 **C** 程序混合在一起。请见“”一章
4. **.h** 文件内使用的恒定定义 **(#define)** 被有效地保留并且包括，例如，C，Z，N，和 V。不要使用这些名称创建程序变量。
5. **编译器优化能够移除未使用的变量和/或者未生效的声明**并能够对调试产生影响。 为了防止这种情况的发生，这些变量可被定义为MMM；例如，
   MMMMMMM iM。

A.3 调试中

调试器是 CSS 的一部分并且可被用于一个单机应用。 当使用单机调试器和使用来自 CCS IDE 的调试器时，这部分适用。

> **注:** **注意 CCS 版本注释**
>
> 对于意外操作的情况，请见 CCS 版本注释文档以了解已知缺陷和当前 CCS 版本的限制。 为了访问这些信息，点击 Start → All Programs → Texas Instruments → Code Composer Studio → Release Notes。

1. **当不能与此器件进行通信时，调试器产生报告。**解决这个问题的可能的解决方案包括：
   - 确保在 Project → Properties → General → Device → Connection 中选择正确的调试接口和相应的端口号。
   - 确保在目标方硬件上正确配置跳线设置。
   - 确保没有其它软件应用（例如，打印机驱动器）已经保留或者控制 COM/并行端口，这将防止调试服务器与此器件通信。
   - 打开器件管理器并确定是否用于 FET 工具的器件已经被正确安装以及 COM/并行端口是否被 Windows 操作系统 (OS) 成功识别。 检查 PC BIOS 中的并行端口设置（请见 FAQ 调试中 #5）。 针对 IBM 或者联想 (Lenovo) ThinkPad® 计算机的用户，尝试端口设置 LPT2 和 LPT3，即便操作系统报告称并行端口位于 LPT1。
   - 重新启动计算机。

   确保 MSP430 器件牢靠地安装在插槽上（这样插槽的“金手指”可以与器件的引脚完全接合），并且它的引脚 1（在顶部表面以一个环形凹口标出）与印刷电路板 (PCB) 上的标记“1”对齐。

   CAUTION

   对器件可能造成的损害

   自始至终只使用一个真空拿取工具处理 MSP430；不要使用您的手指，这是因为您能很容易地把引脚弄弯，致使器件报废。 此外，时刻遵守并按照正确的静电放电 (ESD) 预防措施操作。

2. **调试器能够调试利用中断和低功耗模式的应用。**请见 FAQ 调试中 #7。
3. **器件运行时，调试器不能访问器件寄存器和内存。**要访问器件寄存器和内存，用户必须将器件停止。
4. **调试器报告，此器件的 JTAG 安全保险丝被熔断。** 使用现有的 MSP-FET430PIF 和 MSP430-FET430UIF JTAG 接口工具，当修改由外部供电的目标板时，会有一个缺点。 这会导致一个 MSP430 中的意外保险丝检查并会在 JTAG 安全保险丝没有熔断的情况下被认为已经熔断。 这一情况在 MSP-FET430PIF 和 MSP-FET430UIF 上都会发生，但是主要见于 MSP-FET430UIF 上。

   权变措施：

   - 将器件$\overline{\text{RST}}$/NMI 引脚到 JTAG 头（引脚 11），MSP-FET430PIF/MSP-FET430UIF 接口工具能够拉$\overline{\text{RST}}$线路，这也将器件内部保险丝逻辑复位。
   - 不要同时连接 V_{CC}工具（引脚 2）和 JTAG 头的 V_{CC}目标方（引脚 4）。 为调试器中的 V_{CC}指定一个值，这个值与外部电源电压相等。
5. **并行端口指示器 (LPTx) 具有下列物理地址：LPT1=378h，LPT2= 278h，LPT3=3BCh。**并行端口（ECP、兼容、双向、正常）的配置不十分重要；ECP 看起来运行正常。 解决调试器和器件间通信问题的附加窍门请见FAQ 调试中 #1。

6. 当调试器被启动并且当器件被编辑时，**调试器将RST/NMI 变为高电平有效来使器件复位。** 当器件被手工重新编辑（使用重新载入 (Reload)），当 JTAG 被重新同步时（使用再同步 (Resynchronize) JTAG），此器件也可由调试器的复位 (Reset) 按钮复位。 当RST/NMI未被置成有效（低电平），调试器设定驱动RST/NMI 逻辑电路为高阻抗，并且通过一个 PCB 上的电阻器将RST/NMI 上拉至高电平。

   当调试器被启动时，RST/NMI 被置成有效并在加电之后被置成无效。 然后RST/NMI 被置成有效并在器件初始化完成之后第二次被置成无效。

7. **调试器能够调试一个器件，此器件的程序能够重新配置RST/NMI 引脚的功能至 NMI。**
8. **当调试器复位器件时，XOUT/TCLK 引脚的电平未定义。**驱动 XOUT/TCLK 的逻辑电路在所有其它时间被设定为高阻抗。
9. **当测量器件的电流时，确保 JTAG 控制信号被释放，**否则由 JTAG 引脚上的信号供电且测量结果不正确。 参见 FAQ 调试中 #10。
10. **当调试器取得器件的控制权时，CPU 接通**（也就是说，它不处于低功耗模式），这与状态寄存器中的低功耗模式位的设置无关。 任一低功耗模式条件被存储于 STEP 或者 GO 之前。 因此，在调试器控制此器件时不要测量器件的功耗。 而是运行使用运行时释放 JTAG 的应用程序。
11. 在有内存的地方，MEMORY 窗口正确显示了存储器的内容。 然而，在没有内存的地方，**MEMORY 窗口不正确显示存储器的内容。**存储器应该只用在器件数据表指定的地址范围内。
12. 调试期间，调试器利用系统时钟来控制此器件。 因此，当调试器控制此器件时，**由主系统时钟 (MCLK) 计时的器件计数器和其它组件受到影响。**采取特别的预防措施来大大降低对于安全装置定时器的影响。 CPU 内核寄存器被保持。 所有其它时钟源（SMCLK 和 ACLK）和外设在仿真期间继续正常运行。 换句话说，**闪存仿真工具是部分侵入式工具。**

    调试期间，通过停止时钟，支持时钟控制的器件能够进一步大大减少这些影响 (Project → Properties → CCS Debug Settings → Target → Clock Control)。
13. 当编辑闪存式，**不要在写入闪存操作之后立即在指令上设置断点。**对于这个限制的简单权变措施是在闪存操作之后跟随一个 NOP 并在 NOP 之后在指令上设置一个断点。
14. 清除和编辑闪存存储器需要多个内部机器周期。 **当控制闪存的指令为单步执行时，**在这些操作完成前对器件的控制权被返还给调试器。 因此，**调试器用错误信息升级它的内存窗口。**针对这个运转状态的权变措施是在闪存访问指令后跟随一个 NOP，然后在检查闪存访问指令的效果之前跨过此 NOP。
15. **正常程序执行期间，读取时被清除的位**（也就是说，中断标志）**被清除**

    使用特定带有增强型仿真逻辑的 MSP430 器件，例如 MSP430F43x/44x 器件，数据位并不以这种方式工作（即，数据位并不是由调试器读取操作清除）。
16. **调试器不能用于调试在 F12x 和 F41x 器件 RAM 中执行的程序。**针对此限制的权变措施就是在闪存中调试程序。
17. **虽然单步执行具有激活的和已被启用的中断，但是它只有当中断服务例程 (ISR) 有效时才会出现**（也就是说，非 ISR 代码不会执行，并且单步运行在 ISR 第一行上停止）。 然而，由于器件在处理非 ISR（也即是说，主线路）代码之前处理一个有效且被启用的中断，所以这个状态是正确的。 对于这个运转状态的权变措施是，虽然在 ISR 内部，禁用堆栈上的 GIE 位，这样能够在退出 ISR 之后将中断禁用。 这样使得非 ISR 代码能够被调试（但是没有中断）。 通过设置寄存器窗口中状态寄存器中的 GIE 位，中断可以在晚些时候重新被启用。

    在带有时钟控制的器件上，有可能在单步执行之间将时钟挂起，并推迟中断请求 (Project → Properties → CCS Debug Settings → Target → Clock Control)。

18. 在装有数据传送控制器 (DTC) 的器件上，**数据传送周期的完成会取代一个低功耗模式指令的单步执行。**只有当一个中断被处理之后，此器件才能超过低功耗模式指令继续运行。 直到一个中断被处理，在之前单步执行似乎没有效果。 对于这个情况的权变措施是在低功耗模式指令之后的指令上设置一个断点，然后执行（运行）至这个断点。
19. **当 DTC 对一个单步执行或者一个断点进行响应而停止时，由数据传送控制器 (DTC) 进行的数据传送也许不能恰好停止。**当 DTC 被启用且一个单步被执行时，一个或者更多字节的数据可以被传送。 当 DTC 被启用且被配置用于两数据块传送模式时，当对一个单步执行或者一个断点进行响应而停止时，DTC 也许不能够恰好在数据块边界上停止。
20. **断点。**CCS 支持一定数量的预先定义的断点和监视点类型。 详细的概述请见2.2.2 节。

IAR 2.x/3.x/4.x 到 CCS C 语言迁移 (C-Migration)

TI CCS C 编译器的源代码和 IAR 嵌入式工作台编译器的源代码并不完全兼容。 虽然标准 ANSI/ISO C 代码在这些工具间可移植，但是特定实现扩展不同并需要被移植。 这个附录记录了两个编译器间的主要差异。

Topic **Page**

B.1 中断矢量定义

现在 CCS 完全支持 IAR ISR 声明（使用 #pragma vector=）。 然而，对于所有其它 IAR 预编译指令并非如此。

B.2 内在功能

针对 MSP430 特定处理器内在指令，CCS 和 IAR 工具使用一样的指令。

B.3 数据和功能安置

B.3.1 数据安置在一个绝对的位置上

CCS 编译器不支持在 IAR 编译器中执行的机制（使用 @ 运算符或者预编译位置指令 (#pragma location directive)）：

```
/* IAR C Code */ __no_init char alpha @ 0x0200; /* Place ‘alpha' at address 0x200 */ #pragma
location = 0x0202 const int beta;
```

如果需要绝对数据位置，这一步可通过进入连接器命令文件来实现，然后声明变量作为 C 代码中的伪指令：

```
/* CCS Linker Command File Entry */ alpha = 0x200; beta = 0x202; /* CCS C Code */ extern char
alpha; extern int beta;
```

绝对 RAM 位置必须从 RAM 段中排除；否则，由于连接器自动分配地址，它们的内容有可能被写入覆盖。 RAM 数据块的开始地址和长度必须在连接器命令文件中修改。 对于之前的例子，RAM 开始地址必须被从 0x0200 至 0x0204 移动 4 个字节，这样将长度从 0x0080 减少至 0x007C（对于具有 128 字节 RAM 的 MSP430 器件）：

```
/* CCS Linker Command File Entry */
/*************************************************************************/ /* SPECIFY THE
SYSTEM MEMORY MAP */
/*************************************************************************/ MEMORY /* assuming
a device with 128 bytes of RAM */ { ...  RAMMM = 0x0204MMM = 0x007C/* MMM  M = 0x200MM
M = 0x0080 */ ... }
```

连接器命令文件 (lnk_msp430xxxx.cmd) 中的外设寄存器映射的定义以及与 CCS 一起提供的特定器件头文件 (msp430xxxx.h) 是在绝对位置放置数据的一个示例。

注: 当一个项目被创建时，CCS 从包括目录(<Installation Root>\ccsv5\ccs_base\tools\compiler\MSP430\include) 中拷贝与所选 MSP430 衍生器件相对应的连接器命令文件到项目目录。 因此，确保在项目目录中完成所有连接器命令文件改变。 这允许使用同一器件的不同项目使用特定项目连接器命令文件。

B.3.2 数据放置进入已命名段。

在 IAR 中，有可能使用 @ 运算符或者一个 #pragma directive 来将变量放置到已命名的段中：

```
/* IAR C Code */ __no_init int alpha @ "MYSEGMENT"; /* Place ‘alpha' into ‘MYSEGMENT' */ #pragma
location="MYSEGMENT" /* Place ‘beta' into ‘MYSEGMENT' */ const int beta;
```

使用 CCS 编译器，#pragma DATA_SECTION() directive 必须被使用：

```
/* CCS C Code */ #pragma LOCATION(alpha, "MYSEGMENT") int alpha; #pragma LOCATION(beta,
"MYSEGMENT") int beta;
```

有关如何在 IAR 和 CCS 之间翻译内存段的名称的信息请见B.5.3 节。

B.3.3 函数放置到已命名的段中

使用 IAR 编译器，使用 @ 运算符或者 预编译位置指令 (#pragma location directive) 可将函数放置到一个已命名的段中：

```
/* IAR C Code */ void g(void) @ "MYSEGMENT" { } #pragma location="MYSEGMENT" void h(void) { }
```

使用 CCS 编译器，必须使用以下带有 #pragma CODE_SECTION() directive 的机制：

```
/* CCS C Code */ #pragma CODE_SECTION(g, "MYSEGMENT") void g(void) { }
```

有关如何在 IAR 和 CCS 之间翻译内存段的名称的信息请见B.5.3 节。

B.4 C 调用惯例

CCS 和 IAR C 语言编译器使用不同的调用惯例将参数传递给函数。 当把一个混合 C 语言和编译项目移植到 TI CCS 代码生成工具时，为了反映这些变化，必须修改编译函数。 要获得与调用惯例相关的详细信息，请见《*TI MSP430 优化 C/C++ 编译器用户指南*》（文献号：SLAU132）和《*IAR MSP430 C/C++编译器参考指南*》。

下面的示例是一个函数，此函数将 32 位字'数据'写入到一个大端字节序中的指定内存位置。 可见使用不同的 CPU 寄存器传递参数'数据'。

IAR 版本：

```
;------------------------------------------------------------------------------
 ; void WriteDWBE(unsigned char *Add, unsigned long Data) ; ; Writes a DWORD to the given memory
location in big-endian format. MMMMMMMMMM
MM; ; IN: R12 Address (Add) ; R14 Lower Word (Data) ; R15 Upper Word (Data) ;--------------------
-------------------------------------------------------
 WriteDWBE swpb R14 ; Swap bytes in lower word swpb R15 ; Swap bytes in upper word mov.w
R15,0(R12) ; Write 1st word to memory mov.w R14,2(R12) ; Write 2nd word to memory ret
```

CCS 版本：

```
;------------------------------------------------------------------------------
 ; void WriteDWBE(unsigned char *Add, unsigned long Data) ; ; Writes a DWORD to the given memory
location in big-endian format. MMMMMMMMMM
MM; ; IN: R12 Address (Add) ; R13 Lower Word (Data) ; R14 Upper Word (Data) ;--------------------
-------------------------------------------------------
 WriteDWBE swpb R13 ; Swap bytes in lower word swpb R14 ; Swap bytes in upper word mov.w
R14,0(R12) ; Write 1st word to memory mov.w R13,2(R12) ; Write 2nd word to memory ret
```

B.5 其它差异

B.5.1 初始化静态和全局变量

ANSI/ISO C 标准指定没有明确初始化的静态和全局（伪指令）必须被重新初始化为 0（在程序开始运行之前）。 通常当程序被载入且在 IAR 编译器中被执行时，这个任务被执行：

```
/* IARMglobal variable, initialized to 0 upon program start */ int Counter;
```

然而，TI CCS 编译器不会预初始化这些变量；因此，由应用程序满足这一要求：

```
/* CCS, global variable, manually zero-initialized */ int Counter = 0;
```

B.5.2 定制启动例程

借助于 IAR 编译器，可以定制 C 启动函数，这使得应用程序有机会执行诸如配置外设的初期初始化、或者省略数据段初始化。 一个 customized __low_level_init() 函数实现此操作：

```
/* IAR C Code */ int __low_level_init(void) { = /* Insert your low-
level initializations here */ /*================================== */ /* Choose if segment
initialization */ /* should be done or not. */ /* Return: 0 to omit initialization */ /* 1 to run
initialization */ /*================================== */ return (1); }
```

返回的值将控制数据段是否由 C 启动代码初始化： 借助于 CCS 编译器，定制启动例程名为 _system_pre_init()。 它的使用方法与在 IAR 编译器中的使用方法一样。

```
/* CCS C Code */ int _system_pre_init(void) { /* Insert your low-
level initializations here */ /*================================== */ /* Choose if segment
initialization */ /* should be done or not. */ /* Return: 0 to omit initialization */ /* 1 to run
initialization */ /*================================== */ return (1); }
```

请注意两个编译器省略段初始化的操作将省略明确和不明确的初始化。 运行时，用户必须确保重要的变量在它们被使用前被初始化。

B.5.3 预先定义的内存段名称

针对数据和函数放置的内存段名称由 CCS 和 IAR 工具中特定器件连接器命令文件控制。 然而，使用不同的段名称。 更多详细信息请见连接器命令文件。 下面的表格显示了如何转换最常用的段名称。

描述	CCS 段名称	IAR 段名称
RAM	.bss	DATA16_N DATA16_I DATA16_Z
堆栈 (RAM)	.stack	CSTACK
主内存（闪存或者 ROM）	.text	CODE
信息内存（闪存或者 ROM）	.infoA .infoB	INFOA INFOB INFO
中断矢量（闪存或者 ROM）	.int00 .int01 … .int14	INTVEC
复位矢量（闪存或者 ROM）	.reset	复位

B.5.4 预先定义的宏名称

IAR 和 CCS 编译器均支持一些非 ANSI/ISO 标准预定义宏名称，这将帮助创建可被编译且可被用于不同编译器平台的代码。 使用 #ifdef directive 来检查一个宏名称是否被定义。

描述	CCS 宏名称	IAR 宏名称
MSP430 是目标方吗？MSP430 是所用的特定编译器平台吗？	__MSP430__	__ICC430__
是所用的特定编译器平台吗？	__TI_COMPILER_VERSION__	__IAR_SYSTEMS_ICC__
一个 C 头文件包含在编译源代码内吗？	__ASM_HEADER__	__ IAR_SYSTEMS_ASM__

IAR 2.x/3.x/4.x 到 CCS 汇编程序迁移

TI CCS 汇编程序的源代码和 IAR 汇编程序的源代码并不是 100% 兼容。 指令记忆法是一样的，然而汇编程序的指令有些不同。 这个附录对 CCS 汇编程序指令和 IAR 2.x/3.x 汇编程序指令间的差异进行了注释。

Topic **Page**

C.1 与汇编源代码共享 C/C++ 头文件

IAR A430 汇编程序直接支持特定的 C/C++ 预处理器指令，因此，可将诸如 MSP430 特定器件头文件 (msp430xxxx.h) 的 C/C++ 头文件直接放入汇编代码：

```
#include "msp430x14x.h" // Include device header file
```

使用 CCS Asm430 编译程序时，必须使用采用 .cdecls 指令的不同的机制。 这个指令允许使用混合汇编和 C/C++ 环境的程序设计者共享 C/C++ 头文件，此头文件包含有 C/C++ 和汇编代码间的声明和原型：

```
.cdecls CMLISTM"msp430x14x.h"MMMMMMMMM
```

有关 .cdecls 指令的更多信息请见《*MSP430 汇编语言工具用户指南*》（文献号SLAU131）。

C.2 段控制

CCS Asm430 汇编程序不支持任一 IAR A430 段控制指令，诸如 ORG，ASEG，RSEG，和 COMMON。

描述	Asm430 指令 (CCS)
在 .bss 未初始化区域中保留空间	.bss
在一个已命名的未初始化区域中保留空间	.usect
将程序分配至默认程序区域（以初始化）	.text
将数据分配至一个已命名的初始化区域	.sect

为了使用 CCS 编译程序将代码和数据部分分配至指定的地址，有必要创建/使用在连接器命令文件中定义的内存区域。 下面的示例演示了分配在 IAR 和 CCS 汇编中的中断矢量来突出它们之间的差异。

```
;------------------------------------------------------------------------
 ; Interrupt Vectors Used MSP430x11x1/12x(2) - IAR Assembler ;-------------------------------------
-------------------------------------
 ORG 0FFFEh ; MSP430 RESET Vector DW RESET ; ORG 0FFF2h ; Timer_A0 Vector DW TA0_ISR ; ;----------
-----------------------------------------------------------------
 ; Interrupt Vectors Used MSP430x11x1/12x(2) - CCS Assembler ;-------------------------------------
-------------------------------------
 .sect ".reset" ; MSP430 RESET Vector .short RESET ; .sect ".int09" ; Timer_A0 Vector .short
TA0_ISR ;
```

两个例子都假定使用了标准器件支持文件（头文件、连接器命令文件）。 请注意 IAR 和 CCS 间的连接器命令文件是不同的并且不能被再次使用。 与如何在 IAR 和 CCS 之间转化内存段名称相关的信息请见B.5.3 节。

C.3 将 A430 汇编程序指令转化为 Asm430 指令

C.3.1 简介

总的来说，下面的部分描述了如何将用于 IAR A430 汇编程序 (A430) 的汇编程序指令转化为德州仪器 (TI) 的 CCS Asm430 汇编程序 (Asm430) 指令。 这些部分只用作针对转化的指南。 要获得每条指令的详细说明，请见由德州仪器 (TI) 提供的《*MSP430 汇编语言工具用户指南*》(SLAU131)，或者由 IAR 提供的《*MSP430 IAR 汇编程序参考指南*》。

注: 只有汇编程序命令要求转换

只有汇编程序命令要求转换，汇编程序指令无需转换。 两个汇编程序均使用一样的指令记忆法、操作数、运算符、和特别符号，诸如区域程序计数器 ($) 和命令定界符 (；)。

默认情况下，A430 汇编程序大小写不敏感。 这些部分显示了采用大写的 A430 指令，以便将它们与 Asm430 指令（小写显示）区分开来。

C.3.2 字符串

除了使用不同的命令，每个汇编程序针对不同的字符串使用不同的句法。 A430 针对字符串使用 C 语言句法：使用一个反斜杠字符（作为退出字符）和引号 (\") 一起代表一个引用，并且反斜杠本身由两个连续的反斜杠表示。 在 Asm430 句法中，两个连续的引号 ("") 代表一个引用；请见示例：

字符串	**Asm430 句法 (CCS)**	**A430 句法 (IAR)**
计划 "C"	"计划 ""C"""	"计划 \"C\""
\dos\command.com	"\dos\command.com"	"\\dos\\command.com"
连接字符串（例如，错误 41）	-	“错误” "41"

C.3.3 区域控制命令

Asm430 有三个预定义区域，在这些区域中一个程序的不同部分被汇编在一起。 未初始化数据被汇编至 .bss 区域，初始化数据被汇编至 .data 区域，而可执行代码进入 .text 区域。

A430 也使用区域或者段，但是没有预定义段名称。 通常，遵守 C 编译器所使用的命名规范会更加方便；DATA16_Z 代表未初始化数据，CONST 为常量（已初始化）数据，而 CODE 表示可执行代码。 下面的表格使用这些名称：

一对段可被用于将经过初始化的、可更改的数据变成可被编程。 ROM 段 将包含初始化软件 在这种情况下，所有这些段必须大小一致且布局一样。

描述	Asm430 命令 (CCS)	A430 命令 (IAR)
在 .bss（为经初始化的数据）区域中保留尺寸字节	.bss(1)	(2)
汇编入 .data（已经初始化数据）区域	.data	RSEG 常量
汇编入一个已命名的（已经初始化）区域	.sect	RSEG
汇编入 .text（可执行代码）区域	.text	RSEG 代码
在一个已命名的（未经初始化的）区域中保留空间	.usect(1)	(2)
字节边界对准	.align 1	(3)
字边界对准	.align 2	EVEN

(1) .bss 和 .usect 不要求在初始区域和未经初始化区域间来回切换。 例如：

```
MIAR MMMMMM
RSEG DATA16_NMMMM DATA M
EVENMMMMMMM
ADCResultMDS 2MM RAM MMM 1 MM
MMMDS 1MM RAM MMM 1 MMM
RSEG CODEMMMM CODE M
; CCS MMMMMM #1
ADCResult .usect ".bss",2,2MM RAM MMM 1 MM
MM .usect ".bss",1MM RAM MMM 1 MMM
MCCS MMMMMM #2
.bss ADCResult,2,2MM RAM MMM 1 MM
.bss MMM1MM RAM MMM 1 MMM
```

(2) 首先通过切换至那个段，在一个未经初始化段中保留空间，然后定义适当的内存块，然后切换回初始段。 例如：

```
RSEG DATA16_Z
MMMDS 16MMM 16 MMM
RSEG MM
```

(3) 不支持位域常量 (.field) 初始化，因此，区域计数器一直字节对准。

C.3.4 常量初始化命令

描述	Asm430 命令 (CCS)	A430 命令 (IAR)
对一个或者更多连续字节或者文本串初始化	.byte 或者 .string	DB
初始化一个 32 位 IEEE 浮点常量	.double 或者 .float	DF
初始化一个变量长度域	.field	(1)
在当前区域中保留尺寸字节	.space	DS
初始化一个或者更多文本串	初始化一个或者更多文本串	DB
初始化一个或者更多 16 位整数	.word	DW
初始化一个或者更多 32 位整数	.long	DL

(1) 不支持位域常量 (.field) 初始化。 常量必须被组合进使用 DW 的完整字。

```
MAsm430 MMMA430 MM
.field 5,3\
.field 12,4| ->DW (30<<(4+3))|(12<<3)|5MMM 3941
.field 30,8  /
```

C.3.5 列出控制命令

描述	Asm430 命令 (CCS)	A430 命令 (IAR)
允许	.fclist	LSTCND-
禁止错误条件代码块列表	.fcnolist	LSTCND+
设定源码表的页长度	.length	PAGSIZ
设定源码表的页宽度	.宽度	COL
重新启动源码表	.list	LSTOUT+
停止源码表	.nolist	LSTOUT-
允许宏列表和环路阻止	.mlist	LSTEXP+（宏） LSTREP+（环路阻止）
禁止宏列表和环路阻止	.mnolist	LSTEXP-（宏） LSTREP-（环路阻止）
选择输出列表项	.option	(1)
在源码表中弹出一页	.page	PAGE
允许展开的替代符号列表	.sslist	(2)
禁止展开的替代符号列表	.ssnolist	(2)
在列表页页首内打印一个标题	.title	(3)

(1) 没有与 .option 直接相对应的 A430 命令。 单独的列表控制命令（上述的）或者命令行选项 -c（带有子选项）应该被用于取代 .option 命令。
(2) 没有与 .sslist/.ssnolist 直接相对应的命令。
(3) 列表页页首的标题为源文件名。

C.3.6 文件参考命令

描述	Asm430 命令 (CCS)	A430 命令 (IAR)
包括来自另外文件的源语句	.copy 或者 .include	#include or $
识别在在用模块中定义和所用其它模块中使用的一个或者更多符号	.def	PUBLIC 或者 EXPORT
识别一个或者更多全局（外部）符号	.global	(1)
定义一个宏指令库	.mlib	(2)
识别在在用模块中使用而非在其它模块中定义的一个或者更多符号	.ref	EXTERN 或者 IMPORT

(1) 如果符号在在用模块中被定义，命令 .global 函数用作 .def，否则用作 .ref。 PUBLIC 或者 EXTERN 必须对 A430 汇编程序可用，以替代 .global 命令。
(2) 不支持宏指令库概念。 包括带有宏定义的文件必须被用于这个功能性。

模块可与 Asm430 汇编程序一起使用来创建单独的可连接例程。 一个文件也许包含多个模块或者例程。 除了那些 DEFINE 创建的符号，#define（IAR 预处理器命令），所有其它符号或者 MACRO 在模块末尾为“未定义”。 此外，库模块有条件连接。 这意味着只有模块中的一个公共符号外部被参考时，一个库模块才会被包括在连接的可执行代码中。 下列的命令被用于标记 A430 汇编程序内模块的开头和末尾。

附加的 A430 命令 (IAR)	A430 命令 (IAR)
停止一个程序模块	NAME 或者 PROGRAM
启动一个库模块	MODULE 或者 LIBRARY
终止在用程序或者库模块	ENDMOD

C.3.7 条件汇编命令

描述	Asm430 命令 (CCS)	A430 命令 (IAR)
可选可重复块汇编	.break	(1)
开始条件汇编	.if	IF
可选条件汇编	.else	ELSE
可选条件汇编	.elseif	ELSEIF
终止条件汇编	.endif	ENDIF
终止可重复块汇编	.endloop	ENDR
开始可重复块汇编	.loop	REPT

(1) 没有与 .break 直接相对应的命令。 然而，如果可重复块汇编被用在宏指令中时，EXITM 命令可与其它条件一起使用，显示如下：

```
SEQ MACRO FROM,TOMMMMMMMMMMMM
LOCAL X
X SET FROM
REPT TO-FROM+1   ; M FROM M TO MM
IF X>255MMM X MMMMM
EXITM
ENDIF
DB XMMMMMMM FROM...TO
X SET X+1MMMMMM
ENDR
ENDM
```

C.3.8 符号控制命令

汇编时间符号的范围在两个汇编程序中是不同的。 在 Asm430 中，对于文件，定义可为全局或者对于一个模块或者宏指令，定义可为局部。 本地符号可以为未定义 在 A430 中，对于一个宏指令 (LOCAL)，或者一个模块 (EQU) 来说，为局部符号，或者对于一个文件 (DEFINE)，为全局符号。 此外，预处理器命令 #define 还可被用于定义本地符号。

描述	Asm430 命令 (CCS)	A430 命令 (IAR)
为一个替补符号分配一个符号串	.asg	SET 或者 VAR 或者 ASSIGN
未定义本地符号	.newblock	(1)
使一个值与一个符号相等	.equ 或者 .set	EQU 或者 =
在数字替代符号上执行算法	.eval	SET 或者 VAR 或者 ASSIGN
终止结构定义	.endstruct	(2)
开始一个结构定义	.struct	(2)
为一个标签分配结构属性	.tag	(2)

(1) 没有与 .newblock 直接相对应的 A430 命令。 然而，#undef 可被用于复位一个使用 #define 命令定义的符号。 同样地，因为在一个宏指令或者模块的末尾局部符号暗示未定义，宏或者模块可被用于实现 .newblock 功能性。

(2) 不支持结构类型定义。 通过使用宏指令来分配聚合数据和基地址加上符号区距，可实现相似的功能性，显示如下：

```
MYSTRUCT: MACRO
DS 4
ENDM
LO DEFINE 0
HI DEFINE 2
RSEG DATA16_Z
X MYSTRUCT
RSEG CODE
MOV X+LO,R4
...
```

C.3.9 宏命令

描述	Asm430 命令 (CCS)	A430 命令 (IAR)
定义一个宏	.macro	MACRO
从一个宏永久退出	.mexit	EXITM
终止宏定义	.endm	ENDM

C.3.10 混合命令

描述	Asm430 命令 (CCS)	A430 命令 (IAR)
发送用户定义错误消息至输出器件	.emsg	#error
发送用户定义消息至输出器件	.mmsg	#消息(1)
发送用户定义的警告消息至输出器件	.wmsg	(2)
定义一个加载地址标签	.label	(3)
由绝对列表器产生的命令	.setsect	ASEG(4)
由绝对列表器产生的命令	.setsym	EQU 或者 =(4)
程序末尾	.end	END

(1) #消息命令的句法为：#消息“<字符串>”
汇编/编译期间，这将引起 '#message <string>' 被输出至项目建立窗口。

(2) 警告信息不能由用户定义。#消息可被使用，但是警告计数器不会产生增量。

(3) 不支持载入时间地址的概念 运行时间和载入时间地址假定为一致。 为了实现一样的效果，可通过 EQU 命令为标签指定绝对（运行时间）地址。

```
MAsm430 MMMA430 MM
.label load_start load_startM
Run_startM                  <code>
<code>load_endM
Run_endM run_startM EQU 240H
.label load_end run_endM EQU run_start+load_end-load_start
```

(4) 虽然不是由绝对列表器生成，ASEG 定义了绝对段且 EQU 可被用于定义绝对符号。

```
MYFLAG EQU   23EHM MYFLAG MM 23E
ASEG  240HMMMMMM 240
MAINM MOV #23CHM SPM MAIN MM 240
...
```

C.3.11 Asm430 命令的按字母顺序列表和交叉参考

Asm430 命令 (CCS)	A430 命令 (IAR)	Asm430 命令 (CCS)	A430 命令 (IAR)
.align	ALIGN	.loop	REPT
.asg	SET 或者 VAR 或者 ASSIGN	.macro	MACRO
.break	请见 条件汇编命令	.mexit	EXITM
.bss	请见 符号控制命令	.mlib	请见 文件参考命令
.byte 或 .string	DB	.mlist	LSTEXP+（宏）
.cdecls	对 C 预处理器声明的固有支持。		LSTREP+（环路阻止）
.copy 或 .include	#include or $	.mmsg	#message (XXXXXX)
.data	RSEG	.mnolist	LSTEXP-（宏）
.def	PUBLIC 或 EXPORT		LSTREP-（环路阻止）
.double	不支持。	.newblock	请见 符号控制命令
.else	ELSE	.nolist	LSTOUT-
.elseif	ELSEIF	.option	请见 列表控制命令
.emsg	#error	.page	PAGE
.end	END	.ref	EXTERN 或 IMPORT
.endif	ENDIF	.sect	RSEG
.endloop	ENDR	.setsect	请见 混合命令
.endm	ENDM	.setsym	请见 混合命令
.endstruct	请见 符号控制命令	.space	DS
.equ 或 .set	EQU 或 =	.sslist	不支持
.eval	SET 或 VAR 或 ASSIGN	.ssnolist	不支持
.even	EVEN	.string	DB
.fclist	LSTCND-	.struct	请见 符号控制命令
.fcnolist	LSTCND+	.tag	请见 符号控制命令
.field	请见 常量初始化命令	.text	RSEG
.float	请见 常量初始化命令	.title	请见 列表控制命令
.global	请见 文件参考命令	.usect	请见 符号控制命令
.if	IF	.width	COL
.label	请见 混合命令	.wmsg	请见 混合命令
.length	PAGSIZ	.word	DW
.list	LSTOUT+		

C.3.12 不被支持的 A430 命令 (IAR)

CCS Asm430 汇编程序中不支持下列的 IAR 汇编程序命令：

条件汇编命令	宏命令	
REPTC[1]	LOCAL[2]	
REPTI		
文件参考命令	**混合命令**	**符号控制命令**
NAME 或 PROGRAM	RADIX	DEFINE
MODULE 或 LIBRARY	CASEON	SFRB
ENDMOD	CASEOFF	SFRW
列表控制命令	**C 类型预处理器命令[3]**	**符号控制命令**
LSTMAC (+/-)	#define	ASEG
LSTCOD (+/-)	#undef	RSEG
LSTPAG (+/-)	#if，#else，#elif	COMMON
LSTXREF (+/-)	#ifdef，#ifndef	STACK
	#endif	ORG
	#include	
	#error	

(1) CCS 中没有针对 IAR REPTC/REPTI 命令的直接支持。 然而，相同的功能性可以通过使用 CCS .macro 命令来实现：

```
MIAR MMMMMM
REPTI   zero,"R4","R5","R6"
MOV #0,zero
ENDR
MCCS MMMMMM
zero_regs .macro MM
.var M
.loop
.break ($ismember(item, list) = 0)
MOV #0,item
.endloop
.endm
```

通过调用 "zero_regs R4,R5,R6" 生成代码：

```
MOV #0,R4
MOV #0,R5
MOV #0,R6
```

(2) 在 CCS 中，局部标签是通过使用 $n （其中 n=0...9）或者使用 NAME 定义的吗？ 示例是 $4，$7，或者测试吗？

(3) 通过使用 .cdecls 间接支持 C 类型预处理器命令的使用。 更多与 .cdecls 命令相关的信息请见《*MSP430 汇编语言工具用户指南*》（文献号SLAU131）。

特定 FET 菜单

这个附录对 FET 专用 CCS 菜单进行了说明。

Topic **Page**

D.1 菜单

D.1.1 调试视图：Run→Free Run

调试器使用器件 JTAG 信号来调试器件。 在某些 MSP430 器件上，这些 JTAG 信号与器件端口引脚共享。 正常情况下，调试器将引脚保持在 JTAG 模式，这样的话，器件可被调试。 在此期间，共享端口的功能性不可用。

然而，当选择 Free Run（通过打开一个 Debug View 顶部的 Run 图标旁的下拉菜单进行选择）时，JTAG 驱动器被设置为 3 态，并且当 GO 被激活时此器件被从 JTAG 控制（TEST 引脚被设置为 GND）中释放。 任何被激活的片载断点被保持，而 JTAG 端口引脚返回到它们的端口功能性。

这时，调试器不能访问此器件并且不能确定是否到达一个有效的断点（如果有的话）。 调试器必须手动控制以停止此器件，此时器件的状态被确定（也就是说，达到一个断点了吗？）。

请见 调试中 #9。

D.1.2 Run → Connect Target

当选中时，重新获得对此器件的控制权。

D.1.3 Run → Advanced → Make Device Secure

在目标方器件上熔断 JTAG 保险丝。 保险丝被熔断后，JTAG 不能与器件继续通信。

D.1.4 Project → Properties → Debug → MSP430 Properties → Clock Control

当调试器控制器件时，禁用特定的系统时钟（在一个 STOP 或是断点之后。） 在一个 GO 或者一个单步执行之后，所有的系统时钟被启用 (STEP/STEP INFO)。 只有当调试器暂停不用时才能改变。 请见 调试中 #12。

D.1.5 Window → Show View → Breakpoints

打开 MSP430 断点视图窗口。 这个窗口可被用于设置基本和高级的断点。 通过访问属性，对于每个断点，可单独选择诸如条件触发器 (Conditional Triggers) 和寄存器触发器 (Register Triggers) 的高级设置。 通过打开断点 (Breakpoint) 下拉菜单（此菜单位于窗口顶部 Breakpoint 图标旁），诸如堆栈溢出中断 (Break on Stack Overflow) 的预定义断点。 在 Breakpoint View 窗口中，可通过拖拽将断点组合在一起。 当所有的断点条件满足时，一个组合的断点就会被触发。

D.1.6 Window → Show View → Other... Debug → Trace Control

跟踪视图 (Track View) 使用户可以使用状态存储模块。 状态存储模块只在包含高级仿真模块 (EEM) 完全版本的器件中才会出现（请见表 2-1）。 一个断点被定义后，状态存储视图 (State Storage View) 如配置的那样显示跟踪信息。 当点击窗口右上角的配置属性 (Configuration Properties) 时，可以选择多种跟踪模式。 EEM 细节请见应用报告*《使用 CCS 版本4 的增强型仿真模块 (EEM) 进行高级调试》*（文献号：SLAA393）。

D.1.7 Project → Properties → Debug → MSP430 Properties → Target Voltage

MSP-FET430UIF 的目标方电压可在 1.8V 至 3.6V 之间进行调节。 这个电压在 14 引脚目标方连接器的引脚 2 上提供以从 USB FET 为目标方供电。 如果目标方由外部供电，外部电源电压应该连接到目标方连接器的引脚 4 上，所以 USB FET 能够相应地设置输出信号的电平。 只有当调试器暂停不用时才能被改变。

E.1 MSP430L092

E.1.1 仿真模块

MSP430L092 可运行在两种不同的模式下：L092 模式和 C092 仿真模式。 C092 仿真模式的用途是模拟一个带有高达 1920 代码字节的 C092，此 C092 正处于使用一个 L092 进行掩码生成的最后阶段。 在运行调试器之前，必须在 CCS 中设置调试器的操作模式。 所用选择出现在目标方的配置中：打开您项目中的 MSP430L092.CCXML 文件，在高级目标方配置 (Advanced Target Configuration)，高级设置 (Advanced Setup) 部分点击目标方配置 (Target Configuration)。 MSP430 被选择后，CPU 属性变成可见。图 E-1显示了如何选择 L092 模式以及如何使用加载程序代码 (Loader Code)（请见 加载程序代码）。图 E-2显示了如何选择 C092 模式。

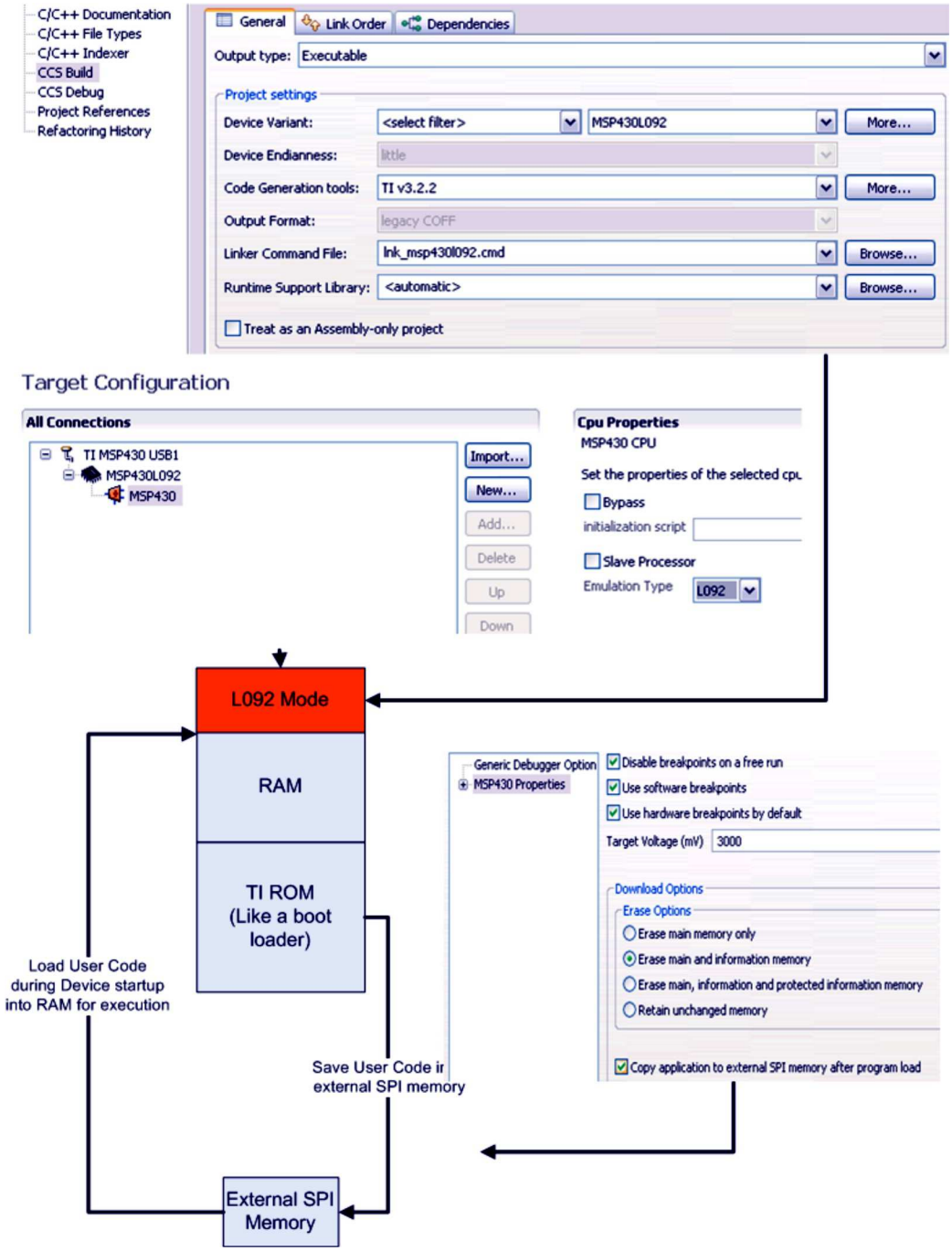

图 **E-1. MSP430L092** 模式

E.1.2 加载程序代码

MSP430L092 中的加载程序代码是 TI 的 ROM 代码，此代码提供一系列的服务。 它使得用户能够在无需开发一个 ROM 掩码的情况下建立匿名应用。 这样一个应用由一个包含加载程序（例如，MSP430L092）的 MSP430 器件和一个 SPI 内存器件（例如，'95512 或者'25640）组成。 那些器件和相似器件可由不同的生产商提供。 带有一个加载程序器件和外部 SPI 内存（用于本地 0.9V 电源电压）的主流应用情况是后期开发，原型开发，和小批量试产 可在 CCS Project Properties→ Debug → MSP430 Properties → Download Options → Copy application to external SPI memory after program load 中设置外部代码下载（请见图 E-1）。

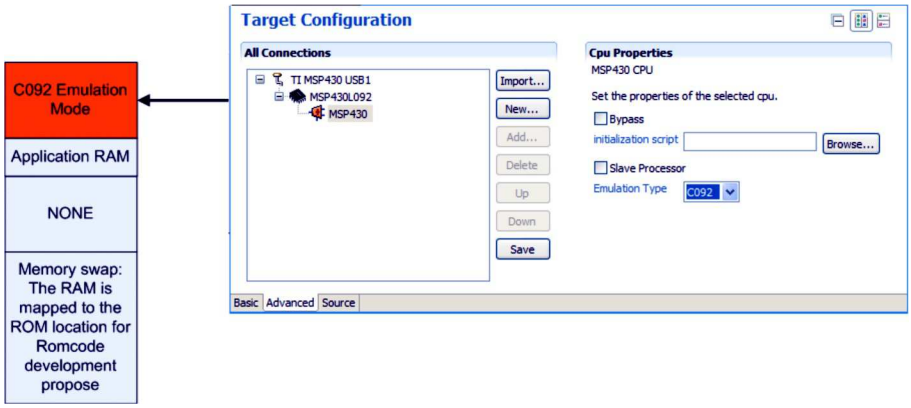

图 **E-2. C092** 仿真模式中的 **MSP430L092**。

E.1.3 C092 密码保护

MSP430C092 是一款用户专用 ROM 器件，此器件受到密码保护。 为了启动一个调试会话，密码必须被提供给 CCS。 打开您项目中的 MSP430C092.CCXML 文件，点击 Advanced Setup section，Advanced Target Configuration 中的 Target Configurations。 MSP430 被选择后，CPU 属性变成可见。图 E-3显示了如何在 CCSv4 目标方配置中提供一个十六进制 (HEX) 密码。

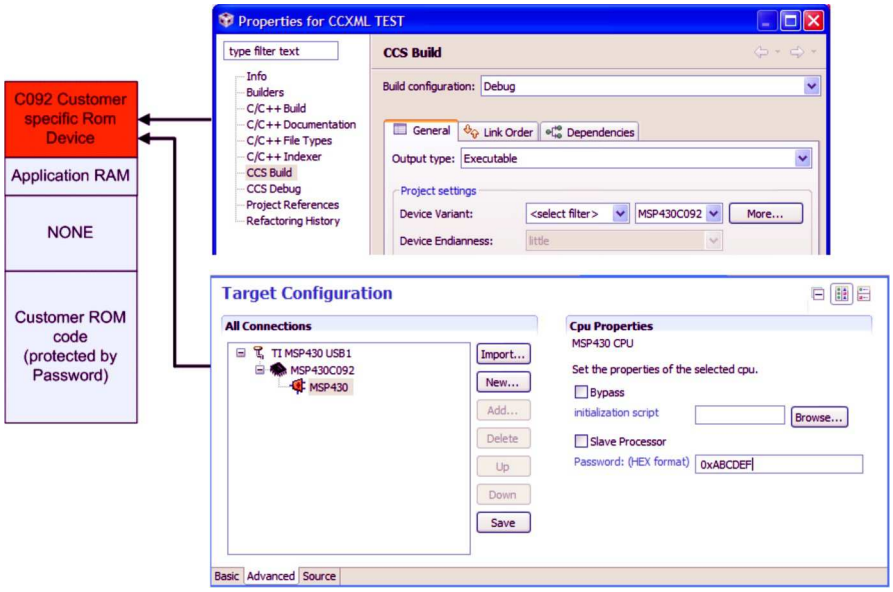

图 **E-3. MSP430C092** 密码访问

E.2 MSP430F5xx/F6xx BSL 支持

大多数 MSP430F5xx 和 'F6xx 器件支持一个默认受到保护的定制 BSL。 为了边界此定制的 BSL，这个保护必须在 CCS Project Properties → Debug → MSP430 Properties → Download Options → Allow Read/Write/Erase access to BSL memory 中被禁用（请见图 E-4）。

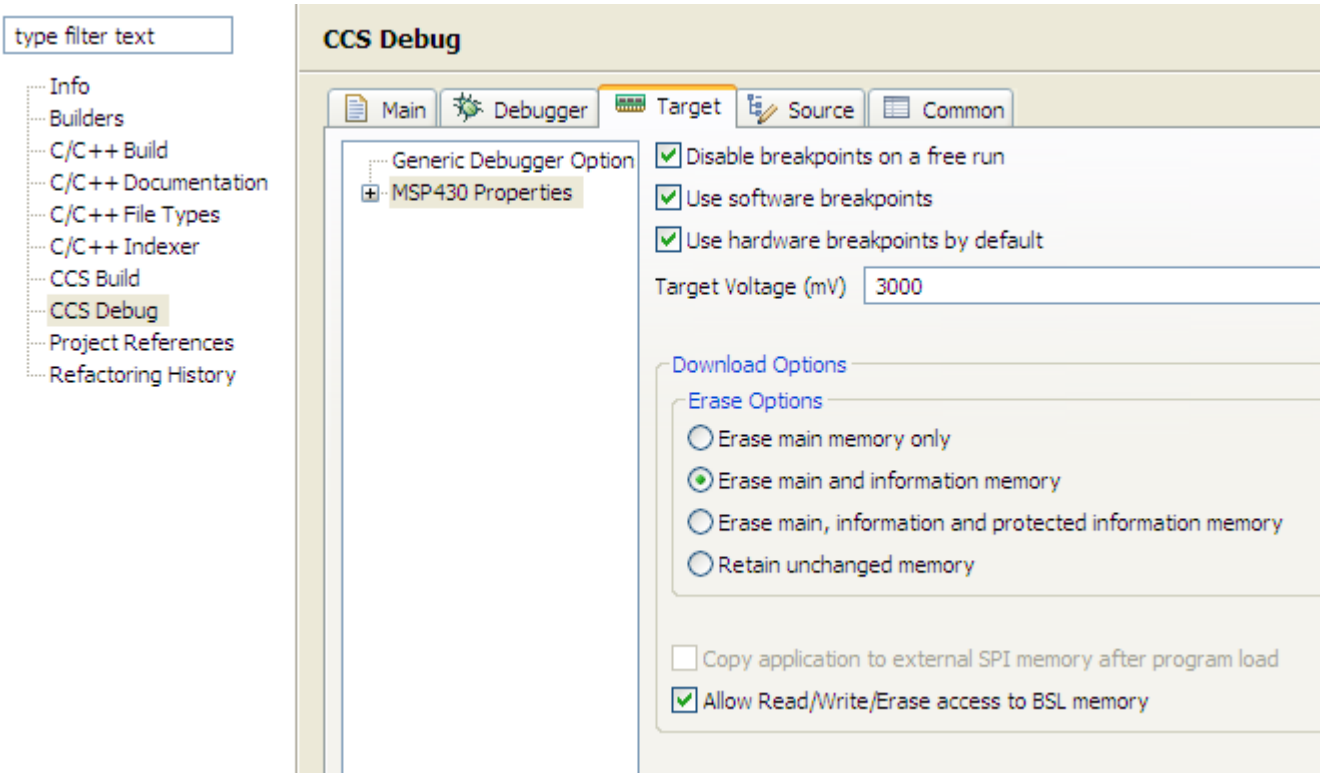

图 **E-4.** 允许对 **BSL** 的访问

E.3 MSP430F5xx/F6xx 密码保护

被选择的 MSP430F5xx 和 'F6xx 器件通过一个用户密码提供 JTAG 保护。 当调试一个 MSP430 衍生器件时，为了启动一个调试会话，必须提供十六进制 JTAG 密码。 打开您项目中的 MSP430Fxxxx.CCXML 文件，点击 Advanced Setup section，Advanced Target Configuration 中的 Target Configurations。 MSP430 被选择后，CPU 属性变成可见（请见图 E-5）。

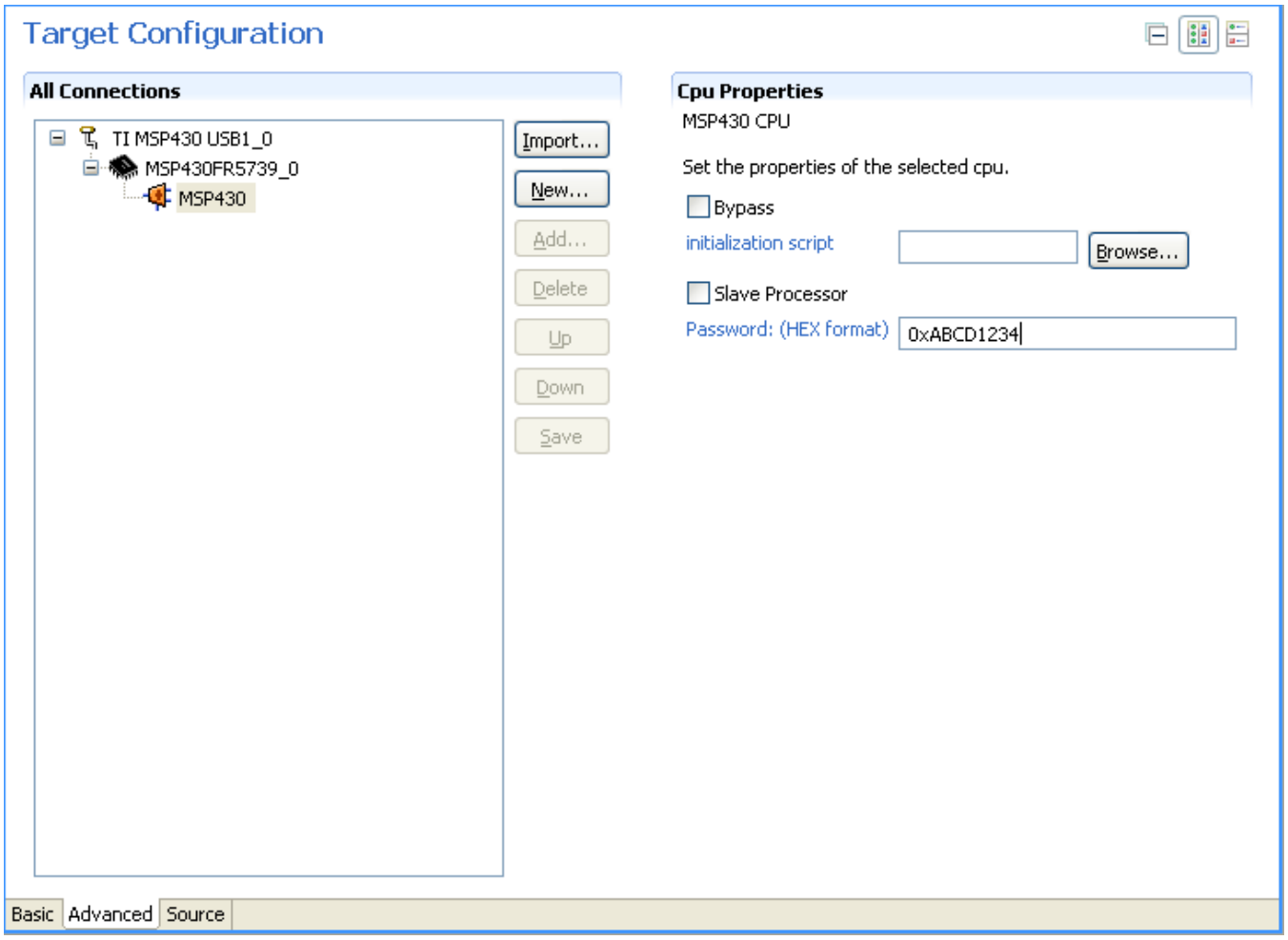

图 **E-5. MSP430** 密码访问

E.4 LPMx.5 CCS 调试支持（只适用于MSP430FR57xx）

LPMx.5 是一个全新的低功耗模式，在这个模式中，对于进入和退出的处理与其它低功耗模式不同。 当正确使用时，LPMx.5 提供器件上可用的最低功耗。 为了实现这一功能，进入 LPMx.5 模式会禁用 PMM 模块的低压降稳压器 (LDO)，从内核上移除电源电压和器件的 JTAG 模块。 由于内核上的电源电压被移除，所有寄存器内容和 SRAM 内容丢失。 从 LPMx.5 模式中退出会引起一个 BOR 事件，这个事件会强制器件完全复位。

E.4.1 正在使用 LPMx.5 进行调试

为了启用 LPMx.5 调试特性，器件唤醒（调试 LPMx.5 模式时需要）复选框上的暂停 (Halt)必须被启用（请见图 E-6）。 为了启用 LPMx.5 调试，点击 Project Properties → Debug → MSP430 Properties → Halt on device wakeup（调试 LPMx.5 模式时需要）。

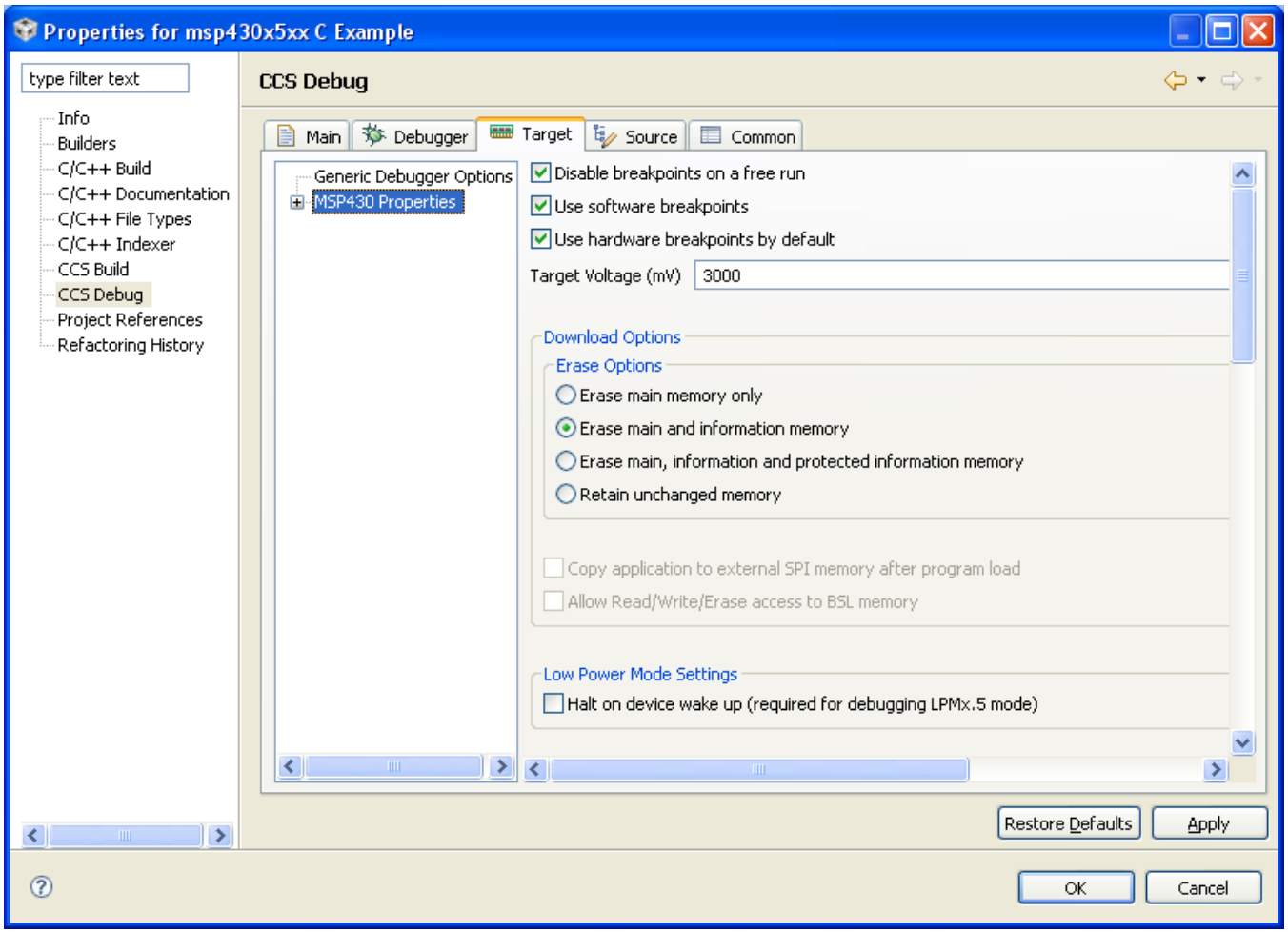

图 **E-6.** 启用 **LPMx.5** 调试支持

如果 LPMx.5 调试模式被启用，目标方器件每次进入和离开 LPMx.5 模式时，在调试器控制台日志文件内会显示一个通知。 按动 CCS 中的 Halt 或者复位 (Reset) 按钮会使目标方器件从 LPMx.5 模式中唤醒并在冷启动时停止该器件。 在 LPMx.5 模式之前有效的所有断点被自动恢复且重新激活。

E.4.2 LPMx.5 调试限制

当一个目标方器件处于 LPMx.5 模式，不能设置或者移除高级条件或者软件断点。 但有可能设置硬件断点。此外，只有在 LPMx.5 模式期间设置的硬件断点能够在 LPMx.5 模式中被移除。 由于会引起器件复位，运行目标方不能与 LPMx.5 模式调试组合使用。

修订历史记录

版本	改变/注释
SLAU157T	全部针对 Code Composer Studio v5.1 的更新信息。 为 CC430F512x，CC430F514x，CC430F614x 添加了仿真特性。
SLAU157S	为 MSP430F52xx，F533x，F643x，F67xx 添加了仿真特性。
SLAU157R	为 MSP430FR57xx，LPMx.5 添加了仿真特性并在附录 E 中添加了通用 MSP430 密码保护指令。
SLAU157Q	为 MSP430F5310 添加了仿真特性。
SLAU157P	为 MSP430AFE253，MSP430F532x，和 MSP430F534x 添加了仿真特性。
SLAU157O	为 MSP430BT5190，MSP430F530x，和 MSP430F563x 添加了仿真特性。 在附录 E 中添加了针对 MSP430F5xx/F6xx 的 BSL 支持。
SLAU157N	在附录 E 中添加了针对 MSP430L092/C092 的仿真特性和内存信息。
SLAU157M	为 MSP430G2xxx，MSP430F51x1，MSP430F51x2，MSP430F550x，MSP430F5510，MSP430F551x，MSP430F552x，MSP430F663x 添加了仿真特性。
SLAU157L	全部为针对 Code Composer Studio v4.1 的更新信息。 为 MSP430F44x1，MSP430F461x，MSP430F461x1 添加了仿真特性。
SLAU157K	为 MSP430F54xxA，MSP430F55xx 添加了仿真特性。 更新和扩展的表 2-1架构信息。
SLAU157J	全部为针对 Code Composer Studio v4 的更新信息。 移除与硬件相关的信息。 它被移至《*MSP430 硬件工具用户指南*》（）。
SLAU157I	在部分 1.7 中添加了 MSP-FET430U100A 工具包以及 MSP-TS430PZ100A 目标方插槽模块电路原理图（图表 B-19）和 PCB（图表 B-20）。 为 CC430F513x，CC430F612x，CC430F613x，MSP430F41x2，MSP430F47x，MSP430FG479，和 MSP430F471xx 添加的仿真特性位于表 2-1中。 更新了 MSP-TS430PN80 目标方插槽模块电路原理图（图表 B-15）中与 MSP430F47x 和 MSP430FG47x 有关的信息。 移除全部与 MSP-FET430Pxx0 和 MSP-FET430X110 工具包有关的信息。
SLAU157H	全部针对 Code Composer Essentials v3 的更新信息。
SLAU157G	添加的 MSP-FET430U5x100 工具包和 MSP-TS430PZ5x100 目标方插槽命令电路原理图。
SLAU157F	将晶振信息添加至部分 1.7。 表 1-1 为添加的调试接口概述。 添加的 eZ430-F2013，T2012，和 eZ430-RF2500。 全部针对 Code Composer Essentials v3 的更新信息。
SLAU157E	添加的 MSP-TS430PW28 目标方插槽模块，电路原理图（图 B-5）和 PCB（图 B-6）。 在部分 1.7 中更新的 MSP-FET430U28 工具包内容信息（支持 DW 或者 PW 封装）。 针对 MSP430F21x2 的仿真特性被添加至表 2-1中。 更新的 MSP-TS430PW14 目标方插槽模块电路原理图（图 B-1）。 更新的 MSP-TS430DA38 目标方插槽模块电路原理图（图 B-7）。
SLAU157D	添加的部分 1.12。 更新的表 2-1。 更新的附录 F。
SLAU157C	更新的附录 F。 在表 2-1中添加的用于 MSP430F22x2，MSP430F241x，MSP430F261x，MSP430FG42x0 和 MSP430F43x 的仿真特性。
SLAU157B	将 MSP-FET430U40 重命名为 MSP-FET430U23x0。 用重命名的 MSP-FET430U23x0 图替代了 MSP-FET430U40 电路原理图和印刷电路板 (PCB) 图。 在部分 A.1 中添加了 FAQ 硬件 #2。 在部分 A.3 中添加了 FAQ 调试中 #4。

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

关于评估板/套件的重要通知

德州仪器 (TI) 在提供整组产品时遵循以下条件：

TI 不会考虑将用于**工程开发、演示或仅作评估用途**的评估板/套件打造为适合普通消费者使用的最终产品。 使用这些产品的人员必须接受电子技术训并严格遵守工程实践标准。 因此，所提供的产品并不具有与所需设计、市场营销和/或制造等方面相关的完整保护注意事项，包括采用此类半导体组件或电路板的最终产品中通常包含的产品安全和环境测量信息。 该评估板/套件并不受与电磁兼容性、限用物质 (RoHS)、回收 (WEEE)、FCC、CE 或 UL 相关的欧盟指令的约束，因此，它们可能不符合这些指令或其它相关指令的技术要求。

如果评估板/套件不符合本用户指南中说明的规范，则可在自发货之日起 30 天内退回该评估板/套件以获取全额退款。 前面所述的保证是零售商向购买者提供的保证，它将替代所有其它的明示或默示的保证或承诺，包括对适用于任何特定用途的商用性或适用性保证。

用户同意对正确安全地使用这些产品承担全部责任和义务。 此外，用户同意 TI 不对由于处理或使用这些产品造成的任何索赔承担责任。 鉴于产品的开放式结构，采取任何适当措施来解决静电放电问题是用户的职责所在。

除了上述赔偿范围以外，任何一方均没有义务对另一方造成的任何间接、特殊、偶然或必然损失承担责任。

目前，TI就相关产品与众多客户进行接洽，因此我们与用户达成的协议**不具备排他性。**

对于**应用帮助、客户产品设计、软件性能或专利权侵犯或此处所描述的服务，TI不承担责任。**

在使用产品之前，请仔细阅读本用户指南，特别是用户指南中的警告和限制通知。 此通知包含有关温度和电压的重要安全信息。 有关 TI 的环境和/或安全计划的其它信息，请联系 TI 应用工程师或访问 www.ti.com/esh。

TI 并未在任何专利权或其它与使用 TI 产品或服务的机器、流程或组合相关的知识产权下授予许可。

FCC 警告

TI 不会考虑将用于**工程开发、演示或仅作评估用途**的评估板/套件打造为适合普通消费者使用的最终产品。 它会生成、使用和发出射频能量，而且尚未依照 FCC 规则第 15 部分子部分中为提供合理的射频干扰保护而制定的计算设备限制执行符合性测试。 在其它环境中操作该设备可能会对无线电通讯造成干扰，在此情况下，用户必须自行承担为更正此干扰而需采取的任何相关措施的费用。

重要声明

德州仪器(TI) 及其下属子公司有权在不事先通知的情况下, 随时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，并有权随时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 保证其所销售的硬件产品的性能符合TI 标准保修的适用规范。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。除非政府做出了硬性规定，否则没有必要对每种产品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 产品或服务的组合设备、机器、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，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。在复制信息的过程中对内容的篡改属于非法的、欺诈性商业行为。TI 对此类篡改过的文件不承担任何责任。

在转售TI 产品或服务时，如果存在对产品或服务参数的虚假陈述，则会失去相关TI 产品或服务的明示或暗示授权，且这是非法的、欺诈性商业行为。TI 对此类虚假陈述不承担任何责任。

TI 产品未获得用于关键的安全应用中的授权，例如生命支持应用（在该类应用中一旦TI 产品故障将预计造成重大的人员伤亡），除非各方官员已经达成了专门管控此类使用的协议。购买者的购买行为即表示，他们具备有关其应用安全以及规章衍生所需的所有专业技术和知识，并且认可和同意，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在关键安全应用中使用其产 品及TI 产品所需的所有法律、法规和安全相关要求。此外，购买者必须全额赔偿因在此类关键安全应用中使用TI 产品而对TI 及其 代表造成的损失。

TI 产品并非设计或专门用于军事/航空应用，以及环境方面的产品，除非TI 特别注明该产品属于“军用”或“增强型塑料”产品。只 有TI 指定的军用产品才满足军用规格。购买者认可并同意，对TI 未指定军用的产品进行军事方面的应用，风险由购买者单独承担，并且独力负责在此类相关使用中满足所有法律和法规要求。

TI 产品并非设计或专门用于汽车应用以及环境方面的产品，除非TI 特别注明该产品符合ISO/TS 16949 要求。购买者认可并同意，如果他们在汽车应用中使用任何未被指定的产品，TI 对未能满足应用所需要求不承担任何责任。

可访问以下URL 地址以获取有关其它TI 产品和应用解决方案的信息：

	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 机动性处理器	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	

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