用于 MSP430™ 的 IAR Embedded Workbench™ 版本 3+

用户指南

Literature Number: ZHCU026AC
June 2004–Revised September 2013

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 IAR Embedded Workbench™ (EW430) 的使用方法，它的使用借助于 MSP430™ 超低功耗微控制器。

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，请见德州仪器 (TI) 提供的相关文档中列出的相关 TI 和 IAR™ 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 开发工具系列的使用。

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（其中包括数据表、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 TI 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

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

MSP430 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

《*MSP430x5xx 和 MSP430x6xx 系列产品用户指南*》，文献号SLAU208

CC430 器件数据表

《*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 器件和 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。 德州仪器 (TI) E2E 社区支持论坛为同行工程师、TI 工程师，和其他专家提供了公开交流的平台。 可在MSP430 网站上找到附加的专用器件信息。

注： **KickStart™ 由德州仪器 (TI) 提供技术支持。**

虽然 Kickstart 是 IAR 的产品，但是德州仪器 (TI) 提供针对它的技术支持。 因此，请不要向 IAR 寻求针对 Kickstart 的技术帮助。 在请求帮助前，请先查阅与 Kickstart 一提供的大量文档。

现在就开始！

这一章提供了安装软件的指令，并显示了如何运行演示程序。

Topic **Page**

1.1 软件安装

按照提供的请先读我 (READ ME FIRST) 文件中的指令来安装 IAR 嵌入式 Workbench™ KickStart。 阅读来自 IAR 的文件 <安装目录>\Embedded Workbench x.x\430\doc\readme.htm 以获得与 Workbench 有关的最新信息。 术语 Kickstart 是指嵌入式工作平台（包括 C-SPY™ 调试器）的受限制版本。 Kickstart 在光盘上与每个 FET 一起提供，而最新版本可从 MSP430 网站上获得。

在前一段（和本文档）中提到的文档可通过：开始→程序→IAR Systems→IAR Embedded Workbench KickStart for MSP430 V3 进行访问。

Kickstart 与 Windows 2000 (SP4)，Windows XP（32 位和 64 位），Windows Vista（32 位和 64 位），以及 Windows 7（32 位和 64 位）兼容。 然而，USB FET 接口只与 Windows XP（32 位和 64 位），Windows Vista（32 位和 64 位），以及 Windows 7（32 位和 64 位）一起工作。

1.2 LED 闪烁

在 FET 上演示的这部分内容相当于 C 语言中的“Hello World！”介绍性程序。 一个使 LED 发光的应用被开发并下载至 FET，然后运行。

1. 启动开发平台（开始 → 程序 → IAR Systems → IAR Embedded Workbench KickStart for MSP430 V3 → IAR Embedded Workbench）。
2. 点击 File → Open Workspace 来打开位于：<安装目录>\Embedded Workbench x.x\430\FET_examples\Flashing the LED.eww 下的文件。 工作平台窗口打开。
3. 点击工作平台窗口底部与 MSP430 器件 (MSP430xxxx) 和所需语言（汇编语言或者 C 语言）相对应的标签来将一个项目激活（请见图 1-1）。

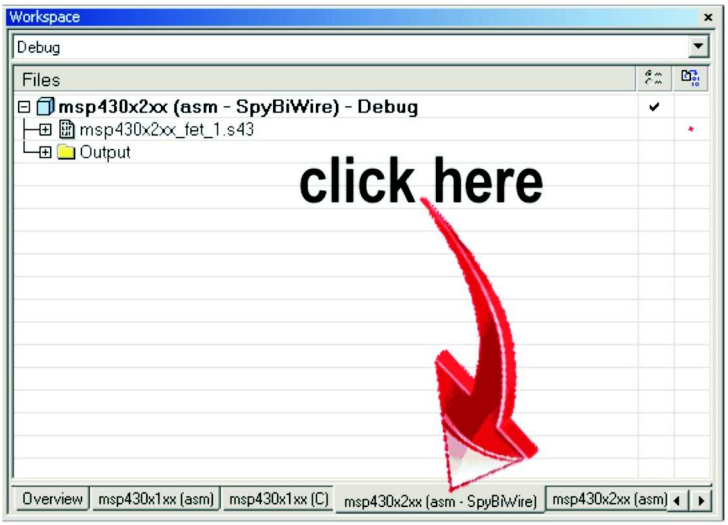

图 **1-1.** 激活项目

或者，在 Workspace Overview 标签内右键点击来激活项目（请见图 1-2）。

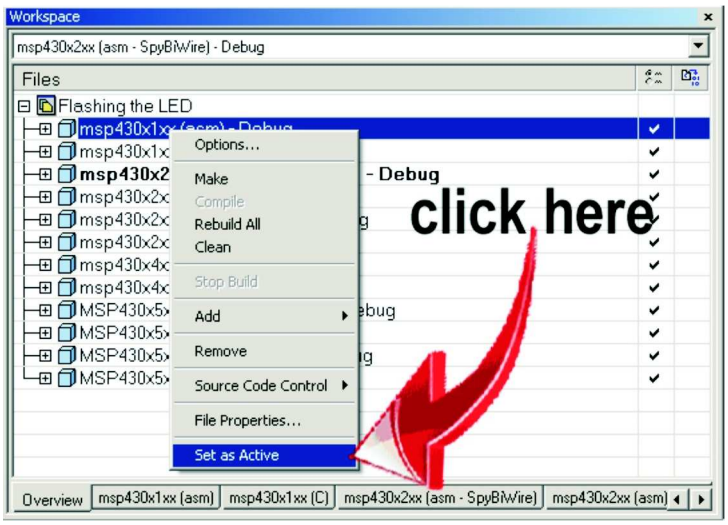

图 **1-2.** 在 **Workspace Overview** 中激活项目

4. 点击 Project → Options → FET Debugger → Setup → Connection 来选择合适的端口：德州仪器 (TI) LPT-IF 用于并行 FET 接口 (MSP-FET430PIF) 或者德州仪器 (TI) USB-IF 用于 USB 接口 (MSP-FET430UIF) 或者用于 eZ430。
5. 点击 Project → Rebuild All 来编译和生成源代码。 通过双击项目，然后双击显示的源文件来查看源代码。
6. 点击 Project → Debug 来启动 C-SPY 调试器。 C-SPY 擦除器件闪存，然后将应用对象文件下载到器件闪存中。

   如果 C-SPY 不能与器件通信，请见常见问题和解答 (FAQ) 调试中 #1。
7. 点击 Debug → Go 来启动应用。 LED 应该闪烁。
8. 点击 Debug → Stop Debugging 来停止调试，退出 C-SPY，并返回工作平台。
9. 点击 File → Exit 来退出工作平台。

恭喜，您刚刚建立并测试了一个 MSP430 应用程序！

1.3 光盘和网络上重要的 MSP430 文档

专用器件数据表和用户手册是 MSP430 信息的主要来源。 生产时这些文档可获得的最新版本与这个工具一起通过只读光盘提供。 MSP430 网站 (www.ti.com/msp430) 上有这些文档的最近版本。

描述 IAR 工具（Workbench 和 C-SPY，汇编程序，C 语言编译器，连接器和库）的 PDF 文档位于 common\doc 和 430\doc 文件夹内。 此文档的附录（即，最新信息）以 HTML 格式存放在同一目录内。430\doc\readme_start.htm 为 IAR 文档导航提供了便捷的开始点。

开发流程

这一章对如何使用 Kickstart 来开发应用软件以及如何使用 C-SPY 来对此软件进行调试进行了说明。

Topic **Page**

2.1 概述

使用 Workbench 在汇编语言或者 C 语言中开发应用，并使用 C-SPY 对它们进行调试。 C-SPY 被无缝集成到工作平台内。 然而，对代码开发环境 （工作平台）和调试器 (C-SPY) 进行区分会使使用更加便捷。 C-SPY 可被配置为与 FET（即，一个真实的 MSP430 器件）一起运行或者与器件的软件模拟器一起工作。 Kickstart 是指 Workbench 和 C-SPY 一起。 Kickstart 软件工具是 IAR 的一个产品。

MSP430 系列和 Kickstart 的文档范围很广。 提供这个工具的光盘包含大量的 MSP430 说明性文档。 MSP430 主页 (www.ti.com/msp430) 是 MSP430 信息的另外一个来源。 KickStart 的组件（工作平台和调试器、汇编程序、编译器、连接器）完全记录在 <安装根目录>\Embedded Workbench x.x\common\doc 和 <安装根目录>\Embedded Workbench\430\doc. .htm 文件中，此文件遍及包含最近更新信息和 PDF 文件附录的 Kickstart 目录树中。 此外，Kickstart 文档也可通过在线帮助文件获得。

IAR 和 TI 提供的请先读我 (Read Me First) 文件和这个文档可通过使用开始 → 程序 → IAR Systems → IAR Embedded Workbench KickStart for MSP430 V3 来访问。

工具	用户指南	最近更新的信息
Workbench，C-SPY	EW430_UsersGuide.pdf	readme.htm，ew430.htm，cs430.htm，cs430f.htm
汇编程序	EW430_AssemblerReference.pdf	a430.htm，a430_msg.htm
编译器	EW430_CompilerReference.pdf	icc430.htm，icc430_msg.htm
C 语言库		CLibrary.htm
连接器和库管理程序	xlink.pdf	xlink.htm，xman.htm，xar.htm

2.2 使用 KickStart

KickStart 版本是一款 IAR Embedded Workbench 的特别启动器套件或评估版本，它在提供的代码尺寸和服务以及支持方面受到限制。 限制：

- C 语言编译器不生成一个汇编代码列表文件。
- 对于传统 MSP430 器件，MSP430 IAR Kickstart C/C++ 编译器的代码容量限制被设定为 4K 字节，而对于 MSP430X 器件，被设定为 8K 字节（与何种 MSP430 器件所采用何种架构相关的详细信息，请见表 2-1）。
- 发布的 IAR 汇编程序是无任何限制的完全版本。
- IAR XLINK 连接器将为传统的 MSP430 器件编译链接最大 4K 字节的 C 语言源代码，而对于 MSP430X 器件，将编译链接8K 字节的 C 语言源代码（与何种 MSP430 器件所采用何种架构相关的详细信息，请见表 2-1），但是对于汇编代码，则没有数量限制。
- IAR Kickstart C-SPY 仿真器将为传统的 MSP430 器件读取最大 4K 字节的 C 语言代码，而对于 MSP430X 器件，将读取 8K 字节的 C 语言代码，但是可读取无数量限制的汇编代码（与何种 MSP430 器件所采用何种架构相关的详细信息，请见表 2-1）。
- MISRA C 不可用。
- 实时运行库源代码不包括在内。

软件工具的完全（也就是说，无限制）版本可从 IAR 购买。 一个中等特性工具集 - 被称为 Baseline，和一个 12K 字节的 C 语言代码长度限制以及基本浮点运算 - 也可从 IAR 获得。 更多信息请见 IAR 的网站（www.iar.se）。

2.2.1 项目设置

配置 Workbench 和 C-SPY 所需的设置很多且十分详细。 当处理项目配置时，请阅读并充分理解 IAR 所提供的文档。 对所提供的汇编程序和 C 语言示例的项目设置进行检查（使用 Project → Options 选择项目名称来访问项目设置）。 当开发您自己的项目时，请将这些项目设置作为模版。 请注意，如果在进行项目设置时没有选择项目名称，那么所做的设置将被应用于所选择的文件（而不是项目）。

建议使用或必须使用以下项目设置：

- 指定目标器件 (General Options → Target → Device)。
- 启用一个汇编程序项目或者一个 C 语言或汇编语言项目 (General Options → Target → Assembler-only project)。
- 生成一个可执行输出文件 (General Options → Output → Output file → Executable)。
- 为了尽可能轻松地调试一个 C 语言项目，请禁用优化 [C/C++ Compiler → Optimizations → Size → None (Best debug support)]。
- 在编译器输出中生成调试信息 (C/C++ Compiler → Output → Generate debug information)。
- 为 C 语言预处理器指定搜索路径 (C/C++ Compiler → Preprocessor → Include Paths)。
- 在汇编程序输出中生成调试信息 (Assembler → Output → Generate Debug Info)。
- 为汇编程序预处理器指定搜索路径 (Assembler → Preprocessor → Include Paths)。
- 要使用 C-SPY 调试项目，指定一个兼容格式 [Linker → Output → Format → Debug information for C-SPY (with runtime control modules/With I/O emulation modules)（带有运行时间控制模块或带有 I/O 仿真模块）]。
- 为任一已使用的库指定搜索路径 (Linker → Config → Search paths)。
- 指定 C-SPY 驱动器。 选择 Project → Options → Debugger → Setup → Driver → FET Debugger 在 FET （即，MSP430 器件）上进行调试。 选择 Simulator 在模拟器上进行调试。 如果 FET 调试器被选择，那么使用 Project → Options → FET Debugger → Setup → Connection 来选择合适的端口：Texas Instruments LPT-IF 用于并行 FET 接口 (MSPFET430PIF) 或者 Texas Instruments USB-IF 用于USB 接口 (MSP-FET430UIF) 或者用于 eZ430。
- 启用器件说明文件。 这个文件使 C-SPY “注意到”正在调制的器件细节。 这个文件与指定目标器件相对应 (Debugger → Setup → Device description file → Override default)。
- 在项目代码下载之前启用主内存和信息内存擦除 (Properties → Download → Erase Main and Information Memory)。
- 为了在调试期间将系统性能最大化，请禁用虚拟断点 (FET Debugger → Breakpoints → Use virtual breakpoints) 并禁用所有系统断点 (FET Debugger → Breakpoints → System breakpoints on)。

注: **使用出厂设置 (Factory Settings) 来快速配置一个项目。**

使用 Factory Settings 按钮将一个项目快速配置为一个可用状态。

下面的步骤可被用于快速配置一个项目。 请注意通用选项 (General Options) 标签上没有 Factory Settings 按钮。

1. 指定目标器件 (General Options → Target → Device)。
2. 启用一个汇编程序项目或者一个 C 语言或汇编语言项目 (General Options → Target → Assembler-only project)。
3. 生成一个可执行输出文件 (General Options → Output → Output file → Executable)。
4. 将编译器置成出厂设置 (C/C++ Compiler → Factory Settings)。
5. 将汇编程序置成出厂设置 (Assembler → Factory Settings)。

6. 将连接器置成出厂设置 (Linker → Factory Settings)。
7. 将 C-SPY 置成出厂设置 (Debugger → Factory Settings)。
8. 在硬件上调试 (Debugger → Setup → Driver → FET Debugger)。
9. 指定用于连接 FET 的可用并行端口（如果不是 LPT1 的话）(FET Debugger → Setup → Connection → Texas Instruments LPT-IF) 或者指定 USB 端口 (FET Debugger → Setup → Connection → Texas Instruments USB-IF)。

注: **在提及文件时，避免使用绝对路径名称。**

相反地，使用相对路径名称关键字 $TOOLKIT_DIR$ 和 $PROJ_DIR$。 这些关键字的说明请见 IAR 文档。 相对路径名称的使用可允许项目轻松迁移，并且当 IAR 系统升级时（例如，从 Kickstart 或者 Baseline 升级到完全版），无需修改项目。

2.2.2 在 IAR EW430 5.60.1 和更新的版本中使用针对 MSP430 的数学函数库

TI 的 Mathlib 是 EW430 5.60.1 和更新版本的一部分。 这个经优化的库在使用浮点标量数学运算中将性能提升了 26 倍多。 更多细节，请参见 MSPMathlib 主页 (http://www.ti.com/tool/mspmathlib)。

可在全部支持的器件上为新建和现有的项目启用 MSPMathlib。 在项目选项中启用或禁用 MSPMathlib (General Options → Library Configuration → MathLib)。

2.2.3 用于 MSP430L092 和 MSP430C092 的附加项目设置

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 正处于掩码生成的最后阶段。

启动调试器之前，运行模式由 EW430 确定。 有两个单选按钮用于模式选择。 缺省情况下，L092 模式被选择（请见图 2-1和图 2-2）。

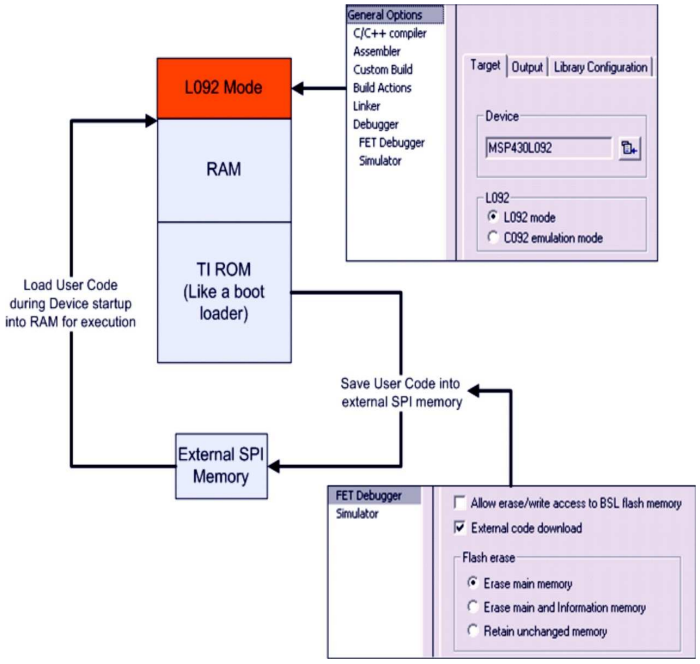

图 **2-1. L092** 模式

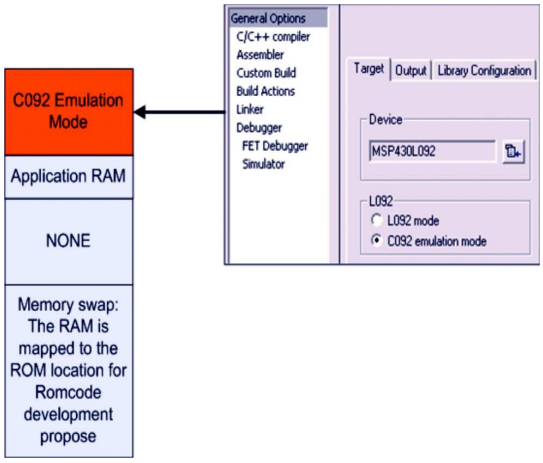

图 **2-2. C092** 仿真模式

2.2.3.1 MSP430L092 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。 这样一个应用包括一个 MSP430 器件，此器件含有加载程序（例如，MSP430L092）和一个 SPI 内存器件（例如，'95512 或者 '25AA40）；可从不同的制造商获得这些器件和相似器件。

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。

图 2-1显示了将应用载入外部 SPI 内存的选择。

2.2.3.2 MSP430C092 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 EW430。图 2-3显示了如何在 EW430 中提供一个十六进制 (HEX) 密码。

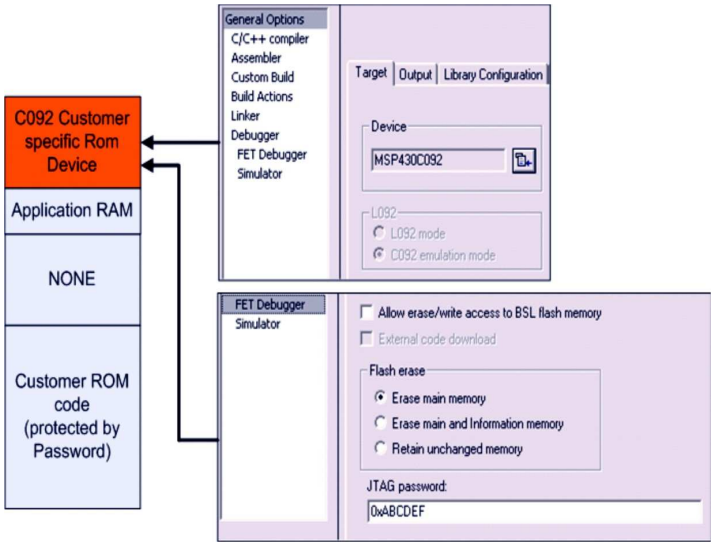

图 **2-3. C092** 密码

2.2.4 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2.2.1 节，项目设置）。 《*MSP430 IAR 嵌入式工作平台 IDE 用户指南*》提供了此过程一个更加综合性的概述。

1. 启动工作平台 (Start → Programs → IAR Systems → IAR Embedded Workbench KickStart for MSP430 V3 → IAR Embedded Workbench)。
2. 创建一个新的文本文件 (File → New → File)。
3. 将程序文本输入到文件。

注: **使用 .h 文件来简化代码开发。**

定义器件寄存器和每一个器件位名称的文件与 Kickstart 一起提供。 这些文件可以大大简化您的程序开发任务。 此文件位于 <安装目录>\Embedded Workbench x.x\430\inc。其中包括与您文本文件中目标器件相对应的 .h 文件（#include "msp430xyyy.h"）。 此外，还提供了文件 io430xxxx.h，此文件因包含 C 源文件而被优化。

---

4. 保存程序文本文件 (File → Save)。

   建议将汇编语言文本文件保存为一个文件类型后缀为 ".s43" 的文件，而将 C 语言文本文件保存为一个文件类型后缀为 ".c" 的文件。
5. 创建一个全新的工作区 (File → New → Workspace)。
6. 创建一个新项目 (Project → Create New Project)。 选择工具链：MSP430，项目模版：清空项目并点击 'OK'。 指定一个项目名称并点击保存 (Save)。
7. 将程序文本文件添加到项目 (Project → Add Files)。 选择程序文件并点击打开 (Open)。 或者，双击文件将其添加到项目。

注: **如何将汇编语言源文件添加到您的项目中**

出现在添加文件 (Add Files) 窗口中的默认文件类型为 "C/C++ Files"。 为了查看汇编语言文件 (.s43)，在 "Files of type" 下拉菜单中选择 "Assembler Files"。

---

8. 保存工作区 (File → Save Workspace)。 指定工作区名称并点击 Save。
9. 配置项目选项 (Project → Options)。 对于每一个子类别（General Options，C/C++ Compiler，Assembler，Linker，Debugger），接受默认 Factory Settings，以下情况除外：
   - 指定目标器件 (General Options → Target → Device)。
   - 启用一个汇编程序项目或者一个 C 语言或汇编语言项目 (General Options → Target → Assembler-only project)。
   - 生成一个可执行输出文件 (General Options → Output → Output file → Executable)。
   - 为了在 FET（即，MSP430）上进行调试，点击 Debugger → Setup → Driver → FET Debugger。
   - 指定用于与 FET 连接的可用端口 (FET Debugger → Setup → Connection)。
10. 建立项目 (Project → Rebuild All)。
11. 用 C-SPY 调试应用 (Project → Debug)。 这样将启动 C-SPY，从而使其获得对目标器件的控制，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 C-SPY 不能与器件通信，请见 FAQ 调试中 #1。
12. 点击 Debug → Go 来启动应用。
13. 点击 Debug → Stop Debugging 来停止应用，退出 C-SPY，并返回工作平台。
14. 点击 File → Exit 来退出工作平台。

2.2.5 用于 LPMx.5 调试的附加项目设置

2.2.5.1 LPMx.5 是什么

LPMx.5 是一个低功耗模式，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)，这将会从内核和器件的 JTAG 模块上移除电源电压。 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。

注: 目前，当 LPMx.5 调试被激活时，Embedded Workbench 不支持 "RELEASE JTAG ON GO" 选项。

LPMx.5 详细信息请见 MSP430 器件系列用户指南。

2.2.5.2 调试 LPMx.5

要启用 LPMx.5 调试特性，必须通过点击 FET Debugger → Setup → Debug LPMx.5 来选择 "Debug LPMx.5" 单选框（请见图 2-4）。

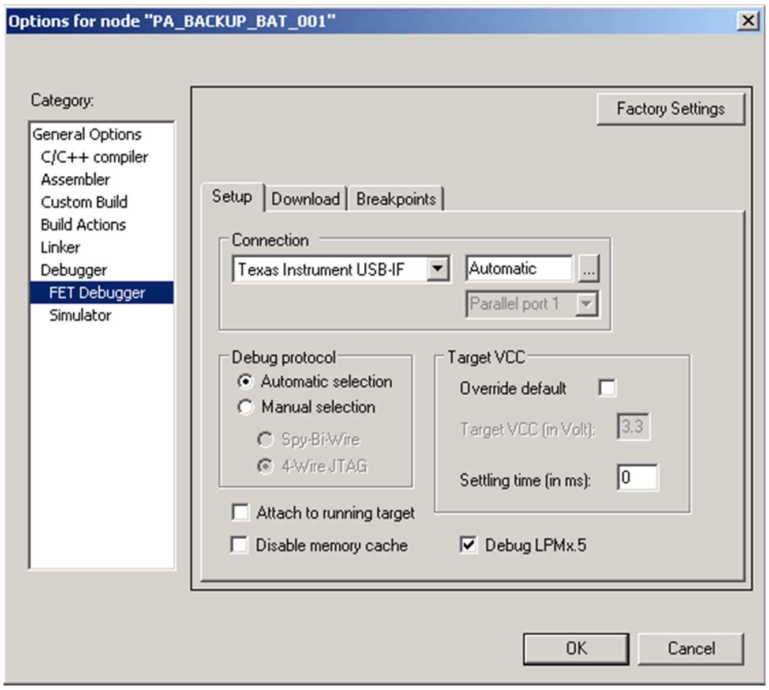

图 2-4. 启用 LPMx.5

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 (Debugger) 的日志文件内会显示一个通知。图 2-5

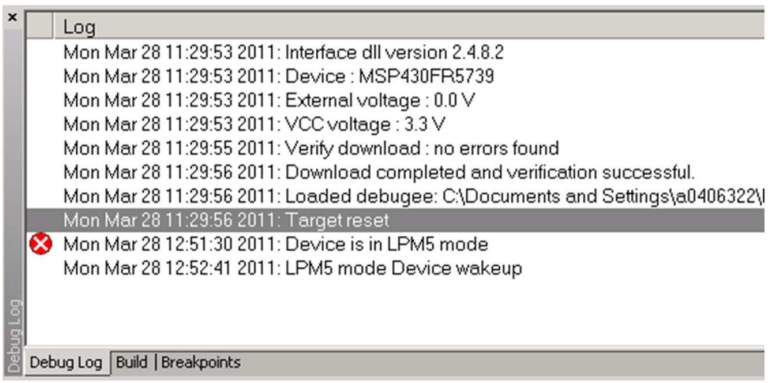

图 **2-5. LPMx.5** 通知

按下 EW430 中的 Halt（停止）或者 Reset（复位）按钮将器件从 LPMx.5 模式中唤醒并在代码启动时将其停止。 在 LPMx.5 模式之前有效的所有断点被自动恢复且重新激活。

2.2.5.3 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才能够在 LPMx.5 模式中被移除。 由于会引起器件复位，连接正在运行的目标方不能与 LPMx.5 模式调试组合使用。

2.2.6 MSP430 器件的密码保护

当调试一个支持用户密码保护的 MSP430 器件时，十六进制 JTAG 密码必须被提供来启动一个调试回话。

点击 FET Debugger → Download → JTAG password 来设定 JTAG 密码（请见图 2-6）。

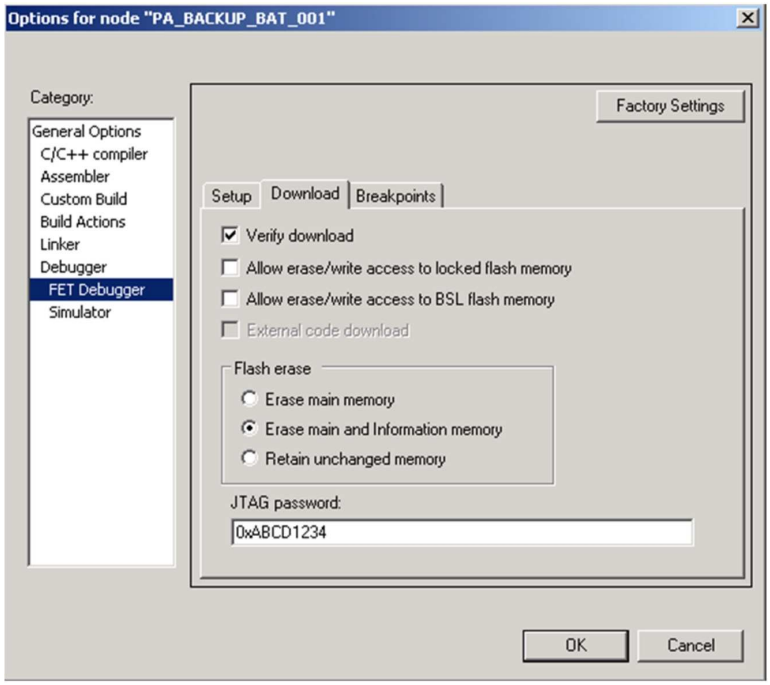

图 **2-6. JTAG** 密码

2.2.7 使用一个现有的 IAR V1.x，V2.x 或 V3.x 项目

可使用一个来自 IAR V1.x，V2.x 或 V3.x 系统（支持全新的 IAR V4.x 系统）的现有项目；请见 IAR 文档 *《EW430 x.xx 的逐步迁移》*。 这个文档位于 <安装根目录>\Embedded Workbench x.x\430\doc\migration.htm。

2.2.8 堆栈管理和 .xcl 文件

通过项目选项对话 (General Options → Stack/Heap) 或者通过对 .xcl 连接器控制文件的直接修改来配置保留的堆栈大小。 这些被输出到连接器的文件包含控制器件内存（RAM，闪存）分配的基本指令。 这些文件完整说明请见 IAR XLINK 文档。 与 FET 一起提供的 .xcl 文件（<安装根目录>\Embedded Workbench x.x\430\config\lnk430xxxx.xcl）定义了一个称为 CSTACK 的浮动段 (RSEG)。 CSTACK 用于定义 RAM 区域，此区域被用于 C 语言程序内的系统堆栈。 CSTACK 也可被用在汇编语言程序中 (`MOV.W #SFE(CSTACK)MSP`)。 CSTACK 被确定为从 RAM 的最后位置延伸 50 个字节（也就是说，此堆栈从 RAM 向下延伸 50 个字节）。

.xcl 文件中其它基本指令定义了其它浮动区域，这些区域可从 RAM 的首位置分配至堆栈底部。 请注意以下关键内容：

- **提供的 .xcl 文件都会为堆栈保留 50 个字节的 RAM，这与是否真正需要这一数量的堆栈（或者保留的数量是否满足需要）无关。**
- **没有针对堆栈的运行时间检查。 此堆栈可溢位 50 个被保留的字节，并有可能写覆盖其它段。 无错误被输出。**

可对提供的 .xcl 文件进行修改以调整堆栈的大小来满足应用的需要；编辑 -D_STACK_SIZE=xx 来为堆栈分配 xx 个字节。 请注意，如果需要的话，.xcl 文件还为堆保留 50 个字节（例如，通过 malloc()）。

2.2.9 如何生成德州仪器 (TI) .TXT（和其它格式）文件

为了与 GANG430 和 PRGS430 编程器一起使用，Kickstart 连接器可被配置成以 TI .TXT 格式输出项目。 点击 Project → Options → Linker → Output → Format → Other → msp430-txt。 也可选择 Intel™ 和 Motorola™ 格式。

更多信息，请见附录 A 中的 FAQ 程序开发 #6。

2.2.10 示例程序概述

<安装根目录>\Embedded Workbench x.x\430\FET_examples 提供了针对 MSP430 器件的示例程序。 每个工具文件夹包含含有汇编程序和 C 语言源代码的文件夹。

<安装根目录>\Embedded Workbench\x.x\430\FET_examples\Flashing the LED.eww 非常便捷地将 FET_1 演示代码组织成为一个工作区。 此工作区包含用于每个 MSP430 器件系列的汇编语言和 C 语言项目。 提供用于每一个项目的调试和发布版本。

<安装根目录>\Embedded Workbench x.x\430\FET_examples\contents.htm 便捷地组织和记录了这些示例。

在MSP430 主页的代码示例下可找到附加的代码示例。 请注意，一些示例程序在 LFXT1 上要求一个 32KHz 晶振，并不是所有 FET 都提供 32KHz 晶振。

2.3 使用 C-SPY

C-SPY 内的 FET 专用菜单说明请见附录 B。

2.3.1 断点类型

C-SPY 断点机制使用少数几个片内调试资源（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，应用程序可以以器件的全速（或者“实时”）运行。 当多于 N 个断点被设定时并启用虚拟断点时 (FET Debugger → Breakpoints → Use virtual breakpoints)，此应用在主机 PC 下运行；系统以低很多的速度运行，但是提供不受限的软件断点数量（或者非实时运行）。 在非实时模式期间，实际上，PC 重复单步执行器件并在每次运行之后询问器件已确定一个断点是否已经被击中。

支持（代码）地址和数据（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	LPMx.5 调试支持
CC430F512x	MSP430Xv2	X	X	2	X	X			X
CC430F513x	MSP430Xv2	X	X	2	X	X			
CC430F514x	MSP430Xv2	X	X	2	X	X			X
CC430F612x	MSP430Xv2	X	X	2	X	X			
CC430F613x	MSP430Xv2	X	X	2	X	X			
CC430F614x	MSP430Xv2	X	X	2	X	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			
MSP430G22xx	MSP430		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			

[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	**LPMx.5** 调试支持
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			
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	3	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	
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	
MSP430SL5438A	MSP430Xv2	X	X	8	X	X	X	X	
MSP430F543x	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	
MSP430F535x	MSP430Xv2	X	X	8	X	X	X	X	X
MSP430F563x	MSP430Xv2	X	X	8	X	X	X	X	
MSP430F565x	MSP430Xv2	X	X	8	X	X	X	X	X
MSP430FR57xx	MSP430Xv2	X	X	3	X	X			X
MSP430FR59xx	MSP430Xv2	X	X	3	X	X			X
MSP430F643x	MSP430Xv2	X	X	8	X	X	X	X	
MSP430F645x	MSP430Xv2	X	X	8	X	X	X	X	X
MSP430F663x	MSP430Xv2	X	X	8	X	X	X	X	X
MSP430F665x	MSP430Xv2	X	X	8	X	X	X	X	X
MSP430F67xx	MSP430Xv2	X	X	3	X	X			X
MSP430i20xx	MSP430	X	X	2		X			
MSP430L092	MSP430Xv2	X		2		X			
MSP430TCH5E	MSP430	X	X	2		X			

2.3.2 使用断点

如果 C-SPY 在开始时设置的断点数量大于 N 并且虚拟断点被禁用，将有一个消息通知用户只有 N 个（实时）断点被启用（此时一个或者多个断点将被禁用）。 请注意工作平台允许设定任意数量的断点，而不用考虑 C-SPY 的使用虚拟断点设置。 如果虚拟断点被禁用，在 C-SPY 可设定的最大断点数量为 N。

如果 Project → Options → Debugger → Setup → Run To 被启用，复位时将临时需要一个断点（请见 FAQ 调试中 #32）。

运行至光标 (Run to Cursor) 操作临时需要一个断点。 因此，如果虚拟断点被禁用，当 Run to Cursor 被使用时，只有 N-1 个断点可被使用（请见 FAQ 调试中 #33）。

如果，在处理一个断点时，一个中断被激活，C-SPY 将停止在中断服务程序的第一条指令上（请见 FAQ 调试中 #26）。

2.3.3 使用单步执行

当调试一个汇编语言文件时，越过子函数、跳出子函数、和执行下一条指令的操作与进入子函数的操作一样；也就是说，当前指令将全速执行。

当调试一个汇编语言文件时，一个调用 (CALL) 的单步操作在被调用函数的第一条指令上停止。

当调试一个汇编语言文件时，一个执行子函数的操作将以在函数末设置断点，并以GO全速执行子函数来实现。

当调试一个 C 语言文件时，一个单步（一步）指令操作执行下一条 C 语言指令。 因此，有可能越过一个函数调用。 如果可能的话，一个硬件断点被放置在函数调用之后，一个 GO 在后台被执行。 这将使得函数被全速执行。 如果无硬件断点可用，此函数在非实时模式下执行。 支持进入子函数。 支持跳出子函数。

在反汇编模式中 (View → Disassembly)，一个非调用指令的单步操作以全速 来执行指令。

在反汇编模式中 (View → Disassembly)，如果可能的话，一个调用指令的实现是将一个硬件断点指令放置在调用指令之后，然后执行 GO。 被调用的函数以全速执行。 如果在 GO 之前没有可用的硬件断点，被调用的函数在非实时模式下执行。 在任一情况下，执行在调用之后的指令上停止。

只有当源指令出现时，才有可能进行单步操作。 当运行没有源代码的代码，必须使用断点（也就是说，将断点放置在对没有源代码的函数的调用之后，然后转到实时模式中的断点）。

在单步操作期间，如果一个中断被激活，那么当前的指令被完成并且 C-SPY 在中断处理例程的第一条指令上停止（请见 FAQ 调试中 #26）。

2.3.4 使用观察窗口

C-SPY 观察窗口机制允许在调试期间允许监控 C 语言变量。 虽然最初设计并非如此，观察窗口机制可被延伸至监控汇编程序变量。

假定观察的变量在 RAM 中定义，例如：

```
RSEG DATA16_I varword ds 2 ; two bytes per word varchar ds 1 ; one byte per character
```

在 C-SPY 中：

1. 打开观察窗口 (View → Watch)。
2. 点击 Debug → Quick Watch。
3. 为了观察 varword，在表达式串口中输入：
   `(__data16 unsigned int *) varword`
4. 为了观察 varchar，在表达式中输入：
   `(__data16 unsigned char *) varchar`
5. 点击添加观察 (Add Watch) 按钮。

6. 关闭快速观察 (Quick Watch) 窗口。
7. 对于已经在观察窗口中创建的条目，点击符号 + 来显示被观察变量的内容（或者值）。

为了改变所显示变量（缺省、二进制、八进制、十进制、十六进制、字符）的格式，选择类型，点击鼠标右键，然后选择所需格式。 通过选择可改变已显示的变量的值，然互输入新的值。

在 C 语言中，可以通过将变量拖拽至观察窗口内进行观察。

由于 MSP430 外设是内存映射的，可将观察变量的概念延伸至观察外设。 请注意当外设被 C_SPY 读写时，有可能会有副作用（请见 FAQ 调试中 #24）。

通过在它们的名称前加上 '#' 符号，可指定观察的 CPU 内核寄存器（即，#PC，#SR，#SP，#R5 等）。

在观察窗口中被观察的变量只有当 C-SPY 获得对器件的控制权时才可被更新（例如，在一个断点命中、一个单步操作、或者一个停止或退出之后）。

虽然寄存器可在观察窗口中被监控，View → Register 是首选的方法。

常见问题和解答

这个附录介绍了程序硬件开发和调试工具方面常见问题的解决方案。

Topic **Page**

A.1 硬件

对于硬件相关 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）

1. 在 **430/tutor** 文件夹中提供的文件只与模拟器一起工作。 不要用 FET 使用这些文件（请见 FAQ 程序开发 #11）。
2. 一个常见的 **MSP430“错误”就是禁用安全装置机制失败**；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（请见 FAQ 程序开发 #14）。
3. 当向一个项目中添加源文件时，不要添加已包括在源文件内的文件，这些文件已经被添加到项目中（例如，一个 .c 或者 .s43 文件中的 .h 文件）。 这些文件被自动添加到项目文件层次中。
4. 在汇编程序中，在双引号内包含一个字符串（“字符串”）自动附加一个零字节到字符串（作为字符串末端的标记）。 在一个单引号内包含的字符串（‘字符串’）不是这样。
5. 当使用编译器或者汇编语言时，如果源代码行的最后一个字符是反斜杠 **(\)**，随后的回车或换行被忽略（也就是说，好像当前代码行和下一代码行是一个单一行）。 当采用这种方法使用时，反斜杠字符是一个“代码行继续”字符。
6. 为了与 **C-SPY** 一起使用，连接器输出格式必须为 **"Debug information for C-SPY" (.d43)**（针对 **C-SPY** 的调试信息）。 否则 C-SPY 不启动，并且一个错误消息被输出。 C-SPY 无法输入一个 .TXT 文件。
7. 使用 Project → Options → General Options → Target → Position-Independent Code可生成浮动地址代码。
8. 在 **C** 语言库内部，**GIE**（全局中断启用）被禁用，之后（和恢复之前）硬件乘法器被使用。 为了禁用这一操作，与 TI 联系以获得针对这些库的源代码。
9. 可在 **Workbench** 中将汇编程序和 **C**程序合成。 请见由 IAR 提供的《C/C++ 编译器参考指南》的汇编语言接口一章。
10. Workbench 可生成一个采用 TI .TXT 格式的项目。 **C-SPY** 不能输入一个 **TI .TXT** 格式的项目。 在这个情况下，一个错误消息被输出。
11. **Kickstart** 文档中给出的示例程序（即，演示、指导等）是不正确的。 此程序只在模拟器中工作。 由于安全装置机制被激活，在真实器件上，此程序运行不正确。 需要对此程序进行修改以禁用此安全装置机制。 使用这个 C 语言基本指令来禁用安全装置机制：
    ```
    WDTCTL = WDTPW + WDTHOLDM
    ```
    或者此条汇编语言指令：
    ```
    mov.w # WDTPW+WDTHOLD,&WDTCTL
    ```
12. 对于使用一个 **8** 位错误的 **MPY** 的访问会被标志为一个错误。 在 .h 文件内部，16 位寄存器采用的定义方式使得对它们的 8 位操作被标志为一个错误。 这一特性通常是有益的并且能够捕捉寄存器访问违反。 然而，在 MPY 情况下，也可使用 8 位 运算符来访问这个寄存器。 如果用 8 位运算符来访问 MPY，通过使用 "MPY_" 来寻址寄存器可使访问违反检查机制失效。 相似地，对 8 位寄存器的 16 位操作也被标志。
13. **.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。

14. 与所有 **C** 语言应用隐式链接的 **CSTARTUP** 不会禁用安全装置定时器。 使用WDTCL=WDTPW+WDTHOLDM 来明确禁用安全装置。 这条指令最好被放置在 __low_level_init() 函数内，此函数在 main() 之前执行。

    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，源屏幕变成空白，由于 C-SPY 不能定位用于 CSTARTUP 的源代码。 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 __low_level_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 seg_init  */
    /*         1 to run seg_init  */
    /*==================================*/
    return (1);
}
```

15. 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。 优化：在 Project → Options → C/C++ Compiler → Code → Optimizations 内不支持 (NONE)。 或者，变量可被宣称为易变。
16. **IAR** 教程假定 **Workbench** 的一个完全或者 **Baseline** 版本。 在一个 Kickstart 系统中，不能将 C 编译器配置成输出汇编语言助记符指令。
17. 可在全新的 IAR 2.x/3.x 系统中使用一个 IAR 1.x 系统中的现有项目；请见针对 EW430 x.x 的 IAR 文档迁移指南。 这个文档位于 <安装根目录>\Embedded Workbench x.x\430\doc\migration.htm
18. 汇编语言项目必须参考代码段(RSEG CODE)来使用 **Linker → Processing → Fill Unused Code Memory**机制。 无需特别步骤即可使用 Linker → Processing → Fill Unused Code Memory with C projects。
19. 确保为纯 **C** 语言和混合 **C** 语言和汇编语言项目选择适当的 **C** 语言运行时间库 (Project → General Options → Library Configuration → Library)。 对于纯汇编语言项目，一定不能链接运行时间库，否则会链接错误，并导致生成失败。（例如，RESET 矢量被分配两次）。
20. 大量的 **C** 语言和 **C++** 运行时间库与**Workbench** 一起提供。

    cl430d：C，64 位双精度

    cl430dp：C，64 位双精度，浮动地址

    cl430f：C，32 位双精度

    cl430fp：C，32 位双精度，浮动地址

    dl430d：C++，64 位双精度

    dl430dp：C++，64 位双精度，浮动地址

    dl430f：C++，32 位双精度

    dl430fp：C++，32 位双精度，浮动地址

    与使用何种库相关的信息请见 IAR MSP430 C/C++ 编译器参考指南。

A.3 调试中 (C-SPY)

1. **使用 C-SPY 进行调试时，如果看上去没有影响到外部连接的 MSP430 器件。** 如果情况是这样的话，检查主调试器菜单条中是否包含一个名为 Simulator 的菜单项。 如果有的话，正在运行的是 C-SPY MSP430 器件的内核仿真器，而没有建立与实际目标器件的真实通信。 **解决方案：确保 C-SPY 驱动器被设定为 FET 调试器** (Project → Options → Debugger → Setup → Driver)。
2. **C-SPY 报告不能与此器件进行通信时。** 解决这个问题的可能的解决方案包括：
   - 确保选择正确的调节接口；使用 Project → Options → FET Debugger → Connection。
   - 在并行端口 MSP-FET430PIF 接口被使用时，确保在 C-SPY 配置中指定了正确的并行端口（LPT1，2，3）；使用 Project → Options → FET Debugger → Connection → Parallel Port → LPT1 （缺省）或者 LPT2 或者 LPT3。 检查 PC BIOS 以获得并行端口地址 (0x378，0x278，0x3bc)，和并行端口配置（ECP，Compatible （兼容），Bidirectional （双向），或者 Normal （正常））（请见 FAQ 调试中 #8）。 对于使用 IBM ThinkPad™ 计算机的用户，即使操作系统报告称并行端口位于 LPT1，也请尝试使用端口 LPT2 和 LPT3。
   - 确保在目标方硬件上正确配置跳线设置。
   - 确保没有其它的软件应用已经保留或控制 COM 或并行端口（例如，打印机驱动程序，ZIP 驱动的驱动程序等）这种情况发生在使用 MSP-FET430PIF并行端口 接口时。 这样一个软件会防止 C-SPY 或 FET 驱动程序访问并行端口并因此禁止与器件的通信。
   - 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 或并行端口是否被 Windows 操作系统 (OS) 成功识别。
   - 也许有必要重新启动计算机来完成所需端口驱动程序的安装。
   - 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**
>
> **有可能对器件造成的损坏**
>
> 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。

3. **C-SPY 报告器件的 JTAG 熔丝被熔断。** 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。 这会启动一个 MSP430 中的熔丝意外检查并会在 JTAG 安全熔丝没有熔断的情况下被认为已经熔断。 这一情况常见于 MSP-FET430PIF 和 MSP-FET430UIF，但是主要见于 MSP-FET430UIF 上。
   权变措施：
   - 将器件 $\overline{\text{RST}}$/NMI 引脚接至 JTAG 头（引脚 11），MSP-FET430PIF 或 MSP-FET430UIF 接口工具能够拉 $\overline{\text{RST}}$ 线路，这也将器件内部熔丝逻辑复位。
   - 不要同时连接 JTAG 头的 V_{CC}工具（引脚 2）和 V_{CC}目标（引脚 4）并且也不要为调试器中的 V_{CC}指定一个等于外部电源电压的值。
4. **C-SPY 能够下载数据到 RAM，信息块，和闪存主存储器。** 如果试图向器件内存空间之外下载数据的话，会输出一个警告消息。
5. **C-SPY 能够调试采用中断和低功耗模式的应用（请见 FAQ** 调试中 #26）。
6. **器件运行时，C-SPY 不能访问器件寄存器和内存。** C-SPY 显示 "-" 来标明一个寄存器或内存文件无效。要访问器件寄存器和内存，用户必须将器件停止。 然后任一显示的寄存器或内存域被更新。

7. 当 **C-SPY** 被启动时，闪存存储器被擦除并且打开的文件被编程，编程方式与在 Project → Options → FET Debugger → Download Control 中设定的下载选项一致。 这个初始的擦除和编程操作可通过选择 Project → Options → FET Debugger → Download Control → Suppress Download 来禁用。 通过使用 Emulator → Init New Device 可以手工初始化闪存编程。
8. 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，Compatible，Bidirectional，Normal) 的配置并不重要；对于解决 C-SPY 和器件间的通信问题的附加提示，ECP 看上去工作的还不错（请见 FAQ 调试中 #1）。
9. 当 C-SPY 被启动并且当器件被编辑时，**C-SPY 将RST/NMI 置为有效来使器件复位**。 当器件被手工重新编程 (Emulator → Init New Device)，当 JTAG 被重新同步时 (Emulator → Resynchronize JTAG)，此器件也可由 C-SPY RESET（复位）按钮复位。 当RST/NMI 未被置成有效（低电平），C-SPY 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 C-SPY 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。

   在 C-SPY 内，Emulator → "Power on" Reset 为目标方循环供电以生成一个加电复位。
10. **C-SPY 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。**
11. 当 **C-SPY** 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。
12. 当对器件进行电流测量时，请确保 **JTAG** 控制信号被释放 (Emulator → Release JTAG on Go)，否则此器件将由 JTAG 引脚上的信号供电，而测量值也将是错误的（请见 FAQ 调试中 #14）。
13. 大多数 **C-SPY** 的设置（断点等）在会话间被保留。
14. 当 **C-SPY** 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将在 STEP 或者 GO 之前被恢复。 因此，在 C-SPY 控制此器件时不要测量器件的功耗。 相反的，在 JTAG 被释放时，运行您使用 GO 的应用（请见 FAQ 调试中 #12）。
15. C-SPY 的 View → Memory → Memory Fill 对话要求十六进制的值用于起始地址、长度，这个值应该以**"0x"** 作为开始。 否则这些值将被认为是十进制。
16. C-SPY 的内存调试视图 (View → Memory) 可被用于查看 RAM，信息内存，和闪存主存储器。 C-SPY 的内存工具可被用于修改 RAM；而内存工具不能修改信息内存和闪存主内存。 信息内存和闪存主内存只有当一个项目被打开且数据被下载到器件上时才可被编辑，或者当 Emulator → Init New Device 被选择时。
17. **C-SPY** 不允许信息内存和闪存主内存的独立段被分开操作；认为信息内存是一个不间断内存，而闪存主存储器为一个第二个不间断内存。
18. 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。
19. 调试期间，C-SPY 利用系统时钟来控制此器件。 因此，当 **C-SPY** 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，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。

    支持时钟控制的器件 (Emulator → Advanced → Clock Control) 通过在调试期间选择停止时钟能够进一步减少这些影响（请见 FAQ 调试中 #24）。

20. 在 **C-SPY** 执行器件复位后有一段时间此时间段处于 [当 C-SPY 会话被首先启动，当闪存被重新编程（通过 Init New Device），并且当 JTAG 被重新同步 (Resynchronize JTAG)] 以及在 C-SPY 重新获得对器件的控制之前此器件正常执行代码。 这一运转状态也许有负面影响。 一旦 C-SPY 重新获得对器件的控制，它执行一个器件复位并重获控制权。
21. 当对闪存编程时，不要在写入闪存操作之后的指令上设置断点。 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 之后的指令上设置一个断点（请见 FAQ 调试中 #23）。
22. 转储内存长度限定符被限制为四个十六进制位（**0** 至 **FFFF**）。 这样将可被写入的字节数量限制在 0 至 65535 之间。 因此，不能在包括 0 至 0xFFFF 在内的区间内写入 内存，因为这要求限定符的长度为 65536（或者 10000h）。
23. 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 C-SPY。 因此，**C-SPY** 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（请见 FAQ 调试中 #21）。
24. 正常程序执行期间，读取时被清除的外设位（也就是说，中断标志）被清除，当读取的同时被调试时时如此（即，内存转储、外设寄存器）。

    当使用特定的 MSP430 器件时（诸如 MSP430F15x，MSP430F16x，MSP430F43x 和 MSP430F44x 器件），位的运行方式不是如此（也就是说，这些位不由 C-SPY 读取操作清除）。
25. **C-SPY** 不能用于调试在 **MSP430F12x** 和 **MSP430F41x** 器件的 **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。
26. 当单步执行时，发生了已使能的中断，在中断服务例程 **(ISR)** 有效时，会出现不执行代码（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，这样处理是正确的，器件总是在处理非 ISR（也即是说，主程序）代码之前优先处理一个有效且被使能的中断。 对于这个运转状态的权变措施是，在 ISR代码内部，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。

    在带有时钟控制仿真特性的器件上，有可能在单步操作之间挂起一个时钟并延迟一个中断请求 (Emulator → Advanced → Clock Control)。
27. 观察窗口变量的基数（十进制、十六进制等）属性在 **C-SPY** 回话之间并不保留；此基数返回缺省格式。
28. 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 (GO) 至这个断点。
29. 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，当执行一个单步指令或者对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。
30. C-SPY寄存器窗口支持指令周期长度计数器。 只有在单步操作的同时周期计数器才被激活。 当器件复位或者器件运行 (Go) 时，此计数复位。 可在任一时候对此计数器进行编辑（通常设定为零）。
31. 有可能使用 **C-SPY** 来获得对正在运行且状态未知的器件的控制。 简单使用 C-SPY 来编辑一个虚拟设备，然后通过在选择的 Go 上的释放 JTAG 来启动应用。 从虚拟设备上移除 JTAG 连接器并将其连接至未知器件。 选择 Debug → Break（或者 Stop （停止手型图标）来停止未知器件。 然后可以询问器件的状态。

32. 如果 Project → Options → Debugger → Setup → Run To 被启用，临时复位一个程序要求一个断点。 如果 N 个或者更多断点被设定，Reset 会设定一个虚拟断点并且运行到 Run To 函数。 因此，也许在程序复位前需要大量的时间（也就是说，在 Run To 函数上停止）。 在这段时间内，C-SPY 表明程序正在运行并且 C-SPY 窗口也许为空（或者未被正确升级）。
33. 运行至光标处 (Run to Cursor) 操作需要一个临时断点。 如果 N 个断点被设定并且虚拟断点被禁用的话，**Run To Cursor** 不正确地使用了一个虚拟断点。 这将导致非常慢的程序执行。
34. 模拟器只是一个 **CPU** 内核模拟器；外设未被模拟，并且中断为统计事件。
35. 在没有数据断点功能的器件上，C-SPY会将断点与一个断点指令表达式进行关联并进行判断(仲裁)。 这一机制可被用于合成一个数据断点。 对于这个复杂断点机制的说明请见 C-SPY 文档。
36. **C-SPY** 文档提及的 **ROM** 监控器只适用于老款的基于 MSP430Exxx (EPROM)的器件；当使用 FET 和基于闪存的 MSP430F 器件时，它可被忽略。
37. 特定功能寄存器(SFR) 和外设寄存器显示在 **View → Register** 中。
38. **putchar()** 和 **getchar()** 断点只有当这些函数出现时才被设定（并且此机制被启用）。 请注意 putchar() 和 getchar() 可被一个库函数间接引用。
39. 闪存程序和下载进度条并不是逐步更新的。 这一运行状态将会出现。 只要一个“大块”内存被写入闪存，进度条就会更新。 开发工具试图大大减少程序块的数量以大大提升编程效率。 因此，例如，有可能将一个 60K 字节的程序缩减为一个单一块，并且直到整个写入操作完成时，进度条才被更新。
40. 在将一个完整的 **EW430** 项目（其中包括工作区，项目，源和生成的目标文件）迁移至一个不同的存储位置后（例如，另外一台 PC），在启动 C-SPY 之前，需要重建目标文件（重建项目）。 连接器在目标文件中存储绝对路径名称，它有可能与新的存储位置不匹配。 C-SPY 会显示一个消息，称源文件不能被定位或者可在调试期间显示其它陌生的结果。

FET 专用菜单

Topic Page

B.1 菜单

B.1.1 Emulator → Device Information

打开一个显示有正在使用的目标器件信息的窗口。 此外，通过在窗口中点击右键，这个窗口在一个 MSP-FET430UIF 接口为目标方供电的情况下允许调整目标方电压。 可在 1.8V 至 5V 的范围内对输出电压进行调整。 这个电压在 14 引脚目标方连接器的引脚 2 上提供以从 MSP-FET430UIF 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 MSP-FET430UIF 能够相应地设置输出信号的电平。

B.1.2 Emulator → Release JTAG on Go

C-SPY 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，C-SPY 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 Release JTAG On Go 时，JTAG 驱动器被设定为三态，并且当 Go 被激活时器件从 JTAG 控制中被释放（TEST 引脚被置为接地 (GND)）。 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，C-SPY 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）。 C-SPY 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 FAQ 调试中 #12）。

B.1.3 Emulator → Resynchronize JTAG

重新获得对器件的控制。

不能在器件运行时重新同步 JTAG。

B.1.4 Emulator → Init New Device

按照 Download Options（下载选项）中的配置初始化器件。 基本上，当前的程序文件被下载至器件内存。 然后器件被复位。 这个选项可被用于使用来自同一 C-SPY 回话内部的同一程序编辑多个器件。

器件运行时不能选择 Init New Device（初始化新器件）。

B.1.5 Emulator → Secure - Blow JTAG Fuse

在目标方器件上熔断熔丝。 熔丝被熔断后，不能与器件继续通信。

B.1.6 Emulator → Breakpoint Usage

列出了所有使用的硬件和虚拟中断，以及所有当前被定义的 EEM 断点。

B.1.7 Emulator → Advanced → Clock Control

当 C-SPY 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（STEP 或 STEP INFO）之后，所有的系统时钟被启用（请见 FAQ 调试中 #19）。

B.1.8 Emulator → Advanced → Emulation Mode

指定将被仿真的器件。 在变为仿真模式之后，此器件必须被复位（或者通过 Init New Device 来重新初始化）。

B.1.9 Emulator → Advanced → Memory Dump

将特定器件内存的内容写入一个指定的文件中。 一个传统对话框被显示，使得用户能够指定一个文件名、一个内存起始地址、和长度。 然后编址存储器被以文本文件的格式写入一个已命名的文件中。 这些选项使得用户能够选择字或者字节文本格式，并且地址信息和寄存器内容也可被附加到文件上。

B.1.10 Emulator → Advanced → Breakpoint Combiner

打开 Breakpoint Combiner（断点组合器）对话框。 Breakpoint Combiner 对话框允许用户指定断点从属关系。 当断点在规定的先后顺序内出现时，一个断点会被触发。

B.1.11 Emulator → State Storage Control

打开 State Storage 对话框。 State Storage 对话框使用户能够使用状态存储模块。 状态存储模块并不出现在所有 MSP430 派生器件上。 有关实施的详细信息，请参阅表 2-1。

请见 MSP430 IAR 嵌入式工作台 IDE 用户指南的 IAR C-SPY FET 调试器部分。

B.1.12 Emulator → State Storage Window

打开 State Storage（状态存储）窗口，并显示由 State Storage 对话配置的已存储的状态信息。

请见 MSP430 IAR 嵌入式工作台 IDE 用户指南的 IAR C-SPY FET 调试器部分。

B.1.13 Emulator → Sequencer Control

打开程序控制器对话框。 程序控制器对话框允许用户配置程序控制器状态机。

请见 MSP430 IAR 嵌入式工作台 IDE 用户指南的 IAR C-SPY FET 调试器部分。

B.1.14 Emulator → "Power on" Reset

对器件的循环供电以引起一个复位。

B.1.15 Emulator → GIE on/off

启用或者禁用所有中断。 需要在 Go 之前手工恢复。

B.1.16 Emulator → Leave Target Running

如果 C-SPY 被关闭，目标方继续运行用户程序。

B.1.17 Emulator → Force Single Stepping

在 Go 上，程序单步执行。 只用在这个模式下，周期计数器才能正常工作。

注: Emulator → Advanced 菜单的可用性

不是所有的 MSP430 器件都支持全部的 Emulator → Advanced 菜单。 这些菜单

修订历史记录

版本	注释
SLAU138AC	表 2-1，已添加针对 MSP430F523x，F524x，F525x（请见 *MSP430F52xx*）和 i2040（请见 *MSP430i20xx*）的仿真特性。 2.2.2 节，已添加 MSPMathlib 支持。
SLAU138AB	已添加针对 MSP430TCH5E 的仿真特性。
SLAU138AA	已在表 2-1中添加针对 MSP430F535x，F565x 和 F645x 的仿真特性。
SLAU138Z	已添加针对 MSP430SL5438A 的仿真特性。 已在表 2-1中更新针对 MSP430F665x 和 MSP430F67xx 的调试支持。
SLAU138Y	已添加针对 MSP430FR59xx 和 MSP430F665x 的仿真特性。 已更新表 2-1中的 LPMx.5 调试支持。
SLAU138X	德州仪器 (TI) 提供的相关文档，已添加 CC430 文档. 表 2-1，已添加 CC430F512x，CC430F514x，CC430F614x，MSP430G22xx。
SLAU138W	已更新 表 2-1。
SLAU138V	已更新 2.2.5 节 和 2.2.6 节。 已更新 表 2-1。
SLAU138U	已更新 表 2-1
SLAU138T	已更新 表 2-1
SLAU138S	已添加 2.2.3 节。 已更新 表 2-1中的 MSP430x092 和 MSP430F438 以及 MSP430F439。
SLAU138R	已为 MSP430G2xxx，MSP430F51x1，MSP430F51x2，MSP430F550x，MSP430F5510，MSP430F551x，MSP430F552x，MSP430F663x 添加仿真特性。
SLAU138Q	已在 1.1 节中更新支持的操作系统。
SLAU138P	已在 表 2-1中更新针对 MSP430F461x 和 MSP430F461x 的特性。
SLAU138O	已删除附录 C：80 引脚 MSP430F44x 和 MSP430F43x 器件仿真
SLAU138N	已添加与 MSP430F5xx 和 CC430 器件相关的信息
SLAU138M	• 已删除与硬件相关的信息。 它被移至《*MSP430 硬件工具用户指南*》（SLAU278）。 • 已在 表 2-1中添加用于 MSP430F55xx 和 MSP430F54xxA 的仿真特性。 • 已更新和扩展 表 2-1架构信息。
SLAU138L	• 已在部分 1.7 中添加 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）。 • 已在表 2-1中为 CC430F513x，CC430F612x，CC430F613x，MSP430F41x2，MSP430F47x，MSP430FG479，和 MSP430F471xx 添加仿真特性。 • 已更新 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。 • 已删除全部与 MSP-FET430Pxx0 和 MSP-FET430X110 工具包有关的信息。
SLAU138K	• 已添加 MSP-FET430U5x100 工具包和 MSP-TS430PZ5x100 目标方插槽命令电路原理图。 • 表 1-1 为添加的调试接口概述。 • 已添加 eZ430-F2013，T2012，和 eZ430-RF2500。
SLAU138J	• 已更新部分 1-7。
SLAU138I	• 已添加 MSP-TS430PW28 目标方插槽模块，电路原理图（图 B-5）和 PCB（图 B-6）。 • 已更新 MSP-FET28U1 工具包内容信息（支持 DW 或者 PW 封装）（部分 1-7）。 • 针对 MSP430F21x2 的仿真特性被添加至表 2-1中。 • 已更新 MSP-TS430PW14 目标方插槽模块电路原理图（图 B-1）。 • 已更新 MSP-TS430DA38 目标方插槽模块电路原理图（图 B-7）。
SLAU138H	• 已更新附录 E
SLAU138G	• 已在表 2-1 中添加用于 MSP430F22x2，MSP430F241x，MSP430F261x，MSP430FG42x0 和 MSP430F43x1 的仿真特性。 • 已在附录 B 中更新图 B-15 和图 B-16

版本	注释
SLAU138F	• 将 MSP-FET430U40 重命名为 MSP-FET430U23x0。 • 已用重命名的 MSP-FET430U23x0 图替代 MSP-FET430U40 电路原理图和印刷电路板 (PCB) 图。 • 已在部分 A.1 中添加 FAQ 硬件 #2。 • 已在部分 A.3 中添加 FAQ 调试中 #3。

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

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

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